Volume 29 Number 4 November 2005 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

Hot Topics in European Agent Research I

Guest Editors:

Andrea Omicini

Paolo Petta

Matjaž Gams

The Slovene Society Informatika, Ljubljana, Slovenia

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

The Second AgentLink III Technical Forum: Main Issues and Hot Topics in European Agent Research

1 Introduction

Together, this and the next issue of *Informatica* present a collection of articles on the edge of agent research in Europe. The papers in this double special issue arise from scientific exchanges and debates that took place at the Second AgentLink III Technical Forum (AL3-TF2) hosted by the Jozef Stefan Institute in Ljubljana, Slovenia, from February 28 to March 2, 2005.

AgentLink III is a European Commission (EC)-sponsored Coordination Action (Project number: IST-FP6-002006CA) to support research and development in agent-based technologies and to strengthen Europe's efforts in this domain. This two-year project (2004-2005) is funded through the Semantic-based Knowledge Systems area of the Information Society Technologies (IST) Thematic Priority of the Sixth Framework Programme (FP6). AgentLink III follows the Thematic Networks AgentLink (1998-2001) and AgentLink II (2001-2003) of the Fifth Framework Programme. Its Management Committee comprises academic and industrial representatives from across the European agent technology community. To support this leadership, the project established in early 2004 a system of membership by which institutions active in agent research or development could apply to join the project. By 31 August 2005, 192 organisations from 21 European and associated states had become members of AgentLink III, out of which there were 125 Universities, 30 Research Institutes and 37 private companies. This high level of membership indicates the considerable support for the project from European organisations, both public and private.

2 AgentLink III Objectives

The long-term goal of AgentLink III is to put Europe at the leading edge of international competitiveness in the increasingly-important area of agent technologies. To realise this goal, AgentLink III has sought to achieve the following objectives:

- To gain competitive advantage for European industry by promoting and raising awareness of agent systems technology.
- To support standardisation of agent technologies and promote interoperability.
- To facilitate improvement in the quality, profile, and industrial relevance of European research in the area of agent-based computer systems, and draw in relevant prior work from related areas and disciplines.
- To support student integration into the agent community and to promote excellence in teaching in the area of agent-based systems.
- To provide a widely known, high-quality European forum in which current issues, problems, and solutions in the research, development and deployment of agent-based computer systems may be debated, discussed and resolved.
- To identify areas of critical importance in agent technology for the broader IST community, and to focus work in agent systems and deployment in these areas.

Further information about AgentLink III and its many activities is available from the AgentLink website at `http://www.agentlink.org`.

3 AgentLink III Technical Fora

In order to support co-ordination and collaboration of European research efforts, AgentLink III established a series of research meetings, called the AgentLink III Technical Fora (TFs). These comprise a number of parallel workshops, called Technical Forum Group (TFG) meetings, on topics suggested in response to a call for proposals issued before each Technical Forum. Soliciting topics for TFGs in this way ensures that the meetings retain flexibility, and can reflect whatever is the current focus of research attention in the agents community. This feature also implies that the standard for acceptance can be quite high, with proposers needing to show evidence of research co-ordination activities before, during, and after each Technical Forum. Examples of such activities include the establishment of persisting web-sites and discussion forums, the production of short and long reports of the activities at the event, integration of related activities, and development of joint survey papers of the respective research area.

Three Technical Fora were organised under AgentLink III:
– TF1: Rome, Italy: 30 June–2 July 2004.
– TF2: Ljubljana, Slovenia: 28 February–2 March 2005.
– TF3: Budapest, Hungary: 15–17 September 2005.
Over a hundred participants have registered for each TF meeting, with attendance not only from European and neighbouring countries, but also Australia, Japan, and the USA.

Each Technical Forum supported between six and nine TF Groups, with most of them meeting for a whole day, and often mixing with other groups in planned, but also spontaneously arranged joint meetings on research topics of common interest. Given that AgentLink seeks to reach out and establish links with related research disciplines and with other research projects, special efforts have been aimed at encouraging the formation of TF Groups which make connections between the agents community and other communities. For instance, there have been TF Groups which have looked at the intersection of agent technologies and the law; biology and bio-informatics; and economics. In addition, joint TF Groups have been held with two related EC-funded projects, *KnowledgeWeb*[1] and *ASPIC*[2].

Further information about AgentLink III Technical Fora and their articulation is available from the AgentLink website at `http://www.agentlink.org/activities/al3-tf/`.

4 Main Issues and Hot Topics in European Agent Research

The double special issue of *Informatica* comprises reviewed invited and selected articles from the TFG meetings held at the Second AgentLink III Technical Forum (AL3-TF2). The aim of this collection is to provide a deep and coherent overview of the main issues and the hottest topics in European research on agent technologies, as they emerged from the work of the AgentLink III TFGs.

To this end, the TFG Chairs were asked to produce first of all a survey of the main issues as they arose in the work and scientific debate at their TFG meetings, and then possibly to invite some of the most prominent participants to illustrate the hottest topics in their specific field of discussion. Both the surveys and the "hot topics" articles were then scrutinised and carefully reviewed by several European experts, who helped the Guest Editors to select the best papers among the many submitted, and thereby contributed significantly to the improvement of the accepted papers. As a result, five surveys from among the most active TFGs were selected for these Special Issues, with the aim of giving a general overview of the current activities and challenges in a number of key areas of agent technologies, and also of revealing the breadth and sophistication of current research and development in Europe. In addition, seven other "hot topic" papers from four of these TFGs were also included, to provide readers of *Informatica* with in-depth insights in some of the most controversial and potentially fertile sub-areas in the agent field.

The contributions collected in the first number of the special issue document work in the TFGs on Agent-Oriented Software Engineering (TFG-AOSE) and Environments for Multiagent Systems (TFG-ENV), and are described briefly in the following. The articles in the second number of the double special issue (30(1)) were provided by the TFGs on Multiagent Resource Allocatoin (TFG-MARA), Programming Multi-Agent Systems (TFG-PROMAS), and Self-Organisation in MAS (TFG-SELFORG).

Agent-Oriented Software Engineering (TFG-AOSE)
An Overview of Current Trends in European AOSE Research, by Carole Bernon, Massimo Cossentino, and Juan Pavón, provides an "aerial" perspective on current research trends in Agent-Oriented Software Engineering (AOSE), with a specific attention to the results coming from European research groups.
Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS, by Ivan Trencansky and Radovan Cervenka, focuses on a semi-formal visual modelling language for specifying, modelling, and documenting systems that incorporate agent features.
The Tropos *Metamodel and its Use*, by Angelo Susi, Anna Perini, John Mylopoulos, and Paolo Giorgini, illustrates the Tropos metamodel from the basic concepts of actor, goal, plan, resource, and social dependency, and discusses its use through an extension toward security.

Environments for Multiagent Systems (TFG-ENV)
On the Role of Environments in Multiagent Systems, by Danny Weyns and Tom Holvoet, suveys MAS research on the long overlooked issue of the role and ontological status of the environment of MAS, and presents a model for MASs that alongside agents puts forward the environment as first-order abstraction.
Towards a Unified View of the Environment(s) within Multi-Agent Systems, by Abdelkader Gouaïch and Fabien Michel, challenges the single-environment hypothesis by allowing for multiple occurrences of the agent-environment relationship within individual MAS.
Coordination Artifacts: A Unifying Abstraction for Engineering Environment-Mediated Coordination in MAS, by Alessandro Ricci and Mirko Viroli, proposes the notion of coordination artifact as a unifying abstraction for engineering environment-based coordination of agents.

Acknowledgements

A warm thanks goes to the AL3-TF2 TFG Chairs who have so actively cooperated in inviting, selecting, and collecting the papers for these Special Issues, and to the reviewers, who have contributed so much to the improvement of the scientific level of every paper in this collection.

We wish to thank all of those who have contributed to the success of the AgentLink III Technical Fora; especially the AgentLink III staff: Catherine Atherton, Becky Earl, Adele Maggs and Serena Raffin. We also thank the AgentLink III Management Committee for their support, and the local organisers and their staff in Ljubljana, Slovenia. We also thank Cristiano Castelfranchi, Peter McBurney, and Laszlo Varga who, along with us, have served as Technical Forum Chairs for AgentLink III.

We hope you will enjoy this—and look forward to the second!—part of the double special issue.

Andrea Omicini, Paolo Petta and Matjaž Gams

[1]See `http://knowledgeweb.semanticweb.org/`.
[2]See `http://www.argumentation.org/`.

An Overview of Current Trends in European AOSE Research

Carole Bernon
IRIT – University Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse, France
E-mail: bernon@irit.fr, http://www.irit.fr/SMAC

Massimo Cossentino
ICAR-CNR, National Research Council, Viale delle Scienze, ed. 11, 90128 Palermo, Italy
E-mail: cossentino@pa.icar.cnr.it, http://www.pa.icar.cnr.it/~cossentino

Juan Pavón
Fac. Informática, Universidad Complutense Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain
E-mail: jpavon@sip.ucm.es, http://grasia.fdi.ucm.es/jpavon

Keywords: Agent Oriented Software Engineering (AOSE), Agent oriented methodologies, Multi-Agent Systems

Received: June 31, 2005

The agent oriented approach is doing great steps towards its (not yet reached) maturity; from a software engineering point of view, it is today positively used for the analysis and design of complex systems. In this paper, which is related to the activity of the AgentLink AOSE TFG (Agent Oriented Software Engineering Technical Forum Group), we provide a perspective of current research trends in this area with a specific attention for results coming from European groups. We start with a discussion of what are agents, specially from the perspective of the software engineer. We present recent trends in modelling agents and multi-agent systems, and then we review the different activities in the agent development process: from analysis and design to implementation, verification and finally testing.

Povzetek: Podan je povzetek evropskega raziskovanja AOSE.

1 Introduction

With the increasing amount of successful applications and techniques based on the agent paradigm, which have validated the feasibility of the approach, there is a big concern on its applicability in an industrial context. This implies the definition of repeatable, reusable, measurable and robust software process and techniques for the development of multi-agent systems (MAS). For this reason, a lot of effort in the agent field has been devoted to the definition of methods and tools for supporting agent oriented software engineering (AOSE). This involves the definition of modelling languages for the specification of MAS, techniques for requirements elicitation and analysis, architectures and methods for designing agents and their organizations, platforms for implementation and deployment of MAS, and validation and verification methods. Taking into account the diversity of influences in the agent paradigm (from distributed objects to knowledge base systems, but also from other fields such as Psychology, Biology and Social sciences) there are many methodological approaches, which should get unified and integrated in a common body of knowledge and practices. This is one of the aims of current actions at EU level, such as the AgentLink (www.agentlink.org) effort, or the collaboration in standardization organizations such as FIPA (www.fipa.org).

In this paper we try to provide a perspective of current research trends in this area, specially in EU groups. This can be useful as a starting reference point to look for specific matters (in this sense there is an extensive bibliography), and is complemented in relevant topics with other papers in this special issue.

The paper starts with a discussion of what are agents, specially from the perspective of the software engineer (section 2). This is followed by a presentation of trends in modelling this kind of systems (section 3). Then, different activities in the development process for MAS are reviewed: analysis and design (section 4), implementation (section 5), verification and testing (section 6). Finally, the conclusions (section 7) provide a view, from the authors of this paper, on what lines of work and trends should follow research in this area.

2 From Objects to Agents and Multi-Agent Systems

When dealing with the agent notion and how to engineer agent-based applications, one question often arises: may agents be considered as an extension of objects and then classical object-oriented software engineering be used as well to build agent-based applications? Several papers have tried to answer this question [76][106], others have compared agents with programs [46] or with components [7]. Many authors agree on the fact that distinguishing agents and objects is difficult because they share some aspects, but they also differ, mainly on notions such as autonomy and interaction. Both agents and objects encapsulate their state, which in objects is determined by the values of a set of variables whilst in agents this can be defined in terms of goals, beliefs, facts, etc., what determines a *mental state*. Objects may have control over their state by using private attributes or methods but any public method of an object can be invoked by another object forcing the former one to perform the action described by the method. An object, contrary to an agent, has then a limited control over its behaviour because the decision on which method to execute is taken by an external actor (the *caller*). An agent can determine which behaviour to follow (depending on its goals, its internal state and its knowledge from the environment) and not because someone else forces it to do something. Therefore, the notion of autonomy is stronger in agents.

This autonomy in agents implies that usually they have their own thread of control, whilst, most of the time, objects are passive entities, becoming active just when one of their methods is invoked by another object. This difference may be alleviated by the notion of active objects in which an object has its own thread of control. However, agents have some features which make them something more than active objects. According to Van Parunak and Odell [76], agents exhibit a *dynamic autonomy* (their behaviour can be *reactive* as they react to changes in their environment, *proactive* as they are able to take initiatives to proceed into goal-directed actions, and *social* as they communicate with other agents in organizations) as well as an *unpredictable autonomy* (their behaviour depends on their state, their individual goals, and their interactions with others). Active objects would become agents if they are able to take "initiatives". However, this distinction is not always well established. For this reason some works in the agent domain, for instance, on formalization of coordination issues, usually are more related to classical concurrency theory and do not consider intentional aspects of agents.

What makes really the difference, according to many authors is the social dimension of agents (for instance, the Huhns-Singh test [58] states that *a system containing one or more reputed agents should change substantively if another of the reputed agents is added to the system)*. Agents cannot be considered in isolation and are social entities, which communicate and interact with other entities that share a common environment. Communication between objects is defined in terms of messages that activate methods, but in the agent domain, this communication is richer both in the diversity of mechanisms and in the language, which is defined at a more abstract level, in terms of ontologies and speech acts, for instance. This social perspective is reflected also in the definition of organizations with social rules and relationships among agents [42].

Therefore, the use of object-oriented software engineering techniques can be applied for the development of MAS, but some extensions are required to deal with social issues (organization, interaction, coordination, negotiation, cooperation), more complex behaviour (autonomy, mental state, goals, tasks), and a greater degree of concurrency and distribution.

2.1 Definition of Agents

In [91], an agent is defined as anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. For Ferber, agents are still plunged into an environment but he endows agents with additional characteristics [41]. An agent becomes then able to communicate directly or not with other agents, it is driven by a set of tendencies, possesses resources of its own, has a limited representation of its environment, possesses skills and can offer services, and may be able to reproduce itself. Its behaviour tends towards satisfying its goals, taking into account the resources and skills available to it in accordance with its perception, its representation and the communication it receives. Depending on the nature of applications in which agents are used, different labels exist for agents [46][77]: agents are qualified as being autonomous, intelligent or mobile, for instance. This plethora of labels makes the term "agent" almost meaningless because it can be used too frequently to characterise anything, so [69] recommends to formally define the notion of agency. In this paper agents are characterized through their essential properties: an agent is able to act, is autonomous, proactive, communicates with others, and perceives its environment[1].

2.2 Definition of Multi-Agent Systems (MAS)

Most of the authors agree on viewing a MAS as a system composed of agents that communicate and collaborate to achieve specific personal or collective tasks. This is related to what was said before, an agent is not an isolated entity but it is only understandable when located in an environment where other agents exist, with which it can interact.

MAS are appropriate to deal with complex and open problems. The organization facilitates managing complexity by determining structures, norms and dependencies. In some cases, the organization is explicitly a subject of analysis and design (e.g., [42] [111]). But in certain approaches, the organization emerges at run time (e.g., [10][36][93]). This allows the analysis of emergent behaviours in systems in which is not easy to know their structure in advance. From the point of view of AOSE, this means that both top-down and bottom-up approaches are feasible when building a MAS, depending on the problem under study.

2.3 MAS Meta-models

Meta-modelling is a means to define concepts used in a system. This can facilitate analysis and design by identifying activities for instantiating the meta-model entities with respect to the target application (i.e., the meta-model identifies which elements should the developer look for, and what relationships and constraints exist for those elements). For instance, Aalaadin defines one of the first meta-models for MAS in terms of three main concepts: Agents, Groups and Roles [42]. With this meta-model, the developer has an organizational-driven approach to build a MAS. An organization is a structural relationship between a collection of agents and is described by a set of interaction modes. Agents are defined by their function in the organisation (Role) and belong to one or more Groups, possibly for gaining some capabilities.

[1] Properties defined during the second meeting of the AgentLink3 AOSE TFG (Ljubljana, February 2005).

Meta-models are also useful to integrate concepts. This is the approach of the MESSAGE project [21], whose aim was to define a methodology for the development of telecom applications using agent technology. MESSAGE adopted concepts and notations from different methodologies in a common framework. Its definition was made using meta-models. Furthermore, these meta-models were used to build graphical editors [51]. In order to cope with complexity of MAS, MESSAGE structured the specification of meta-models in five viewpoints: organization, agent, goals/tasks, domain, and interactions.

In the object world, the notion of object is clearly defined by a set of criteria and almost all developers agree on what makes a system object oriented. Meta-modelling is then possible relying on standard notations such as UML [90]. On the contrary, no universally accepted structural representation of the elements (agent, role, behaviour, ontology, etc.) that compose an actual MAS, with their relationships, exists yet. This has led several existing agent-oriented methodologies to propose their own concepts and system structure illustrated by a particular MAS meta-model. This lack of unification at the MAS meta-model level, and then at the agents concepts level, therefore prevents developers from reusing fragments of existing agent-oriented methodologies to build their own methodology especially dedicated to their needs (this is the methodology composition process suggested by the method engineering approach [17][54]; this proposes to create a new methodology starting from existing methodology parts, called method fragments, that a method engineer defines and stores in a method base).

A further step in this direction would be the standardisation of the process that is necessary to follow in order to build a new methodology. This would be desirable to make agent-oriented engineering used in the industrial world. From this perspective, some initial attempts have been made to find a unified meta-model based on several methodologies [11], or by trying to reach an agreement in the agent community with the work of the FIPA Modelling TC or the AgentLink III AOSE TFG.

3 Modelling Agents

Modelling agents and MAS needs adapted modelling languages, notations and tools. Agents, as said above, are not far from objects and most of the modelling methods are based on tools coming from the object-oriented domain. The most generally accepted modelling language used for object-oriented software engineering is UML. UML is a *de facto* standard, and most modelling tools are already based on it, which facilitates the development of tools. However, UML does not provide all the notation elements to model all the specific features of agents.

UML extension abilities (i.e., stereotypes, tagged values, constraints) have been used to support agent-oriented modelling. For instance, Agent-UML (AUML) [3] extends UML sequence diagrams to specify Agent Interaction Protocols by providing mechanisms to define agent roles, agent lifelines (interaction threads, which can split into two or more lifelines and merge at some subsequent points using connectors like AND, OR or XOR), nested and interleaved protocols (patterns of interaction that can be reused with guards and constraints), and extended semantics for UML messages (for instance, to indicate the associated communicative act, and whether messages are synchronous or not).

Also in the context of AUML there are proposals for extending class diagrams into agent class diagrams [2]. Here an agent class consists of several elements:

- An agent name used to differentiate objects from agents in a diagram and providing an agent with three information: instance, role and class.
- A state description that looks similar to the attribute compartment in class diagrams but expresses well-formed formulae for logical descriptions of the state, it may be used to model beliefs, desires and intentions of agents, for instance.
- Actions that can be reactive or proactive.
- Methods implementing services, as in UML classes.
- Capabilities describing what an agent can do.
- Organisation belonging, which specifies the different groups in which an agent evolves, the roles it plays and under which constraints it evolves in these groups.
- Agent head automata, which define the behaviour of an agent.

AUML is under study in the FIPA Modelling TC, and being modified in order to take into account new features in UML 2.0 [57]. For example, communication between agents are now captured by enhanced sequence diagrams, which become interaction diagrams, in which agents can change their role, add or delete roles during interactions, and notions of loop or break are added to the AND, OR and XOR connectors that were available. AUML is being smoothly introduced as an add-on into different agent-oriented toolkits, such as OpenTool for ADELFE [10] and the INGENIAS Development Kit [82].

Another proposal for agent-oriented modelling as an UML profile is AORML (Agent-Object-Relationship Modelling Language) [104]. Here, agents are considered from two perspectives: external and internal. The external AOR model describes the perspective of an external observer who is watching the agents and their interactions. Agent Diagrams are used to depict the agent types and objects of the domain and their relationships, while interactions are modelled using Interaction Frame Diagrams (possible interactions between two agent types), Interaction Sequence Diagrams (instances of interaction processes) and Interaction Pattern Diagrams (general interaction patterns). An internal model adopts the view of a particular agent to be modelled and depicts, using three kinds of diagrams (Reaction Frame Diagrams, Reaction Sequence Diagrams, Reaction Pattern Diagrams), the world represented by the mental state of this agent.

A more recent extension of UML for MAS is AML, which is described in another paper of this special issue [25]. Two UML profiles for AML are given and enable implementing AML in CASE tools based on UML 1.* or UML 2.0. Furthermore, using these AML profiles, a designer is free to customise AML through the definition of extensions to this language.

There are also approaches based on OPM (Object Process Methodology) [37]. OPM considers processes and objects as equally important classes of things, which together describe the function, structure and behaviour of systems. A single diagramming tool, Object-Process Diagrams (OPDs), is enough for modelling the system. This has been extended in OPM/MAS [99] by taking MAS building blocks from Gaia methodology. For instance, organization, society, platform, rule, role, user, protocol, belief, desire, fact, goal, intention, and service are modelled as OPM objects. And the behavioural concepts such as agent, task, and messaging are modelled using the process concept. Another approach [74], taking inspiration from OPM, allows zooming through different abstraction layers and apply this feature to SODA [79], a methodology that addresses the coordination aspects of agent societies. The analysis of complexity is also considered from the perspective of interactions in [83].

In section 4.3 we discuss how the management of complexity can be also addressed by considering complementary aspects of a MAS.

4 Analysing and Designing Agents

According to Sommerville [97], all the different kinds of software development processes share some fundamental activities. These include specification (consisting in the definition of software functionalities and constraints, i.e., requirements analysis), design and implementation (consisting in the production of the software), validation (where the produced software should be validated against customer requirements) and evolution (the software evolves according to customer new needs). In this section we discuss the first two items of this list: specification analysis and design.

During the specification phase, the designer collects and analyzes the software requirements, which are usually considered from two perspectives: User and System. The latter being the detailed and more technical expression of what the customer specifies in the User Requirements. System Requirements consist of functional (services the software should provide), non–functional (constraints on the services) and domain requirements (coming from the application domain).

Design consists in converting system specifications into an executable system. This is usually achieved by structuring the software into modules, defining the data to be managed and the interfaces between components. Sometimes a specific attention is given to the algorithms that are necessary to solve the problem.

A fundamental contribution in defining the impact of agents in these phases has been argued by Jennings [63] in the sense that agents can be a successful solution for two major problems of contemporary approaches: rigidity of components interactions and limitedness of available system's organizational structures.

In the next sub-sections we present existing contributions in this area (with a specific attention for European ones) according to their key features. Specifically, we consider formal and non formal approaches, multi-views paradigms, agent design life cycles and some other remaining issues.

4.1 Formal approaches

Many authors looked at the problem of analysis and design of agent-oriented systems with a formal approach. This usually includes the adoption of a mathematical formalism to obtain a correct specification of the system to be; the output of a formal method is a formal specification that can be used for implementing the system, verifying its correspondence with user requirements or evaluating the final result [85].

Several of these works adopt a kind of logic (usually a modal logic [96]) to represent the system. As an example, LORA (Logic of Rational Agents) [107] which is founded on a first-order logic, includes a BDI (Belief-Desire-Intention) [87] component (used for the agent architecture), a temporal component (used for specifying the system dynamics), and an action component (used to represent agents' actions). LORA is adopted by MABLE (a language for the design of MAS) that allows an automatic verification of the agent system [108].

Situation Calculus [71] is another expression of this field of research; it is a first-order logic (with some extensions to second-order logic) capable of representing dynamic domains. IndiGolog [34] is a recent implementation of situation calculus, supporting the high-level programming of robotic intelligent agents that can perform online planning and plan execution in partially unknown environments. In IndiGolog (that is part of the GOLOG [67] family), environment dynamics is modelled using situation calculus while the agent behaviour is designed in a procedural way.

Another formal approach is due to M. Luck and M. D'Inverno [69] and it is an application of the Z language [98] to the specification of agents. Z is based on first order predicate calculus with the original introduction of the concept of schema. A schema is composed of a declarative part (declaration of variables and their types) and another part where variables are related and their constraints expressed. Agents in Z are defined within a four-layer hierarchy that includes: entities (inanimate objects with attributes), objects (entities with capabilities), agents (objects with goals), autonomous agents (agents with motivations). In this work the authors take profit of the great number of existing experiences in Z for inheriting a great number of tools that include code production and model checking capabilities. Another approach that uses the Z formalism (and statecharts) can be found in [56].

4.2 Non-formal Approaches

Non-formal approaches to the specification and design of agent systems are mostly based on the use of structured natural language and graphical notations. Among these, for system requirements specification, UML-related diagrams like use-case and sequence diagrams are very common use. These approaches are mainly requirement-oriented and they often aim at capturing system functionalities through a set of heuristics and views.

Several agent-oriented design methodologies perform the specification in this way; they generally include a complete design process, not only system specification aspects. We can fundamentally identify three categories of non-formal specifications: functional-oriented [62] (often adopting use-case diagrams), goal-oriented approaches [103] (that aim at identifying the goals of the system and eventually dividing them among agents), and, finally, role-oriented approaches [65] (they adopt the role as the key abstraction for specifying a MAS, they are often also concerned about designing roles/agents coordination). While the functional and goal-oriented specifications are well-known and widely adopted in the object-oriented context, role-guided specifications are more specific of the agent community.

Functional specifications (mostly looking at European works) are adopted in the PASSI methodology [29] and the ROADMAP [64], an extension of Gaia [109], both of them adopting use-case diagrams.

PASSI starts analysis with use-cases and arrives to code production and testing in an iterative process. It includes an extensive patterns reuse practice and it is conceived to be supported by a specific design tool (PTK), since several of its activities are partially automated.

Identification and modelling of system goals is part of the MESSAGE methodology [21], which is based on a set of meta-models supporting five different views of the MAS: organization, agent, tasks/goals, interactions, and environment. INGENIAS [82] refines and extends these meta-models, and uses them to build support tools for all stages of the development cycle. Furthermore, for requirements elicitation, INGENIAS proposes to base on Activity Theory to analyse intentional and social issues of the system, by providing a set of contradiction patterns that guide the developer in the identification of conflicts in the specification about the agent and the organization goals [47].

Tropos [16] starts from the i* framework [110], which has been developed mainly thinking on information systems, actors, beliefs, commitments and goals are used to model system organization. Tropos uses this requirements analysis approach and incorporates it in a complete process that moves from the specification to detailed design.

One of the key features of agency consists in interaction; we can even note that this is also the fundamental aspect of some standardization attempts coming from FIPA (Abstract Architecture Specification [43]) or OMG MAF (Mobile Agent Facility [78]). As a consequence, many authors devoted their attention to capturing interaction aspects often by modelling agents' roles [65][18].

European methodologies that give a prominent importance to role modelling are Gaia [109], SODA [79] and RICA [94] (but also the cited MESSAGE, INGENIAS, and PASSI).

Gaia has been, probably, the most influent methodology concerning the analysis of the system as a society/organization consisting on a set of roles that are later assigned to agents. Gaia's roles are related with one another, and participate in pre-defined patterns of interactions with other roles. Implementation issues are not dealt in this methodology since considered depending on the chosen deployment agent platform. Although initially Gaia suffered from the limitation of being conceived for closed systems and ignoring the possibility of self-interesting agents, a new release of it [111] included concepts like organizational rules as the way to manage more complex open systems.

SODA (Societies in Open and Distributed Agent Spaces) [79] aims at modelling the behaviour of agent societies (considered as not deducible from the behaviour of single agents) and their environments (that can be open, distributed, dynamic and unpredictable). It has a specific attention for agent interactions (starting from a role model) but does not face the design of the agent's inner structure.

Another methodology that puts in a prominent position roles is RICA (Role/Interaction/Communicative Action) [94]. It integrates relevant aspects of Agent Communication Languages (ACL) and Organisational Models and it is itself based on the concepts of Communicative Roles and Interactions.

Other authors concentrated their efforts to coordination among agents [27][80]. A coordination-based approach should consider system openness, the presence of self-interested agents and MAS social laws that rule the overall behaviour of the agents thus encompassing single-role modelling issues.

Coordination is sometimes pursued by adopting a programmable coordination media (like the MARS system presented in [19]), but other authors specifically conceived their design methodologies for dealing with coordination.

Another interesting methodology specifically conceived for coordination of robotic agents is Cassiopeia [38]. Cassiopeia design process is based on the concept of role, agent, dependency, and group; an agent is seen as a set of roles (there can be individual roles, relational roles and organizational roles). The methodology enumerates several different layers, among them the organizational roles layer describes the dynamics of the groups by defining the roles that the agents have to play to let the group appear. Dependencies among roles can be of three types: functional, resource-based or goal-based and in this sense the methodology partially recalls the already cited i* framework.

Cassiopeia assumes that agents are cooperative and this is the same hypothesis that is behind the ADELFE methodology, which focuses on adaptive MAS [9]. Adaptive software can be profitably used in situations in which the environment is unpredictable or the system is open. Contrary to Cassiopeia, in ADELFE agents are not characterised by roles but by the cooperation rules they follow. These rules are described in a proscriptive way, they express what are non cooperative situations, and make an agent locally decide why and when changing its interactions with others. Cooperation is thus viewed as the engine of adaptation according to the AMAS (Adaptive Multi-Agent System) theory [22].

Other contributions about non-formal agent design come from MaCMAS/UML [84], which is a fragment of methodology devoted to deal with large/complex MAS, and the works on modelling electronic institutions and their norms in Islander [95].

4.3 Multi-view Approaches

Multi-views, multi-perspectives, multi-level approaches base their philosophy on three well-known methods for tackling complexity, already mentioned by Booch [12]: Abstraction, Decomposition, Hierarchy. After all, as it can be deduced from the discussion in sections 2 and 3, agent-oriented systems can be more complex than object-oriented ones and therefore a well structured way to manage this complexity is necessary.

The structuring of a MAS in several viewpoints appears in many methodologies. One of the first to propose this was Vowels Engineering, which has been the basis for the MAGMA approach [35]. It considers the five Latin vowels (initially only the first four): Agent, Environment, Interactions, Organization, and User. Different techniques can be applied to analyse and design each aspect. Agents can be conceived as simple automata or complex knowledge-based systems. Interactions can be studied as physical models, e.g., wavelength propagation, or as speech acts. Organizations can be inspired in biological models or ruled by sociological models. The purpose of this methodology is to consider component libraries that provide solutions for each aspect, so that the designer can instantiate an agent model, an organization model, and so on. The methodology proposes to consider vowels (aspects) in a certain order, depending on the kind of system being developed. For instance, if social relationships are important, the development process should start with the organization. If the process starts with agents, then the system will have an organization that probably emerges as a result of the interactions of individual agents. These viewpoints have been applied similarly in the MESSAGE [21] and INGENIAS methodologies [81], which redefine viewpoints as organization, agent, domain/environment, goals/tasks, and interactions.

The concept of level in agency is also another way of considering several views. It has been initially introduced by Newell [75] and Jennings [63] recalled the knowledge level and complemented it with a new social level. The knowledge level is concerned with the agent seen as an asocial problem solver while the social level looks at the agent organization as its main focus.

Other works in this direction presented different *perspectives* [28][32], which are more directed to the representation of the system from a different point of view (architectural, social, knowledge, computer, resource, autonomy) rather than a different level of abstraction.

Other examples of methodologies that emphasize the modelling of the MAS from different viewpoints are MAS-CommonKADS [60] (organization, tasks, experience, agents, communications, coordination, and design), ODAC [50], which uses the five ODP viewpoints (enterprise, information, computational, technology and engineering) [61], and MASSIVE [68] (that includes seven views: environment, task, role, interaction, society, architectural, system).

4.4 Agent Design Life Cycle Models

The whole set of activities and phases needed to develop and maintain a software system is usually addressed as a *Software (Engineering or Development) Process*. Fuggetta in [48] defines it as “the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain (evolve) a software product”, sometimes this is also known as a *Software Life Cycle Process* [59]. Usually the sequence of phases (here we mean high level activities or set of activities) that compose a Software Process is ruled by a software life cycle model. Examples of software life cycle models are the waterfall model, the prototyping model, the evolutionary development, the incremental/iterative delivery, the spiral model, and so on.

A classification of many agent-oriented methodologies according to the software life cycle model they adopt, can be found in [24]. The paper remarks that current research in the area of AOSE methodologies underestimate the importance of the process model in the development of MAS; according to the authors, this is confirmed by the fact that in many cases, AOSE methodologies do not make explicit reference to the underlying process model. Anyway, most of them propose iterative and incremental development process in the same way as the Unified Process.

Some novelties about life cycle models for agents come from the application of the Extreme Programming [5] and Agile Manifesto [4] principles to agents. Proposed design approaches [26][66] seem to show that besides the respect for the main principles of this research stream (attention for code rather than documentation, central role of customer, and so on) a fundamental importance in MAS agile design is played by its ontological aspects (both of the cited approaches give great importance to drawing some ontological models of the problem domain).

4.5 Other Issues In Designing Agents

Despite of the number of works we have discussed, we are still leaving apart some specific areas. These for instance include the design of Internet specific applications by means of agents (see [112]); the importance of this field is growing up in conjunction with the studies on web-services [72] (and their extensions to agent-services [33]).

Another important aspect of design is evaluation. In the last years several works have been proposed on this topic. Some look at specific attributes of the methodology to evaluate it (this is the case of [23][100]) while some others more generically try to identify the elements that a methodology should include to deal with specific aspects of agency like for instance managing complexity [83].

Finally, we would like to report some studies on the composition of new methodologies based on the reuse of existing portions of them (usually called *method fragments*). These works start with experiences from classical software engineering [17][86] and have their primary justification in the claim that one single design methodology cannot be suited to face all problems and developing contexts. According to this paradigm, each class of problems should be faced by a specific methodology that properly considers the skills of the development group and other factors affecting the software production (like for instance strategic choices about implementing environment and technologies).

Actually, a wide repository of method fragments coming from diffused agent methodologies (Gaia, MaSE, PASSI, Prometheus, Tropos) is included in the Open Process Framework [44]. A similar approach is pursued by the FIPA Methodology Technical Committee, whose results can also be found in works of some of its members [31][45]. Although some experiences exist in supporting tools for object-oriented approaches [102], the lack of specific agent-oriented instruments and the intrinsic complexity of the approach has still limited the diffusion of this paradigm.

5 Implementing Agents

Agent systems can be implemented and deployed on a variety of target platforms. There are agent-oriented platforms that conform to some standards such as FIPA or MAF [78], but it can be the case that a MAS is finally realized on more conventional technology, for instance, as Java distributed objects or components. Here we describe both agent platforms (section 5.1) and proposals for transformation from MAS design models to implementation (section 5.2). Finally, in section 5.3 we consider agent-oriented programming languages.

5.1 Agent Platforms

Agent platforms support developers by providing a set of reusable components and services for the implementation and deployment of agents. Most of them are compliant with standards. In Europe, JADE can be considered as the reference FIPA compliant platform. Other platforms are more focused to support agent coordination, such as TuCSoN and Islander.

JADE (Java Agent DEvelopment Framework) [6] originates as a collaboration between the research labs of Telecom Italia (TILAB) and Univ. Parma, and currently is distributed as open source software under the terms of the LGPL (Lesser General Public License Version 2). JADE illustrates well the implementation of FIPA management architecture components: the Agent Communication Channel, the Agent Management System, and the Directory Facilitator. Agent communication is performed through message passing, where FIPA ACL is the language to represent messages, and with libraries that implement FIPA protocols, which can be used as reusable components when building agent-based applications. This facilitates the task of developers who can rely on agent lifecycle management by JADE and have some guarantee of interoperability with other FIPA compliant agent systems. JADE supports both reactive and deliberative agents by defining a structure for agent behaviours, which can be Java classes implementing state machines or rule systems, by an integration of JESS (Java Expert System Shell, available at http://herzberg.ca.sandia.gov/jess/) in the platform. Furthermore, JADE provides some tools for agent debugging (sniffer agents) and monitoring, and other common services such as naming and yellow pages. As a result of the EU IST project LEAP (Lightweight Extensible Agent Platform), JADE incorporated facilities for agent mobility and can be deployed on mobile lightweight Java environments down to J2ME-CLDC. Currently, LEAP libraries are distributed as an add-on of JADE distribution from version 3.0 onwards. A board has been constituted recently with the purpose of driving its evolution and consolidating JADE as a *de-facto* standard middleware for agent-based applications.

Another approach for agent communication, instead of message passing, is the use of a tuple spaces, a classic mechanism for coordination. This is illustrated by TuCSoN (Tuple Centres Spread over the Networks), by the Univ. Bologna [88]. An interesting feature of this kind of systems is the ability to define coordination laws (something that is not common for tuple space approaches in general). Islander+AMELI [40] also provides a coordination middleware, by exploiting the concept of electronic institutions to implement complex negotiation processes.

5.2 Transformation from Design to Implementation

As a modelling paradigm agents contribute to the use of abstract concepts that are close to those used when reasoning about human behaviours and organizations. This can facilitate analysis and design activities but the gap to implementation is greater than with other paradigms, which are closer to current computational frameworks. In this sense, although there are well-established agent platforms, such as JADE, it is common to see agent systems that are implemented on more conventional platforms, usually depending on the application environment and constraints (for instance, a robotic system or a J2EE server). In order to solve this kind of situations, some integrated development environments (IDEs) provide tools for modelling with agent concepts and a process for transforming agent specifications into code for the target platforms.

Finally, when considering multiple target platforms, the trend is to follow the OMG Model Driven Architecture (MDA) approach [73]. Basically, the idea is to specify the meta-model of a MAS modelling language, which is platform independent, and those of the target platforms. Mappings define rules or algorithms that determine how instances of types in the MAS meta-model result in the generation of instances of types in the meta-model specifying a target platform. This approach has been discussed in [1] and is used by the INGENIAS Development Kit (IDK) to generate code on JADE, Servlets, Robocode tanks, and other systems [51]. It is also proposed by MetaDIMA [52] and Agent Factory [30].

5.3 Agent-Oriented Programming Languages

The use of agent-oriented programming languages facilitates the understanding of agent features. There is an extensive review of this in an accompanying paper of this special issue [13], which considers imperative, declarative and hybrid approaches. Basically, the different proposals consider an agent model that makes emphasis either on mobility issues, or on an intentional behaviour model, or on a communication model. CLAIM [39] is probably the most complete in considering all these issues and being applied to real applications. Many provide support for a BDI model, such as dMARS, 3APL, or Coo-BDI.

6 Verification and Testing

Verification and testing techniques for MAS usually apply known results from concurrent and distributed computing.

Verification is normally based on formal theories, that allow the analysis of a system in order to determine whether certain properties hold. These can be *liveness* (whether the system will progress) or *safety* properties (whether the system will do right things), thus answering to the question *is the system being built right?* When the property consists on whether the application fulfils the requirements, we usually refer to it as *validation*. Testing, on the other hand, is usually defined as the activity of looking for errors in the final implementation.

What is interesting to note in the case of MAS when discussing verification and testing is whether organizational, cognitive, development, evolution, and motivational concepts are considered, because the consequences of having concurrent and distributed processes are already a subject extensively covered in the literature since the seventies. Winograd & Flores [105] already criticised that many approaches try to work with these properties through techniques that were conceived for other purposes, without taking advantage of specific agent characteristics. In this context, verification and testing of MAS have not just imported techniques from other paradigms, but they have also created new approaches to solve this problem.

An example of the first formal approaches for verification in the agent domain is DESIRE [15], a design and specification framework that describes agents and the MAS itself as networks of tasks organized in a hierarchy. The interaction and coordination among agents is specified as interchanges of pieces of information and control dependencies. Properties to be verified are represented with temporal logics: what is a conflict among goals or how to choose among design alternatives. Checking properties consists of demonstrating that these are satisfied in a concrete problem using the DESIRE representation of the system. Although this allows proving complex properties of the system and the domain, it has the limitation of the agent model as being task-oriented.

Other formal approaches have shown limited scope because they are assuming a fixed agent model, usually more as a kind of reactive process rather than intentional, and demand too detailed specifications, which makes these techniques work for toy examples but unaffordable for real cases, apart of the learning curve that they imply for developers. For these reasons, there are several approaches that try to mix the goodness of formal languages with the expressive power of semi-formal (usually graphical) languages.

An example of this is the use of model checking techniques to verify the satisfaction of requirements in Tropos [49]. Specifications with the graphical language of Tropos are translated into Formal Tropos, adding temporal logic constructs. This offers the possibility of verifying the specification with formal methods.

Recently we start to see the application of theories coming from other fields, such as Sociology. Activity Theory, for instance, has been applied to the identification of contradiction patterns (e.g., conflicts between individual goals and community goals) by translating concepts for the social science to agent concepts, in this case for INGENIAS and Tropos [47]. Activity Theory is also being considered for analysing social coordination in the TuCSoN platform [89].

Concerning testing, apart of debugging tools that help the developer to follow messages exchange and in some cases to introspect agents (as in the case of MadKit [53]) an interesting approach is the use of data mining tools for analysing and presenting results to the developer. This is used for the JADE platform in the ACLAnalyser tool [14]. Another work, specifically conceived for the JADE platform and including both a test method (aimed at testing single agent features with a regression testing approach) and a supporting tool is presented in [20].

7 Conclusion

The agent-oriented approach, from a software engineering point of view, is mainly used for analysis and design of complex systems. Implementation and deployment of these systems may take a variety of forms, sometimes following agent related standards (such as FIPA and MAF) but usually as conventional distributed objects or component based software. Thus, the main benefit of agent-orientation at present seems to be at the level of modelling. The coupling with that diversity of target platforms is motivating approaches in the AOSE community which are in line with the OMG Model Driven Architecture (MDA) approach. Following this, and considering the state-of-the-art as reported in this work, we think the agent approach can be profitably used for modelling the solution at a platform independent level, and then some tools could provide proper transformations to specific target platforms.

Acknowledgement

We would like to thank all the members of the AgentLink AOSE Technical Forum Group for their active participation during the AL3 Technical Fora and their contribution in the off-line work.

References

[1] Amor M., Fuentes L. and Vallecillo A. (2005). Bridging the Gap Between Agent-Oriented Design and Implementation Using MDA. In: *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*. Lecture Notes in Computer Science 3382, Springer Verlag, pp. 93—108.

[2] Bauer, B. (2002). UML Class Diagrams Revisited in the Context of Agent-Based Systems. In: *Agent-Oriented Software Engineering II: Second International Workshop*, *AOSE 2001*. Lecture Notes

in Computer Science 2222, Springer-Verlag, pp. 101—118.

[3] Bauer B, Müller J., and Odell J. (2001). Agent UML: A Formalism for Specifying Multiagent Interaction. In: *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000*. Lecture Notes in Computer Science 1957, Springer-Verlag, pp. 91—103.

[4] Beck K., et al. *Manifesto for Agile Software Development*. http://www.agilemanifesto.org.

[5] Beck K., and Andres C. (2004). *Extreme Programming Explained: Embrace Change , 2nd Edition*. Addison-Wesley.

[6] Bellifemine F., Poggi A., and Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software Practice and Experience 31* (2), pp. 103—128.

[7] Bergenti F., and Huhns M. (2004). On the Use of Agents as Components of Software Systems, In: [8], chapter 2, pp. 19—32.

[8] Bergenti F., Gleizes M.-P., and Zambonelli F., editors (2004). *Methodologies and Software Engineering for Agent System: The Agent Oriented Software Engineering Handbook*. Kluwer Academic Publisher, New York.

[9] Bernon C., Camps V., Gleizes M.-P., and Picard G. (2005). Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. In: [55], chapter VII, pp. 172—202.

[10] Bernon C., Camps V., Gleizes M.-P. and Picard G. (2004). Tools for Self-Organizing Applications Engineering. In: *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering [revised and extended papers presented at the Engineering Self-Organising Applications Workshop, ESOA 2003]*. Lecture Notes in Artificial Intelligence 2977, Springer Verlag, pp. 283—298.

[11] Bernon C., Cossentino M., Gleizes M-P., Turci P., and Zambonelli F. (2005). A Study of some Multi-agent Meta-models. In: *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*. Lecture Notes in Computer Science 3382, Springer Verlag, pp. 62—77.

[12] Booch, G. (1994). *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, Reading, MA.

[13] Bordini, R., Braubach, L., El Fallah-Seghrouchni, A., Dastani, M., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A., and Ricci, A. (2005). A Survey on Languages and Platforms for MAS Implementation. *Informatica 29* (this issue).

[14] Botía J., López-Acosta A., and Gómez-Skarmeta A. (2004). ACLAnalyser: A Tool for Debugging Multi-Agent Systems. *Proc. 16th European Conference on Artificial Intelligence, ECAI 2004*, pp. 967—968.

[15] Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., and Treur, J. (1997). DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems 6(1)*. pp. 67—94.

[16] Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J., and Perini A. (2004). TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers 8(3), pp. 203—236.

[17] Brinkkemper S., Lyytinen K., and Welke R. (1996). *Method Engineering: Principles of Method Construction and Tool Support*. Chapman &Hall.

[18] Cabri G., Ferrari L., and Zambonelli F. (2004). Role-based Approaches for Engineering Interactions in Large-Scale Multiagent Systems. In: *Post-Proceedings of Advances in Software Engineering for Large-Scale Multiagent Systems (SELMAS 03)*, Lecture Notes in Computer Science 2940, Springer-Verlag, pp. 243—263.

[19] Cabri G., Leonardi L., and Zambonelli F. (2003). Engineering Mobile Agent Applications via Context-Dependent Coordination. *IEEE Transactions on Software Engineering* 28(11), pp. 1039-1055.

[20] Caire G., Cossentino M., Negri A., Poggi A, and Turci P. (2004). Multi-agent Systems Implementation and Testing, In: *From Agent Theory to Agent Implementation - Fourth International Symposium (AT2AI-4)*, Vienna, Austria.

[21] Caire G., Evans R. Massonet P., Coulier W., Garijo F.J., Gomez J., Pavón J., Leal F., Chainho P., Kearney P.E., and Stark J. (2002). Agent Oriented Analysis using MESSAGE/UML. In: *The Second International Workshop on Agent-Oriented Software Engineering (AOSE 2001)*, Lecture Notes in Computer Science 2222, Springer-Verlag, pp. 119-135.

[22] Capera D., Georgé J.P., Gleizes M.P., and Glize P, (2003). The AMAS Theory for Complex Problem Solving based on Self-organizing Cooperative Agents. In: *Proc. of the 1st International Workshop on Theory And Practice of Open Computational Systems (TAPOCS03@WETICE'03)*, Linz, Austria, pp.283—288.

[23] Cernuzzi L., and Rossi G. (2002). On the Evaluation of Agent Oriented Methodologies. In: *Proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, pp. 21-30.

[24] Cernuzzi L., Cossentino M., and Zambonelli F. (2005). Process Models for Agent-based Development. *Engineering Applications of Artificial Intelligence* 18(2), pp. 205-222.

[25] Trencansky I., Cervenka R.. (2005). Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS, *Informatica 29* (this issue).

[26] Chella A., Cossentino M., Sabatucci L., and Seidita V. (2004). From PASSI to Agile PASSI: Tailoring a Design Process to Meet New Needs. In: *2004 IEEE/WIC/ACM International Joint Conference on Intelligent Agent Technology (IAT'04)*, Beijing, China. pp. 471-474.

[27] Ciancarini P. (1996). Coordination Model and Languages as Software Integrators, *ACM Computing Surveys*, 28(2), pp. 300-302.

[28] Cossentino M. (2002). Different Perspectives in Designing Multi-agent Systems, *AGES'02 workshop at NODe02*, Erfurt, Germany. pp. 61-73.

[29] Cossentino M. (2005). From Requirements to Code with the PASSI Methodology. In: [55], chapter IV, pp. 79—106.

[30] Cossentino M., Sabatucci L., and Chella A. (2003). A Possible Approach to the Development of Robotic Multi-Agent Systems. In: *IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03)*, pp. 13-17.

[31] Cossentino M., and Seidita V. (2004). Composition of a New Process to Meet Agile Needs Using Method Engineering, In: *Software Engineering for Large Multi-Agent Systems vol. III*, Lecture Notes in Computer Science 3390, Springer-Verlag, pp. 36-51.

[32] Cossentino M., and Zambonelli F. (2004). Agent Design from the Autonomy Perspective. In: *Agents and Computational Autonomy: Potential, Risks, and Solutions*, Lecture Notes in Computer Science 2969, Springer-Verlag, pp. 140-150.

[33] Dale J., and Ceccaroni L. (2002). Pizza and a Movie: A Case Study in Advanced Web Services. In: *Agentcities: Challenges in Open Agent Environments Workshop*, AAMAS Conference 2002, Bologna, Italy.

[34] De Giacomo G., Lespérance Y., Levesque H.J., and Sardina, S. (2004). On the Semantics of Deliberation in IndiGolog - From Theory to Implementation. *Annals of Mathematics and Artificial Intelligence* 41(2-4), pp. 259-299.

[35] Demazeau Y. (1995). From Cognitive Interactions to Collective Behaviour in Agent-Based Systems, *1st European Conference on Cognitive Science*, Saint-Malo, France, pp. 117-132.

[36] Di Marzo Serugendo, G., Gleizes, M.-P., Karageorgos, A. (2005). Self-Organisation and Emergence in MAS: An Overview. *Informatica 29* (this issue).

[37] Dori, D. (2002). Object-Process Methodology: A Holistic System Paradigm. Springer.

[38] Drogoul A., and Collinot A. (1998). Applying an Agent-Oriented Methodology to the Design of Artificial Organisations: a Case Study in Robotic Soccer. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1), pp. 113-129.

[39] El Fallah Seghrouchni A. and Sun A. (2003). Claim: A Computational Language for Autonomous, Intelligent and Mobile Agents. In: *Proceedings of ProMAS'03,* Lecture Notes in Artificial Intelligence 3067, Springer Verlag, pp. 90–110.

[40] Esteva M., Rosell B., Rodríguez-Aguilar J.A., and Arcos, J.L. (2004). AMELI: An Agent-based Middleware for Electronic Institutions. In: *Third International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'04).* pp. 236—243.

[41] Ferber J. (1999). *Multi-Agent Systems*, Addison-Wesley: Reading, MA.

[42] Ferber J., and Gutknecht O. (1998). A Meta-model for the Analysis and Design of Organizations in Multi-agent Systems. In Proc. of the *3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pp. 128–135.

[43] FIPA. Abstract Architecture Specification. Document SC00001L. Available online at http://www.fipa.org/specs/fipa00001/SC00001L.html.

[44] Firesmith D.G., and Henderson-Sellers B. (2002). *The OPEN Process Framework*. Addison-Wesley.

[45] Fortino G., Garro A., and Russo W. (2004). From Modeling to Simulation of Multi-Agent Systems: an Integrated Approach and a Case Study. In: Proceedings of the *Second German Conference on Multiagent System Technologies (MATES'04),* Lecture Notes in Artificial Intelligence 3187, Springer-Verlag, pp. 213-227.

[46] Franklin S, and Graesser A. (1996) Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In: *Intelligent Agents III – Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence, 1193, Springer Verlag, pp. 21—35.

[47] Fuentes R., Gómez-Sanz J.J., and Pavón, J. (2004). Social Analysis of Multi-Agent Systems with Activity Theory. In: *Proceedings of CAEPIA 2003*, Lecture Notes in Artificial Intelligence 3040, Springer-Verlag, pp. 526-535.

[48] Fuggetta A. (2000). Software Process: a Roadmap. In Proceedings of the *Conference on the Future of Software Engineering*, ACM Press, New York (USA), pp. 25-34

[49] Fuxman A., Pistore M., Mylopoulos J. and Traverso P. (2001). Model Checking Early Requirements Specifications in Tropos. In: *Proceedings 5th IEEE International Symposium on Requirements Engineering (RE 2001)*, pp. 174-181.

[50] Gervais M. (2003). ODAC: An Agent-Oriented Methodology based on ODP. *Journal of Autonomous Agents and Multi-Agent Systems* 7(3), pp. 199–228.

[51] Gomez-Sanz J. J., and Pavón, J. (2002). Meta-modelling in Agent-Oriented Software Engineering. In: *Advances in Artificial Intelligence - IBERAMIA 2002*, Lecture Notes in Artificial Intelligence 2527, Springer-Verlag, 606-615.

[52] Guessoum Z., and Jarraya, T. (2005). Meta-Models & Model-Driven Architectures, *Contribution to the AOSE TFG AgentLink3 meeting*, Ljubljana, 2005.

[53] Gutknecht O., Ferber J., and Michel F. (2001). Integrating Tools and Infrastructures for Generic Multi-agent Systems. In: *Proceedings of the fifth international conference on Autonomous agents (Agents 2001)*, ACM Press, pp. 441–448.

[54] Henderson-Sellers, B., and Debenham, J. (2003). Towards Open Methodological Support for Agent Oriented Systems Development. In: *Proceedings of the First International Conference on Agent-Based Technologies and Systems*. University of Canada, Canada. pp. 14–24.

[55] Henderson-Sellers, B. and Giorgini, P., editors (2005). *Agent-Oriented Methodologies*. Idea Group Publishing.

[56] Hilaire V., Koukam A., Grue, P., and Muller J.-P. (2000). Formal Specification and Prototyping of Multi-agent Systems. In: *Engineering Societies in the Agents' World (ESAW'00)*, Lecture Notes in Artificial Intelligence 1972, Springer Verlag, pp. 114—127.

[57] Huget M.-P., and Odell J. (2005). A Study of some Multi-agent Meta-models. In: *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*. Lecture Notes in Computer Science 3382, Springer Verlag, pp. 16—30.

[58] Hunhns M., and Singh M.P. (1999). A Multiagent Treatment of Agenthood. *Applied Artificial Intelligence: An International Journal* 13(1-2), pp. 3-10.

[59] IEEE Computer Society (2004). *SWEBOK. Guide to the Software Engineering Body of Knowledge*. Online at: http://www.swebok.org/.

[60] Iglesias C., Garijo M., Gonzales J., and Velasco J. R. (1998). Analysis and Design of Multi-agent Systems using MAS-CommonKADS. In: *Intelligent Agents IV, Proc. of the Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL'97)*, Lecture Notes in Artificial Intelligence 1365, Springer-Verlag, pp. 313–326.

[61] ISO/IEC X.900 (1995). IS 10746-x ITU-T Rec. X90x, *ODP Reference Model Part x*.

[62] Jacobson I. (1992). *Object-Oriented Software Engineering*, Addison-Wesley.

[63] Jennings N.R. (2000). On Agent-based Software Engineering. *Artificial Intelligence 117*(2), pp. 277—296.

[64] Juan T., Pearce A., and Sterling L. (2002). ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In: *First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, pp. 3—10.

[65] Kendall E. A. (2000). Role Modeling for Agent System Analysis, Design, and Implementation. *IEEE Concurrency.* Volume 8 , Issue 2. pp. 34-41.

[66] Knublauch H. (2002). Extreme Programming of Multi-Agent Systems. In: *First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, pp. 704—711.

[67] Levesque H. J., Reiter R., Lespérance Y., Lin F., and Scherl, R. B. (1997). GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming* 31 (1-3), pp. 59–83.

[68] Lind J. (2001). *Iterative Software Engineering for Multiagent Systems: The MASSIVE Method.* Lecture Notes in Computer Science 1994, Springer-Verlag.

[69] Luck M., and d'Inverno M. (2001). A Conceptual Framework for Agent Definition and Development. *The Computer Journal* 44(1), pp. 1—20.

[70] Luck, M., Ashri, R., D'Inverno, M. (2004). *Agent-Based Software Development*. Artech House Publishers.

[71] McCarthy J., and Hayes P.J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: *Machine Intelligence 4*, Edinburgh University Press, pp. 463—502.

[72] McIlraith S., Son T. C., and Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems* 16(2), pp. 46-53.

[73] Miller J., and Mukerji, J. (eds) (2003). *MDA Guide Version 1.0.1,* omg/2003-06-01.

[74] Molesini, A., Omicini, A., Ricci, A., and Detti, E. (2005). Zooming Multi-Agent Systems. In: *6th International Workshop Agent-Oriented Software Engineering (AOSE 2005),* pp. 193-204.

[75] Newell A. (1982) The Knowledge Level, *Artificial Intelligence*, 18, pp. 87–127.

[76] Odell J. (2002) Objects and Agents Compared. *Journal of Object Technology* 1(1), pp. 41—53.

[77] OMG (2000). *Agent Technology – Green paper*, Agent Platform Special Interest Group, OMG Document agent/00-09-01, version 1.0, 1 September 2000, http://www.objs.com/agent/index.html.

[78] OMG (2000). *Mobile Agent Facility, version 1.0.* OMG Document - formal/00-01-02, online at http://www.omg.org/cgi-bin/doc?formal/2000-01-02.

[79] Omicini A. (2001). SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In: Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Lecture Notes in Computer Science 1957, Springer-Verlag, pp. 185—193.

[80] Omicini A., Papadopoulos, G. A. (2001). Why Coordination Models and Languages in AI?. *Applied Artificial Intelligence 15(1)*, pp. 1—10 .

[81] Pavón J., and Gómez-Sanz J. (2003). *Agent-Oriented Software Engineering with INGENIAS.* In: *Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03),* Lecture Notes in Computer Science 2691, Springer Verlag, pp. 394-403.

[82] Pavón J., Gómez-Sanz J. and Fuentes, R. (2005). The INGENIAS Methodology and Tools. In: [55], chapter IX, pp. 236—276.

[83] Pena J., and Corchuelo R. (2005). Towards clarifying the importance of interactions in agent-oriented software engineering. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* 25 (1), pp. 19-28.

[84] Peña J., Corchuelo R., and Arjona J. L. (2003). A Top Down Approach for MAS Protocol Descriptions. In: *ACM Symposium on Applied Computing SAC'03*, ACM Press, pp. 49-54.

[85] Pressman Roger S. (1982). *Software Engineering: A Practitioner's Approach*, McGraw-Hill Series in Software Engineering and Technology, McGraw-Hill, New York, 6th edition.

[86] Ralyte J., and Rolland C. (2001). An Approach for Method Reengineering. In: *Proc. Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling*, Lecture Notes in Computer Science 2224, pp. 471—484.

[87] Rao A.S., and Georgeff M. P. (1995). BDI Agents: from Theory to Practice. In: *Proc. of the First International Conference on Multi-Agent Systems (ICMAS'95)*, The MIT Press, pp. 312-319.

[88] Ricci A., and Omicini A. (2003). Supporting Coordination in Open Computational Systems with TuCSoN. In: *12th IEEE International Workshops on Enabling Technologies (WETICE 2003), Infrastructure for Collaborative Enterprises.* IEEE Computer Society, pp. 365—370.

[89] Ricci A., Omicini A. and Denti E. (2003). Activity Theory as a Framework for MAS Coordination. *Engineering Societies in the Agents World III, 3rd international Workshop (ESAW'02),* Lecture Notes in Computer Science 2577, Springer-Verlag, pp. 96—210 .

[90] Rumbaugh J., Jacobson I., and Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Addison Wesley. Reading, MA.

[91] Russell S., and Norvig P. (1995). *Artificial Intelligence; A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

[92] Schreiber A., Wielinga J., Akkermans J., and de Velde W. V. (1994). *CommonKADS: A Comprehensive Methodology for KBS Development*. Technical report, Univ. of Amsterdam, Netherlands Energy Research Foundation ECN and Free Univ. of Brussels.

[93] Schillo, M., and Fischer, K. Holonic Multiagent Systems. *Zeitschrift für Künstliche Intelligenz*, no. 3 (in printing).

[94] Serrano J. M., and Ossowski S., (2004). On the Impact of Agent Communication Languages on the Implementation of Agent Systems. In: *Cooperative Information Agents VIII, 8th International Workshop, CIA 2004,* Lecture Notes in Computer Science 3191, Springer-Verlag, pp. 92—106.

[95] Sierra C., Rodríguez-Aguilar J.A., Noriega P., Esteva M., and Arcos J.L. (2004). Engineering Multi-agent Systems as Electronic Institutions. *Upgrade, The European Journal for the Informatics Professional, V(4)*, pp. 33—39.

[96] Singh M. (1997). Formal Methods in DAI: Logic Based Representation and Reasoning. In: *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, pp. 331–376.

[97] Sommerville I. (2004). *Software Engineering 7th edition.* Addison Wesley.

[98] Spivey J. (1992). *The Z Notation: A Reference Manual*. Prentice Hall, Hemel Hempstead, 2nd edition.

[99] Sturm, A., Dori, D., and Shehory, O. (2003). Single-Model Method for Specifying Multi-Agent Systems. In: *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, ACM Press, pp. 121—128.

[100] Sturm A., and Shehory O. (2004) A Framework for Evaluating Agent-Oriented Methodologies. In: *Agent-Oriented Information Systems, 5th Int. Bi-Conference Workshop, AOIS 2003*. Lecture Notes in Computer Science 3030, Springer-Verlag, pp. 94—109.

[101] Tavares da Silva J.L., and Demazeau Y. (2002). Vowels Co-ordination Model. In: *First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, pp. 1129—1136.

[102] Tolvanen, J.-P., and Lyytinen, K. (1993) Flexible Method Adaptation in CASE - the Metamodeling Approach. *Scandinavian Journal of Information Systems*, Vol. 5. IRIS Association. pp. 51-77.

[103] van Lamsweerde A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. In: *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE 2001)*, IEEE Computer Society, pp. 249.

[104] Wagner G. (2003). The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior, *Information Systems* 28 (5), pp. 475—504.

[105] Winograd T., and Flores C.F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, NJ: Ablex.

[106] Wooldridge M., and Ciancarini P. (2001). Agent-Oriented Software Engineering: The State of the Art. In: *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000*. Lecture Notes in Computer Science 1957, Springer-Verlag, pp. 1—28.

[107] Wooldridge, M. (2000). *Reasoning about Agents*. The MIT Press, Cambridge, MA.

[108] Wooldridge M., Fisher M., Huget M.-P., and Parsons S. (2002). Model Checking Multi--agent Systems with MABLE. In: *First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, pp 952—959.

[109] Wooldridge M., Jennings N. R., and Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems* 3(3), pp. 285-312.

[110] Yu E. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, pp. 226—235.

[111] Zambonelli F., Jennings N., and Wooldridge M. (2003). Developing Multiagent Systems: the Gaia Methodology. *ACM Transactions on Software Engineering and Methodology* 12(3), pp. 417-470.

[112] Zambonelli F. and Jennings N. R., Omicini A. and Wooldridge M. (2001). Agent-Oriented Software Engineering for Internet Applications. In: *Coordination of Internet Agents: Models, Technologies, and Applications*, Springer Verlag, pp. 326-346.

Last access date for web links reported in the paper: 30-08-2005

Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS

Ivan Trencansky and Radovan Cervenka
Whitestein Technologies, Panenska 28, 811 03 Bratislava, Slovakia
Tel +421 (2) 5443-5502, Fax +421 (2) 5443-5512
E-mail: {itr,rce}@whitestein.com

Keywords: agent, multi-agent system, modeling language, agent-oriented software engineering

Received: May 6, 2005

The Agent Modeling Language (AML) is a semi-formal visual modeling language for specifying, modeling and documenting systems that incorporate features drawn from multi-agent systems theory. It is specified as an extension to UML 2.0 in accordance with major OMG modeling frameworks (MDA, MOF, UML, and OCL). The ultimate objective of AML is to provide software engineers with a ready-to-use, complete and highly expressive modeling language suitable for the development of commercial software solutions based on multi-agent technologies. This paper presents an overview of AML. The scope of the language, its structure and extensibility mechanisms are discussed, and the core AML modeling constructs and mechanisms are introduced and demonstrated by examples.

Povzetek: Opisana je vizualizacija agentnega jezika za modeliranje.

1 Introduction

The Agent Modeling Language (*AML*) [3, 5, 4] is a semi-formal[1] visual modeling language for specifying, modeling and documenting systems that incorporate concepts drawn from *Multi-Agent Systems* (*MAS*) theory.

The most significant motivation driving the development of AML was the extant need for a ready-to-use, comprehensive, versatile and highly expressive modeling language suitable for the development of commercial software solutions based on multi-agent technologies. To qualify this more precisely, AML was intended to be a language that: (1) is built on proved technical foundations, (2) integrates best practices from agent-oriented software engineering (AOSE) and object-oriented software engineering (OOSE) domains, (3) is well specified and documented, (4) is internally consistent from the conceptual, semantic and syntactic perspectives, (6) is versatile and easy to extend, (7) is independent of any particular theory, software development process or implementation environment, and (8) is supported by Computer-Aided Software Engineering (CASE) tools.

Given these requirements, AML is designed to address the most significant deficiencies with current state-of-the-art and practice in the area of MAS oriented modeling languages, which are often: (1) insufficiently documented and/or specified, or (2) using proprietary and/or non-intuitive modeling constructs, or (3) aimed at modeling only a limited set of MAS aspects, or (4) applicable only to a specific theory, application domain, MAS architecture, or technology, or (5) mutually incompatible, or (6) insufficiently supported by CASE tools.

[1]The term "semi-formal" implies that the language offers the means to specify systems using a combination of natural language, graphical notation, and formal language specification.

The objective of this paper is to present the approach applied to specification of AML, and a brief overview of the various modeling constructs AML provides to model MASs. Due to limitations in paper length, a comprehensive description of AML abstract syntax, semantics, and notation is not provided.

The rest of the paper is structured as follows: Section 2 presents the approach applied to specification of AML and the available extensibility mechanisms. Section 3 explains the AML fundamental entities and their features, sections 4, 5, 6, 7 and 8 present an overview of AML approach to modeling different aspects of agents and MASs, like social aspects, different kinds of interactions, capabilities, mobility, and mental attitudes. In the end the conclusions are drawn.

2 The AML Approach

Toward achieving the stated goals and overcoming the deficiencies associated with many existing approaches, AML has been designed as a language, which:

– incorporates and unifies the most significant concepts from the broadest set of existing multi-agent theories and abstract models (e.g. DAI [24], BDI [17], SMART [9]), modeling and specification languages (e.g. AUML [1, 11, 12], TAO [18], OPM/MAS [20], AOR [23], UML [15], OCL [14], OWL [19], UML-based ontology modeling [7], methodologies (e.g. MESSAGE [10], Gaia [25], TROPOS [2], PASSI [6], Prometheus [16], MaSE [8]), agent platforms (e.g. Jade, FIPA-OS, Jack, Cougaar) and multi-agent driven applications,

- extends the above with new modeling concepts to account for aspects of multi-agent systems thus far covered insufficiently, inappropriately or not at all,
- assembles them into a consistent framework specified by the AML meta-model (covering abstract syntax and semantics of the language) and notation (covering the concrete syntax), and
- is specified as an extension to UML in accordance with the OMG modeling frameworks (MDA, MOF, UML, and OCL).

2.1 The Language Definition

AML is built upon the Unified Modeling Language (UML) 2.0 Superstructure [15], augmenting it with several new modeling concepts appropriate for capturing the typical features of multi-agent systems (see Fig. 1).

The main advantages of this approach are:

- Reuse of well-defined, well-founded, and commonly used concepts of UML.
- Use of existing mechanisms for specifying and extending UML-based languages (metamodel extensions and UML profiles).
- Ease of incorporation into existing UML-based CASE tools.

The abstract syntax, semantics and notation of the language are defined at the *AML Metamodel and Notation* level. The *AML Metamodel* is further structured into two main packages: *AML Kernel* and *UML Extension for AML*.

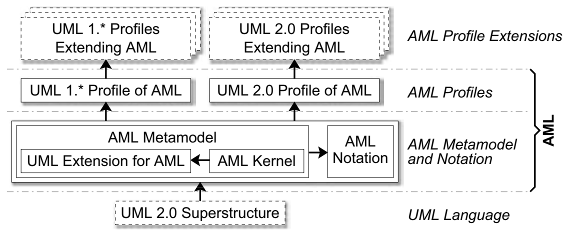


Figure 1: Levels of AML definition

The *AML Kernel* is a conservative[2] extension of UML 2.0, comprising specification of all the AML modeling elements. It is logically structured into several packages, each of which contains specification of modeling elements dedicated for modeling specific aspect of MAS.

The *UML Extension for AML* package adds some meta-properties and structural constraints to the standard UML elements. It is thus a non-conservative extension of UML, and therefore an optional part of the language. However, the extensions contained within are simple and can be easily implemented in most existing UML-based CASE tools.

Upon the AML Metamodel and Notation two UML profiles of AML are specified: *UML 1.* Profile for AML* (based on UML 1.*) and *UML 2.0 Profile for AML* (based on UML 2.0). The primary objective of these profiles is to enable implementation of AML into existing UML 1.* and UML 2.0 based CASE tools, respectively.

[2]A conservative extension of UML is an extension of UML which retains the standard UML semantics in unaltered form [22].

2.2 Extensibility of AML

AML is designed to encompass a broad set of relevant theories and modeling approaches, it being essentially impossible to cover all inclusively. In those cases where AML is insufficient, several mechanisms can be used to extend or customize it as required:

- *Metamodel extension* offers first-class extensibility (as defined by MOF [13]) of the AML metamodel and notation.
- *AML profile extension* offers the possibility to adapt AML for a given domain, platform or development method by means of UML Profiles, without the need to modify the underlying AML Metamodel and Notation.
- *Concrete model extension* allows to employ alternative MAS modeling approaches as complementary specifications to the AML model.

3 Modeling MAS Entities

In general, *entities* are objects that can exist independently of others. In order to maximize reuse and comprehensibility of the metamodel AML defines several auxiliary abstract metamodeling concepts called *semi-entities* and their types. Semi-entity types are specialized UML classes used to specify coherent set of features, logically grouped according to particular aspects of MASs. They are used to specify features of other types of modeling elements.

3.1 AML Semi-entities

AML defines the following semi-entities:

Behaviored semi-entities represent elements, which can own capabilities, observe and/or effect their environment by means of perceptors and effectors, provide and use services, and can be (de)composed into behavior fragments.

Socialized semi-entities represent elements, which can form societies, can participate in social relationships and can own social properties.

Mental semi-entities represent elements which can be characterized in terms of their mental attitudes, e.g. which information they believe in, what are their objectives, needs, motivations, desires, what goal(s) they are committed to, when and how a particular goal is to be achieved, which plan to execute, etc.

3.2 AML Fundamental Entities

The fundamental entities that compose MASs are: agents, resources, and environments. AML therefore defines three modeling concepts, which can be used to model the above mentioned fundamental entities at both type and instance levels:

Agent type is used to specify the type of agents, i.e. self contained entities that are capable of interactions, observations and autonomous behavior within their environment.

Resource type is used to model the type of resources within the system, i.e. physical or informational entities with which the main concern is their availability (in terms of its quantity, access rights, conditions of usage/consumption, etc.).

Environment type is used to model the type of a system's inner environment[3], i.e. the logical or physical surroundings of entities which provide conditions under which the entities exist and function.

In AML, all the aforementioned entity types are specialized UML classes, and thus can utilize all the features defined for UML classes, i.e. can be instantiated, can own structural and behavioral features, behaviors, can be structured into parts and ports, participate in interactions, can participate in various kinds of relationships (e.g. associations, generalizations, dependencies), etc. The instances of the entity types (called entities) can be modeled by means of UML instance specifications classified according to the corresponding types.

Furthermore, all the AML fundamental entity types inherit features of behaviored semi-entities, and in addition to these, agent and environment types are also socialized and mental semi-entities.

Fig. 2 shows an example of a definition of an abstract class `3DObject` that represents spatial objects, characterized by shape and position, existing inside a containing space. An abstract environment type `3DSpace` represents a three dimensional space. This is a special `3DObject` and as such can contain other spatial objects. `3DSpace` provides a service `Motion` to the objects contained within (for details about services see Sect. 5.4). Three concrete `3DObjects`, an agent type `Person`, a resource type `Ball` and a class `Goal` are defined as specialized `3DObjects`. `3DSpace` is further specialized into a concrete environment type `Pitch` representing a soccer pitch containing two goals and a ball.

[3]*Inner environment* is that part of an entity's environment that is contained within the boundaries of the system.

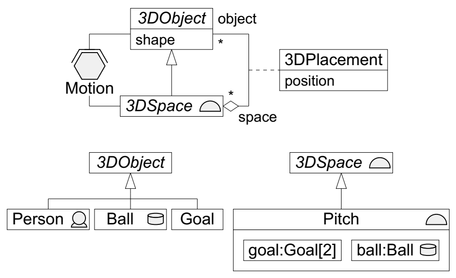


Figure 2: Example of entities, their relationships, service provision and usage

4 Modeling Social Aspects

MASs are commonly perceived as systems comprised of a number of autonomous agents, situated in a common environment, and interacting with each other in order that the desired functionality and properties of the systems could emerge. These properties of MAS are not always derivable or representable solely on the basis of properties and capabilities of individual agents, but are usually given also by their mutual relationships, interactions, coordination mechanisms, social attitudes, etc. Such aspects of MASs are commonly referred to as *social aspects*.

From the social perspective the following aspects of MAS are commonly considered in MAS models (for detaisl see [4]):

– *Social structure* concerning mainly with the identification of societies which can evolve within the system, specification of their properties, structure, identification of comprised roles, individual entities that can participate in such societies, what roles they can play, their mutual relationships, etc.

– *Social behavior* covering such phenomena as social dynamics (i.e. the ability of a society to react to internal and external events), norms (i.e. rules or standards of behavior shared by members of a society), social interactions (how individuals and/or societies interact with others in order to exchange information, coordinate their activities, etc.), and social activities of individual entities and societies (e.g. how they change their attitudes, roles they play, social relationships), etc.

– *Social attitudes* addressing the individual and/or common tendencies (usually expressed in terms of motivations, needs, wishes, intentions, goals, beliefs, commitments, etc.) to anything of a social value.

In this section the focus is on modeling social structure of multi-agent systems. AML modeling constructs which can be used to model social behavior and social attitudes are outlined in the subsequent sections, mainly 5, 6, and 8.

In order to accommodate special needs for modeling social aspects, AML utilizes concepts of: organization units, social relationships, entity roles, and role properties.

4.1 Organization Units

Organization unit type is a specialized environment type, and thus inherits features of behaviored, socialized and mental semi-entity types. They are used to specify the type of societies that can evolve within the system from both the external as well as internal perspectives.

From an *external perspective*, organization units represent coherent autonomous entities, which can be characterized in terms of their mental and social attitudes, can perform behavior, participate in different kinds of (social) relationships, can observe and interact with their environment, offer and use services, play roles, etc. Their properties and behavior are both (1) emergent properties and behavior of all their constituents, their mutual relationships, observations and interactions, and (2) the features and behavior of organization units themselves.

For modeling organization units from external perspectives, in addition to features defined for UML classes (structural and behavioral features, owned behaviors, relationships, etc.), also all the features of behaviored, socialized, and mental semi-entities can be utilized.

From an *internal perspective*, organization units are types of environment that specify the social arrangements of entities in terms of structures, interactions, roles, constraints, norms, etc.

For this purpose organization unit types usually utilize the possibilities inherited from UML structured classifier, and model their internal structure by contained parts and connectors, in combination with entity role types used as types of the parts.

For an example of an organization unit see Fig. 3 (b).

4.2 Social Relationships

Social relationship is a particular type of connection between social entities related to or having dealings with each other. For modeling such relationships, AML defines a special type of UML property, called social property. The social property can be used either in the form of an owned social attribute, or as the end of a social association, and can specify its social role kind[4].

For an example of modeling social relationships see Fig. 3.

4.3 Roles and Role Properties

Roles are used to define a normative behavioral repertoire of entities, and thus provide the basic building blocks of MAS societies. For modeling roles, AML provides *entity role type*, a specialized behaviored, socialized and mental semi-entity type. Entity role types are used to model abstractions of coherent set of features, capabilities, behaviors, observations, relationships, participation in interactions, and services offered or required by entities participating in a particular context. Each entity role type should be realized by a specific implementation possessed by an entity that can play that entity role type. An instance of an entity role type is called entity role and exists only while some behavioral entity plays it.

For modeling the ability of an entity to play an entity role type, AML provides *role properties*. Role property is a specialized UML property, used to specify that an instance of its owner (i.e. a behavioral entity) can play one or several roles of a particular entity role type. The role property can be used either in the form of a role attribute or as the end of a play association.

One entity can at each time play several entity roles. These entity roles can be of the same as well as of different types. The multiplicity defined for a role property constraints the number of entity roles of given type the particular entity can play concurrently. Additional constraints which govern playing of entity roles can be specified by UML constraints.

To allow explicit manipulation of entity roles in UML activities and state machines, AML defines a set of actions for entity role creation and disposal, particularly create role action and dispose role action.

Fig. 3 (a) contains the diagram depicting an agent of type `Person` which can play entity roles of type `Player`, `Captain`, `Coach`, and `Referee`. The possibility of playing entity roles of a particular type is modeled by play associations. Fig. 3 (b) depicts an organization unit `SoccerMatch`, which comprises three `referees` (of the `Referee` entity role type) and two `teams` (of the `SoccerTeam` organization unit type). The `SoccerTeam` itself consists of one to three `coach`es, and eleven to fifteen `players` of which one is the `captain`. The `players` are peers to each other (the `cooperate` connector), and subordinates to the `coaches` (the `manage` connector), and the `captain` (the `lead` connector). The `referees` are superordinate to the both `SoccerTeams` (the `control` connector).

Fig. 4 shows the instantiation of the previously defined types in a model of a system's snapshot, where the agent Lampard, of type Person, plays the entity role player, and the agent Terry, also of type Person, plays the entity role captain and leads Lampard. The agent `Mourinho`, playing the entity role `coach` manages both players Lampard and Terry.

5 Modeling Interactions

To support modeling of interactions in MAS, AML provides a number of UML extensions, which can be logically subdivided into: (1) generic extensions to UML interactions, (2) speech act based extensions to UML inter-

[4]AML predefines *peer*, *subordinate* and *superordinate* social role kinds, but this set can be extended as required.

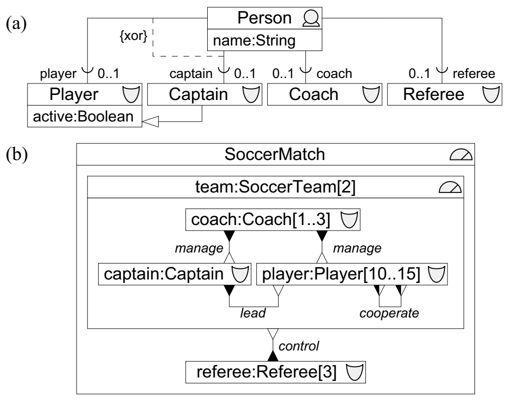


Figure 3: Example of social structure modeling

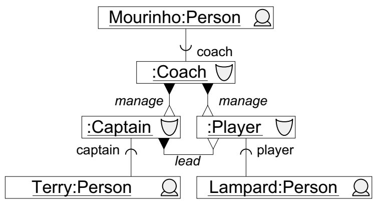


Figure 4: Example of the entity role instantiation and playing

actions, (3) observations and effecting interactions, and (4) services.

5.1 Generic Extensions to UML Interactions

Generic extensions to UML interactions provide means to model: (1) interactions between groups of entities (multi-message and multi-lifeline), (2) dynamic change of object's attributes to express changes in internal structure of organization units, social relationships, or played entity roles, etc., induced by interactions (attribute change), (3) modeling of messages and signals not explicitly associated with the invocation of corresponding methods and receptions (decoupled message), (4) mechanisms for modification of interaction roles of entities (not necessary entity roles) induced by interactions (subset and join dependencies), and (5) modeling the actions of dispatch and reception of decoupled messages in activities (send and decoupled message actions, and associated triggers).

Multi-message is a specialized UML message which is used to model a particular communication between (unlike UML message) multiple participants, i.e. multiple senders and/or multiple receivers.

Multi-lifeline is a specialized UML lifeline, used to represent (unlike UML lifeline) multiple participants in interactions.

Decoupled message is a specialized multi-message used to model the asynchronous dispatch and reception of a message payload without (unlike UML message) explicit specification of the behavior invoked on the side of the receiver. The decision of which behavior should be invoked when the decoupled message is received is up to the receiver what allows to preserve its autonomy in processing messages.

Attribute change is a specialized UML interaction fragment used to model the change of attribute values (state) of interacting entities induced by the interaction. Attribute change thus enables to express addition, removal, or modification of attribute values, and also to express the added attribute values by sub-lifelines. The most likely utilization of attribute change is in modeling of dynamic change of entity roles played by behavioral entities represented by lifelines in interactions, and the modeling of entity interactions with respect to the played entity roles (i.e. each sub-lifeline representing a played entity role can be used to model interaction of its player with respect to this entity role).

Subset is a specialized UML dependency between event occurrences owned by two distinct (superset and subset) lifelines used to specify that since the event occurrence on the superset lifeline, some of the instances it represents (specified by the corresponding selector) are also represented by another, the subset lifeline.

Similarly, *join* dependency is also a specialized UML dependency between two event occurrences on lifelines (subset and union ones), used to specify that a subset of instances, which have been until the subset event occurrence represented by the subset lifeline, is after the union event occurrence represented by the ŞunionŤ lifeline. The union lifeline, thus after the union event occurrence represents the union of the instances it has been representing before, and the instances specified by the join dependency.

Send decoupled message action is a specialized UML send object action used to model the action of dispatching a decoupled message, and *accept decoupled message action* is a specialized UML accept event action used to model reception of a decoupled message action that meets the conditions specified by the associated decoupled message trigger.

A simplified interaction between entities taking part in a player substitution is depicted in Fig. 5. Once the main `coach` decides which players are to be substituted (`p1` to be substituted and `p2` the substitute), he first notifies player `p2` to get ready and then asks the main `referee` for permission to make the substitution. The main `referee` in turn replies by an `answer`. If the `answer` is "yes", the substitution process waits until the game is interrupted. If so, the `coach` instructs player `p1` to exit and `p2` to enter. Player `p1` then leaves the pitch and joins the group of inactive players and `p2` joins the pitch and thereby the group of active players.

Fig. 6 shows an example of the communicative interaction in which the attribute change elements are used to model changes of entity roles played by agents. The diagram realizes the scenario of a captain change caused by the original captain (`player2`) substitution.

At the beginning of the scenario the agent

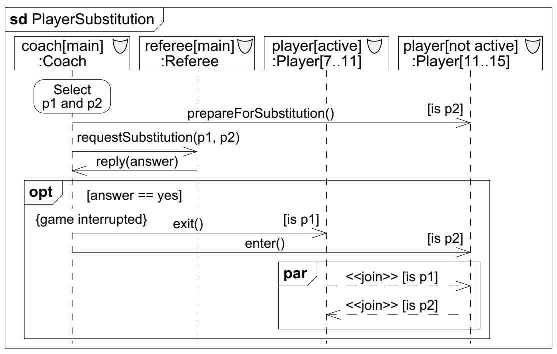


Figure 5: Example of a communicative interaction

`player2` is captain (modeled by its role property `captain`). During the substitution, the main `coach` gives the `player2` order to hand the captainship over (`handCaptainshipOver()` message) and the `player1` the order to become the captain (`becomeCaptain()` message). After receiving these messages, the `player2` stops playing the entity role `captain` (and starts playing the entity role of ordinary `player`) and the `player1` changes from ordinary `player` to `captain`.

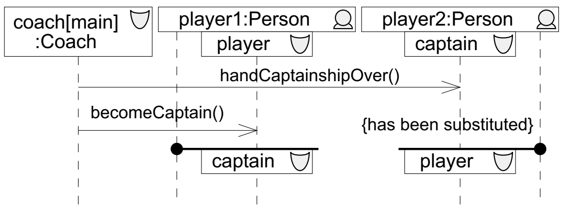


Figure 6: Example of a social interaction with entity role changes

5.2 Speech Act Specific Extensions to UML Interactions

Speech act specific extensions to UML interactions comprise modeling of speech-acts (communication message), speech act based interactions (communicative interactions), patterns of interactions (interaction protocols), and modeling the actions of dispatch and reception of speech-act based messages in activities (send and accept communicative message actions, and associated triggers).

Communication message is a specialized decoupled message used to model communicative acts of speech act based communication within *communicative interaction*s (a specialized UML interaction) with the possibility of explicit specification of the message performative and payload. Both the communication message and communicative interaction can also specify used agent communication and content languages, ontology and payload encoding.

Interaction protocol is a parametrized communicative interaction template used to model reusable templates of communicative interactions.

5.3 Observations and Effecting Interactions

AML provides several mechanisms for modeling observations and effecting interactions in order to (1) allow modeling of the ability of an entity to observe and/or to bring about an effect on others (perceptors and effectors), (2) specify what observation and effecting interactions the entity is capable of (perceptor and effector types and perceiving and effecting acts), (3) specify what entities can observe and/or effect others (perceives and effects dependencies), and (4) explicitly model the actions of observations and effecting interactions in activities (percept and effect actions).

Observations are in AML modeled as the ability of an entity to perceive the state of (or to receive a signal from) an observed object by means of *perceptors*, which are specialized UML ports. *Perceptor types* are used to specify (by means of owned *perceiving acts*) the observations an owner of a perceptor of that type can make.

Perceiving acts are specialized UML operations which can be owned by perceptor types and thus used to specify what perceptions their owners, or perceptors of given type, can perform.

The specification of which entities can observe others, is modeled by a *perceives* dependency. For modeling behavioral aspects of observations, AML provides a specialized *percept action*.

Different aspects of effecting interactions are modeled analogously, by means of *effectors*, *effector types*, *effecting acts*, *effects* dependencies, and *effect actions*.

An example is depicted in Fig. 8 (a) which shows an entity role type `Player` with two eyes–perceptors called `eye` of type `Eye`, and two legs–effectors called `leg` of type `Leg`. Eyes are used to see other players, the pitch and the ball, and to provide localization information to the internal parts of a player. Legs are used to change the player's position within the pitch (modeled by changing of internal state implying that no effects dependency need be placed in the diagram), and to manipulate the ball.

5.4 Services

The AML support for modeling services comprises (1) the means for the specification of the functionality of a service and the way a service can be accessed (service specification and service protocol), (2) the means for the specification of what entities provide/use services (service provision, service usage, and serviced property), and (if applicable) by what means (serviced port).

A *service* is a coherent block of functionality provided by a behaviored semi-entity, called service provider, that can be accessed by other behaviored semi-entities (which can be either external or internal parts of the service provider), called service clients.

Service specification is used to specify a service by means of owned service protocols, i.e. specialized interaction protocols extended with the ability to specify two mandatory, disjoint and nonempty sets of (not bound) parameters, particularly: provider and client template parameters.

The *provider template parameters* of all contained service protocols specify the set of the template parameters that must be bound by the service providers, and the *client template parameters* of all contained service protocols specify the set of template parameters that must be bound by the service clients. Binding of these complementary template parameters specifies the features of the particular service provision/usage which are dependent on its providers and clients.

Service provision/usage are specialized dependencies used to model provision/use of a service by particular entities, together with the binding of template parameters that are declared to be bound by service providers/clients.

Fig. 7 shows a specification of the `Motion` service defined as a collection of three service protocols. The `CanMove` service protocol is based on the standard FIPA protocol `FIPA-Query-Protocol`[5] [21] and binds the `proposition` parameter (the content of a `query-if` message) to the capability `canMove(what, to)` of a service provider. The `participant` parameter of the `FIPA-Query-Protocol` is mapped to a service provider and the `initiator` parameter to a service client. The `CanMove` service protocol is used by the service client to ask if an object referred by the `what` parameter can be moved to the position referred by the `to` parameter. The remaining service protocols `Move` and `Turn` are based on the `FIPA-Request-Protocol` [21] and are used to change the position or direction of a spatial object.

Binding of the `Motion` service specification to the provider `3DSpace` and the client `3DObject` is depicted in Fig. 2.

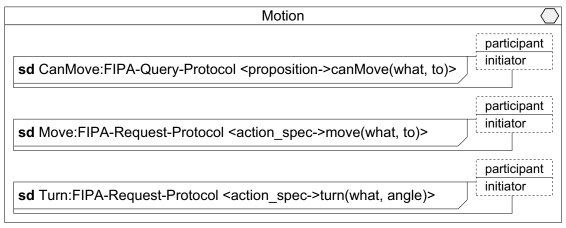

Motion
participant
initiator
sd CanMove:FIPA-Query-Protocol <proposition->canMove(what, to)>
participant
initiator
sd Move:FIPA-Request-Protocol <action_spec->move(what, to)>
participant
initiator
sd Turn:FIPA-Request-Protocol <action_spec->turn(what, angle)>

Figure 7: Example of service specification

[5]The AML specification of the interaction protocol can be found in [3].

6 Modeling Capabilities and Behavior

AML extends the capacity of UML to abstract and decompose behavior by another two modeling elements: capability and behavior fragment.

Capability is an abstract specification of a behavior which allows reasoning about and operations on that specification. Technically, a capability represents a unification of the common specification properties of UML's behavioral features and behaviors expressed in terms of their inputs, outputs, pre- and post-conditions.

Behavior fragment is a specialized behaviored semi-entity type used to model a coherent re-usable fragment of behavior and related structural and behavioral features. It enables the (possibly recursive) decomposition of a complex behavior into simpler and (possibly) concurrently executable fragments, as well as the dynamic modification of an entities behavior in run-time. The decomposition of a behavior of an entity is modeled by owned aggregate attributes of the corresponding behavior fragment type.

Fig. 8 (a) shows the decomposition of the `Player` entity role type's behavior into a structure of behavior fragments. In part (b) two fragments, `Mobility` and `BallHandling`, are described in terms of their owned capabilities (`turn`, `walk`, `catch`, etc.).

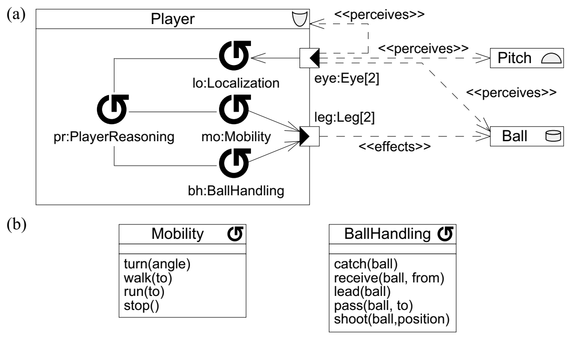


Figure 8: Example of behavior fragments, observations and effecting interactions

7 Modeling MAS Deployment and Mobility

The means provided by AML to support modeling of MAS deployment and agent mobility comprise: (1) the support for modeling the physical infrastructure onto which MAS entities are deployed (agent execution environment), (2) what entities can occur on which nodes of the physical infrastructure and what is the relationship of deployed entities to those nodes (hosting property), (3) how entities can get to a particular node of the physical infrastructure (move and clone dependencies), and (4) what can cause the entity's movement or cloning throughout the physical infrastructure (move and clone actions).

Agent execution environment type is a specialized UML execution environment used to model types of execution environments within which MAS entities can run. While it is a behaviored semi-entity type, it can explicitly, for example, also specify a set of services that the deployed entities can use or should provide at run time.

Agent execution environment can also own *hosting properties*, which are used to classify the entities which can be hosted by the owning agent execution environment. The hosting property's *hosting kind* specifies the relation of the referred entity type to its owning agent execution environment (i.e. either *resident* of *visitor*).

Hosting association is a specialized UML association used to specify hosting property in the form of an association end.

Move is a specialized UML dependency between two hosting properties used to specify that the entities represented by the source hosting property can be moved to the instances of the agent execution environments owning the destination hosting property. Likewise the *clone* dependency is used.

Move and *clone actions* are specialized UML add structural feature actions used to model actions that cause movement or cloning of an entity from one agent execution environment to another one. Both the actions thus specify: (1) which entity is being moved or cloned, (2) the destination agent execution environment instance where the entity is being moved or cloned, and (3) the hosting property where the moved or cloned entity is being placed.

8 Modeling Mental Aspects

Mental semi-entities can be characterized in terms of their mental attitudes, i.e. motivations, needs, wishes, intentions, goals, beliefs, commitments, etc. To allow modeling all the above, AML provides: goals, beliefs, plans, contribution relationships, mental properties and associations, mental constraints, and commit/cancel goal actions.

Goal is a specialized UML class used to model goals, i.e. conditions or states of affairs with which the main concern is their achievement or maintenance. Goals can thus be used to represent objectives, needs, motivations, desires, etc.

Belief is a specialized UML class used to model a state of affairs, proposition or other information relevant to the system and its mental model.

The attitude of a mental semi-entity to a belief or commitment to a goal is modeled by the belief or the goal instance being held in a slot of the corresponding *mental property* (owned by the mental semi-entity, or a mental association relating the belief or the goal to the mental semi-entity).

Plan is a specialized UML activity used to model: predefined plans, or fragments of behavior from which the plans can be composed.

Mental constraint is a specialized UML constraint used to specify properties of owning beliefs, goals and plans which can be used within reasoning processes of mental semi-entities. Supported kinds of mental constraints are pre- and post-conditions, commit conditions, cancel conditions and invariants.

Contribution is a specialized UML relationship used to model logical relationships between goals, beliefs, plans and their mental constraints. The manner in which the specified mental constraint (e.g. post-condition) of the contributor influences the specified mental constraint kind of the beneficiary (e.g. pre-condition) as well as the degree of the contribution can also be specified.

Actions to model commitments to and de-commitments from goals within activities are also provided.

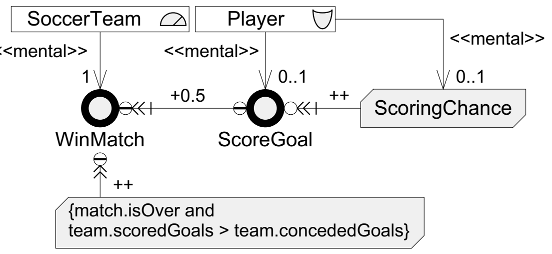


Figure 9: Example of a mental model

Fig. 9 shows an example of a snapshot of the mental model of a soccer team (represented by the `SoccerTeam` organization unit type) and its players (`Player` entity role type). The soccer team has the goal to win a match (modeled by the `WinMatch` goal). The goal `WinMatch` is accomplished, when the soccer match is over and the team has scored more goals than conceded. This is expressed by the sufficient contribution of the belief `{match.isOver and team.scoredGoals > team.concededGoals}` to the postcondition of the goal `WinMatch`. The soccer team players may have goals to score a goal (`ScoreGoal`) which it is feasible to commit to, when they are in a scoring chance. This is expressed by the necessary contribution of the belief `ScoringChance` to the precondition of the goal `ScoreGoal`.

9 Conclusion

The limitation in paper length has not allowed to present all the modeling elements and mechanisms AML provides (e.g. support for ontologies, contexts, etc.). Nevertheless, we believe that from what has been presented in this paper, it is evident that AML provides a rich set of modeling constructs for modeling applications that embody and/or exhibit characteristics of multi-agent systems. It integrates best modeling practices and concepts from existing agent oriented modeling and specification languages into a unique framework built on foundations of UML 2.0 and OCL 2.0. The structure of the language definition together with the MDA/MOF/UML "metamodeling technology" (UML profiles, first-class metamodel extension, etc., gives AML the advantage of natural extensibility and customization. AML is also supported by CASE tools.

We feel confident that AML is sufficiently detailed, comprehensive and tangible to be a useful tool for software architects building systems based on, or exhibiting characteristics of, multi-agent technologies. In this respect we anticipate that AML may form a significant contribution to the effort of bringing about widespread adoption of intelligent agents across varied commercial marketplaces.

Acknowledgement

The authors are indebted to Stefan Brantschen, Monique Calisti, and Dominic Greenwood, for their support and fruitful comments which have inspired many ideas and thus substantially influenced the current version of AML.

References

[1] B. Bauer, J.P. Muller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Interaction. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer-Verlag, Berlin, 2001.

[2] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 2(3):203–236, 2004.

[3] R. Cervenka and I. Trencansky. Agent Modeling Language: Language Specification. Version 0.9. Technical report, Whitestein Technologies, 2004.

[4] R. Cervenka, I. Trencansky, and Calisti. Modeling Social Aspects of Multiagent Systems: The AML Approach. In J.P. Muller and F. Zambonelli, editors, *The Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 05). Workshop 7: Agent-Oriented Software Engineering (AOSE)*, pages 85–96, Universiteit Utrecht, The Netherlands, 2005.

[5] R. Cervenka, I. Trencansky, M. Calisti, and D. Greenwood. AML: Agent Modeling Language. Toward Industry-Grade Agent-Based Modeling. In J. Odell, P. Giorgini, and J.P. Muller, editors, *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*, pages 31–46. Springer-Verlag, Berlin, 2005.

[6] M. Cossentino, L. Sabatucci, and A. Chella. A Possible Approach to the Development of Robotic Multi-Agent Systems. In *IEEE/WIC Conference on Intelligent Agent Technology (IAT'03)*, pages 539–544, Halifax, Canada, 2003.

[7] S. Cranefield, S. Haustein, and M. Purvis. UML-Based Ontology Modelling for Software Agents. In *IProceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, 2001.

[8] S.A. DeLoach. Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems. In *Agent-Oriented Information Systems '99 (AOIS'99)*, Seattle, WA, 1999.

[9] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, Berlin, 2001.

[10] R. Evans, P. Kearny, J. Stark, G. Caire, F. Garijo, J.J. Gomez-Sanz, F. Leal, P. Chainho, and P. Massonet. MESSAGE: Methodology for Engineering Systems of Software Agents. Technical Report P907, EURESCOM, 2001.

[11] J. Odell, H.V.D. Parunak, and B. Bauer. Extending UML for Agents. In G. Wagner, Y. Lesperance, and E. Yu, editors, *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pages 3–17, Austin, Texas, 2000.

[12] J. Odell, H.V.D. Parunak, M. Fleischer, and S. Brueckner. Modeling Agents and their Environment. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002*, pages 16–31. Springer-Verlag, Berlin, 2002.

[13] OMG. Meta Object Facility (MOF) Specification. Version 1.4, formal/2002-04-03, april 2002.

[14] OMG. UML 2.0 OCL Specification. ptc/03-10-14, October 2003.

[15] OMG. Unified Modeling Language: Superstructure version 2.0. ptc/03-08-02, 2003.

[16] L. Padgham and M. Winikoff. Prometheus: A Methodology for Developing Intelligent Agents. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002*, pages 174–185. Springer-Verlag, Berlin, 2002.

[17] A.S. Rao and M.P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In J.F. Allen, R. Fikes, and E. Sandewall, editors, *Knowledge Representation and Reasonning (KR&R-91): Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann Publishers, San Mateo, California, 1991.

[18] V. Silva, A. Garcia, A. Brandao, C. Chavez, C. Lucena, and P. Alencar. Taming Agents and Objects in Software Engineering. In A. Garcia, C. Lucena, J. Castro, A. Omicini, and F. Zambonelli, editors, *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications*, volume LNCS 2603, pages 1–25. Springer-Verlag, Berlin, 2003.

[19] M.K. Smith, D. McGuinness, R. Volz, and C. Welty. Web Ontology Language (OWL), Guide Version 1.0, W3C Working Draft. URL: http://www.w3.org/TR/2002/WD-owl-guide-20021104, 2002.

[20] A. Sturm, D. Dori, and O. Shehory. Single-Model Method for Specifying Multi-Agent Systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 121–128. ACM Press, New York, NY, 2003.

[21] The Foundation for Intelligent Physical Agents. FIPA Specifications Repository. URL: http://www.fipa.org/repository/index.html, 2004.

[22] W.M. Turski and T.S.E. Maibaum. *The Specification of Computer Programs*. Addison-Wesley, London, 1987.

[23] G. Wagner. The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. *Information Systems*, 28(5):475–504, 2003.

[24] G. Weiss. *Multiagent Systems–A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 3^{rd} edition, 2001.

[25] F. Zambonelli, N.R. Jennings, and M. Wooldridge. Developing multiagent systems: the Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, 2003.

The *Tropos* Metamodel and its Use

Angelo Susi, Anna Perini and John Mylopoulos
ITC-irst, Via Sommarive, 18, I-38050 Trento-Povo, Italy
E-mail: susi@itc.it, perini@itc.it, jm@cs.toronto.edu

Paolo Giorgini
Department of Information and Communication Technology
University of Trento, via Sommarive 14, I-38050 Trento-Povo, Italy
E-mail: paolo.giorgini@dit.unitn.it

Keywords: Agent Oriented Software Engineering Methodology, Metamodel

Received: May 9, 2005

Tropos is a software development methodology founded on the key concepts of agent-oriented software development. Specifically, Tropos emphasizes concepts for modelling and analysis during the early requirements phase. This phase precedes the prescriptive requirements specification of the system-to-be. In this paper, we present the Tropos metamodel starting from the basic concepts of actor, goal, plan, resource and social dependency and then we illustrate its use by introducing an extension intended to introduce concepts for modelling security concerns. We also sketch the Tropos modelling environment and compare with the metamodels of other software development methodologies.

Povzetek: Podana je programska metodologija Tropos, temelječa na agentnih pristopih.

1 Introduction

Software development paradigms have exploited a wealth of models to capture requirements and design information about a software system (the "system-to-be") throughout its development process. Structured software development used SADT and Data Flow Diagrams. Object-oriented software development has used a range of modelling languages which have been integrated into UML. Not surprisingly, agent-oriented software development is following on the same footsteps.

To formally analyze software models, we need a means to define their syntax and semantics. *Metamodels* have been used for the former task. Metamodels define a set of possible instantiations, which are all and only the syntactically correct models in some modelling language. As such, metamodels have been used for more than two decades as a basis for defining the syntax of (usually graph-theoretic) modelling languages, such as UML as well as *Tropos*.

The objective of this paper is to introduce the *Tropos* metamodel, discuss some of its uses, and compare it to other metamodels of agent/goal-oriented software development methodologies. Section 4 of the paper sketches the *Tropos* methodology, while Section 3 presents the metamodel and explains its features. Section 4 presents one extension of the metamodel to include security-related concepts. In Section 5 we sketch the *Tropos* development environment, which uses the metamodel in its basic core. Section 6 relates the proposed metamodel to others in the same family of modelling languages, while Section 7 concludes the paper.

2 Models and Methodology

Tropos is founded on the idea of using the agent paradigm and related mentalistic notions during all phases of the development software process. The methodology [6] adopts the *i** [26] modelling framework, which proposes the concepts of (social) *actor*, *goal*, *task*, *resource* and social *dependency* to model both the system-to-be and its organizational operating environment. The *i** framework includes the strategic dependency model (actor diagrams in *Tropos*) for describing the network of inter-dependencies among actors, as well as the strategic rationale model (goal diagrams in *Tropos*) for describing and supporting the means-ends analysis conducted by each actor as it attempts to ensure that – through delegations to other actors – its goals will eventually be fulfilled.

An *actor diagram* is a graph whose nodes represent actors (*agents*, *positions*, or *roles*), while edges represent dependencies among them. A dependency represents an agreement between two actors where one actor (the depender) depends on another (the dependee) to fulfill a goal, perform a task or deliver a resource (the dependum). Dependencies may also involve softgoals (such as "having a good quality meeting") which represent vaguely-defined goals, with no clear-cut criteria for their fulfillment.

A *goal diagram* is also a graph where nodes represent goals or plans[1], while edges represent goal/plan relationships, such as AND/OR-decomposition (i.e., a goal/plan can be decomposed into a set of other goals/plans. Goals/plans can also be related to softgoals through qualitative relationships (labelled "+" or "-") to indicate that the goal/plan contributes positively or negatively to the fulfillment of the softgoal. Goal diagrams appear inside a balloon associated with a single actor. This is the actor whose goals/plans are being analyzed to determine how they can be fulfilled/executed.

The *Tropos* methodology supports four phases of software development: Early Requirements Analysis, Late Requirements Analysis, Architectural Design, and Detailed Design. *Early requirements* is concerned with understanding the organizational context within which the system-to-be will eventually function. During early requirements analysis, the requirements engineer identifies the domain stakeholders (who have a stake in the system-to-be) and models them as social actors, who have goals and depend on each other for goals to be fulfilled, plans to be performed, and resources to be furnished. *Late requirements*, on the other hand, is concerned with a definition of the functional and non- functional requirements of the system-to-be. This is accomplished by treating the system as another actor (or a small number of actors) who are dependers/dependees in dependencies that relate them to external actors. The shift from early to late requirements occurs when the system actor is introduced and it participates in delegations from/to other actors.

Architectural design is concerned with the global structure of the system-to-be. Unsurprisingly, subsystems and system components are represented as actors too, and their dependencies to other system components are social, rather than procedural/structural. This means that system components need to have the ability to monitor dependencies to other actors to make sure they will be fulfilled. As well, system components need to be able to cancel dependencies that seem ineffective and replace them with new ones through planning, negotiation, etc. As with conventional software architectures, architectural styles constitute critical support for the software developer. Since the fundamental concepts of Tropos architectures are intentional and social, we have turned to theories which study social structures to define architectural styles: namely Organization Theory and Strategic Alliances.

Detailed design focuses on the specification of actor communication and behavior. To support this phase, we have adopted existing agent communication languages such as FIPA-ACL [20] or KQML [11]; also message transportation mechanisms and other related concepts and tools. We have also proposed and defined a set of stereotypes, tagged values, and constraints to accommodate *Tropos* concepts within UML [5].

Through the models constructed during these phases, one can answer "why" questions, in addition to "what" and "how" ones, regarding system functionality. For example, one can ask "Why does this component of the system need to notify library users when a book becomes available". Answers to why questions ultimately link system functionality to stakeholder needs, preferences and objectives. Such answers serve as ultimate justifications for all elements of a proposed design.

[1]Plans in *Tropos* correspond to tasks in *i**.

3 The Metamodel

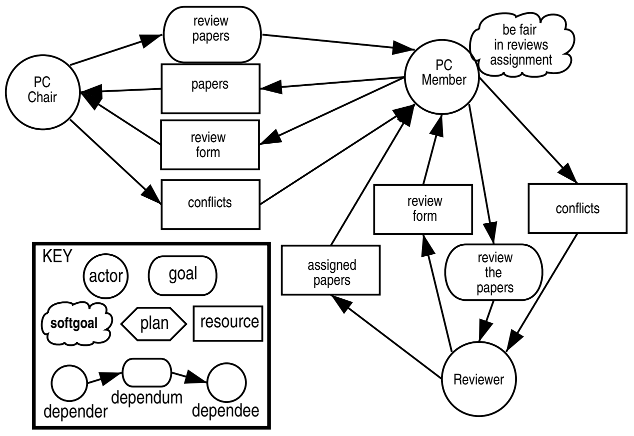


Figure 2: The *Tropos* actor diagram describing a sketch of the conference review process.

Figure 1 shows the portion of the *Tropos* metamodel, where agent, role and position are specialization of the concept of actor. A position can *cover* $1 \dots n$ roles, whereas an agent can *play* $0 \dots n$ roles and can *occupy* $0 \dots n$ positions. An actor can have $0 \dots n$ goals, which can be both hard and softgoals and are wanted by 1 actor.

An actor *dependency* is a quaternary relationship and relates respectively a depender, dependee, and dependum (i.e. goal, plan, resource). It is possible to specify also a reason for the dependency (labeled as *why*).

A model is an instance of the metamodel and can have a graphical representation in terms of actor and goal diagrams.

Figure 2 depicts an example of an actor diagram for the domain of the Conference Review Process and represents a model that can be obtained instantiating the metamodel discussed so far. Three actors are involved: the Program Committee Chair (`PC Chair`), the Program Committee Member (`PC Member`) and the `Reviewer`. Dependencies take place between them; in particular the goal `review papers` is delegated by the `PC Chair` to the `PC Member`, moreover the `PC Chair` also expects to have the information of the possible `conflicts` (a resource dependency) between the `PC Member` and the authors of the papers. On the other hand, the `PC Member` depends on the `PC Chair` to obtain the `papers` to distribute and the `review form`. Many critical goal and resource dependencies occur between the `PC Member` and the `Reviewer`. In particular, the `PC`

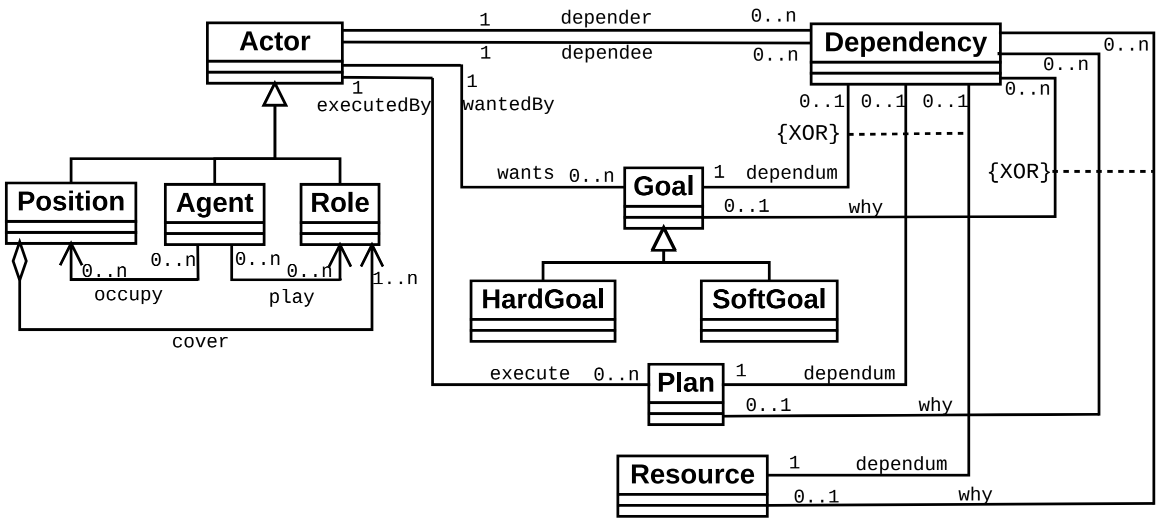


Figure 1: The UML class diagram specifying the actor concept and the dependency relationship in the *Tropos* metamodel. UML notation is compliant with the OMG MOF 1.4.

`Member` depends on the `Reviewer` for `review the papers` and to obtain the information about the possible `conflicts` on `assigned papers`. The `Reviewer` depends on the `PC Member` in order to obtain a set of `assigned papers` as well as the `review form`. Finally, the `PC Member` wants to `be fair in the review assignment`, and this is represented as a softgoal wanted by the `PC Member`.

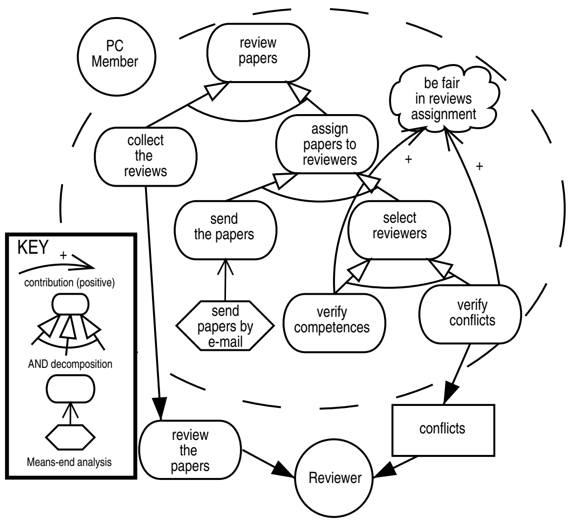


Figure 4: The *Tropos* goal diagram related to the actor `PC Member`.

The concepts related to the *Tropos* goal diagram are depicted in Figure 3. The central concept of goal is represented by the class *Goal*. Goals can be analyzed, from the point of view of an actor, by *Means-end analysis*, *Contribution analysis* and *Boolean decomposition*. *Means-end Analysis* is a ternary relationship defined among an *Actor*, whose point of view is represented in the analysis, a goal (the end), and a *Plan*, *Resource* or *Goal* (the means). *Contribution Analysis* is a ternary relationship between an *actor*, whose point of view is represented, and two goals. Contribution analysis strives to identify goals that can contribute positively or negatively towards the fulfillment of other goals (see association relationship labeled *contribute* in Figure 3). A contribution can be annotated with a qualitative metric, as proposed in [8], denoted by $+, ++, -, --$. In particular, if the goal *g1* contributes positively to the goal *g2*, with metric $++$ then if *g1* is satisfied, so is *g2*. Analogously, if the plan *p* contributes positively to the goal *g*, with metric $++$, this says that *p* fulfills *g*. A $+$ label for a goal or plan contribution represents a partial, positive contribution to the goal being analyzed. With labels $--$, and $-$ we have the dual situation representing a sufficient or partial negative contribution towards the fulfillment of a goal. *Decomposition*, whose metamodel is described in Figure 3, is also a ternary relationship which defines a generic boolean decomposition of a root goal into subgoals, that can be in particular an *AND*- or an *OR-decomposition* specified via the attribute Type in the class *Boolean Decomposition* specialization of the class *Decomposition*.

The concept of plan in *Tropos* is specified in Figure 2 and 3. *Means-end analysis* and *AND/OR decomposition*, defined above for goals, can be applied to plans also. In particular, *AND/OR decomposition* allows for modelling the plan structure.

Figure 4 gives a sketchy view of goal diagram for the actor `PC Member` and for the goal `review papers` and for the softgoal `be fair in the review assignment`.

The goal `review papers` has been AND-decomposed in two sub goals: `assign papers to reviewers` and `collect the reviews`. This latter represents the “Why” for the dependency `review`

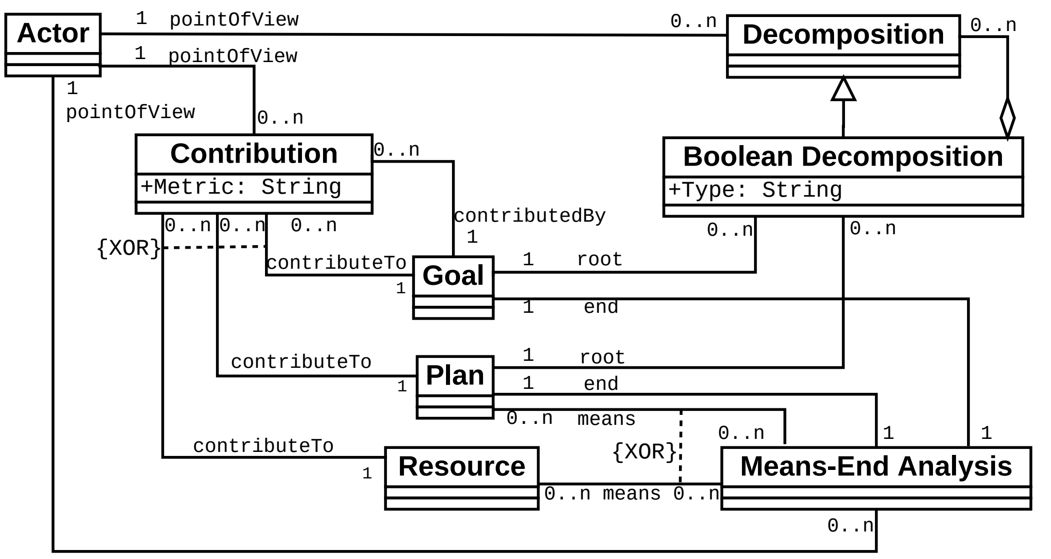


Figure 3: The UML class diagram specifying the concepts related to the goal diagram in the *Tropos* metamodel.

the papers between PC Member and Reviewer, as shown in Figure 1. The goal assign papers to reviewers is decomposed in two subgoals: send the papers, that is operationalized as send papers by e-mail, and select reviewers decomposed in verify the competences and verify conflicts. This latter represents the “Why” for the resource dependency conflicts between the PC Member and the reviewer. Moreover, the fulfillment of these two sub-goals can contribute positively to the fulfillment of the softgoal be fair in the review assignment as described by the positive contribution relationships in the diagram.

4 Metamodel Extension

Secure *Tropos* has been proposed in [16] as a formal framework for modelling and analyzing security. It enhances *Tropos* introducing four new concepts and relationships behind *Tropos* dependency: *trust*, *delegation*, *provisioning*, and *ownership*. The basic idea of ownership is that the owner of a resource (goal or plan) has full authority concerning access and disposition of his resource (goal or plan). The distinction between owning a resource makes it clear how to model situations in which, for example, a client is the legitimate owner of his/her personal data and a Web Service provider that stores customers’ personal data, provides the access to her/his data. We use the relation for delegation when in the domain of analysis there is a formal passage of authority (e.g. a signed piece of paper, a digital credential is sent, etc.). The trust relations have their intuitive meaning among agents, namely the believe of an agent that the actor does not misuse some resources.

Figure 5 shows the the new part of the *Tropos* metamodel concerning trust and ownership. An actor (the *truster*) trusts another actor (the *trustee*) about the achievement of a goal, the fulfillment of a plan or the delivering of a resource. The content of the trust relationship is called *trustum*. An actor can be the owner of a resource, a plan and goal and he/she has authority concerning the use of the resource, the execution of the plan and achievement of the goal, respectively.

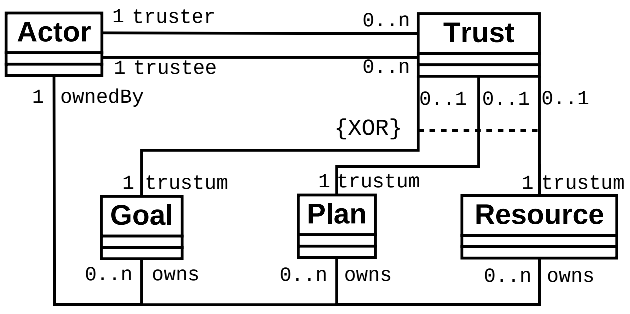


Figure 5: The *Tropos* metamodel related to the concept of Trust.

The metamodel describing delegation relationships is basically identical to the metamodel for the dependency relationship as presented in Figure 1. The *delegater* delegates the *delegatee* for the achievement of a goal, the execution of a plan or the delivering of a resource. As for the dependency relationship, it is also possible here to specify the reason (*why*) of a delegation.

We have shown in [17] how the original concept of *Tropos* dependency can be expressed in terms of trust and delegation. Roughly, when an actor depends on another actor to achieve a goal (to fulfill a task or to deliver a resource), it is implicitly intended that the actor trusts the other actor and delegates it for such activities. A precise formalization of dependency refinement in terms of trust and delegation has been presented in [17].

Figure 6 presents an example of application the extended metamodel. The `Author` trusts the `PC Chair` to `implement a fair review process` and he/she is the owner of the paper sent to the `PC Member` and reviewed by the `Reviewer`. The `PC Chair` trusts and delegates `PC Member` to review a certain number of papers, and in turn the `PC Member` trusts and delegates the `Reviewer` to review the papers. The `PC member` (`Reviewer`) depends on the `PC Chair` (`PC Member`) to receive the paper to review.

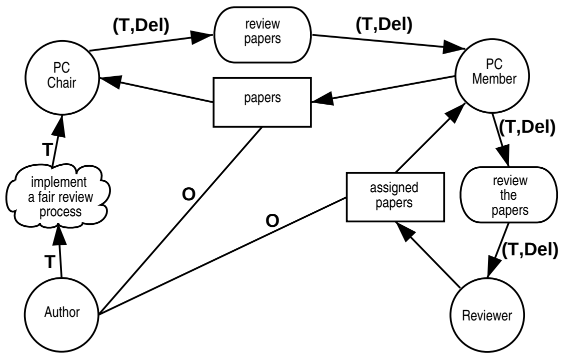


Figure 6: The *Tropos* actor diagram with the trust concepts.

5 A Modelling Environment

In order to support the specific analysis techniques adopted in *Tropos*, different tools have been developed, such as a tool for the verification of requirements specification through model-checking technique (T-Tool) [13], a tool which supports forward and backward reasoning on the goal analysis structures (GR-Tool) [15]. In this section, we will give details of a modelling environment, called TAOM4e (Tool for Agent-Oriented Modelling for Eclipse), which is based on an implementation of the metamodel described in the previous sections. The metamodel has been specified following the OMG's MDA [21] standard for metamodel interoperability, that is the Meta Object Facility (MOF)[2] which offers a mechanism for automatically deriving a concrete syntax based on XML DTDs and/or schemas known as XML Model Interchange (XMI). This is a preliminary step towards the adoption of the model-to-model transformation approach proposed by MDA.

Among the main requirements we considered in developing this tool are the following [23]:

- *Visual Modelling*. The modelling environment should support the user during the specification of an AO model (e.g., according to the *Tropos* visual notation). Moreover, the environment should allow us to represent new entities that will be included in the *Tropos* metamodel, language variants, such as those presented in Section 4, as well as to restrict its use to a subset of entities of the modelling language.

[2]http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF

- *Specification of model entities properties*. The modelling environment should allow us to easily annotate the visual model with model properties like invariants, creation or fulfillment conditions that are typically used in Formal *Tropos* specification.
- *Automatic Model Translation*. The modelling environment should allow us to save a model in a standard format (e.g., XML and XMI), and provide automatic transformation into a different specification language. The model-to-model transformation approach should be also compliant with Query/View/Transformation (QVT) requirements [14], as discussed in [24].
- *Extensibility*. The modelling environment should be extensible and allow for different configurations by easily integrating other tools at will.

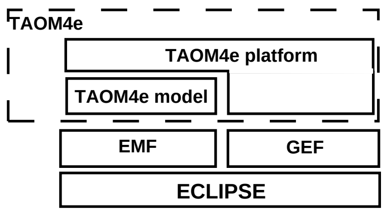


Figure 7: The architecture of TAOM4e.

An effective solution to the requirement of a flexible architecture and to the component integration issue is offered by the Eclipse Platform.

New tools are integrated into the platform through plug-ins that provide the environment with new functionalities. A plug-in is the smallest unit of function in Eclipse and the Eclipse Platform itself is organized as a set of subsystems, implemented in one or more plug-ins, built on the top of a small runtime engine. The TAOM4e architecture is depicted in Figure 7. It follows the Model View Controller pattern and has been devised as an extension of two existing plug-ins. First, the EMF plug-in[3] offers a modelling framework and code generation facilities for building tools and other applications based on a structured data model. Given an XMI model specification, EMF provides functions and runtime support to produce a set of Java classes for the model. Most importantly, EMF provides the foundation for interoperability with other EMF-based tools and applications. The resulting plug-in, called *TAOM4e model* implements the *Tropos* metamodel. It represents the Model component of the MVC architecture. Second, the Graphical Editing Framework (GEF) plug-in[4] allows developers to create a rich graphical editor around an existing metamodel. The functionality of the GEF plug-in helps to cover the essential requirement of the tool, that is supporting a visual development of *Tropos* models by providing some standard functions like drag & drop, undo-redo, copy & paste and others. The resulting plug-in, called *TAOM4e*

[3]http://www.eclipse.org/emf/
[4]http://www.eclipse.org/gef/

platform represents both the Controller and the Viewer components of the tool. In Figure 8 a snapshot of the modeler: the diagram editor window on the right, the project and model browsers on the left, the entity properties window at the bottom.

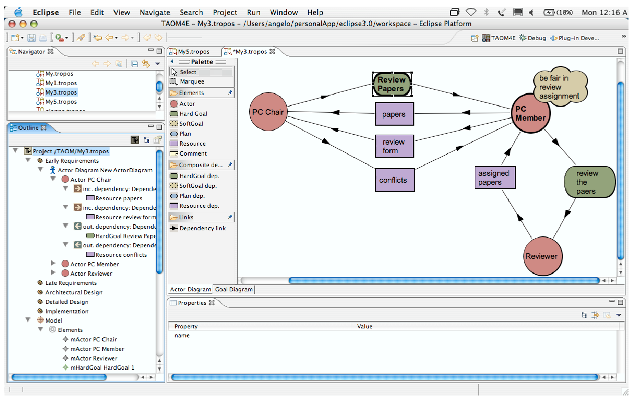


Figure 8: The Graphic User Interface of TAOM4e.

6 Related Work

Many Agent-Oriented Software Engineering methodologies have been proposed and compared over the last few years [18, 25]. An analysis of the metamodels of three methodologies, ADELFE [4], GAIA [27] and PASSI [7] has been presented in [3]. The aim of this work was to face interoperability issues between different methodologies.

In this section we extend this analysis including *Tropos*. We will focus on four dimensions: Agent Structure, Agent Interaction, Agent Organization and Agent Development (e.g., CASE tools at support of the development process). Table 1 summarizes the comparison. In ADELFE the concept of agent (`Cooperative Agent`) is defined as the composition of aptitudes, skills, characteristic, communication and representation. Not explicit concept of role is given, the concept of goal is implicitly used to identify agent skills, but it is not representable as well as the concept of plan, since a plan is an entity that will be built at run time and which is not representable at design time. In GAIA, an agent (`Agent Type`) is specified as a composition of roles. Each role is responsible of a specific set of activities associated with the role. Goals cannot be explicitly modeled, but they are implicitly used to characterize a role. In PASSI, an agent (`Agent`) is defined as the composition of roles and each role is defined as the manifestation of the agent activity in some scenario. Goals are implicitly considered when specifying non-functional requirements attached to agent duties. In *Tropos*, the concept of `Actor` generalizes the concepts of agent and role (or set of roles), an actor can have individual goals and it can be able to execute plans to satisfy goals. Goal analysis in *Tropos* drives the modelling process, as discussed in Section 4 and allows us to represent goal decomposition, means to satisfy a goal or contribution towards goal satisfaction through different goal relationships.

The concepts used to specify the interactions of an agent with another agent or with the environment are similar in ADELFE, GAIA and PASSI. Basically, they use the concept of communication, role, and protocols. *Tropos* adopts the Agent Unified modelling Language (AUML) Agent Interaction Diagram, described in [2, 22] (proposed by the FIPA –Foundation for Physical Intelligent Agents– [12] and the OMG Agent Work group) where agent communicative acts are represented as messages in a UML sequence diagram.

In GAIA, the concept of organization is a primary concept, organization rules specify constraints that the organization should observe. In PASSI, agent organization aspects are modeled implicitly in terms of services that can be accessed by agents in a given scenario. In ADELFE, agent organization and society emerges from the evolving interactions between the agents which are compliant with cooperation rules.

In *Tropos* the strategic dependencies between actors in a domain makes explicit the organizational dimension and provide basic entities to model organizational patterns [19]. Moreover, the *Tropos* metamodel has been extended to include concepts of business processes and security.

Both ADELFE and PASSI provide CASE tools at support of modelling and for ad-hoc analysis on part of the resulting specification. *Tropos* provides modelling and analysis tools (details can be found in http://www.troposproject.org) as well as code generation tools [10].

This comparison shows that different metamodels (methodologies) may allow us to model different properties of a system (e.g., organizational aspects, communications and protocols). On the other hand, it shows that even if metamodels share a comparable set of concepts, they can be used in a different way by the different methodologies. This can be found also considering requirements engineering methodologies based on metamodels. For instance, in KAOS [9], the concept of agent is used to assign leaf goals resulting from goal analysis.

Finally, other related work on $i*$ and *Tropos* metamodels are worth to be mentioned. The $i*$ metamodel [26] represents the basis for the *Tropos* metamodel. Other extensions of the $i*$ metamodel have been proposed. For instance, in [1] where a methodology for COTS selection is proposed.

7 Conclusion

We have presented an overview of the *Tropos* metamodel. Like other software development methodologies, *Tropos* supports a variety of models that need to be analyzed for syntactic and semantic consistency. The metamodel serves as a basis for checking for syntactic consistency. Making

Agent Structure	**ADELFE**	**GAIA**	**PASSI**	***Tropos***
Agent	`Cooperative Agent`	`Agent Type`	`Agent`	`Actor`
Role	Not explicit	`Role` in a organization	`Role` in a scenario	Specialization of `Actor`
Goal	Not explicit	Not explicit	Not explicit	`Goal` and goal relationships
Plan	Not explicit	`Activity` of a Role	Ontology of `Action`	`Plan` and plan relationships
Agent Interaction				
Comm. & Protocol	Agent `Communication` `Agent Interaction` `Protocols` associated to communication	`Communication` associated to a role and protocols associated to a communication	Communication associated to a role and Messages as components of communication	Not in the current metamodel. AUML interaction diagram UML sequence diagram messages for communication acts
A. Organization				
Structure & Rules	`Cooperation rules`	`OrganizationStructure`, `Organization`, `OrganizationalRule`	Not explicit	Strategic `Dependency`, Ownership, Delegation and `Trust` Organizational patterns
A. Development				
Modeler	Open-Tool	—	PASSI Toolkit	TAOM, OME, DW-Tool, ST-Tool
Analysis tooos	Open-Tool	—	PASSI Toolkit	GR-Tool, DW-Tool, ST-Tool,T-Tool
Code Generation	—	—	PASSI Toolkit	SKwyRL

Table 1: Comparison of the meta-models of four Agent-Oriented methodologies.

it richer, could also help in supporting some forms of semantic consistency currently conducted through a series of tools offered within the *Tropos* software development environment.

References

[1] C. Ayala, C. Cares, J. P. Carvallo, G. Grau, M. Haya, X. Franch G. Salazar, E. Mayol, and C. Quer. A Comparative Analysis of *i**-Based Agent-Oriented Modeling Language. In *Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05)*, pages 43–50, Taipei, Taiwan, 2005. KSI Press.

[2] B. Bauer, J.P. Muller, and J. Odell. Agent UML: A formalism for specifying multiagent interaction. In P. Ciancarini and M. Wooldridge, editors, *Proc. of the 1st Int. Workshop on Agent-Oriented Software Engineering (AOSE'00)*, volume 1957 of *LNCS*, pages 91–104, Limerick, Ireland, 2001. Springer.

[3] C. Bernon, M. Cossentino, M. P. Gleizes, P. Turci, and F. Zambonelli. A Study of Some Multi-agent Meta-models. In J. P. Muller J. Odell, P. Giorgini, editor, *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*, volume 3382 of *LNCS*, pages 62–77, New York, USA, NY, 2004. Springer.

[4] C. Bernon, M.P. Gleizes, S. Peyruqueou, and G. Picard. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Third International Workshop on Engineering Societies in the Agents World (ESAW-2002)*, volume 2577 of *LNCS*, pages 156–169, Madrid, Spain, 2003. Springer.

[5] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language: User Guide*. Addison-Wesley, 1999.

[6] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. Kluwer Academic Publishers.

[7] A. Chella, M. Cossentino, and L. Sabatucci. Tools and patterns in designing multi-agent systems with PASSI. *WSEAS Transactions on Communications*, 3(1):352–358, 2004.

[8] L.K. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Publishing, 2000.

[9] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993. Elsevier.

[10] T. T. Do, M. Kolp, and S. Faulkner. Agent Oriented Design Patterns: The SKwyRL Perspective. In *Proc. of the 6th International Conference on Enterprise Information Systems (ICEIS 2004)*, pages 48–53, Porto, Portugal, 2004.

[11] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J.M. Bradshaw, editor, *Software Agents*, pages 291–316. MIT Press, Menlo Park, CA, 1997.

[12] FIPA. The Foundation for Intelligent Physical Agents. At http://www.fipa.org, 2001.

[13] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. Model checking early requirements specifications in Tropos. In *Int. Symposium on Requirements Engineering*, pages 174–181, Toronto, CA, 2001. IEEE Computer Society.

[14] T. Gardner, C. Griffin, J. Koehler, and R. Hauser. A review of omg mof 2.0 query / views / transformations submissions and recommendations towards the final standard. In *MetaModelling for MDA Workshop*, pages 178–197, York, UK, England, 2003.

[15] P. Giorgini, J. Mylopoulous, and R. Sebastiani. Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology. *Engineering Applications of Artificial Intelligence*, 18(2):159–171, 2005.

[16] P. Giorgini, F. Massacci, J. Mylopoulous, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of iTrust'04*, volume 2995 of *LNCS*, pages 176–190. Springer-Verlag, 2004.

[17] P. Giorgini, F. Massacci, J. Mylopoulous, and N. Zannone. Modeling Security Requirements Through Ownership, Permission and Delegation. In *Proc. of the The 13th IEEE Requirements Engineering Conference (RE'05)*, Paris, France, 2005. IEEE Computer Society.

[18] B. Henderson-Sellers and P. Giorgini, editors. *Agent-Oriented Methodologies*. Idea Group Inc., Hershey, PA, USA, 2005.

[19] M. Kolp, P. Giorgini, and J. Mylopoulos. A goal-based organizational perspective on multi-agents architectures. In *Proc. of the 8th Int. Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages, ATAL'01*, volume 2333 of *LNCS*, pages 128–140, Seattle, USA, 2002. Springer.

[20] Y. Labrou, T. Finin, and Y. Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, 1999. IEEE.

[21] Stephen J. Mellor, Kendall Scott, Axel Uhl, and Dirk Weise. *MDA Distilled*. Addison-Wesley, 2004.

[22] J. Odell, H. Van Dyke Parunak, and B. Bauer. Extending UML for agents. In *Proc. of the 2nd Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS'00*, pages 3–17, Austin, USA, 2000.

[23] A. Perini and A. Susi. Developing Tools for Agent-Oriented Visual Modeling. In G. Lindemann, J. Denzinger, I.J. Timm, and R. Unland, editors, *Multiagent System Technologies, Proc. of the Second German Conference, MATES 2004*, volume 3187 of *LNAI*, pages 169–182, Erfurt, Germany, 2004. Springer.

[24] A. Perini and A. Susi. Automating Model Transformations in Agent-Oriented modelling. In *Agent-Oriented Software Engineering VI: AOSE 2005*, LNCS, Utrecht, The Netherlands, 2005. Springer.

[25] A. Sturm and O. Shehory. A Framework for Evaluating Agent-Oriented Methodologies. In M. Winikoff P. Giorgini, B. Henderson-Sellers, editor, *Proc. of the Int. Bi-Conference Workshop on Agent-Oriented Information Systems, AOIS 2003*, volume 3030 of *LNCS*, pages 94–109. Springer, 2003.

[26] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.

[27] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, 2003. ACM.

On the Role of Environments in Multiagent Systems

Danny Weyns and Tom Holvoet
AgentWise, DistriNet, Katholieke Universiteit Leuven
Celestijnenlaan 200 A, B-3001 Leuven, Belgium
E-mail: {danny.weyns, tom.holvoet}@cs.kuleuven.be

Keywords: multiagent systems, environment, engineering environments

Received: June 30, 2005

For a long time, the role of the environment has been underestimated in multiagent systems research. Originating from research on behavior-based agents and situated multiagent systems, the importance of the environment is now gradually being accepted in the multiagent system community in general. In this paper, we elaborate on the role of environments in multiagent systems. We present a model for multiagent systems that puts forward agents and the environment as first-order abstractions. Starting from this model, we elaborate on the logical functionalities of the environment. Competence in engineering environments is a prerequisite to apply environments in practical multiagent system applications. We briefly discuss how current agent-oriented methodologies deal with the environment, and we discuss an approach for engineering environments that puts forward artifacts as building blocks for environments. After that we present the concern-based approach for engineering environments developed in our research group. This approach models the environment as a set of modules that represent different functional concerns of the environment. We illustrate how we have applied this approach in a real-world multiagent system application. The paper concludes with a number of research challenges that are important for the further exploration of environments for multiagent systems.

Povzetek: Opisuje vlogo okolij v multiagentnih sistemih.

1 Introduction

Multiagent systems are an approach to build complex distributed applications. A multiagent system consists of a population of autonomous entities (agents) situated in a shared structured entity (the environment). One classic definition of an autonomous agent is: *an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives* [56]. This definition stresses the importance of the environment, an agent is not an isolated entity but *exists in* an environment in which it senses and acts. In spite of the fundamental role of the environment in agent systems, most researchers neglect to integrate the environment as a primary abstraction in models and tools for multiagent systems, or minimize its responsibilities [49]. Typically, the responsibilities of the environment are reduced to a message transport system or broker infrastructure. Restricting interaction to inter-agent communication neglects a rich potential of possibilities for the paradigm of multiagent systems.

Opportunities that environments offer have been demonstrated in the domain of behavior-based agents and situated multiagent systems. In behavior-based agent systems, interaction in the environment has been considered as an essential feature for intelligent behavior for a long time [9, 25, 1]. Originally, the main focus of this research community was on systems where agents interact in a physical environment, such as robots. Gradually, this work has influenced the software agent community. Today, researchers working in the domain of what is known as *situated multiagent systems* consider logical environments as essential parts of their multiagent systems [29, 10, 26, 47]. These researchers have shown that the environment can serve as a robust, self-revising shared memory, and an excellent medium for indirect coordination of agents [49]. Several practical applications have shown how indirect interaction trough the environment increases the power and expressiveness of multiagent systems, enabling solutions that would otherwise be impossible or at least impractically complex. There are examples in domains such as supply chain systems [41], network support [6], peer-to-peer (P2P) systems [2], manufacturing control [37], and support for automatic logistic services [55].

Originating from research on behavior-based agents and situated multiagent systems, the importance of the environment in multiagent systems is now gradually being accepted in the multiagent system community in general. For example, [8] argues that the multiagent research community should not only focus attention on making agents smarter but also on making the environment more capable of managing and protecting the conditions in which agents have to operate. Recently, the environment has begun to emerge as the focus of research in its own right [14, 48, 34].

This paper is structured as follows. In Sect. 2, we present a model for multiagent systems that puts forward agents and the environment as first-order abstractions. Starting from this model, we discuss logical functionalities of the environment in Sect. 3. Section 4 discusses engineering issues of environments and in Sect. 5 we illustrate a real-world application in which the environment plays a central role. Finally, in Sect. 7 we draw conclusions and list a number of research challenges for the further exploration of environments for multiagent systems.

2 The Environment Abstraction

In line with [49], we put forward agents and the environment as *first-order abstractions* in multiagent systems. This allows to clearly define the environment responsibilities that differ from the agent responsibilities. A first-class module can be defined as *a program building block, an independent piece of software which [...] provides an abstraction or information hiding mechanism so that a module's implementation can be changed without requiring any change to other modules*[1]. Just as the agents, the environment should therefore be an independent building block that encapsulates its own clear-cut responsibilities in a multiagent system. Motivations to put forward the environment as first-order abstraction include the following:

1. Several aspects of multiagent systems that conceptually do not belong to agents themselves should not be assigned to, or hosted inside agents. Examples are infrastructure for communication and coordination, the topology of a spatial domain, or support for the action model.

2. The above (and other) aspects should be explicitly considered. The environment is the natural candidate to encapsulate these aspects.

3. The environment can be a creative part of a designed solution of a multiagent system, helping to manage the huge complexity of engineering complex real-world applications.

One problem with the specification of environments is the confusion between the logical entity of an environment in the application and the underlying infrastructure of the multiagent system. To unravel this confusion, we discuss a model for multiagent-based applications that describes the position of agents and the environment at three levels [54], see figure 1:

– the *multiagent system (MAS) application* layer at the top (i.e., the application logic);

– the *execution platform* (i.e., middleware infrastructure and the operating system);

– and the *physical infrastructure* at the bottom (i.e., processors, network, etc.).

[1]The Free Online Dictionary of Computing, http://foldoc.doc.ic.ac.uk/foldoc/, 8/2005

Below we elaborate on each layer and illustrate that the abstraction of the environment as well as the agents, crosscut the three layers in the model. Before that we introduce a simple file searching system in a P2P network that we use as a running example to illustrate the three-layer model [53]. The idea of this application is to let mobile agents act on behalf of users and browse a shared distributed file system to find requested files. Each user is situated in a particular node (its base). Users can offer files at their base and can send out agents to find files for them. Agents can observe the environment, however, to avoid network overload, agents can perceive the environment only to a limited extent, e.g. two hops from the agent's current position. An agent can perceive nodes and connecting links, bases on nodes, and files available on nodes. Agents can also sense signals. Each base emits such a signal. The intensity of the signal decreases with every hop. Sensing the signal of its base enables an agent to "climb up" the gradient, i.e. move towards its base or alternatively "climb down", i.e. move away from it. Finally, agents can sense pheromones. An agent can drop a file-specific pheromone in the environment when it returns back to its base with a copy of a file. Such a pheromone trail can not only help the agent later on when it needs a new copy of the file, it can also help other agents to find their way to that file. Pheromones evaporate, thereby limiting their influence over time. This is an important property to avoid that agents are misled when a file disappears from a certain node.

2.1 Multiagent System Application Layer

The Multiagent System Application layer contains the *Application Specific Logic*, i.e. Application Agents (AAs) and the Application Environment (AE) of the multiagent system. The AAs are the autonomous entities in the multiagent system, the AE offers a domain specific abstraction to AAs, hiding the complexity of resource access, interaction handling, and consistency management. The AE imposes the rules that regulate domain dynamics. Section 3 elaborates on responsibilities of the AE.

The AAs in the P2P file searching system are the logical entities that are created by the users to search for files in the network. The AE is the logical entity that represents the space in which the AAs perform their job. The AE offers a representation to the AAs of the neighboring nodes and connecting links of the network. The AE also represents the available files, the gradient fields emitted by the bases, and the file-specific pheromones dropped by the agents.

The application logic is typically deployed on top of a *Multiagent System Framework*. The multiagent system framework supports predefined multiagent systems abstractions, such as a particular engine for agent's decision making, support for communication, a model for action, etc. These abstractions can be reused over different applications. In the P2P file searching system, the multiagent system framework layer should provide a pheromone in-

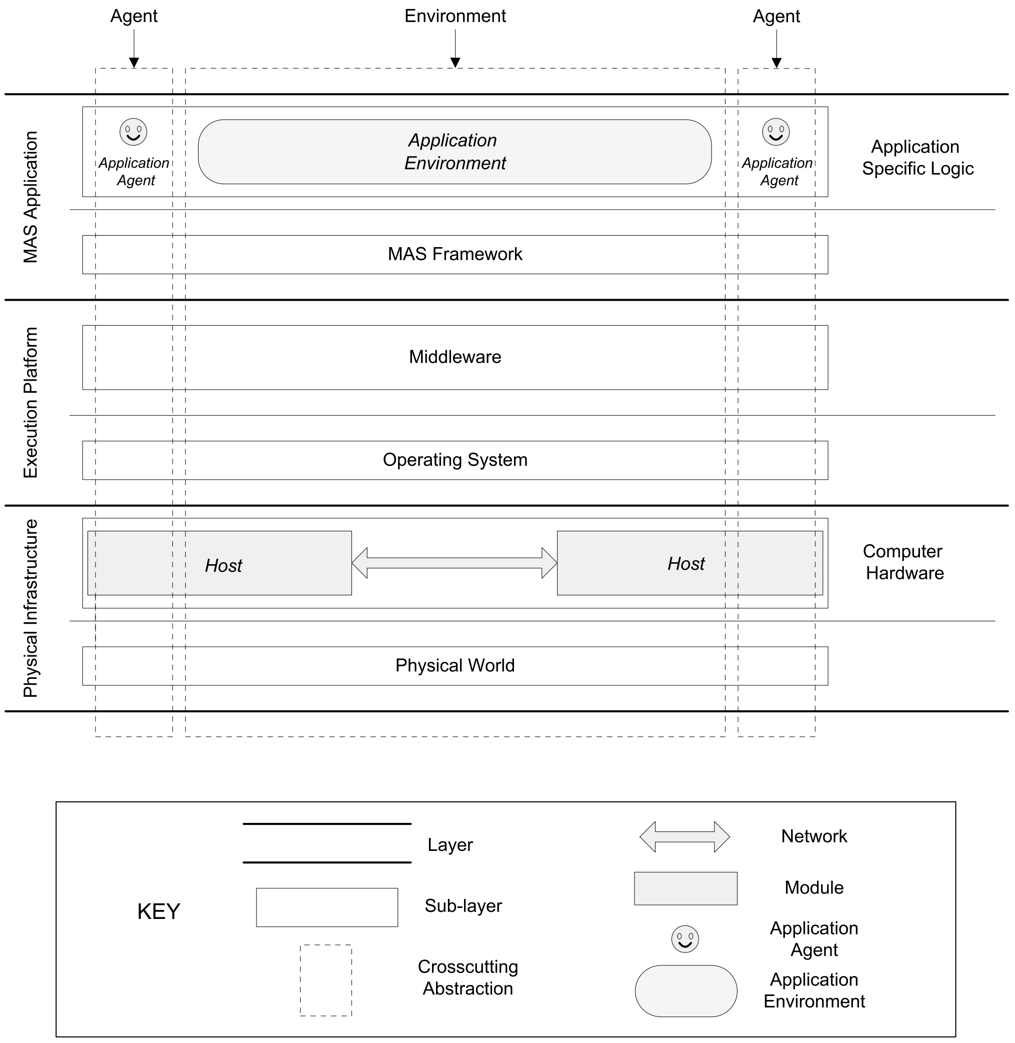


Figure 1: Three-Layer Model for Multiagent Systems.

frastructure and infrastructure for gradient fields. Another example is support for mobility of the agents.

2.2 Execution Platform

The Execution Platform is composed by a *Middleware* on top of an *Operating System*. Middleware serves as the glue between (distributed) components. It provides support for remote procedure calls, threading, transactions, persistence, load balancing, generative communication, etc. In general, middleware offers a software platform on which distributed applications can be executed. The operating system enables the execution of the application on the physical hardware, it offers basic functionality to applications, hiding low-level details of the underlying physical platform. The operating system manages memory usage and offers transparent access to lower level resources such as files, it provides network facilities, it handles the intervention of the users, it provides basic support for timing, etc.

An example of middleware support in the P2P file searching system is a distributed tuple-space infrastructure that provides a basic substrate for the pheromone and gradient field infrastructure. The operating system provides many basic functions, one example is the file system.

2.3 Physical Infrastructure

The Execution Platform runs on top of the *Physical Infrastructure*, which is composed of the *Computer Hardware* with hosts and a network, and the *Physical World*, if present in the application. In the P2P file sharing system, the physical infrastructure consists of a computer machine on each node and a connecting network. Each machine is an access point to the system for a user.

We refer the interested reader to [54] in which the three-layer model for multiagent systems is applied to several other practical applications.

2.4 Related Models

To our best knowledge, no deployment models for multiagent systems were previously proposed that explicitly discusses the position of agents and the environment. However, several layered models for multiagent system infrastructure are discussed in literature, prominent examples are Retsina [44] and JADE [4]. Here we look at two other examples, the spatial computing stack model applied to TOTA [26], and a model with multiple environments for multiagent systems proposed in [23].

TOTA. In [26], Mamei and Zambonelli introduce the notion of "spatial computing stack" and apply it to the TOTA middleware (Tuples On The Air). The spatial computing stack defines a framework for spatial computing mechanisms at four levels: the physical level at the bottom, the structure level above it, then follows the navigation level, and finally the application level at the top. The "physical level" deals with how components find each other and start communication with each other. In the case of TOTA, a node detects in-range nodes via one-hop message broadcast. The "structure level" is the level at which a spatial structure is built and maintained by components in the physical network. In TOTA, a tuple can be injected from a node. A TOTA tuple is defined in terms of a content and a propagation rule. The content represents the information carried on by the tuple and the propagation rule determines how the tuple should be propagated across the network. Once a tuple is injected it propagates and creates a centered spatial structure in the network representing some spatial feature relative to the source. At the "navigation level" components exploit basic mechanisms to orient their activities in the spatial structure and to sense and affect the local properties of space. TOTA defines an API to allow application components to sense TOTA tuples in their one-hop neighborhood and to locally perceive the space defined by them. Navigation in the space consists of agents acting on the basis of the local shape of specific tuples. At the "application level", navigation mechanisms are exploited by application components to interact and organize their activities. TOTA enables complex coordination tasks in a robust and flexible way. An example is a group of agents that coordinate their respective movements by following locally perceived tuples downhill or uphill resulting in specific formations.

The spatial computing stack model extends over the three layers of the model presented in this paper. The physical level is situated in the Physical infrastructure, the structure and navigation level are situated in the Middleware layer, and the application level finally is situated in the multiagent system Application layer.

Multiple Environments. In [23], Gouaich and Michel make a statement to model different "aspects" of the environment with different environments. Essentially, the authors consider different instances of an environment within a single multiagent system. As an illustrative example they refer to the three-layer model described in this paper and associate with each layer in this model a separate environment. The authors state that considering different environments for different aspects improves modularity and extensibility of multiagent systems. Another example discussed in the paper is the AGRE [18] model. AGRE considers a spatial, a temporal, and an organizational aspect in one environment abstraction. Gouaich and Michel state that when a new aspect is identified and must be integrated in the AGRE model, the entire model must be revised.

Unfortunately, the authors do not explain what the added value is of considering different environments for different aspects instead of dealing with different aspects *in an disciplined manner* in one environment abstraction. The authors also keep silence on crosscutting issues related to different aspects, and how the approach with multiple environments deals with this problem.

Explicitly dealing with different *concerns* of a software system is good software engineering practice. However, it is unclear whether it is useful to associate *separate* environments with different environmental concerns in a multiagent system. One way to match the approach of different environments with the three-layer model is to consider the environment as a multidimensional entity with different dimensions for different aspects/concerns, rather than a separate environment for each aspect/concern.

3 The Role of Environments in Multiagent Systems

Having clarified how the agents and the environment are first-order abstractions that span the application logic, the execution platform and the physical infrastructure, we now elaborate on the logical functionalities of the environment.

The functionalities of the environment we discuss in this section are located in the multiagent system Application layer, i.e. the top layer in Fig. 1. Several functionalities may seem quite natural responsibilities of environments. We want to stress, however, that in practice the functionalities we put forward are often dealt with in an implicit or ad hoc way. Our goal is to make the logical functionalities *explicit*, i.e. as concerns of environments as first-order abstractions. Not every functionality we discuss is relevant for every possible environment. In practice, it is up to the designer to decide which functionalities should be integrated in the environment model for the domain at hand. Finally, we want to underline that the proposed list of functionalities is not intended to be complete but rather serves as a start to explore the many-sided role of environments in multiagent systems.

3.1 Structuring

The environment is first of all a shared "space" for the agents, resources and services, which structures the whole system. Resources are objects with a specific state. Services are considered as reactive entities that encapsulate functionality. The agents as well as resources and services are dynamically interrelated to each other. It is the role of the environment to define the rules which these relationships have to comply to. As such the environment acts as a *structuring* entity for the multiagent system. This structuring can take different forms: it can be spatial, see e.g. [10][3], but also organizational, e.g. [17][57], or the environment can be structured as a mediating entity as e.g. in [20][24]. Specific properties can be defined separately for each space, such as positions, locality, groups or roles. Structuring is a fundamental functionality of the environment. The structure of the environment is a design choice that depends upon the requirements of the domain at hand, and the designer should deal with it explicitly.

3.2 Managing Resources and Services

Besides structuring, the environment is also in charge of enabling and controlling the access to resources and services. In general, resources can be read/perceived, written/modified or consumed by agents. Services on the other hand provide functionality to the agents on their request.

The extent to which agents are able to access a particular resource or service may depend on several factors such as the nature of the resource or service, the capabilities of the agent, the (current) interrelationships with other resources, services or agents, etc. In general, the access to the resources and services can be described by a set of laws defined by the domain at hand, see e.g. [19][47].

3.3 Providing Observability

Contrary to agents, the environment must be observable, i.e. agents must be able to inspect their neighborhood. Besides the observation of resources and services, agents may even be able to observe the actions of other agents [45]. In general, agents should be able to inspect the environment according to their current preferences. Examples of selective perception are [53] where "foci" are proposed to enable agents to perceive their environment related to their current tasks, and [24, 42] where "views" are proposed as selector for perception. Perception is constrained not only by agents' capabilities, but also by environmental properties (which in fact reflect properties of the problem domain). In [53] the environmental constraints are made explicit in the form of "perceptual laws".

Related to observability is the semantic description of the domain. This can be done by defining an environment ontology, see e.g. [12]. The ontology must cover the structure of the environment as well as the observable characteristics of resources, services and agents, their interrelationships, and possibly the regulating laws. In an open system, it would be useful for agents to be able to understand at run-time a new environment they are discovering. For symbolically-oriented agents, an explicit ontology should be available to the agents to enable them to interpret their environment and reason about it. For reactive/behavior-based/stigmergic agents, the designer/developer applies the ontology to encode the agents' internal structures. As such, these kinds of agents have an implicit ontology that enables them to make decisions.

3.4 Enabling Communication

Communication is inextricably bound up with multiagent systems. The environment defines concrete means for agents to communicate. Communication can take different forms. The most used scheme is a message-passing style from one agent to the other. In generative or indirect communication, agents produce communication objects in the environment and consume them to read them. Well-known properties of generative communication are name, space and time decoupling. [22] extends this list of properties with locality and non-intentionality. An important other approach of communication is based on stigmergy [36]. Each of these types of communication has its own pros and cons. Designers should be aware of the potency as well as the impact of each type of communication for their solution. Selecting a particular type of communication should be an architectural choice, determined by the requirements of the problem domain at hand.

3.5 Maintaining Environmental Processes

Besides the activity of the agents, the environment can assign particular activities to resources as well. A digital pheromone, for example, is a dynamic structure as it aggregates with additional pheromone that is dropped, it diffuses in space and it evaporates over time. Other examples are a rolling ball that moves on, or the local temperature that evolves over time. Maintaining such dynamics is an important functionality of the environment, it is useful for self-organization, see e.g. [10, 43].

3.6 Ruling the Multiagent System

The environment can define different types of rules or laws on all entities in the multiagent system. Environment rules are a powerful tool to express the capabilities an environment needs to ensure consistency in the system. Rules may restrict access to specific resources or services to particular types of agents, or determine the outcome of agents' interactions.

Dealing with interactions in multiagent systems in general is a very complex matter. In [27], Minsky and Ungureanu point out the difficulties to control the activities of agents operating in distributed systems and propose coordination policies to deal with control. According to the authors, coordination policies need to be formulated explicitly rather than being implicit in the code of the agents involved and they should be enforced by means of a generic, broad spectrum mechanism. The environment is the natural candidate to embed such control mechanism.

In electronic institutions [28], agents interact through agent group meetings that are called scenes. Interactions in a scene have to follow a well-defined communication protocol. Scenes can be composed in a performative structure. The specification of a performative structure contains a description of how the different roles can legally move from scene to scene. Agents within a performative structure may participate in different scenes at the same time with different roles. Agent actions in the context of an institution may have consequences that either limit or enlarge its subsequent acting possibilities. Such consequences will impose obligations to the agents and affect its possible paths within the performative structure. The environment can define and enforce the rules imposed on the movements and interactions of agents in an electronic institution.

A particular problem is the regulation of simultaneous actions. If we allow multiple agents to act in the environment in parallel, we need explicit models to deal with actions that range far beyond the scope of state changes based on simple individual manipulation of objects. [19] and [47] discuss models for simultaneous actions. Central to these models are (1) the distinction between the products of the agents' behavior on the one hand and the reaction of the environment on the other hand, and (2) a set of explicitly defined laws that govern the effects of the actions of the agents. These models resolve a number of fundamental issues with respect to actions in multiagent systems, however, dealing with actions in multiagent systems needs extensive further research to grow into full maturity.

4 Engineering Environments

An important condition to apply environments in practical multiagent system applications is competence in the engineering environments. Disciplined design practices for agents in general are in their infancy, and extending these techniques to environments greatly increases the scope of work to be done [49]. In this section, we first give a brief overview how current agent oriented methodologies deal with the environment. After that we discuss two proposals for engineering environments.

4.1 Environments in Agent-Oriented Software Methodologies

Popular methodologies such as Prometheus [35], Tropos [21] or Adelfe [11] offer support for some basic elements of the environment, however, they do not consider the environment as a first-order abstraction. Two methodologies that explicitly cope with the environment are SODA [30] and GAIA v.2. [57].

4.1.1 SODA

SODA takes the environment into account and provides specific abstractions and procedures for the design of agent infrastructures. In SODA, the environment is the space in which agents operate and interact. SODA provides a resource model that models the application environment in terms of the available services, associated with abstract resources. The environmental model maps resources onto infrastructure classes. An infrastructure class is characterized by the services, the access modes, the permissions granted to roles and groups, and the interaction protocols associated to its resources. Infrastructure classes can be further characterized in terms of other features: their cardinality (the number of infrastructure components belonging to that class), their location (with respect to topological abstractions), and their owner (which may be or not the same as the one of the agent system, given the assumption of decentralized control).

4.1.2 GAIA v.2.

According to GAIA v.2. (hereafter GAIA), "modelling the environment involves determining all the entities and resources that the multiagent system can exploit, control or consume when it is working towards the achievement of the organizational goal" [57] pp. 12.

In GAIA, the identification of the environmental model is part of the analysis phase and is intended to yield an abstract, computational representation of the environment in which the multiagent system will be situated. During the subsequent architectural design phases, the output of the environmental model (together with a primary role model, a preliminary interactions model, and a set of organizational rules) is integrated in the system's organizational structure that includes the real-world organization (if any) in which the multiagent system is situated. The organizational structure is then used to complete the preliminary role and interaction models. During the detailed (and final) design phase, the definition of the agent model and services model are derived from the completed role and interaction models. GAIA does not commit itself to specific techniques for modelling roles, environment and interactions, etc. The outcome of the GAIA process is a technology-neutral specification that should be easily implemented using an appropriate agent-programming framework or an object or a component-based framework. With respect to the development of the environmental model, [57] pp.23 states "it is difficult to provide general modelling abstractions and general techniques because the environments for different applications can be very different in nature and also because they are somehow related to the underlying technology." Therefore a "reasonable general approach is proposed (without the ambition to be universal), that describes the environment in terms of *abstract computational resources*, such as variables or tuples, made available to the agents for sensing (e.g. reading their values), for affecting (e.g. changing their values) and for consuming (e.g. extracting them from the environment).” As such the environmental model is represented as a list of resources, each associated with a symbolic name, characterized by the type of actions that the agent can perform on it and possibly associated with additional textual comments and descriptions. The authors of [57] confirm that in realistic development scenarios, the analyst would choose to provide a more detailed and structured view of environmental resources.

4.2 Summary

Although SODA and GAIA explicitly put forward the environment as a first-order abstraction in the methodological process, the interpretation of what the environment comprises is meagre. Design support is limited to the representation of resources and simple access control to the resources.

4.3 Engineering Approaches for Environments

In this section, we zoom in on two approaches to engineer environments. The first approach is inspired by social science, and models the environment as a set of mediating artifacts that agents can use. The second approach models the environment as a composition of modules that represent different functional concerns of the environment, such as communication, perception, actions and interaction.

4.3.1 Artifacts as Building Blocks for Engineering Environments

Inspired by Activity Theory [33], and building upon the work on coordination artifacts [32, 39], the notion of *artifact* has been proposed as an abstract building block for modeling and engineering environments [34, 46]. Contrary to an agent that is basically an autonomous, goal-oriented entity with social abilities, an artifact is a software entity designed to provide some kind of function or service that agents can *use* to achieve their goals. This characterization fits the basic distinction made in Distributed Artificial Intelligence [13] between goal-oriented entities (agents) which pro-actively interact, and function-oriented entities (artifacts) designed with a clear interface and working modalities to be used by goal-oriented entities to achieve their objectives. An artifact can be specified by: (1) its *function*, i.e. *what* services the artifact provides; (2) its *usage interface*, i.e. the set of the operations which agents can invoke to use the artifact and exploit its function; and (3) a set of *operating instructions*, i.e. descriptions that explain *how* the artifact can be used to exploit its functionality.

Artifacts can be useful from two different perspectives: (1) analytical, i.e. as a way to describe, discuss, compare existing environment models and approaches keeping a certain level of abstraction and uniformity; and (2) from an engineering perspective, i.e. as a concrete way to design and build multiagent systems. As a first rough classification, artifacts can be classified in three categories. A first class are *resource artifacts*. A resource artifact mediates the access to a specific resource, or directly represents a resource in the multiagent environment. Resource artifacts provide a representation of computational or physical entities (from objects to services, such as a web service) at the abstraction level of the agents. A second class are *coordination artifacts*. A coordination artifact provides a coordinating function or service, it can be used by agents as a tool for communication, coordination and, more generally, it support social activities in the multiagent system [33, 32]. Finally, a third class of artifacts are *organization artifacts*, which have an organizational or security function. An example of an organization artifact is a boundary artifact. A boundary artifact can be used to characterize and control the presence of an agent in an organization context, reifying and enacting a *contract* between the agent and the organization. E.g., boundary artifacts can be used as “filters”, allowing only agent actions that satisfy the contract for the specific role(s) the agent plays in an organization.

Concrete examples of artifacts in the context of the general purpose coordination infrastructure TuCSoN [26] are a *Tuple Centre* [21] and a *Agent Coordination Context* [40]. A tuple centre is an example of a coordination artifact which coordinating behavior can be specified dynamically in a language called ReSpecT. An agent coordination context is an example of a boundary artifact. An agent coordination context enables (and filters) agent actions (and patterns of actions) according to (1) the role(s) the agent plays, and (2) the organizational rules of the organization context where the agent is situated.

4.3.2 Concern-Based Engineering of Environments

The second approach models the environment as a set of modules that represent different functional concerns of the environment [47, 50]. Fig. 2 depicts a high-level module view of the environment architecture.

The *PerceptGenerator* module is responsible for perception [53]. When an $agent_i$ is interested in perceiving its neighborhood, it invokes a $sense_i$ command on the environment. Such a sense command contains one or more *foci* that expresses the agent’s current interests of perception. The *PerceptGenerator* then composes a $representation_i$ based on the foci, the current *state* of the environment and a set of *perceptual laws*. A perceptual law constrains the composition of a representation according to the requirements of the modelled domain. An example is a perceptual law that specifies how an area behind an obstacle is out of scope of a perceiving agent.

The *MessageDelivering* module is responsible for message transfer. When a message arrives, the *MessageDelivering* module passes the message to the list of addressees indicated in the message. It is possible to provide *communication laws* that are applied when messages are transferred. An examples is a communication law that specifies

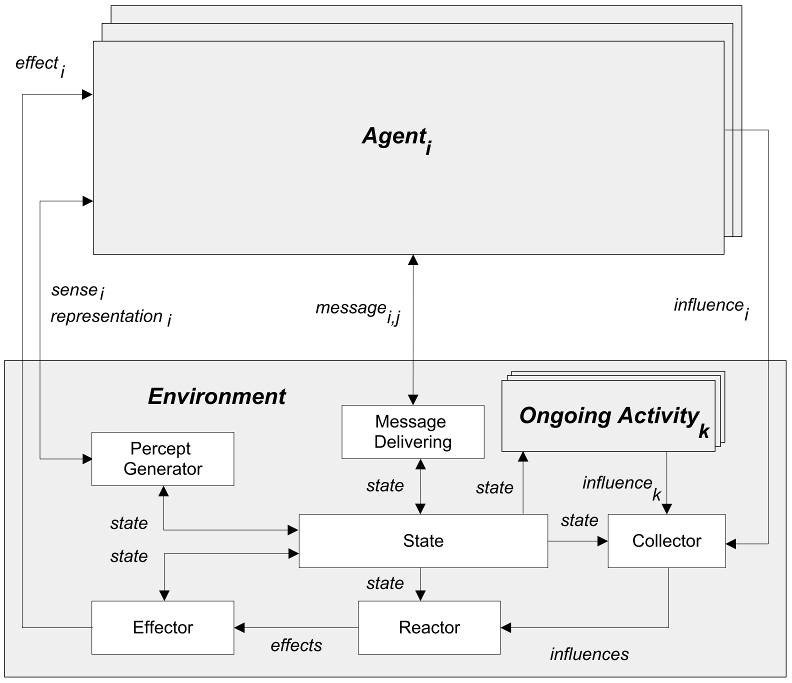


Figure 2: Concern-based modularization of the environment.

the maximal distance that messages can be delivered. Communication laws are interesting for simulation purposes, but can also be a useful instrument for designers, e.g. to regulate the message transfer.

The *Collector–Reactor–Effector* modules take care of action handling. The action model is based on the influence–reaction model of J. Ferber and J.P. Müller [19]. According to this model, agents produce influences into the environment and subsequently the environment reacts by combining the influences to deduce a new state of the world from them. The reification of actions as influences enables the environment to combine simultaneously performed activity in the system. The *Collector* module collects the influences of all simultaneously performed activity in the multi-agent system and passes them to the Reactor module. The simultaneity of activity can be based on transactional semantics, or it can be determined by a synchronization mechanism [16, 47]. The collector passed the influences to the *Reactor* module that calculates, according to a set of domain specific *interaction laws*, the reaction, i.e. state changes in the environment and effects for the agents. An example of an effect is an agent that receives a packet that it has picked up. An example of an interaction law is a law that determines the effects of two RoboCup football players that kick the ball simultaneously. The reactor finally passes the effects to the *Effector* module that applies the outcome of the interaction, i.e. it updates the state of the environment and passes the effects to the applicable agents.

Ongoing Activities correspond to environmental processes as discussed in Sect. 3. An ongoing activity is defined by an *Operation* that produces influences in the environment according to the state of the world. Examples of ongoing activities are a moving ball, an evaporating pheromone, a self-managing gradient field, or an automatic garbage collector for objects.

It is important to notice that the module view of the environment architecture as depicted in Fig. 2 abstracts from distribution. For a practical application, the state of the environment, the delivering of messages, ongoing activities, etc. will be implemented according to the domain at hand, i.e. centralized or distributed. Another important remark is that the presented model also abstracts from real-word resources, external to the multiagent system. The *state* of the environment may represent external resources. Support to keep the state of the representation consistent with external resources is not covered by the presented model. In the next section, we discuss an example where the state of the environment represents resources in the physical world.

5 Applying the Environment in a Real-World Application

In this section, we illustrate how we have applied the approach of concern-based engineering of environments to an automated transportation system for warehouse logistics. This real-world application is developed in a joint R&D project between the AgentWise research group and Egemin, a manufacturer of automating logistics services in warehouses and manufactories [15, 52].

The automated transportation system uses automatic guided vehicles (AGVs) to transport loads through a warehouse. Typical applications are distributing incoming goods to various branches, or distributing manufactured

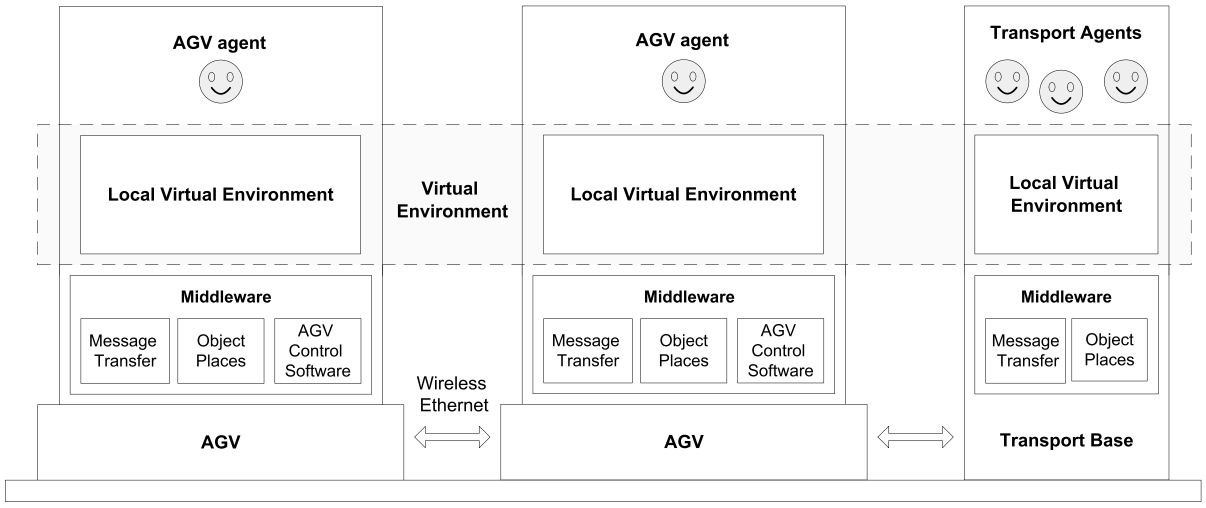


Figure 3: High-level model of the AGV transportation system.

products to storage locations. An AGV is provided with a battery as its energy source. AGVs can move through a warehouse, following fixed paths on the factory floor, typically guided by a laser navigation system, or by magnets or cables that are fixed in the floor. The low-level control of the AGVs in terms of sensors and actuators (such as staying on track on a path, turning, and determining the current position, etc.), is handled by the AGV control software. Fig. 3 depicts a high-level model of the situated multiagent system. The situated multiagent system consists of two kinds of agents, *transport agents* and *AGV agents*. Transport agents are located at *transport bases*. AGV agents are located in AGVs that are situated on the factory floor. The communication infrastructure provides a wireless network that enables mobile AGVs to communicate with each other and with transport agents on transport bases.

A transport agent represents a transport that needs to be handled by an AGV. AGV agents are responsible for executing the assigned transports. AGVs are situated in a physical environment, however, this environment is very constrained: AGVs cannot manipulate the environment, except by picking and dropping loads. This restricts how AGV agents can exploit their environment. Therefore, a virtual environment was introduced for agents to live in. This virtual environment offers a medium that agents can use to exchange information and coordinate their behavior. Besides, the virtual environment serves as a suitable abstraction that shields the AGV agents form low-level issues, such as the physical control of the AGV. The AGV control software that deals with the low-level control of the AGVs is fully reused. As such, the AGV agents control the movement and actions of AGVs on a fairly high level.

In the AGV application, the only physical infrastructure available to the AGVs is a wireless network for communication. In other words, the virtual environment is necessarily distributed over the AGVs and transport bases. In effect, each AGV and each transport base maintains a *local virtual environment*, which is a local manifestation of the virtual environment. Local virtual environments are merged with other local virtual environments opportunistically, as the need arises. In other words, *the* virtual environment as a software entity does not exist; rather, there are as many local virtual environments as there are AGVs and transport bases. Some of these local virtual environments may have been synchronized recently with each other, while others may not. From the agent perspective, the virtual environment appears as one entity. The synchronization of the state of neighboring local virtual environments is supported by the ObjectPlaces middleware [42].

We now illustrate the use of the virtual environment with a couple of examples.

Routing. For routing purposes, the virtual environment has a static map of the paths through the warehouse. This graph-like map corresponds to the layout used by low-level AGV control software. To allow agents to find their way through the warehouse efficiently, the virtual environment provides signs on the map that the agents use to find their way to a given destination. These signs can be compared to traffic signs by the road that provide directions to drivers. At each node in the map, a sign in the virtual environment represents the cost to a given destination for each outgoing segment. The cost of the path is the sum of the static costs of the segments in the path. The cost per segment is based on the average time it takes for an AGV to drive over the segment. The agent perceives the signs in its environment, and uses them to determine which segment it will take next.

Traffic Information. Besides the static routing cost associated with each segment, the cost is also dependent on dynamic factors, such as congestion of a segment. To warn other agents that certain paths are blocked or have a long waiting time, agents mark segments with a dynamic cost on a *traffic map* in the virtual environment. Agents mark the traffic map by dropping pheromones on the applicable segments. When AGVs come in each others neighborhood, the information of the traffic maps is exchanged and merged to provide up-to-date information to the AGV agents. Since pheromones evaporate over time, outdated information automatically vanishes over time. AGV agents take the information on the traffic map into account when they decide how to drive through the warehouse.

Collision Avoidance. AGV agents avoid collisions by coordinating with other agents through the virtual environment. AGV agents mark the path they are going to drive in their environment using *hulls*. The hull of an AGV is the physical area the AGV occupies. A series of hulls then describes the physical area an AGV occupies along a certain path. If the area is not marked by other hulls (the AGV's own hulls do not intersect with others), the AGV can move along and actually drive over the reserved path. Afterwards, the AGV removes the markings in the virtual environment. [51] discusses collision avoidance through the virtual environment in detail.

In summary, the virtual environment serves as a flexible coordination medium, which hides much of the distribution of the system from the agents: agents coordinate by putting marks in the environment, and observing marks from other agents. The virtual environment creates opportunities beyond a physical environment that situated AGV agents can exploit.

6 Conclusions and Challenges

There is a growing awareness in the multiagent research community that the environment plays a crucial role in multiagent systems. In this paper, we discussed the role of environments in multiagent systems. Important responsibilities of the environment are: (1) the environment structures the multiagent system as a whole; (2) the environment is in charge to managing resources and services; (3) contrary to agents, the environment must be observable; (4) the environment must define concrete means for the agents to communicate; (5) the environment is responsible to maintain ongoing processes in the system; and finally (6) the environment can define different types of rules on all the entities in the multiagent system.

The research track on environments is still young and many issues are open for future research, we have just started to explore the possible responsibilities of environments in multiagent systems. The term "environment" is vague and ill-defined in relation to multiagent systems. An ongoing research challenge will be developing a clearer understanding of what we mean by an "environment." In this paper we have discussed an initial model for multiagent systems that considers agents and the environment as first-order abstractions. These abstractions span the application logic, the execution platform and the physical infrastructure of the mutiagent system. However, the exact nature of the relationship between the agent software, the environment software, and the software and hardware that make up the computational substrate needs further clarification. Recent initiatives tackle these and related research questions, see [14, 34].

The engineering of environments is still in its infancy. In this paper, we discussed two initial models for engineering environments: artifacts and concern-based modularization. Study of agent-oriented methodologies shows that current methodologies offer little support for designing environments, a whole domain of work is waiting to be tackled. From a methodological point of view, the environment should be considered as a first-order abstraction in design models and description languages. Initial work in that direction has been conducted, e.g. [5]. Agent-oriented programming has led to the proliferation of frameworks and development platforms for agents. Recognition of the importance of environments will stimulate extensions to these tools, or even the development of new tools that can support environments within which agents from different platforms can interact. Exploring work in that direction is on its way, see e.g. [7].

Besides the research work, we have to apply environments in real-world multiagent system applications. In this paper, we discussed a practical application and showed how a virtual environment creates opportunities for agents to exchange information and coordinate their behavior in a way that would be impossible in the physical environment. Encountering the complexity of real applications will urge us to invent new ways to exploit environments.

Acknowledgement

We would like to express our appreciation to the attendees of the workshops on Environments for Multiagent Systems in New York, 2004 and Utrecht 2005, and the AgentLink III Technical Forum in Ljubljana, 2005 for the inspiring discussions that have considerably contributed to the work presented in this paper. A word of appreciation also goes to the anonymous reviewers for their usefully comments to improve this paper.

References

[1] R.C. Arkin. *Behavior-based robotics*. Massachusetts Institute of Technology, MIT Press, Cambridge, MA, USA, 1998.

[2] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A framework for the development of agent-based Peer-to-Peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 15–22, Vienna, Austria, 2002. IEEE Computer Society, Digital Library.

[3] S. Bandini, S. Manzoni, and G. Vizzari. A spatially dependent communication model for ubiquitous systems. In Weyns et al. [48], pages 74–90.

[4] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Software - Practice and Experience*, 31(2):103–128, 2001.

[5] C. Bernon, M. Cossentino, and J. Pavón. An Overview of Current Trends in European AOSE Research. *In this volume*.

[6] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunications networks with ant-like agents. In *Proceedings of the 2nd International Workshop on Intelligent Agents for Telecommunication a Applications*, pages 60–71, Paris, France, 1998. Springer, London, UK.

[7] R. Bordini, L. Braubach, A. El Fallah-Seghrouchni, M. Dastani, J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A survey on languages and platforms for MAS implementation. *In this volume*.

[8] J. M. Bradshaw, N. Suri, A. Ca nas, R. Davis, K. Ford, R. Hoffman, R. Jeffers, and T. Reichherzer. Terraforming Cyberspace. *Computer*, 34(7):48–56, 2001.

[9] R. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139Ű–159, 1991.

[10] S. Brueckner. *Return from the ant, Synthetic ecosystems for manufacturing control*. Ph.D Dissertation, Humboldt University, Berlin, Germany, 2000.

[11] S. Peyruqueou G. Picard C. Bernon, M. P. Gleizes. ADELFE: A methodology for adaptive multiagent systems engineering. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *Lecture Notes in Computer Science*, pages 156–169, Madrid, Spain, 2003. Springer, Berlin, Heidelberg, Germany.

[12] P. Chang, K. Chen, Y. Chien, E. Kao, and V. Soo. From reality to mind: A cognitive middle layer of environment concepts for believable agents. In Weyns et al. [48], pages 57–73.

[13] R. Conte and C. Castelfranchi, editors. *Cognitive and social action*. UCL Press, University College, London, UK, 1995.

[14] E4MAS. International workshop series on Environments for Multiagent Systems. *http://www.cs.kuleuven.ac.be/$\sim$distrinet/ events/e4mas/*, 8/2005.

[15] Egemin Modular Controls Concept. EMC ^{2} project, Flemish Institute for the Advancement of Scientific-Technological Research in the Industry, IWT, Belgium. *http://emc2.egemin.com/*, 8/2005.

[16] J. Ferber. *An introduction to distributed artificial intelligence*. Addison-Wesley, London, UK, 1999.

[17] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multiagent systems. In P. Giorgini, J. P. Müller, and J. Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230, Melbourne, Australia, 2003. Springer, Berlin, Heidelberg, Germany.

[18] J. Ferber, F. Michel, and J. Baez. AGRE: Integrating environments with organizations. In Weyns et al. [48], pages 48–56.

[19] J. Ferber and J. P. Müller. Influences and reaction: A model of situated multiagent systems. In M. Tokoro, editor, *Proceedings of the 2th International Conference on Multi-agent Systems*, pages 72–80, Kyoto, Japan, 1996. American Association for Artificial Intelligence, AAAI Press, Menlo Park, California, USA.

[20] D. Gelernter and D. Carrierro. Coordination languages and their significance. *Communications of the ACM*, 35(2), 1992.

[21] F. Giunchiglia, J. Mylopoulos, and A. Perini. The TROPOS software development methodology: Processes, models and diagrams. In C. Castelfranchi and W. L. Johnson, editors, *Proceedings of the 1st Joint Conference on Autonomous Agents and Multiagent Systems*, pages 35–36, Bologna, Italy, 2002. ACM Press, New York, NY, USA.

[22] D. Goldin and D. Keil. Toward domain-independent formalization of indirect interaction. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 393–394, Modena, Italy, 2004. IEEE Computer Society, Digital Library.

[23] A. Gouaich and F. Michel. Towards a unified view of environment(s) within multiagent systems. *In this volume*.

[24] C. Julien and G. C. Roman. Egocentric context-aware programming in ad hoc mobile environments. In *Proceedings of the 10th Symposium on Foundations of Software Engineering*, pages 21–30, Charleston, South Carolina, USA, 2002. ACM Press, New York, NY, USA.

[25] P. Maes. Modeling adaptive autonomous agents. *Artificial Life*, 1(1-2):135–162, 1994.

[26] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *Proceedings of the 2nd International Conference on Pervasive Computing and Communications*, pages 263–276, Orlando, Florida, 2004. IEEE Computer Society, Washington, DC, USA.

[27] N. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering Methodologies*, 9(3):273–305, 2000.

[28] P. Noriega and C. Sierra. Electronic institutions: Future trends and challenges. In *Proceedings of the 6th International Workshop on Cooperative Information Agents*, volume 2446 of *Lecture Notes in Computer Science*, pages 14–17. Springer-Verlag, London, UK, 2002.

[29] J. Odell, V. Parunak, M. Fleischer, and S. Breuckner. Modeling agents and their environment. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *Agent-Oriented Software Engineering III*, volume 2585 of *Lecture Notes in Computer Science*, pages 16–31, Bologna, Italy, 2003. Springer, Berlin, Heidelberg, Germany.

[30] A. Omicini. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computer Science*, pages 185–193, Limerick, Ireland, 2001. Springer, Berlin, Heidelberg, Germany.

[31] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, 2001.

[32] A. Omicini, A. Ricci, M. Viroli, C. Cristiano, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In N. Jennings, M. Tambe, C. Sierra, L. Sonenberg, S. Parsons, and E. Sklar, editors, *3rd Joint Conference on Autonomous Agents and Multiagent Systems*, pages 286–293, New York, NY, USA, 2004. IEEE Computer Society, USA.

[33] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and multiagent systems*, 2(3):251–269, 1999.

[34] AgentLink Technical Forum Group on Environments for Multiagent Systems. *http://www.cs.kuleuven.ac.be/∼distrinet/events/e4mas/tfg2005/*, 8/2005.

[35] L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *Agent-Oriented Software Engineering III*, volume 2585 of *Lecture Notes in Computer Science*, Bologna, Italy, 2003. Springer, Berlin, Heidelberg, Germany.

[36] V. Parunak. Go to the ant: Engineering principles from natural agent systems. *Annals of Operations Research*, 75:69–101, 1997.

[37] V. Parunak. The AARIA Agent architecture: From manufacturing requirements to agent-based system design. *Integrated Computer-Aided Engineering*, 8(1), 2001.

[38] A. Ricci, A. Omicini, and E. Denti. Activity theory as a framework for MAS coordination. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *Lecture Notes in Computer Science*, pages 96–110, Madrid, Spain, 2003. Springer, Berlin, Heidelberg, Germany.

[39] A. Ricci and M. Viroli. Coordination artifacts: A unifying abstraction for engineering environment-mediated coordination in MAS. *In this volume*.

[40] A. Ricci, M. Viroli, and A. Omicini. Agent coordination context: From theory to practice. *Cybernetics and Systems*, 2:618–623, 2004.

[41] J. Sauter and V. Parunak. ANTS in the supply chain. In *Proceedings of the Workshop on Agent-Based Decision Support Managing Internet-Enabled Supply Chain*, pages 1–9, Seattle, WA, USA, 1999.

[42] K. Schelfthout and T. Holvoet. Views: Customizable abstractions for context-aware applications in MANETSs. In A. Garcia, R. Choren, C. Lucena, A. Romanovsky, T. Holvoet, and P. Giorgini, editors, *Software Engineering in Large-Scale Multiagent Systems*, St. Louis, USA, 2005. ACM Press, Digital Library.

[43] G. Di Marzo Serugendo, M. P. Gleizes, and A. Karageorgos. Self-organisation and emergence in MAS: An overview. *In this volume*.

[44] K. Sycara, M. Paolucci, M van Velsen, and J. Giampapa. The Retsina MAS infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):29–48, 2003.

[45] L. Tummolini, C. Castelfranchi, A. Omicini, A. Ricci, and M. Viroli. "Exhibitionists" and "Voyeurs" do it better: A shared environment for flexible coordination with tacit messages. In Weyns et al. [48], pages 215–231.

[46] M. Viroli, A. Ricci, and A. Omicini. Engineering MAS environment with artifacts. In D. Weyns, V. Parunak, and F. Michel, editors, *Proceedings of 2nd International Workshop on Environments for Multiagent Systems*, pages 1–16, Utrecht, The Netherlands, 2005.

[47] D. Weyns and T. Holvoet. Formal model for situated multiagent systems. *Fundamenta Informaticae*, 63(2-3):125–158, 2004.

[48] D. Weyns, V. Parunak, and F. Michel, editors. *Proceedings of the 1st International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, Germany, 2005. Springer.

[49] D. Weyns, V. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multiagent systems, State-of-the-art and research challenges. In Weyns et al. [48], pages 1–47.

[50] D. Weyns, K. Schelfthout, and T. Holvoet. Architectural design of a distributed application with autonomic quality requirements. In D. Garlan, M. Litoiu, H. Ḿller, J. Mylopoulos, D. Smith, and K. Wong, editors, *Design and Evolution of Autonomic Computing Software*, St. Louis, USA, 2005. ACM Press, Digital Library.

[51] D. Weyns, K. Schelfthout, and T. Holvoet. Exploiting a virtual environment in a real-world application. In D. Weyns, V. Parunak, and F. Michel, editors, *Proceedings of 2nd International Workshop on Environments for Multiagent Systems*, pages 1–18, Utrecht, The Netherlands, 2005.

[52] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *4th Joint Conference on Autonomous Agents and Multiagent Systems, Industry Track*, pages 67–75, Utrecht, The Netherlands, 2005. ACM Press, New York, NY, USA.

[53] D. Weyns, E. Steegmans, and T. Holvoet. Towards active perception in situated multiagent systems. *Applied Artificial Intelligence*, 18(9-10):867–883, 2004.

[54] D. Weyns, G.. Vizzari, and T. Holvoet. "Environments for multiagent systems: Beyond infrastructure. In D. Weyns, V. Parunak, and F. Michel, editors, *Proceedings of 2nd International Workshop on Environments for Multiagent Systems*, pages 1–17, Utrecht, The Netherlands, 2005.

[55] Whitestein Technologies, Living Systems. *http://www.whitestein.com/pages/index.html*, 8/2005.

[56] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[57] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The GAIA methodology. *Transactions on Software Engineering and Methodology*, 12(3):317–370, 2003.

Towards a Unified View of the Environment(s) within Multi-Agent Systems

Abdelkader Gouaïch
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France
gouaich@lirmm.fr and www.lirmm.fr/∼gouaich

Fabien Michel
Laboratoire d'Etudes et de Recherches Informatiques
LERI, Rue des Crayères, B.P. 1035, 51687 REIMS Cedex 2, France
fmichel@leri.univ-reims.fr and www.univ-reims.fr/Labos/LERI/membre/fmichel

Keywords: MASs, environment, multi-environments

Received: May 6, 2005

Within the Multi-agent systems (MASs) paradigm, the concept of the environment plays a central role. In fact, the autonomous agents do only exist when they are deployed on an environment. Still, there is an implicit hypothesis in current trends of the MASs considering that the agents are related to only one environment that captures all the different aspects of the application domain. In this paper we challenge this implicit hypothesis by enabling multiple occurrences of the agent-environment relationship. This brings clarity and modularity for the design and implementation of complex MASs since each environment targets a specific aspect of the application. Thanks to the proposed characterization of the agent-environment relationship, the agents are still offered a unified view about all the environments.

Povzetek: Analizira povezave MAS z okoljem in pokaže, da jih je bolje imeti več.

1 Introduction

The paradigm of Multi-agent systems (MASs) naturally implies the idea that the agents are embedded in an environment. Indeed, as J. Odell and colleagues have pointed out: "without an environment, an agent is effectively useless" [11]. In fact, even if designers do not always consider the environment as a primary abstraction when engineering MAS applications, the main agency definitions do always mention that an agent is an entity that is operating in an environment using perception and action means, see e.g. [15, 3, 21, 20]. As highlighted in [19], this is even more obvious when considering situated MASs where the agents are placed within an environment that may comprise processes that do modify the state of the world independently from the actions of the agents. Consequently, such processes are modeled as parts of the environment which is thus considered as a first-class entity. Recent works have shown that the environment has clearly a rich potential not only for situated MASs but for the paradigm of MASs as a whole [18].

In this paper, we clearly embrace the idea that the environment has to be considered as a first order abstraction. However, the implicit hypothesis stating that all the agents of a MAS **share a common environment** has to be revised. Indeed, the agency definitions implicitly propose a vision where the agents only belong to one and unique environment that captures all the different aspects of the application domain.

Complex MASs often need to define different environments to capture different aspects of the application domain. However, due to the usual *single environment view*, this is generally done in an ad hoc manner. This has contributed to the confusion related with the notion of environment within MASs. The main goal of this of paper is to promote the idea that several environments can coexist within a single MAS application. To achieve this, we propose to characterize the relationship that maps an agent to an environment by the following features: *(i)* ontology of the environment, *(ii)* perceptions means, *(iii)* action means, *(iv)* interaction functions and *(v)* localization function in case of situated environments.

Once the relationship that links an agent to an environment has been characterized, it is then possible to consider multiple occurrences of this relationship between an agent and several environments. The agents are still offered a unified view of what is an environment, but each specific environment has its own way to implement the features of the agent-environment relationship.

2 Background

In [11], J. Odell and colleagues identify different types of environments that have been used within MASs applications. This work clearly identifies that MASs applications, depending on their application domain, need different kind of environments. However, J. Odell and colleagues have studied the characteristics of the environments rather than studying the characteristics of the relationship that maps agents to environments. Furthermore, the different classes of environments were studied and analyzed separately and the coexistence of several environments within the same MAS has not been considered.

In [19], D. Weyns and colleagues present the 3-Layer model which is a first step towards a better understanding of the different concerns which have to be considered when studying the environments of MASs. Figure 1 presents the 3-Layer model that distinguishes between three different classes of environments that exist at three different layers: *(i)* the environment of the MAS application layer, namely the *application environment*, *(ii)* the environment defined by the *execution platform* (a generic middleware) and *(iii)* the environment defined by the *physical infrastructure*. This decomposition identifies that there are several kinds of environments within a single MAS application. Still it implicitly considers that the agents, at the MAS application layer, are in relation with only one single environment. Furthermore, the relationship between the agents and their different environments is not studied. In this paper, we try to characterize this relationship independently from the layer where the environment is defined. For instance, the agent-environment relationship is the same for the environments that are defined within the application layer and those defined within the execution platform.

J. Ferber and colleagues were facing a domain of application, social simulations, where the autonomous agents are both situated in a spatial environment and in an organizational environment. They have developed a model that includes both the spatial and the organizational aspects in a single environment, namely the AGRE model [5] (cf. figure 2). The AGRE model merges the concepts of two aspects of the application domain: the organizational aspects and the spatial and temporal aspects to represent the agents within a virtual space. But this approach is limited since if another aspect of the application domain is identified, then the entire AGRE model has to be revised in order to include new concepts.

S. Bandini and colleagues propose the *Multilayered Multi-Agent Situated Systems* (MMASS) model for the definition of structured environments for situated MASs. The MMASS model relies on decomposing the environment in several different layers that represent physical or conceptual abstractions, specifying different aspects of the whole system [1]. Such an approach also highlights the fact that the environment may be defined according to many different aspects. Indeed, MASs may be composed by heterogeneous agents that may need different action and perception means achieving their goals. The MMASS model thus provides an interesting framework that allows to explicitly take into account different aspects of the application. Still the MMASS model focuses on situated MASs aiming to provide an explicit representation of agent environments and interaction mechanisms that are strongly dependant on the position of agents and on the spatial structure of the environment.

3 Revising some Assumptions on Environments

This section presents in an informal manner how some implicit assumptions on environments are revised. The idea is to start from a well established diagram that shows the agent-environment relationship. This diagram is then modified to obtain the vision of the agent-environment proposed in this paper.

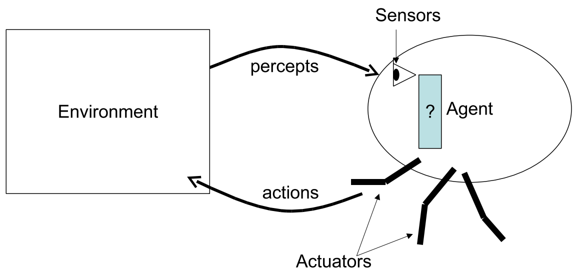


Figure 3: The original agent-environment diagram presented in [15].

Figure 3 shows the original diagram which has been presented in S. Russell and P. Norvig book [15]. This diagram gives an idea of what is the relationship between an agent and an environment. Still, many implicit hypotheses and assumptions are found in this diagram. The first point concerns where the agent's actuators and sensors are defined. Figure 3 situates the actuators and sensors on the side of the agent. This means that the actuators and sensors are defined by the ontology of the agent which makes the environment dependent on the ontology of the agents. This is not suitable since the environment has to be independent from the specific model of an agent. In fact, an environment can hold heterogeneous agents that use different ontologies and reasoning models.

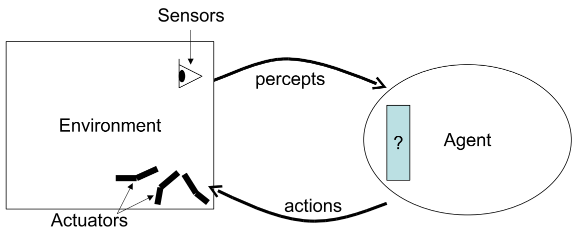


Figure 4: Defining the action and perception means of the agent on the environment side.

By contrast to figure 3, figure 4 places the actuators and sensors of the agent on the environment side. This also introduces the need for the ontology of the environment. In

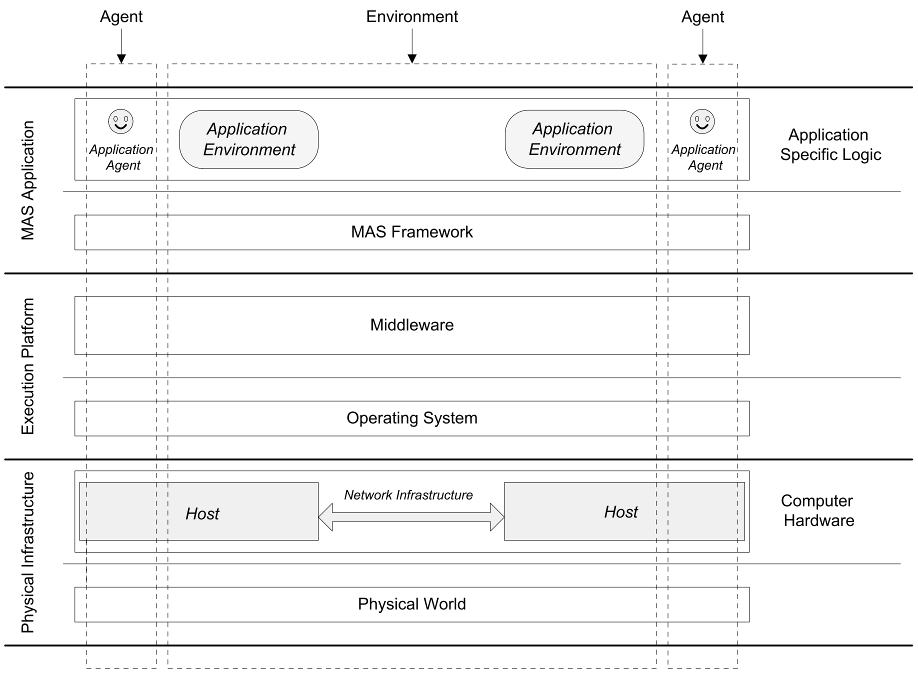


Figure 1: 3-Layer model for MASs [19].

fact, the actuators and sensors have to be explicitly defined by the ontology of the environment as means offered to the agents in order to perceive and act on the environment. The environment also defines the interaction functions that describe the relationship between the action means and perception means. In other words, the environment describes what is the result of the interaction between the actuators and sensors. It is important to notice that the interaction has been shifted from the agents to the action and interaction means. So, the agents do not directly interact, but their action and perception means interact within the environment.

Figure 5 presents a first step towards a general vision of the agent-environment relationship. In fact, an agent can be related to more than one environment. Each environment defines its specific ontology, perception and action means and interaction functions. If the perception and action means were not extracted from the agent and then placed in the environment, as it is the case in figure 3, the multiple instantiation of the agent-environment relationship would have been more difficult since every environment would have to follow the ontology of each agent.

The agent-environment relationship has to be distinguished from the means which are offered to the agent in one of its environment. In fact, any communication medium that enables the communication between the agent and the environment can be used. Particularly, some environments can be used as communication media. Figure 6 shows this case, where the agent is related to two environments 'environment 1' and 'environment 2'. This agent directly accesses 'environment 1'. However, the access to

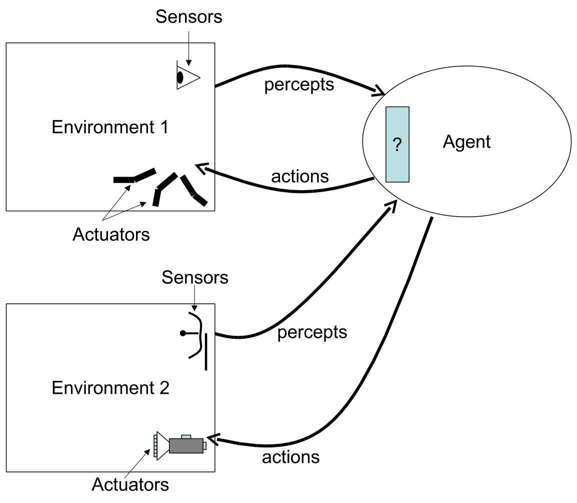


Figure 5: Multiple occurrences of the agent-environment relationship.

'environment 2' is done through 'environment 1' which is considered, in this case, as a communication medium. It is important to notice that 'environment 2' is also related to 'environment 1' in order to act and retrieve the perception results. This schema is similar to the 3-Layer model, where the agents use the 'execution platform' environment in order to implement their relation with the 'application environment'.

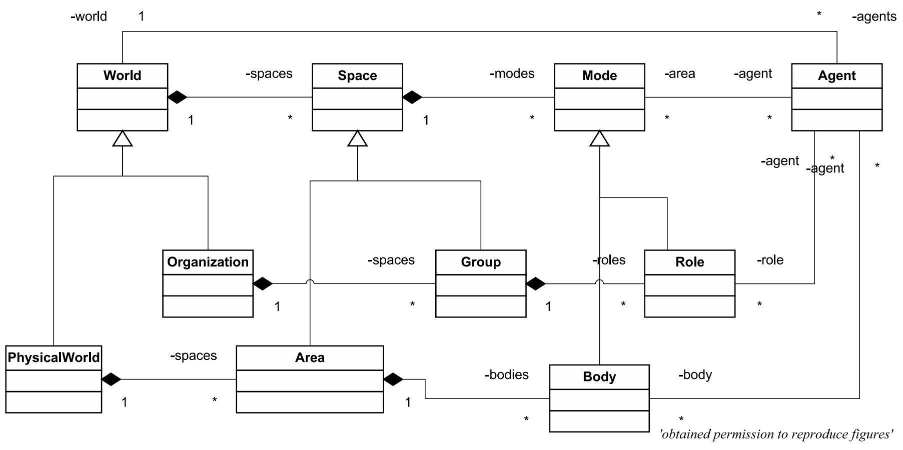


Figure 2: The UML meta-model of the AGRE model [5].

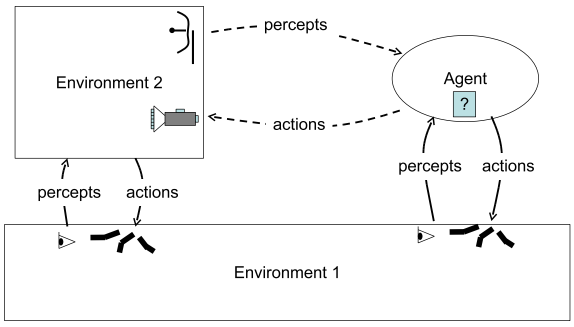


Figure 6: Using an environment as a communication medium to access another environment.

4 Generalizing the Agent-Environment Relationship

In the previous section, we have seen that there is an implicit hypothesis that considers that the autonomous agents do exist in a single environment. Consequently, this environment is supposed to contain all the different aspects and logics of the MAS application. Still, both from a conceptual point of view and from an engineering point of view, this hypothesis is inappropriate when dealing with complex software systems. Moreover, such an approach does not help to understand the role played by the environment within the MAS framework. In fact, each application domain has its own view of what is an environment and what are the functionalities implemented by an environment. Current approaches that have faced the problem of designing a MAS application where the autonomous agents exist in an environment that captures more than one aspect of the application domain have suggested to merge these aspects in a single environment. These approaches are limited since each time a different aspect of the application domain is identified then this aspect is appended to the environment in an ad hoc manner. As a result, the environment centralizes all the different aspects of the targeted application. Such an environment contradicts the modularity and separation of concerns principles which have been proved to be useful when designing complex software systems.

We suggest an approach that: *(i)* challenges the implicit hypothesis that considers only one environment which is commonly shared by the autonomous agents; *(ii)* generalizes current approaches that have identified that several environments are required to capture all the aspects of the application domain.

Such an approach may seem to contrast with the 3-layer approach proposed in this volume by D. Weyns and T. Holvoet, where only one environment crosscuts the three layers (cf. figure 1) [17]. In fact, such a contrast comes from the fact that the two approaches do not have the same objectives: the 3-layer model aims to highlight that several concerns have to be taken into account when engineering environments for MASs. Rather than that, our goal is to characterize the agent-environment relationship and our claim is that this relation has to be instantiated as many times as required. Further study is necessary to clarify the distinction between the two models.

The problem now is to have a more precise idea of what is meant by the existence of agents in an environment. Initially, the term existence was used in [15] by S. Russell and P. Norvig in order to highlight the central role played by the environment within the MASs paradigm. However, up until now there is not a consensual definition of this existence relationship that links the autonomous agents and their environments.

4.1 Characterizing the Agent-Environment Relationship

In order to offer a general approach, we study the relationship between an agent and an environment rather than defining a specific model of an environment. Once this relationship has been characterized, then different environments implement it differently according to the aspect of the application domain which is captured. Thus, the agent is offered a unified view of the existence relationship that links the agent to the different environments.

Starting from the state of the art proposed by D. Weyns and colleagues in [19] and from previous works on the development of environments for autonomous agents [8], the following elements are suggested as a characterization of the agent-environment relationship:

– the *ontology of the environment* has to be distinguished from the ontology of the agents. In fact, the environment defines its own concepts and their logics. To operate in an environment, the agents need to understand some parts of the ontology of the environment. When the agents use an internal ontology that differs from the ontology of the environment, it is their responsibility to map the concepts of their own ontology to the ontology of the environment. For instance, we can conceive the use of cognitive agents, such as BDI agents, in a spatial grid environment. The ontology of the spatial grid environment defines concepts such as: pheromones, cells, movement, position and so on. This ontology does not have any concept of mental states which are present in the BDI model for instance. So, it is the responsibility of the cognitive agent to translate the concepts of the spatial grid environment into some logical predicates which are included in its mental states. Obviously, it is not always easy to translate the concepts defined by the ontology of the environment into the agent's ontology. For instance, even if it is theoretically possible to imagine BDI agents deployed on a spatial grid environment where they can move and drop pheromones, this is very hard to achieve in practice as the ontology of the BDI agents and the ontology of the spatial grid environment are dissimilar.

– the *perception means* are the concepts defined by the ontology of the environment and that enable the agents to perceive their surroundings.

– the *action means* are the concepts expressed by the ontology of the environment and that enable agents to influence their surroundings.

– the *interaction functions* define the relationship between the action means and perception means. It is important to notice that the interaction is not defined between the agents, but between the action and interaction means that are defined by the ontology of the environment.

Besides, for a situated environment, an additional element characterizes this agent-environment relationship:

– the *localization function* is specifically provided by situated environment. In a situated environment, one can define the location of an agent in terms of coordinates within the environment. The location of an agent is also defined as a concept of the ontology of the environment.

This characterization of the agent-environment relationship offers a unified view of the environments which are considered either as: *(i)* an infrastructure or *(ii)* an application-level environment.

4.2 Illustrating some Agent-Environment Relationships

To illustrate the suggested characterization of the agent-environment relationship, let us consider the review of some existing environments.

4.2.1 Tuple Spaces

Tuple Spaces (TSs) have been introduced by researchers at the Yale University where *Linda* [6] –the first tuple space-based system– has been developed. A TS system is composed by the following elements:

– a *tuple* is basically a list of typed fields. Fields may be *actual* when they hold a value or *formal* otherwise.

– a *tuple space* is an abstract storage location where tuples are deposited and retrieved by the software entities which are called processes.

– the *processes* store and retrieve the tuples using the following primitives:

  – out: this primitive inserts a tuple into a tuple space which becomes visible to all the processes that have access to that tuple space.
  – in: this primitive extracts a tuple from a tuple space, with its argument acting as the template, *anti-tuple*, against which to match. When all the corresponding fields of a tuple match the template the tuple is withdrawn from the tuple space.
  – read: this primitive is equivalent to the 'in', except that a matched tuple is not withdrawn from the tuple space and remains visible to the other processes.
  – eval: this is similar to the 'out', except that it creates an independent process yielding the tuple which is inserted in the tuple space.

The Tuple Space as an Environment

– the ontology of the tuple space defines the concepts that follow: tuple, anti-tuple, tuple space, out, in, read and eval. So, in order to exist in a tuple space, an agent has to understand these concepts and to be able to translate them into the concepts of its own ontology.

– the action means offered by a tuple space are represented by the out primitive. In fact, an agent can influence other agents by putting a tuple within the tuple space using the out primitive.

– the perception means offered by a tuple space are represented by the concept of anti-tuple and by the in and read primitives. So, an agent can perceive within a tuple space by using a template represented by the anti-tuple as argument for the in or read primitives.

– the interaction functions defined by the tuple space are the rules that make an anti-tuple matching a tuple. These rules make the link between the action means and perception means of the agents. Some tuple space architectures such as TuCSoN [12] change the interaction functions of the tuple spaces and add some *reaction rules* in order to dynamically change the interactions that take place between the tuples and anti-tuples.

4.2.2 MIC*

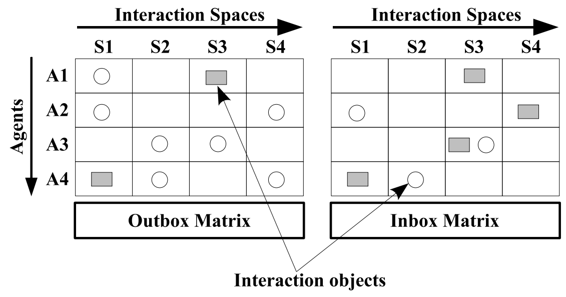


Figure 7: The MIC* structure.

MIC* is an algebraic model of a MAS infrastructure holding autonomous and interacting agents [8]. The MIC* structure (cf. figure 7) is composed of two matrices where rows represent the agents $i \in \mathcal{A}$ and the columns represent interaction spaces $j \in \mathcal{S}$. The elements of both matrices are called *interaction objects*. Interaction objects represent information carriers but also model the interaction means (i.e. actuators and sensors) of the agents within the system. Moreover, interaction objects are formally defined so that their structure is a commutative group $(\mathcal{O}, +)$: a composition of interaction objects is also an interaction object (see [7] for a complete mathematical description of the MIC* model). The outbox matrix and the inbox matrix can be described as follows:

1. The outbox matrix: each element of the outbox matrix $o_{(i,j)} \in \mathcal{O}$ is an interaction object that models the actuators and sensors of the agent i in the interaction space j. In other words, the elements of the outbox matrix model the means that enable an agent to perceive and influence the universe in a particular interaction space. So, having an interaction object in the outbox matrix is the only way for an agent to exist and operate in the MAS. Particularly, when $o_{(i,j)} = 0$, the agent i neither influences nor perceives the universe in the interaction space j: agent i does not exist in j. Moreover, the means used to perceive the universe are distinguished from the result of the perceptions. The perception results are placed in the inbox matrix.

2. The inbox matrix: each element of the inbox matrix $o_{(i,j)} \in \mathcal{O}$ represents the result of the perceptions of the agent i in the interaction space j.

MIC* as an Environment

– the ontology of the MIC* environment defines the following concepts: interaction object, interaction space, inbox matrix, outbox matrix, and the operators of movement, interaction and computation.

– the actions means offered by MIC* are represented by the interaction objects, interaction space, outbox matrix and the computation operator.

– the perception means offered by MIC* are represented by the interaction objects, interaction space, inbox matrix and the interaction operator.

– interaction functions: MIC* defines interaction operators within the scope of an interaction space to calculate the result of the interaction of an actuator on a sensor. Both the actuator and the sensor are uniformly represented as interaction objects. The actuator and the sensor are located in the outbox matrix and the result of the interaction is stored in the row corresponding to the sensor in the inbox matrix. Hence, an agent can retrieve its perception whenever he wants in order to deliberate and emit other interaction objects.

– localization function: MIC* is a situated environment since each agent owns an identifiable location. This location is defined by the rows which are occupied by the agent within the inbox and outbox matrices of MIC*.

4.2.3 Spatial Grid Environment

Spatial grid environments have always been considered as a very useful tool for the modeling of physical environments within MAS applications, see e.g. [13]. A spatial grid environment defines a two-dimensional world which is spatially discretized into *cells* (or *patches*) which define the agents' location and that contain local environmental properties such as a pheromone concentration for instance. MAS platforms such as STARLOGO [14], NETLOGO [16] or TURTLEKIT [10] are some examples which are explicitly based on this kind of environment. Within such environments, the agents have the ability to perceive and act on the environment according to the perception/action means which are afforded by the cell on which they are: the agents can move to another cell, perceive the pheromone concentration of the cell and drop some pheromones on the cell. Additionally, the environment owns some processes that define the pheromone propagation and evaporation phenomena. It is these processes that define the interaction functions since they enable the agents to coordinate their behaviours through the environment.

Spatial Grid as an Environment

- the ontology of a spatial grid environment defines the following concepts: cell, pheromone, propagation function, evaporation function and movement.
- the action means offered by a spatial grid environment are represented by the movement actions and the drop of pheromones.
- the perception means offered by a spatial grid environment are afforded by the cell that represents the current location of an agent.
- the interaction functions defined by a spatial grid environment maps the intensity of the pheromones to the locations of the agents. The spatial grid environment computes the intensity of the pheromones according to the evaporation and propagation functions.
- localization function: within the spatial grid environment, each agent is located by the coordinates of the current inhabited cell.

4.2.4 AGR Organizational Environment

In [4], J. Ferber and O. Gutknecht have proposed the Agent/Group/Role (AGR) model. The main organizational concepts of AGR are described as follows:

- agent: an agent represents the active entity that tries to achieve its design goals by interacting with other agents.
- group: the organization of the MAS is structured by groups. A group is defined as a set of agents that play specific roles. It is important to notice that the agents can interact by sending messages only when they belong to the same group.
- role: the role represents an abstraction of the function of an agent within a group.

The MadKit platform [9] was then developed as an infrastructure that implements the AGR concepts.

AGR as an Environment

- the ontology of the AGR environment defines the concepts of group, role and message.
- the actions means offered by the AGR environment are represented by the action of acquiring/leaving a role within a group and by the action of sending a message.
- the AGR environment considers the roles played by the agent as its perception means.
- the AGR environment defines the interaction functions as message routing and delivery. Hence, messages are delivered to the agents by using the social position of an agent, namely the role of an agent within a group.
- localization function: within the AGR environment, each agent is located by the roles played within groups.

5 Experiment

This section presents an experiment of a MAS that has been entirely built using the multi-environments approach. This MAS is concerned with a social simulation derived from the *SugarScape* system [2]. The simulated agents are located in a spatial 2D grid that contains some resources. The resources are consumed by the agents and regenerated after a period of time. The agents can either move between the cells of the grid or consume the available resources to augment their energy. When the energy of the agents reaches zero these agents die.

Obviously, the simulation of such a system requires the intervention of other meta-agents that manage the simulation process and dynamically collect the information from the simulated system at the runtime. Even if these agents are considered at the meta-level for the *SugarScape* system, they have to be considered as being part of the MAS since they can change the dynamics of the whole system. Among the meta-agents that are required for the simulation, we have identified the following:

- the *scheduler* implements the dynamics of the simulation process. For that, this entity delivers events to the agents to steer the simulation process.
- the *observer* collects some information from the spatial grid environment and displays it to the end user.

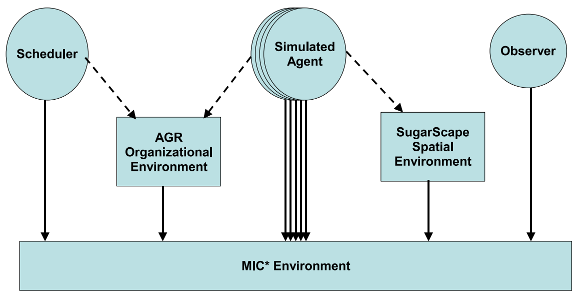

Scheduler
Simulated Agent
Observer
AGR Organizational Environment
SugarScape Spatial Environment
MIC* Environment

Figure 8: Agents and their environments in the application.

As presented in figure 8, several environments have been identified in order to capture the different aspects of the system. These environments are: *(i)* the AGR organizational environment,*(ii)* the MIC* environment, *(iii)* and the SugarScape grid environment.

The AGR organizational environment has been presented in section 4.2.4. This environment holds the simulated agents and the scheduler. In fact, a special group named *sugarscape* is created: the scheduler plays the role of *scheduler* within this group and the agents play the role of *simulation_agent*.

The MIC* environment has been presented in section 4.2.2. This environment is used as an infrastructure and holds all the agents and other environments. It is important to notice that a specific interaction space is associated to each aspect of the application domain. For instance, a first interaction space is created to allow the observer to communicate with the *SugarScape* spatial environment, a second interaction space is created between the simulated agents and the AGR organizational environment and a third interaction space is created between the scheduler and the AGR organizational environment.

Finally, the characterization of the *SugarScape* grid environment is given as follows:

– the ontology of this environment defines the following concepts: location, cell, resource, consumption of resource, and movement.

– the action means offered by this environment are represented by the movement actions and the consumption of resources.

– the perception means offered by this environment are represented by the location of the agents. In fact, according to their location the agents can perceive the available resources within their vicinity.

– the interaction functions defined by this environment maps the resources to the locations of the agents. The *SugarScape* grid environment calculates for each agent what resources are available within its vicinity.

– localization function: each agent is located within this environment by the cell that it occupies.

This system has been implemented and the outputs of the observer agent are presented in figure 9. The process of building such a MAS that merges several aspects such as: the management of the dynamics of the simulated system, the management of the simulation process and the visualization and interpretation of simulation outputs, is made clearer at both the design and implementation levels. This is due to the separation of concerns and the modularity brought by the multi-environments approach. Moreover, each environment is concerned only with a specific aspect and can be developed independently from other environments. To include an additional environment that models another aspect of the application, one has only to describe how this environment implements the agent-environment relationship and to define the set of the deployed agents. Existing environments have not to be redefined or modified.

6 Conclusion

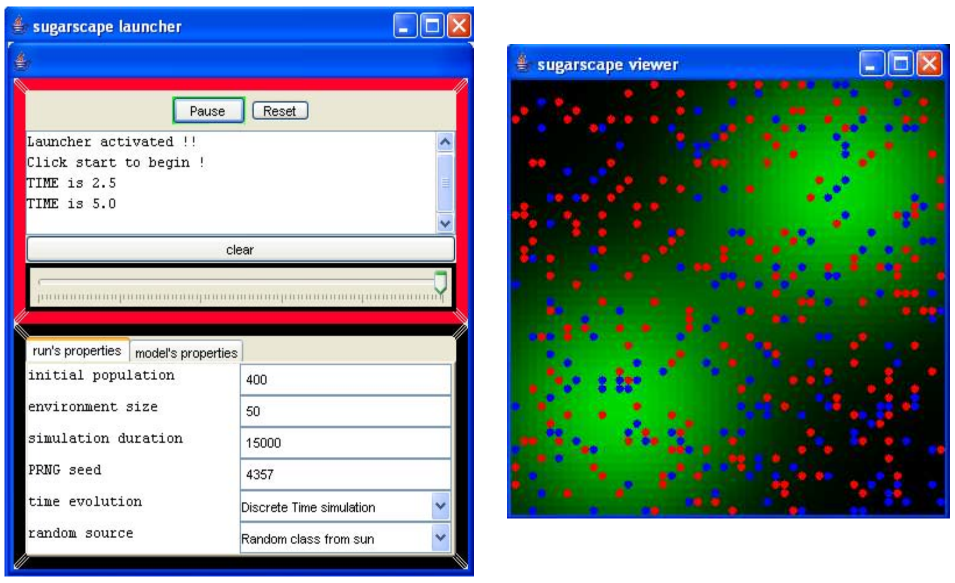
In this paper we have challenged an implicit hypothesis of MASs stating that the agents exist in a single, common and shared environment.

In fact, an agent can be associated with several environments. Each environment captures a specific aspect of the application domain. To reach this point, we have characterized the agent-environment relationship by the following: *(i)* ontology of the environment, *(ii)* perception means, *(iii)* action means and *(iv)* interaction functions. Besides, situated environments define another feature which is the localization function.

Once, the agent-environment relationship has been characterized, it becomes conceivable to allow its multiple instantiation. So, the agents can exist in several and independent environments. We have also seen that the agent-environment relationship has to be distinguished from the means which are used by an agent to access its environments. In fact, an agent can exist in an environment A and use an environment B as a communication medium to access A. This is typically the schema that is generally used to access an application level environment using an infrastructure environment. Still, all these types of environments are captured uniformly using the proposed characterization.

From an engineering point of view, the multi-environments approach brings the necessary modularity and separation of concerns to build MASs that address multi-aspects problems and domains. This has been shown for instance by the *SugarScape* simulation where several aspects of the application have to be considered. If the design models and the implementation of such a system do not explicitly reflect *all* these aspects then some parts of the system would have been implemented in an *ad hoc* manner. Consequently, it would be impossible to capture the dynamics of the whole MAS. For instance, if the infrastructure was ignored in the presented system, then some behaviors of the global MAS would be neither explained nor understood. For instance, the infrastructure, as an environment, acts actively and influences the dynamics of the entire MAS. As illustrated by the example, the multi-environment approach to build MASs can bring an appreciable flexibility within MASs to address complex domains of applications that are not reducible to only one aspect.

As highlighted by D. Weyns and T. Holvoet in this volume [17], the engineering of environments for MASs is still in its infancy and further investigations have to be done considering the environment as a first order abstraction. Concerning the multi-environments approach proposed in this paper, an important research track will be to establish a classification of agent-environment relationships with respect to the five points which have been identified, enabling reusability of agent-environment relationship classes both at the conceptual level and implementation level. For instance, environments that exploit pheromone infrastructures may be considered as a particular instance of the physical environment class. So, one of the primary objec-



```
Launcher activated !!
Click start to begin !
TIME is 2.5
TIME is 5.0
```

initial population	400
environment size	50
simulation duration	15000
PRNG seed	4357
time evolution	Discrete Time simulation
random source	Random class from sun

Figure 9: The outputs of the observer during the simulation process.

tives of our future work is to establish a hierarchy of classes for each part of the agent-environment relationship. Such a taxonomy is necessary developing a clearer understanding of what is really beyond the notion of "environments for MASs".

References

[1] S. Bandini, S. Manzoni, and G. Vizzari. A Spatially Dependant Communication Model for Ubiquitous Systems. In Weyns et al. [18], pages 74–90.

[2] J. M. Epstein and R. L. Axtell. *Growing Artificial Societies*. Brookings Institution Press, Washington D.C., 1996.

[3] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[4] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Y. Demazeau, editor, *Proceedings of the 1998 International Conference on Multi-Agent Systems ICMAS98, Cité des Sciences - La Villette, Paris, France, July 4-7*, pages 128–135. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[5] J. Ferber, F. Michel, and J.-A. Báez-Barranco. AGRE: Integrating Environments with Organizations. In Weyns et al. [18], pages 48–56.

[6] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[7] A. Gouaïch. *Movement, Interaction, Calculation as Primitives for Everywhere and Anytime Computing*. PhD thesis, Université Montpellier II, Montpellier, France, 2005.

[8] A. Gouaïch, F. Michel, and Y. Guiraud. MIC*: A deployment environment for autonomous agents. In Weyns et al. [18], pages 109–126.

[9] O. Gutknecht, J. Ferber, and F. Michel. Integrating tools and infrastructures for generic multi-agent systems. In E. André, S. Sen, C. Frasson, and J. P. Müller, editors, *Proceedings of the fifth international conference on Autonomous agents, AA 2001, Montreal, Quebec, Canada, May 28-June 1*, pages 441–448. ACM Press, New York, NY, USA, 2001.

[10] F. Michel. An Introduction to TurtleKit : a Platform for Building Logo Based Multi-Agent Simulations with MadKit. Technical Report RR LIRMM 002215, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, LIRMM, CNRS, Montpellier, 2002.

[11] J. Odell, H. V. D. Parunak, M. Fleischer, and S. Brueckner. Modeling agents and their environment. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002. Revised Papers and Invited Contributions*, volume 2585 of *Lecture Notes in Computer Science LNCS*, pages 16–31. Springer, Berlin, 2003.

[12] A. Omicini and F. Zambonelli. TuCSoN: a coordination model for mobile information agents. In D. G. Schwartz, M. Divitini, and T. Brasethvik, editors, *1st*

International Workshop on Innovative Internet Information Systems (IIIS'98), Pisa, Italy, June 8–9, pages 177–187. IDI – NTNU, Trondheim (Norway), 1998.

[13] M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In T. Dietterich and W. Swartout, editors, *Proceedings of the Eighth National Conference on Artificial Intelligence, Boston, MA, July 29–August 3, 1990*, pages 183–189. AAAI Press, Menlo Park, CA, 1990.

[14] M. Resnick. *Turtles, termites, and traffic jams: explorations in massively parallel microworlds*. MIT Press, Cambridge, MA, 1994.

[15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[16] The NetLogo system. http://ccl.northwestern.edu/netlogo/ (date of last access: 2005/08/25).

[17] D. Weyns and T. Holvoet. On the Role of Environments in Multiagent Systems. *In this volume*.

[18] D. Weyns, H. V. D. Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, July 19, 2004, New York, NY, USA, Revised Selected Papers*, volume 3374 of *Lecture Notes in Computer Science LNCS*. Springer, Berlin, 2005.

[19] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for Multiagent Systems: State-of-the-Art and Research Challenges. In Weyns et al. [18], pages 1–47.

[20] M. Wooldridge and P. Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000. Revised Papers*, volume 1957 of *Lecture Notes in Computer Science LNCS*, pages 1–28. Springer, Berlin, 2001.

[21] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Coordination Artifacts: A Unifying Abstraction for Engineering Environment-Mediated Coordination in MAS

Alessandro Ricci and Mirko Viroli
DEIS, Alma Mater Studiorum–Università di Bologna
47023 Cesena (FC), Italy
E-mail: {a.ricci,mirko.viroli}@unibo.it

Keywords: Multi-Agent System Coordination, Multi-Agent System Environment, Multi-Agent System Engineering, Coordination Artifacts

Received: May 10, 2005

Similarly to human organizations, where the environment plays a fundamental role in supporting social activities, the environment of a multi-agent system (MAS) is the natural place where understanding and designing agent coordination. Accordingly, we propose the notion of coordination artifact as a unifying abstraction for engineering environment-based coordination of agents. This is meant to capture at the MAS level abstractions and concepts like services, tools, and artifacts, *which are typically shared and exploited by the collectivity of individuals for achieving individual as well as global objectives. In this work we describe this framework, by defining a model for the coordination artifact abstraction, and discussing the infrastructures and technologies currently available for engineering MAS applications with coordination artifacts.*

Povzetek: Zajema enotno abstrakcijo inženirskega okolja v MAS z namenom koordinacije.

1 Introduction

Direct interaction and explicit communication are not always the best approaches to achieve coherent systemic behaviour in the context of MAS and agent societies. This is quite evident when taking into account the main approaches dealing with environment-based coordination such as stigmergy and, more generally, mediated interaction frameworks and infrastructures based on forms of coordination / cooperation without direct communication (see [43] for a recent survey).

Mediated interaction and environment-based coordination are highly debated also in other research fields outside MAS and CS, where collaborative and cooperative activities are studied in complex social contexts: notable examples are CSCW (Computer Supported Cooperative Work) and HCI Ê(Human Computer Interaction) [36], recently focussing on cognitive and social theories which explicitly take into account the role of environment in coordination, such as Distributed Cognition [15] and Activity Theory [19]. There, a relevant issue is to understand what makes an environment a good place for actors to work together: Ê

How to design the agent environment to suitably support the social activities of a possibly open agent society?

This question can be considered of primary importance also in MAS, and it involves issues that are not fully considered by current approaches dealing with coordination through the environment. In particular:

"Not only ants" | Approaches dealing with environment-based coordination typically consider *reactive* agents, either embedding all the intelligence into the environment or obtaining it as emergent phenomenon (well known examples are stigmergy coordination and swarm intelligence [28, 38, 3]). Here instead we are interested on the one side devising an environmental support that can be useful to amplify the intelligence of individual agents, possibly exploiting their cognitive capabilities. On the other side, we are interested in considering intelligence not only as an emergent phenomenon, but promoting the engineering of intelligence by designing and building suitable environmental abstractions.

"Not only special-purpose coordination" | Existing environment-based approaches to coordination — such as stigmergy — typically provide solutions only to specific coordination problems, without the abstraction required to use and systematise coordination in the wide range of social activities. Here instead we are interested in conceiving general purpose environment abstractions that could be suitably specialised and dynamically configured / tuned for addressing specific and heterogeneous coordination activities.

"Toward engineering" | Frequently, investigations in literature only concern simulation and abstract models (a notable exception can be found in [13], where a model for situated MAS is provided for the engineering of systems). Here we are interested instead in methodologies and infrastructures, i.e. in identifying models, languages, architectures and middleware

technologies to be exploited at the design stage in agent oriented software engineering, as well as for development and online management of MAS.

In this paper we describe the conceptual and engineering framework based on the notion of *coordination artifact*, which aims at addressing the above issues. The framework provides a systematic view of environment-based coordination for general coordination problems, and extends the scope of applicability to heterogeneous, *cognitive / intelligent* agents. Coordination artifacts are runtime abstractions encapsulating and providing coordination services, to be exploited by agents within a given social context. They can be exploited then as basic building blocks for designing and developing suitable working environments for heterogeneous multi-agent systems, supporting their coordination for collaboration or competition. Accordingly, coordination artifacts can be considered a kind of *first-order environmental abstractions or modules* as defined in the paper [42], part of this special issue.

We here gather the main results of our previous investigations [43, 24, 40], and provide a self-contained description of the role of coordination artifacts in the engineering of MAS environments. In particular, the remainder of the paper is organised as follows. Sect. 2 recalls the conceptual framework inspired by Activity Theory, as a background for the approach described in the paper, focussing on the importance of the environment in supporting social activities. Sect. 3 presents in detail the coordination artifact abstraction, along with its main properties and Sect. 4 remarks the impact of the framework on MAS engineering. Then, Sect. 5 discusses the framework as a unifying tool for understanding environment-based approaches in general, and in particular focuses on TuCSoN as a model / infrastructure / technology supporting the main features of the coordination artifact approach. Finally, related works are discussed in Sect. 6 and conclusions in Sect. 7.

2 Environment and Activity Theory

The environment support for both the analysis and the development of activities in complex systems — such as human society — is among the main issues studied by socio-psychological approaches such as Activity Theory (AT) and Distributed Cognition.

Activity Theory, defined also Cultural-Historical Activity Theory, is a social psychological theory initiated with the work of Lev Vygotsky (1926–62) in the context of Soviet Psychology (SP) [41]. From its origins, AT was furthered in the Soviet Union by Vygotsky's students — Alexey Leontiev in particular — in the first half of the 20th century. It then spread also outside the Soviet Union, first to Scandinavia and Germany and finally — at the end of the 1990s — to the United States. Nowadays it has been applied also in the context of computer science related fields, such as Computer Supported Cooperative Work (CSCW) and Human Computer Interaction (HCI) (see [19] for a survey).

AT is a very general framework for conceptualising human activities — how people learn and society evolves — based on the concept of human *activity* as the fundamental unit of analysis. The approach was developed in contrast to purely cognitive approaches which were dominating the first years of the 20th century: according to them, human individual and social activities could be analysed and understood focussing only on the internal (mentalistic) representation of the individuals, in other words on the individual information-processing capabilities. On the contrary, the basic inspiration principle of AT is the *principle of unity and inseparability of consciousness (human mind) and activity*: human mind comes to exist, develops, and can only be understood within the context of a meaningful, goal-oriented, and socially determined interaction between human beings and their material environment. From the beginning, a fundamental aspect for AT was the *interaction* between the individuals and the *environment* where they live, in other words their *context*. After an initial focus on the activity of the individuals, the AT research has evolved toward the study of human collective work and social activities, then facing issues such as the coordination and organisation of activities in human society.

Here the investigation of AT is of particular relevance because it remarks the fundamental role of the environment in the development of complex systems. According to AT any activity carried on by one or more components of a systems — individually or cooperatively — cannot be conceived or understood without considering the tools or *artifacts* mediating the actions and interactions of the components. Artifacts on the one side mediate the interaction between individual components and their environment (including the other components), on the other side embody the part of the environment that can be designed and controlled to support components' activities. Moreover, as an observable part of the environment, artifacts can be monitored along the development of the activities to evaluate overall system performance and keep track of system history. In other words, mediating artifacts become first-class entities for both the analysis and synthesis of individual as well as cooperative working activities inside complex systems.

The complexity of the activities of the social systems focussed by AT can be found nowadays in MAS and agent societies. With analogous consideration, we consider it fundamental to frame the role of the environment for the analysis and synthesis of social activities inside MAS, and in particular of the artifacts mediating such activities. In this work we describe the framework of *coordination artifacts* as an approach to systematise this vision and make it effective for the engineering of systems as MASs — from design to development and runtime, including their dynamic observation and management.

3 The Coordination Artifact Abstraction

Coordination artifacts can be conceived as persistent entities specialised in providing a coordination service in a MAS [33, 24]. The term coordination should be here understood in its most general sense, as the management of dependencies among separate activities [16], shaping and constraining the (agent) interaction space [5]. Coordination artifacts are *infrastructural* abstractions meant to improve coordination activities automation; they can be considered then as basic building blocks for creating effective shared collaborative working environments, alleviating the coordination burden for the involved agents. Human society is full of entities like coordination artifacts, engineered by humans in order to support and automate coordination activities: examples range from blackboards, maps, schedulers and paper trays, to traffic lights, clocks, and so on. These and other kinds of computerised artifacts are currently under investigation in the context of CSCW and cognitive sciences, where their importance in supporting human individual and cooperative activities is being recognised [37, 32].

Basically, a coordination artifact *(i)* entails a form of mediation among the agents using it, and *(ii)* effectively embeds and enact some coordination policy. Accordingly, two basic aims can be identified: *(i) constructive*, as an abstraction essential for creating and composing social activities, *(ii) normative*, as an abstraction essential for ruling social activities.

3.1 A First Model

Also taking inspiration from our society, a basic abstract model can be devised, where a coordination artifact features:

- a *usage interface*, defined in terms of a set of *operations*. Agents *use* coordination artifacts by executing operations provided by the artifact, and by eventually perceiving information about the operation completion. Notice that due to the nature of coordination artifacts and their interaction schema, agent actions executing operations are more similar to *practical acts* rather than *communicative acts* — which makes our approach sensibly different from direct, ACL-based interaction;
- a set of *operating instructions*. This information describes (formally) how to use the artifact in order to exploit its coordination service. For instance, operating instructions might specify the protocol of interactions to be used, and the mentalistic semantics of actions and perceptions [40];
- a *coordinating behaviour specification*. This information describes (formally) the coordinating behaviour of the artifact, in terms of coordination rules required for enacting the coordination service.

In particular, taking the agent viewpoint, to exploit a coordination artifact simply means to follow its operating instructions, on a step-by-step basis. It is worth noting that, since a considerable coordination burden can be charged upon the artifact and be hidden from the agents, operating instructions are generally quite simple when compared to the interactive behaviour required in the case of direct communication (protocols). Hence, our approach to interaction can be fruitfully leveraged by intelligent agents, which can exploit an artifact through its operating instructions so as to take part to complex coordination scenarios.

A simple but effective example of coordination artifact is a *task scheduler* in cooperative working environments, which can be found in concurrent systems as well as in human society. The coordination problem concerns ruling the order of execution of a dynamic set of tasks taken in charge by some agents, according to some scheduling policy. A coordination artifact can be designed to provide one such scheduling service. A possible usage interface would consist — for instance — in two basic operations[1]:

- *taskStart(-Token)*, to manifest agent intent to start executing the task. The completion of the operation means that the agent can start the task according to the scheduling policy of the artifact. A token is returned to the agent for identifying its activity;
- *taskCompleted(+Token)*, to signal the completion of the task.

Operating instructions simply consist in: first, invoking the *taskStart* operation to manifest the intention to start a task; then, invoking *taskCompleted* to signal the completion of the task. The coordinating behaviour of the artifact concerns the enactment of the scheduling policy, queueing requests and serving them according their position in the queue — for instance using a FIFO policy.

We conclude this section remarking both the philosophical / conceptual and engineering difference between agents and coordination artifacts. Agents are *goal-governed / goal-oriented* entities, and accordingly agent models / languages / architectures are suitable for defining pro-active and autonomous behaviour of agents. Coordination artifacts are *function-oriented* entities, i.e. entities designed to provide some kind of functionality or service. From this point of view, coordination artifacts are much more similar to *objects* as found in the object-oriented paradigm: differently from agents, they have a well-defined interface, providing operations that can be invoked by agents. On the contrary, agents do not provide interfaces with operations that can be invoked from external entities. More on this point can be found in [35, 39], where a generalisation of the notion of coordination artifact is introduced — the *artifact* abstraction —, representing any device populating agent working environments and that can provide functionalities other than coordination.

[1]The basic Prolog notation is adopted for describing argument of operations: + means an output argument, - an input argument, ? an input / output argument.

3.2 Basic Properties

Generally speaking, as devices exploited by agents to support their coordination activities, coordination artifacts have some basic properties which are indeed different from autonomy, pro-activeness, reactivity, and social ability which characterise instead the agent abstraction [44]:

Focus on interaction management | Coordination artifacts are specialised in automating coordination activities. For this purpose, they typically adopt a computational model suitable for effective and efficient interaction management, whose semantics can be easily expressed with concurrency frameworks such as process algebras [2] or Petri nets [31], see e.g. the work in [40].

Encapsulating coordination | Coordination artifacts encapsulate a coordination service, allowing user agents to abstract from how the service is implemented. As such, a coordination artifact is perceived as an individual entity, but actually it can be distributed on different nodes of the MAS infrastructure, depending on its specific model and implementation. Encapsulation is the key to achieve reuse of coordination. Agent society engineers can create and exploit handbooks or catalogs of coordination artifacts, embodying the solutions to general coordination problems in organisations, analogously to an handbook of handbook of organisation/coordination processes [17]. Also, a coordination artifact provides a certain *quality of coordination*, in particular in terms of the scalability with respect to the dimensions identified by Durfee in [11], which are related to performance, robustness, reliability, and so on. The description of such dimensions is important to identify the range of applicability of the artifact in the engineering of agent societies.

Malleability | Coordination artifacts are meant to support coordination in open agent systems, characterised by unpredictable events and dynamism. For this purpsose, their coordinating behaviour can be adapted and changed dynamically, either *(i)* by engineers (humans) willing to sustain the MAS behaviour, or *(ii)* by agents responsible for managing the coordination artifact, with the goal of flexibly facing possible coordination breakdowns or improving the coordination service provided.

Inspectability and controllability | A coordination artifact typically supports different levels of inspectability: *(i)* inspectability of its operating instructions and coordinating behaviour specification, in order to let user agents to be aware of how to use it or what coordination service it provides; *(ii)* inspectability of its dynamic state and coordinating behaviour, in order to support testing and diagnosing (debugging) stages for the engineers and agents responsible of its management. Controllability is also fundamental for runtime management of a coordination artifact, by making it possible to freeze its behaviour, to trace it, supporting step-by-step execution while watching its state, to restart it, and so on. So, from an operational point of view, a coordination artifact can be understood as a sort of *virtual machine of coordination*, executing some form of coordination specification, fully inspectable and controllable by coordination artifact administrators [21].

Linkability | This term is borrowed from coordinative artifacts studied in the context of CSCW [36]. It refers to the capability of linking artifacts together, in order to support a dynamic form of composition useful to scale with coordination activity complexity. This property is fundamental for supporting also the scenarios depicted in the paper [14] in this special issue, where multiple environments are considered and an environment can act as a medium for agents to interact with an other environment. Analogously, a coordination artifact can be used by an agent as a (coordination) medium to interact with other artifacts, which are linked to the first one.

Spatial extension | Differently from agents, coordination artifacts can have a spatial / topological extension, meaning that — in a MAS model with some topological structure — the same artifact can be present (simultaneously) in different nodes of the topology. In other words, while typically in a MAS a single agent cannot be distributed (it is located on a specific node of the MAS), on the contrary a single artifact can be distributed among different nodes of a MAS.

Summing up, coordination artifacts are conceived to be engineering abstractions used for designing, building and supporting at runtime coordination in agent societies, suitably instrumenting their dynamic working environment. Also, they can be useful to support forms of scientific investigation of collective behaviours. As mediating entities, coordination artifacts typically reify and manage agent communication events; accordingly, they can be used to trace and log the overall interaction behaviour of the agent societies exploiting them. Thus, they can act as kinds of *social memory*, which can then be inspected for possible scientific analysis about global behaviours.

4 Engineering Social Activities

The introduction of coordination artifacts impacts the methodology adopted for engineering social activities in agent societies. Taking inspiration from Activity Theory, we can identify three different stages characterising any social activities supported by coordination artifacts (see Fig. 1):

Co-construction | In this stage, engineers and scientists understand and reason about the social objectives of the society, and define a model of the social tasks

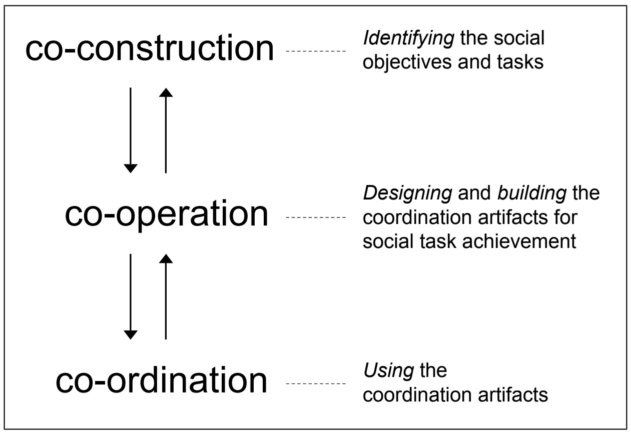

co-construction
Identifying the social objectives and tasks
co-operation
Designing and *building* the coordination artifacts for social task achievement
co-ordination
Using the coordination artifacts

Figure 1: Levels of a social activities

required to achieve them. This implies understanding the shape of the agent interaction space, by possibly identifying also the dependencies that need to be managed (dependency detection is a fundamental aspect of coordination, according to the theory of coordination [16] and to cognitive theories of agent societies [6]).

Co-operation | In this stage, society engineers — and possibly intelligent agents — design and build the coordination artifacts according to the objective identified in the previous stage (co-construction). This implies understanding how to manage the dependencies previously identified, and defining a coordinating behaviour useful for that purpose. A model of coordination artifact must be chosen, according to its ability of embedding and enacting the required coordinating behaviour.

Co-ordination | In this stage, coordination artifacts are exploited, supporting the execution of the social activity. Here, the focus is on the efficient execution and automation of the coordination activities.

As in the case of AT, the three levels are distinct analytical moments that can be applied continuously, since a social activity is considered to be always under development, given the intrinsic openness of the environment and the dynamism of organisations.

4.1 Activity Levels as Engineering Stages

Activity Theory is primarily used as an analytical tool for understanding collaborative work in complex organisational contexts, and as a design tool to improve them. In such contexts, AT makes it possible to face the social complexity first by separating individual and collective activities, then by identifying and designing the artifacts required to support both of them.

Along this line, we can devise a correspondence between the three collaborative stages in Fig. 1, and the engineering stages as typically found in (agent-oriented) software engineering methodologies, i.e., analysis, design, development and deployment / runtime. Generally speaking, individual and social tasks are identified and described in the analysis and design stages of such methodologies. Each individual task is typically associated with one specific competence of the system. Each agent in the system is assigned to one or more individual tasks, and assumes full responsibility for their correct and timely completion. From an organisational perspective, this corresponds to assigning each agent a specific role in the organisation. Conversely, social tasks represent the global responsibilities of the agent system. In order to carry out such tasks, several possibly heterogeneous competences usually need to be combined. The design of social tasks leads to the identification of global *social laws* that have to be respected / enforced by the society of agents, to enable the society itself to function properly and in accordance with the expected global behaviour.

Given this picture, it is possible to identify a correspondence between the analysis stage (where individual and social tasks are identified) and the co-construction level, where the social objectives of the activities are shaped. Then, the identification of the social laws required to achieve the social tasks can be seen as a first step in the co-operation level. This level roughly corresponds to the design and development stages of the engineering process: coordination artifacts are the abstractions which make it possible to design and develop social tasks. At the co-operation level such artifacts are designed and developed to embody and enact — as governing abstractions provided by the infrastructure — the social laws and norms previously identified. Finally, the deployment and runtime stages correspond to the co-ordination level, when the coordination artifacts are instantiated and exploited.

The dynamism among the levels, that are compared here to the engineering stages of a system, promote then a new approach in the engineering of MASs that we can call here *online engineering*: coordination artifacts can be re-designed, manipulated, tested, debugged, analysed dynamically, at runtime. In order to support the online engineering methodology two aspects are essential: first, working with abstractions featuring suitable properties such as inspectability, controllability and malleability, which are necessary for their online analysis and synthesis; second, designing and building infrastructures that support services enabling, access and exploitation (co-ordination stage), and tools for their inspection, control, adaptation (co-operation stage) — see Sect. 4.3.

4.2 The Organisation Perspective: Structuring the Working Environment

Coordination artifacts can be suitably used in a structured and ruled organisation. Coordination artifacts become the entities around which the social activities are built, inducing a natural form of organisation structuring and modelling. By abstracting from details, several independent collaborative and cooperative activities are carried over inside an organisation, each one charged upon a group of agents and a suitable coordination artifact. The same coordination artifact can be used in different ways according to the *roles* of the agents: moreover, operating instructions can be in principle partitioned according to the role of the agent using the artifact.

Following the organisation perspective, coordination artifacts are the key to shape agent working environment, as *(i)* tools for pure coordination, and *(ii)* interfaces mediating and coordinating agent access to the resources and the services provided by the environment itself. As mediating interfaces, coordination artifacts can encapsulate the policies for resource management, involving the coordination of both the users and the resources or the providers of the services.

The two issues above point out the fundamental role of artifacts in the design and construction of an effective working environment, supporting agent activity toward the achievement of their individual and social tasks. This is particularly relevant in the context of cognitive theories applied for CSCW, such as Distributed Cognition [15]. In the design and construction of a good working environment for the organisation, the tension between subjective and objective approaches emerges again in terms of the dichotomy between *flexibility* — the capability of individuals to adapt to contingent situations — and *automation* — the capability of making the execution of activities fluid. On the one side, given the complexity and the openness of agent organisations, a working environment keeps evolving and requires flexibility in order to allow for supporting changes and adaptations. The lack of flexibility dramatically impacts on all system activities. On the other side, a good working environment should assist workers as much as possible in their coordination, providing services to alleviate their coordination burden and let them focus on their individual work. The lack of system coordination typically makes organisations unable to govern the complexity of the activities: the final result is typically a weak control of activities, and poor performances in their execution.

4.3 Toward Infrastructures for Coordination Artifacts

Coordination artifact infrastructures (or middlewares) provide services for their access and use, effectively supporting the co-operation and co-ordination levels, and the reflection / reification transitions. Services range from artifacts creation and discovery to inspection and dynamic adaptation of their state and coordinating behaviour. Referring to the 3-Layer Model defined in [42], a coordination artifact infrastructure is part of the Execution Platform layer — at the Middleware level — while coordination artifacts themselves are specialised and used at the MAS Application Layer, programmed according to the application specific logic.

In the overall, coordination artifacts can be seen then as a fundamental abstraction for *governing* infrastructures [22], i.e. infrastructures providing flexible and robust abstractions to model and shape the agent interaction space, according to the social and normative objectives of systems.

Infrastructures also represent an effective approach to the general problem of formalisability of complex systems, which may come either for pragmatical or theoretical issues. By their very nature, infrastructures intrinsically encapsulate key portions of systems — often in charge of the critical system behaviour. In this case, governing infrastructures encapsulate agent interaction and coordination through coordination artifacts. As a result, providing well-specified infrastructures, and in particular formally-defined coordination artifacts promotes the discovery and proof of critical system properties. Most notably, a system property can be assessed at design-time through the formal definition of some design abstraction. Then, by ensuring compliance of the corresponding run-time abstraction provided by the infrastructure, such a property can be enforced at execution time and be automatically verified for any system based on the infrastructure.

5 A Unifying Abstraction for MAS Environment-based Coordination

The notion of coordination artifact can be considered as a unifying abstraction from different point of views. On the one side, one of the main roles of coordination artifacts is as engineering tools for directly designing and developing building blocks, specialised to provide coordination functionalities (the glue) — general-purpose enough to be suitably programmed and configured according to the specific coordination problems to be solved. On the other side, as the agent abstraction is meant to unify all the specific approaches dealing with autonomous, pro-active and goal-governed / oriented behaviour, the coordination artifact abstraction can be used to represent any first-class device supporting interaction through agents. Accordingly, any device could be described in terms of a coordination artifact with a specific usage interface, a coordinating behaviour and possibly some operating instructions.

Among the main example, the pheromone infrastructure in stigmergy-based coordination approaches ([29] for instance) can be described as a coordination artifact, providing as a usage interaface operations for deposit and sensing pheromones, and with a coordinating behaviour defined by the diffusion / aggregation / evaporation law of pheromones. The notion of coordination artifact can be used as a theoretical foundation to these approaches, identifying and generalising environmental entities which are not described as agents.

Another example is given by e-Institutions [12], that are middlewares where agent interaction is governed and ruled by norms imposed by the *Institution*, as an entity external to the agents. The *institution* can be modelled as a coordination artifact, with the coordinating behaviour specified by the norms ruling agent communication.

Coordination media introduced in the context of coordi-

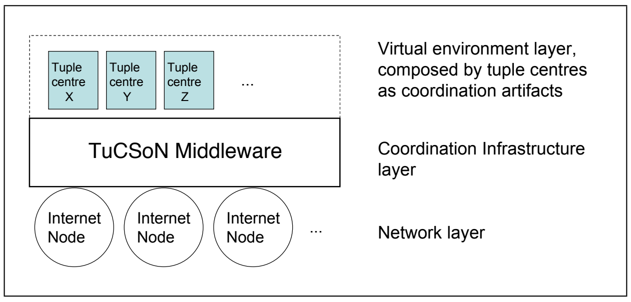


Figure 2: A logical view of TuCSoN infrastructure. The view is analogous to the 3-Layer model defined in [42]

nation models and languages (examples are tuple spaces, channels, etc. see [27] for a survey) can be described at the agent level as coordination artifacts: coordination primitives define the usage interface and the coordinating behaviour is defined by the coordination law defining the semantics of coordination media.

It is worth noting that these examples do not provide any explicit idea of operating instructions, which is instead a main property of coordination artifacts and fundamental for supporting intelligent agent coordination in open environment. This reveals an intrinsic inadequacy of existing approaches in filling the gap between agent rationality and environment-based coordination — see [40].

5.1 A concrete example: TuCSoN

As a concrete example of a model / infrastructure bringing some of the main principles that characterise the coordination artifact framework, here we consider the TuCSoN coordination infrastructure for MASs [26] [2].

The infrastructure enables agent interaction and coordination by means of *tuple centres*, which can be considered as a kind of coordination artifact. Technically, tuple centres are *programmable tuple spaces* — reactive, logic-based blackboards that agents associatively access by writing, reading, and consuming *tuples* (ordered collections of heterogeneous information chunks represented as first-order logic terms) via simple communication operations (*out*, *rd*, *in*, *inp*, *rdp*) [21]. While the behaviour of a tuple space in response to communication events is fixed, the behaviour of a tuple centre can be tailored to the application needs by defining a set of *specification tuples* expressed in the ReSpecT language, which define how a tuple centre should react to incoming / outgoing communication events. So, unlike tuple spaces, tuple centres can be programmed with reactions so as to encapsulate coordination laws directly in the coordination media. From the topology point of view, tuple centres are collected in infrastructure nodes, distributed over the network, organised into articulated domains (see Fig. 2 for a logical view).

So, tuple centres can be conceived as general-purpose coordination artifacts, which can be customised (programmed, tuned) dynamically to entail a specific coordinating behaviour. Generally speaking, tuple centres exhibit the properties that characterise coordination artifacts. First, they provide different levels of inspectability, since both the communication and the coordination state can be inspected at runtime. Second, different levels of malleability and controllability can be provided — both by allowing to dynamically change the artifact coordinating behaviour, and to control its execution by means of proper infrastructure tools [10]. The linkability property is supported by a primitive (*out_tc*) of the ReSpecT language, which makes it possible to directly insert a tuple from a tuple centre to another [34]. Also, we can identify the basic elements that characterise the abstract model of coordination artifacts: the usage interface is composed by the basic coordination primitives plus the primitives to inspect and change tuple centre behaviour (*set_spec* and *get_spec*). The coordinating behaviour specification is given by the ReSpecT specification. The notion of operating instructions is not directly supported in tuple centres, even if the ReSpecT specification tuples implicitly contain a description of how to exploit the tuple centre in order to obtain the coordinating service.

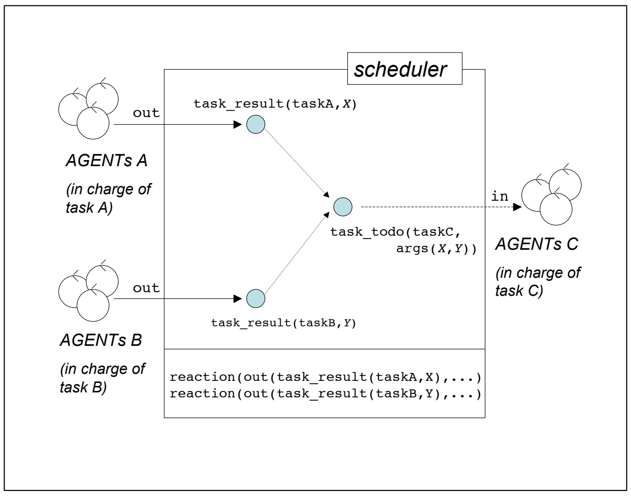
5.2 Coordination Artifacts in TuCSoN

Coordination artifacts can be considered as units of reuse for engineering cooperative working environments: as agents encapsulate skills and competences concerning the execution of some task, the achievement of some goal or the solution of some problem, coordination artifacts encapsulate strategies for constructing and ruling coordination activities. Tuple centre can be then suitably programmed to realise coordination artifacts with different coordinating purposes, such as flexible communication, knowledge mediation, resource sharing, and so on.

As a specific and representative example, here we consider workflow management, which is characterised by different kinds of coordination issues. Distributed workflow management concerns the automated integration and coordination of heterogeneous and independent distributed activities involved in the same global business process. Among the others, it includes activity scheduling and synchronisation, information and control flow management, exception management, and so on. In the context of service-oriented architectures — in particular Web Services — the workflow management idea is applied to the so-called *orchestration* [30].

Typically, special purpose languages — examples are XPDL and BPEL — can be used to define the workflow specification; their specification is executed by the *workflow engine*, the core component of a Workflow Management System. A workflow engine — also called orchestration engine — can be framed here as a general purpose coordination artifact, which is dynamically programmed to enact a coordinating behaviour according to the workflow specification.

[2]The TuCSoN technology is available as an open source project at the TuCSoN web site `http://tucson.sourceforge.net`



```
reaction(out(task_result(taskA,X)),(
  in_r(task_result(taskA,X)),in_r(task_result(taskB,Y)),
  out_r(task_todo(taskC,args(X,Y))) )).

reaction(out(task_result(taskB,Y)),(
  in_r(task_result(taskB,Y)),in_r(task_result(taskA,X)),
  out_r(task_todo(taskC,args(X,Y))) )).
```

Figure 3: Scheduler tuple centre *(top)* with its coordinating behaviour expressed in ReSpecT *(bottom)*

In the context of MASs, a tuple centre then can be programmed to provide the services from a simple task scheduler up to a full-fledged general purpose workflow engine. As an example, here we consider the realisation of a simple scheduler of three activities — A, B and C — coordinated according to a join pattern: task C can only start when both tasks A and B have been completed. Tasks are executed by independent agents, typically unaware of the global workflow and focussed on the achievement of their specific job. The tuple centre `scheduler` shown in Fig. 3 is an example of a coordination artifact providing such a scheduling service. The operation of the usage interface can be:

- `in(task_todo(`*`+TaskName,-TaskInfo`*`))`, for taking in charge the execution of a task. The presence of a tuple `task_todo` manifests the fact that a specific task has to be done, according to current workflow.
- `out(task_result(`*`TaskName,TaskResult`*`))`, for communicating the result of the execution of a task, signaling its completion.

In the example, *`TaskName`* can be `taskA`, `taskB` or `taskC`. The operating instructions of this coordination artifact, to be followed by agents in charge of task execution, would consist first in getting information about the task, then in providing the result. Fig. 3 shows also the ReSpecT specification realising the scheduling behaviour: basically, a suitable `task_todo` tuple is automatically generated in the tuple set as soon as the results of the execution of both tasks A and B are available.

6 Related Work

The coordination artifact abstraction brings in MAS ideas and concepts that have played a central role in other (un)related fields. From concurrent and distributed systems, coordination artifacts can be considered the generalisation of traditional coordination abstractions, from low level ones such as semaphores, monitors, to high-level ones, such as tuple spaces and, more generally, coordination media as found in coordination models and languages [27].

In particular, the notion of coordination artifact is strictly related to the *programmable coordination medium* abstraction defined in [9], on which the tuple centre model is based. According to the frequently adopted meta-model described in [4], a coordination model can be described by identifying the *coordinables* — the entities participating to coordination activities —, and the *coordination media* — the entities enabling and managing agent communication according to some coordination laws defining the semantics of the coordination activities. Programmable coordination media extend the basic notion of coordination medium with the idea of programming the internal behaviour with some specific language, so as to flexibly specify the coordination rules according to the application needs. So, programmable coordination media share some properties which characterise coordination artifacts, such as encapsulation of coordination and malleability of the behaviour. Instead, differently from programmable coordination media and coordination media in general, coordination artifacts do not necessarily manage *communications* among agents, but — more generally — *interactions*, caused by the execution of operations provided by the usage interface. Also, the coordination artifact framework introduces some structural properties — such as operating instructions — which are new with respect to the classic coordination meta-model, and which are indeed important in the context of open agent societies.

Blackboards as defined in Distributed Artificial Intelligence context can be framed and modelled in MAS as coordination artifacts, toward the integration of the two different points of view (traditional multi-agent and blackboard systems) in designing collaborating-software engineering space [7].

Actually, coordination artifacts can be exploited as an analytical tool for describing existing approaches based on some form of mediated / environment-based interaction. For instance, the environment provided by the pheromone infrastructure in [29] supporting stigmergy coordination can be interpreted as a coordination artifact exploited by ants to coordinate with each other: as such, it provides operations for depositing and sensing pheromones, and the coordinating behaviour is given by the environmental laws ruling the diffusion, aggregation and evaporation of pheromones. Analogously, the *field* abstraction in the co-field approach [18] — a recent approach for engineering of swarm intelligent systems — can be seen as a coordination artifact, mediating mobile agents interaction and supporting their coordinated navigation inside some kind of space.

Also some coordination and organisation approaches developed in the context of intelligent / cognitive agents can be framed in terms of artifacts. A main example is is given by electronic institutions ([12] is an example), where agent societies live upon an infrastructure (middleware) which governs agent interaction according to the norms established for the specific organisation, representing both organisation and coordination rules. The institution then can be framed as a kind of shared coordination artifact, characterised by an interface with operations that agents use to communicate, and providing a normative function on the overall set of agents.

7 Conclusions

In the context of human activities and CSCW, Activity Theory and Distributed Cognition remark the importance of the environment — and in particular of the tools available in the environment — for governing the complexity of cooperative / social work, in particular for its analysis and construction. Analogously, the framework of coordination artifacts aims at providing an engineering key for instrumenting a MAS working environment with first-class abstractions which could help agents of a MAS to cooperate and coordinate. Such first-class abstractions are meant to be exploited in the various stages of the MAS engineering process: at the design stage, as modelling entities for designing social activities; at development and runtime stage, as runtime abstractions — supported by suitable infrastructures — to be used by agents to execute the social activities; and at runtime stage also for online engineering of systems, as inspectable, malleable abstractions which can be dynamically observed, controlled, adapted — by human as well as by intelligent agents — to support online debugging and evolution of the activities.

Recently, the coordination artifact concept has been generalised toward the notion of *artifact*, as first-class abstraction representing tools or objects (devices) that agents can either individually or collectively *use* to support their activities, and that can be designed to encapsulate and provide different kind of functionalities [35, 39, 23]: coordination artifacts can be framed then as artifacts designed to specifically provide coordination services. Artifacts are currently investigated as basic building blocks for programming MAS [35], engineering MAS environment [39], and — more generally — to re-frame the notion of intelligent agents as goal-oriented / driven users of artifacts [23]: as happen in the human case [20], artifacts can act not only as amplifiers of agent (human) capabilities, but as entities that can significantly change the nature of the tasks to be done, enhancing the overall performances.

In conclusion, the notion of (coordination) artifact and related conceptual / modelling / engineering frameworks seem to be one promising way to put the environment in-the-loop when modelling and engineering agent-based systems. Indeed, the work can be still considered in its infancy and many aspects need to be further explored and developed: from (formal) theories including artifacts in agent cognition and reasoning models, to models and languages for designing and developing artifacts, to full-fledged infrastructures supporting artifacts and related services at runtime (such as creation, discovery, management, etc.), possibly integrated with existing agent-based platforms. First investigations about the integration between artifacts and existing agent models / platforms can be found in [25] and in [35], which discuss the use of TuCSoN tuple centres in the context of JADE FIPA-compliant platform [1] and of 3APL agents [8], respectively.

References

[1] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: a FIPA 2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 216–217, New York, NY, USA, 2001. ACM Press.

[2] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, Amsterdam, The Netherlands, 2001.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[4] P. Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys*, 28(2):300–302, June 1996.

[5] P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent system engineering: The coordination viewpoint. In N. R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer-Verlag, 2000.

[6] R. Conte and C. Castelfranchi. *Cognitive and Social Action*. University College London, London, UK, 1995.

[7] D. D. Corkill. Collaborating software: Blackboard and multi-agent systems & the future. In *Proceedings of the International Lisp Conference*, New York, NY, USA, 2003.

[8] M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: an approach illustrated using the 3APL language. In J. S. Rosenschein, T. Sndholm, M. Wooldridge, and M. Yokoo, editors, *2nd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 97–104, New York, USA, 14–18 July 2003. ACM Press.

[9] E. Denti, A. Natali, and A. Omicini. Programmable coordination media. In D. Garlan and D. Le Mé-

tayer, editors, *Coordination Languages and Models – Proceedings of the 2nd International Conference (COORDINATION'97)*, volume 1282 of *LNCS*, pages 274–288, Berlin (D), 1–3 Sept. 1997. Springer-Verlag.

[10] E. Denti, A. Omicini, and A. Ricci. Coordination tools for MAS development and deployment. *Applied Artificial Intelligence*, 16(9/10):721–752, Oct./Dec. 2002.

[11] E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7), July 2001.

[12] M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. Ameli: An agent-based middleware for electronic institutions. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 236–243, New York, USA, 19–23 July 2004. ACM.

[13] J. Ferber and J.-P. Müller. Influences and reaction: a model of situated multiagent systems. In *2nd International Conference on Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, 1996.

[14] A. Gouaich and F. Michel. Towards a unified view of the environment(s) within multi-agent systems. In this issue.

[15] D. Kirsh. Distributed cognition, coordination and environment design. In *Proceedings of the European Conference on Cognitive Science (ECCS '99)*, pages 1–11, 1999.

[16] T. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[17] T. W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. S. Osborn, A. Bernstein, G. Herman, M. Klein, and E. O'Donnell. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45(3):425–443, 1999.

[18] M. Mamei, F. Zambonelli, and L. Leonardi. Cofields: Towards a unifying approach to the engineering of swarm intelligent systems. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 68–81. Springer-Verlag, Apr. 2003.

[19] B. A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.

[20] D. A. Norman. Cognitive artifacts. In J. Carroll, editor, *Designing Interaction: psychology at the human-computer interface*, pages 17–38. Cambridge University Press, New York, NY, USA, 1991.

[21] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[22] A. Omicini and S. Ossowski. Objective versus subjective coordination in the engineering of agent systems. In M. Klusch, S. Bergamaschi, P. Edwards, and P. Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, volume 2586 of *LNAI: State-of-the-Art Survey*, pages 179–202. Springer-Verlag, Mar. 2003.

[23] A. Omicini, A. Ricci, and M. Viroli. *Agens Faber*: Toward a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Sciences*, 2005. 1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 Apr. 2005. Post-proceedings.

[24] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 19–23 July 2004. ACM.

[25] A. Omicini, A. Ricci, M. Viroli, M. Cioffi, and G. Rimassa. Multi-agent infrastructures for objective and subjective coordination. *Applied Artificial Intelligence*, 18(9/10):815–831, Oct./Dec. 2004.

[26] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999.

[27] G. A. Papadopoulos and F. Arbab. Coordination models and languages. *Advances in Computers*, 46:329–400, 1998.

[28] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell. A preliminary taxonomy of multi-agent interactions. In J. S. Rosenschein, T. Sndholm, M. Wooldridge, and M. Yokoo, editors, *2nd International Joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 1090–1091. ACM Press, 2003.

[29] H. V. D. Parunak, S. Brueckner, and J. Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In C. Castelfranchi and W. L. Johnson, editors, *1st International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS'02*, pages 449–450. ACM Press, 2002.

[30] C. Peltz. Web services orchestration and choreography. *IEEE Computer*, 36(10):46–52, Oct. 2003.

[31] J. L. Peterson. Petri nets. *ACM Computer Surveys*, 9(3):223–252, 1977.

[32] J. Rambusch, T. Susi, and T. Ziemke. Artefacts as mediators of distributed social cognition. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, Mahwah, NJ, 2004. Erlbaum.

[33] A. Ricci, A. Omicini, and E. Denti. Activity Theory as a framework for MAS coordination. In P. Petta,

R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 96–110. Springer-Verlag, Apr. 2003.

[34] A. Ricci, A. Omicini, and M. Viroli. Extending ReSpecT for multiple communication flows. In *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, Las Vegas, USA, 24–27,June 2002.

[35] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In *Workshop on Programming Languages for Multi-Agent Systems (PROMAS), 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, Utrecht, The Netherlands, 2005.

[36] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *International Journal of Computer Supported Cooperative Work (CSCW)*, 5(2–3):155–200, 1996.

[37] K. Schmidt and I. Wagner. Ordering systems: Coordinative practices and artifacts in architectural design and planning. *International Journal of Computer Supported Cooperative Work (CSCW)*, 13(5–6):349–408, 2004.

[38] L. Steels. The artificial life roots of Artificial Intelligence. *Artificial Life Journal*, 1(1):89–125, 1994.

[39] M. Viroli, A. Omicini, and A. Ricci. Engineering MAS environment with artifacts. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, AAMAS 2005, Utrecht, The Netherlands, 26 July 2005.

[40] M. Viroli and A. Ricci. Instructions-based semantics of agent mediated interaction. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 19–23 July 2004. ACM.

[41] L. S. Vygotsky. *Mind and Society*. Harvard University Press, 1978.

[42] D. Weyns and T. Holvoet. On the role of the environment in multiagent systems. In this issue.

[43] D. Weyns, H. V. D. Parunak, and F. Michel, editors. *Environments for MultiAgent Systems*, volume 3374 of *LNAI*. Springer-Verlag, Feb. 2005. 1st International Workshop (E4MAS 2004), New York, NY, USA, 19 July 2004. Revised Selected Papers.

[44] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Modeling Link Qualities in a Sensor Network

Jure Leskovec
Center for Automated Learning and Discovery, School of Computer Science,
Carnegie Mellon University, Pittsburgh, PA, USA, and
Department of Knowledge Technologies,
Jožef Stefan Institute, Ljubljana, Slovenia
E-mail: jure@cs.cmu.edu, http://www.cs.cmu.edu/~jure/

Purnamrita Sarkar and Carlos Guestrin
Center for Automated Learning and Discovery, School of Computer Science,
Carnegie Mellon University, Pittsburgh, PA, USA
E-mail: {psarkar, guestrin}@cs.cmu.edu

Keywords: Sensor Networks, Machine Learning

Received: June 5, 2005

Sensor networks are ad-hoc wireless networks of small, low-cost sensors, which can measure characteristics of their environment. Autonomous low-cost sensors often have limited battery life, and are prone to failures and communication losses. It is thus important to devise efficient power usage, communication and message routing schemes. In this work, we concentrate on estimating the link qualities between pairs of sensors in a natural environment. The estimation is a basic component of algorithms that optimize the power of radio transmission signal, communication schedules, and a routing scheme. Our results show that simple regression models give estimates with only 6% error. We also show the dimensionality reduction techniques help us understand the topology of the communication network and identify potential bottlenecks in the network.

Povzetek: Z uporabo metod za zmanjšanje dimenzije podatkov in nelinearnih modelov v omrežju senzorjev je možno doseči zmanjšanje napake za 6%.

1 Introduction

A sensor network node is a small autonomous unit, often running on batteries, with hardware to sense environmental characteristics, such as temperature, vibrations and humidity. Such nodes usually communicate using a wireless network. A sensor network is composed of a large number of sensors deployed in a natural environment. The sensors gather environmental data and transfer the information to the central base station with external power supply [11].

Owing to the limited battery power of these sensors a very common strategy to maximize the expected lifetime is to use a better communication strategy. For this strategy to be globally optimized, we must model link qualities (LQs) between pairs of sensors. More precisely, the probability that sensor *j* will receive a message transmitted by node *i*.

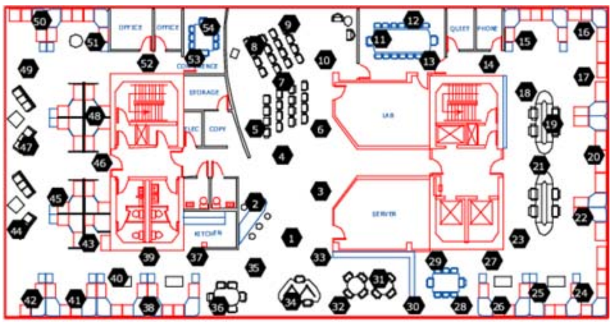


Figure 1: *Sensors in the Intel Berkeley Research lab*

Precise models of link qualitites are the basis of many optimization and networking algorithms. For example, these models can be used to refine the communication protocols, or to decrease the number of packet collisions by tuning radio power appropriately. A proper model for link quality can also be used to select the density and positions of the sensors to ensure efficient communication. These models can also help us ensure robustness of the underlying network by finding the most unstable parts of the network, and the sensors which are critical for communication.

2 The Dataset

The data comes from a deployment of 54 sensors positioned inside the Intel lab in Berkeley [3]. We have 33 days worth of data. For every 30 seconds we have a reading of binary link qualities between all pairs of sensors. There are more than 2.3 million readings in total (1 GB of data). During the data collection period some nodes died and there are about 1% of readings with missing values. The readings from sensors are highly noisy and skewed due to power failures, crashes of base station, sensor failures and rapid changes in the environmental conditions.
Figure 1 shows the map and the positions of 54 sensors inside the Intel Berkeley lab. The lab has a ring structure. The two 'holes' on the map correspond to the kitchen and the elevators. Near to the upper right corner of the map there was a cell phone base station. For this reason link qualities in the upper right part of the building are lower and decay faster with the distance than the link qualities for other parts of the building.

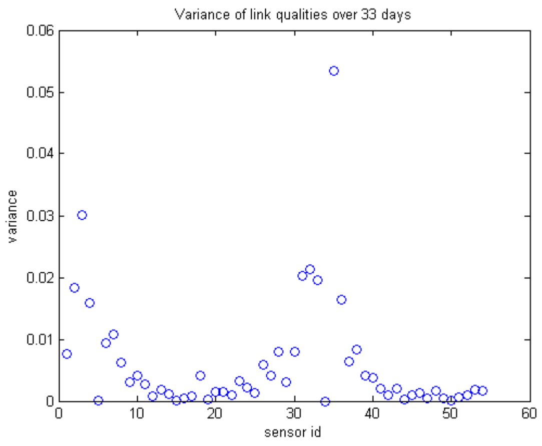


Figure 2: Variance of link quality over time of sensor 34 to all other sensors. Notice the small variance.

3 Analysis of Link Quality

There are two obvious variables influencing the link quality: one is time and the other is location. Figure 2 shows the variance of link qualities over time of sensor 34 to all other sensors. We can observe that the variance is very low on the average. For sensors between 30 and 35, which are closest to sensor 34, we observe that the variance of link quality is higher. As sensors get farther apart the variance of link quality also gets smaller. From figure 2, other experiments and measurements, we concluded that link qualities do not change significantly over time.

We mainly concentrate on spatial link qualities. In this paper we try to relate physical position of the 2 sensors with the link quality. Given the (x,y) positions of the sensors we model the decay of link quality with the distance and build a link quality map.

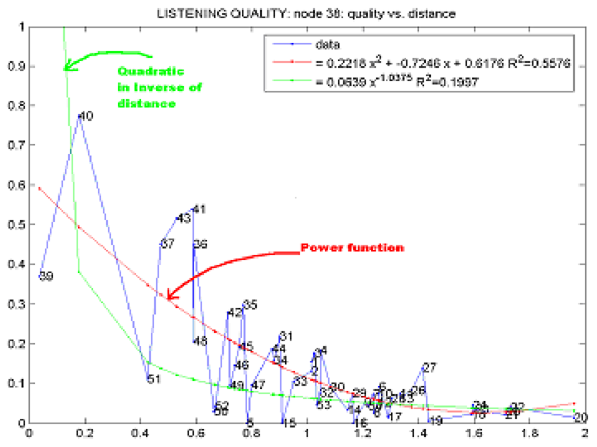


Figure 3: Link quality of node 38 to all other nodes, sorted by the distance.

3.1 Link Quality as a Function of Distance

Theoretically the strength of radio signal should drop with the square of the distance, so we expected link quality to follow the same law.

We analyzed a typical situation and tried to fit a function to the link qualities. In figure 3 we took sensor number 38 and plotted the link qualities to other sensors versus their Euclidean distance (*LQ=f(d)*, where *d* is the distance). A quadratic function had a very poor quality fit (square of the correlation coefficient between the independent and dependent variables, R^2=0.55), a power function ($LQ(d)=d^{-c}$), with c around 2 performed even worse with R^2=0.19. On the average the quadratic regression was the best, with the average R^2 around 60%. The fits for power function were not at all very satisfying, having an average R^2 of 20%.

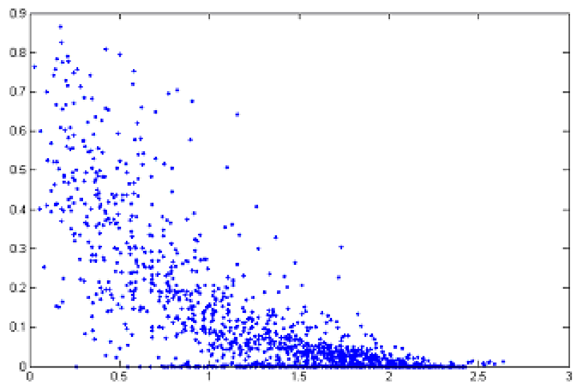


Figure 4: For all possible pairs of nodes we plot link quality versus the distance between the sensors.

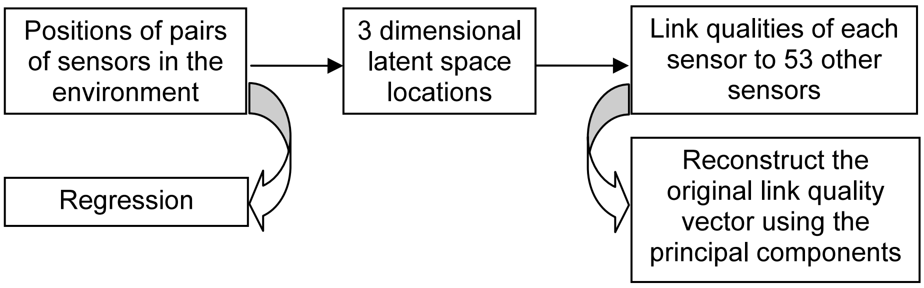

Positions of pairs of sensors in the environment
3 dimensional latent space locations
Link qualities of each sensor to 53 other sensors
Regression
Reconstruct the original link quality vector using the principal components

Figure 5: The 2 level model. We learn 3 regression models to map from (X,Y) positions to latent space positions. We then use principal components to map from the latent space to the link qualities.

Investigating even further we tried to fit a second degree polynomial to the link quality vs. distance between each pair of sensors. Figure 4 shows the amount of noise in the data. We plot the link quality versus the distance. Each point on the plot is a pair of sensors at some distance having some link quality. Observe the high noise in the data. The distance itself is not a good predictor of the link quality between a pair of sensors in the environment.

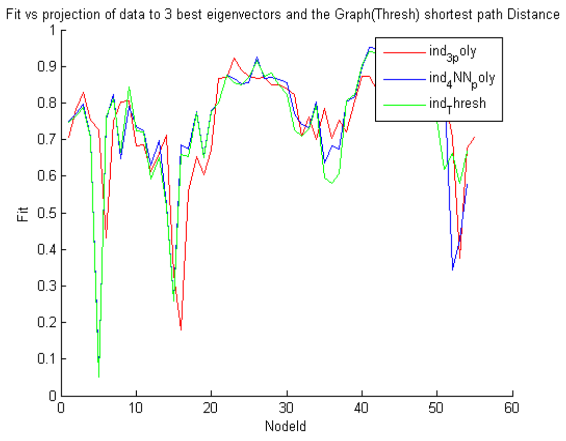


Figure 6: Comparison of predictive accuracy using the 3 distance metrics: distance in 3 dimensional PCA space (top), graph with each node have degree 4 (middle), and graph in which all sensors within a constant radius of a node are connected (bottom).

We also used distance metrics other than Eucledian distance. We connected the sensors into a graph similar to one shown of Figure 1. We explored few different techniques to connect the nodes into a graph based on sensor positions and link qualities. We then measured the shortest path distance between the nodes in the graph.

Figure 6 shows the comparison between various distance metrics. We compare 3 different distance metrics: distance in 3 dimensional space obtained from Principal Component Analysis (PCA) [2].(top), graph where each node has degree 4 (middle), and graph in which all sensors within a constant radius of a node are connected (bottom). The results show that the 3 different approaches are pretty much the same, though the threshold distance metric is in most of the cases equal or better than the others.

Our reasoning was that two nodes can be really close together but if there is a wall in between they won't be able to hear each other. Using a graph distance we would prevent this kind of problem. Unfortunately this did not improve the results.

3.2 Dimensionality Reduction

So far we have been working with full 54 by 54 who-talks-to-whom link quality matrix. One way to reduce the amount of noise in the data is by using dimensionality reduction techniques.

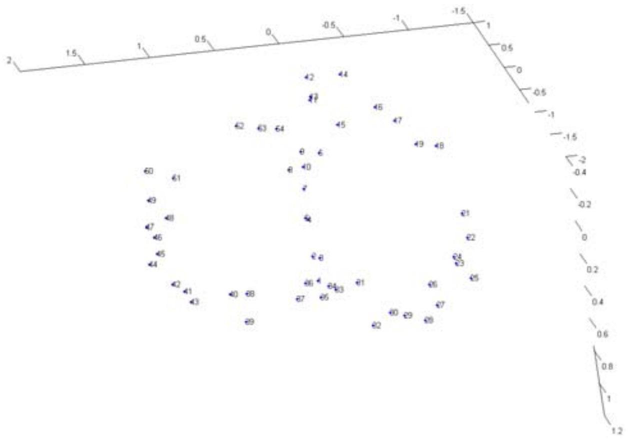


Figure 7: The projection of link quality data on first 3 principal components. Notice the two rings very similar to actual map of the lab.

We perform PCA on 54 by 54 link quality matrix. We essentially do a metric multidimensional scaling [1] on the data to learn the underlying coordinates in a 3-

dimensional Euclidean space, namely the latent space. The first 3 eigenvectors explain around 70% of the variance of the data. Increasing the latent space to 4 dimensions it covers additional 5% more variance so we decided to continue experimenting with 3 dimensional latent space.

Figure 7 shows the latent coordinates of the sensors in a 3-dimensional space. We can clearly observe the two rings we had seen on the map of the lab (Figure 1). This means we are able to reconstruct the map of the lab using only the link quality data. This also implies that the sensor locations should be good attributes for modeling the link qualities. Notice also the big gaps in the rings. This shows two “holes” mentioned in Section 2 and suggests deploying more sensors in that part of the lab to avoid a potential bottle neck in the communication network.

A close inspection of Figure 8 reveals a set of nodes outlining a big hole in the graph. If any two of these nodes die, the communication between two halves of the network might be seriously ruptured. For example nodes 5, 52 and 14 connect the left and right halves of the sensor network. These nodes are critical for the communication of the network. This suggests that one needs to deploy more sensors in this part of the lab to increase the robustness of the network. Thus we see that the multidimensional scaling approach reveals some very important and interesting patterns in the data, besides matching the true map of the sensor locations.

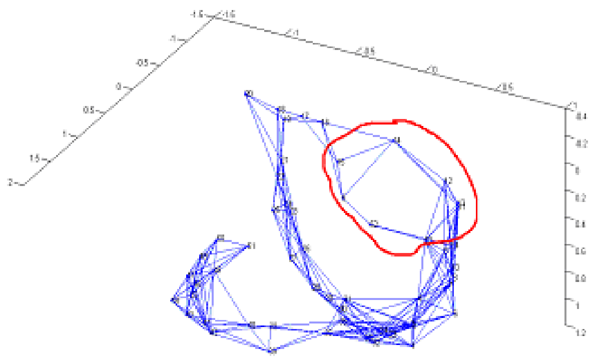


Figure 8: This is same as Figure 7, with the only difference that we have connected nodes if they are within a certain distance from one another in the 3-dimensional latent space. This distance was chosen empirically so that the graph is fairly connected, midway between dense and sparse.

3.3 Link Qualities via Dimensionality Reduction

Now we use the notion of latent space to construct a 2 level model for link quality prediction. We will first learn a regression model to map from lab coordinates(x, y) of a sensor to the 3 dimensional latent space position. We then use the principal components to map from 3 dimensional latent space to the original 53 dimensional vector of link qualities. Figure 5 more clearly depicts the idea.

Note that we learn 3 separate regression models, each from mapping from (x,y) lab location to of the sensors to a particular latent space dimension. We use linear, quadratic and cubic regression. Also note that we have only 54 training instances. We performed leave one out cross validation and report the mean absolute error. Table 1 shows results on test and training set for the 3 models. Notice that the quadratic model performs best and the cubic model overfits the data. Quadratic model gains from 15% to 2% on test accuracy in comparison to the linear one.

Regression type	**Training Set** Mean Error	**Test Set** Mean Error
Linear		
LS dimension 1	0.073	0.078
LS dimension 2	0.229	0.244
LS dimension 3	0.124	0.131
Quadratic		
LS dimension 1	0.045	0.051
LS dimension 2	0.078	0.090
LS dimension 3	0.071	0.083
Cubic		
LS dimension 1	0.038	0.050
LS dimension 2	0.077	0.098
LS dimension 3	0.062	0.099

Table 1: Performance of the regression mapping from XY lap sensor positions to the latent space positions. Quadratic regression performs best, cubic overfits.

Figure 9 shows the scatter plot of predicted latent space position and true latent space position of a particular sensor for a quadratic model. We observe similar plots also for other latent space dimensions. We observe that the residuals are well distributed, and concluded that the quadratic model is suitable.

So far we build the model to map from physical sensor positions to 3 dimensional latent space positions. The last step of the procedure shown on figure 5 is to use principal components to map from 3 dimensional latent space to original 53 dimensional vectors of link qualities. Using quadratic regression model and 3 dimensional latent space the final mean square error of link qualities is 0.14. If we increase the number of dimensions in the latent space to 4, the error increases to 0.145. Increasing the number of dimensions further to 10, gives the average mean error of 0.20.

Notice we are observing an interesting interplay between the two stages of our model. As we pick more latent space dimensions the mapping from latent space to the link quality gets more accurate. On the other hand the mapping from (x,y) positions gets less accurate and the combination of both results in worse performance. The problem with learning mapping to higher latent space dimensions is that they contain more noise, so the regression gets unstable with large errors. Using cross validation we get the best results when using 3 dimensional latent space.

Figure 10 shows the performance of predicting the 2nd, 3rd, and 4th principal component (dimension of latent space). Notice that we can very well fit 2nd and 3rd dimension, while accuracy on 4th latent dimension drops significantly.

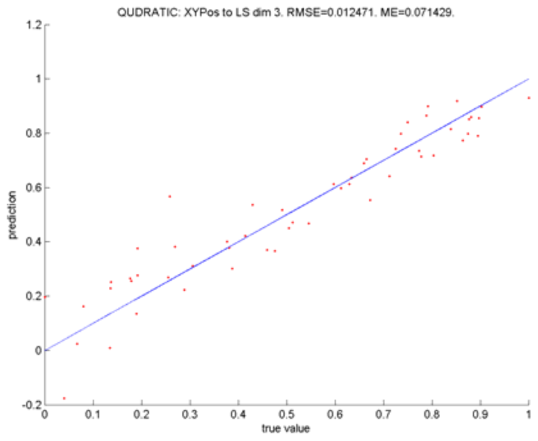


Figure 9: Scatter plot of true and predicted latent space positions for the quadratic model and the 3rd latent space dimension.

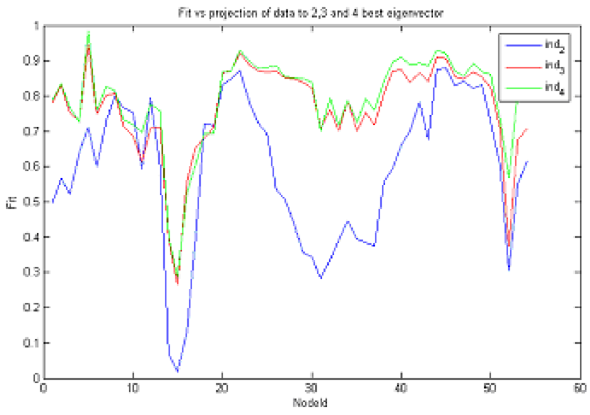


Figure 10: Fit of the regression to the data for the 2nd, 3rd, and 4th best principal component. Figure plots the fit of regression versus sensor id. Notice that we can very well fit 2nd and 3rd dimension, while accuracy on 4th latent dimension is much worse.

3.4 Direct Approach

We also considered a more direct approach. Instead of using Principal Component Analysis to reduce dimensionality of the class and reduce the noise we learn the link quality between a pair of sensors given the lab coordinates of them, using a regression model. In this case we have 2862 (54 squared) training examples each having 4 real attributes (locations of the two sensors). We perform 10 fold cross validation and report average mean error on training and test set.

We compared 3 classes of algorithms: normal least squares polynomial regression, a variant of logit transform and regression Support vector machines (SVM) [4] using polynomial and radial kernels. For the logit transformation our idea was to transform the link qualities (which are probabilities and thus reside on interval (0,1)) to the whole real space. Our hypothesis was that it may be easier to learn the link qualities spread out over the whole real space. In this case we transformed the link quality *LQ* with the equation *LQ' = log(LQ/(1-LQ))*. We then learned the regression model, performed the inverse logit transform and measured the mean error.

Regression type	**Training set** Mean Error	**Test set** Mean Error
Normal		
Linear	0.108	0.108
Quadratic	0.087	0.088
Cubic	0.086	0.088
Logit transform		
Linear	0.409	0.409
Quadratic	0.412	0.412
Cubic	0.411	0.411
SVM		
Linear	0.119	0.119
Quadratic	0.093	0.093
Cubic	0.090	0.090
6 deg polynomial	0.082	0.083
Radial	0.061	0.062

Table 2: The performance of various regression techniques.

Table 2 shows the results for the 3 classes of regression algorithms we tested. Our first observation is that even simple linear regression outperforms our 2 level model by 4%. We observe a 2% improvement of quadratic and cubic model over the linear model. Next observation is that logit transform performs far the worse. It performs a bit better than random guessing which has the mean error of 0.5. The SVM with polynomial kernels have similar performance as normal least squares regression using the same degree polynomial as in SVM kernel. We observe that even very high degree polynomial kernel of degree 6 does not help to fit the data very well.

The radial kernel outperforms all other techniques with a mean error of around 6% on both training and test set. Radial kernel is especially appropriate for this task, since it has a bell shape, which means that a link quality basically decays in a bell shape with the distance.

3.5 Link Quality Map

Having built the model we can now look at the link quality map for a particular sensor. We fix the location of the first sensor and then for every position of the second sensor we use the model to obtain the link quality. We call this the link quality map.

Figures 11 and 12 show the two examples of link quality maps generated using the SVM radial kernel. Figure 11 shows the case when we positioned the sensor in the center of the lab. We observe a bell shape decay of link qualities. Notice how the link qualities are very low on the left middle part of the figure. Notice also that on the left side in the top corner link quality is better than left middle and left bottom corner. This is because left bottom part is further away and better hidden behind the wall. There is also a cell-phone base station on the left part of the map, which further decreases link qualities.

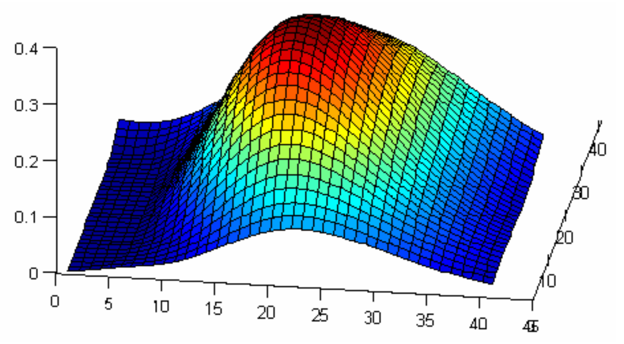


Figure 11: Link quality map for a sensor in the center of the Intel lab.

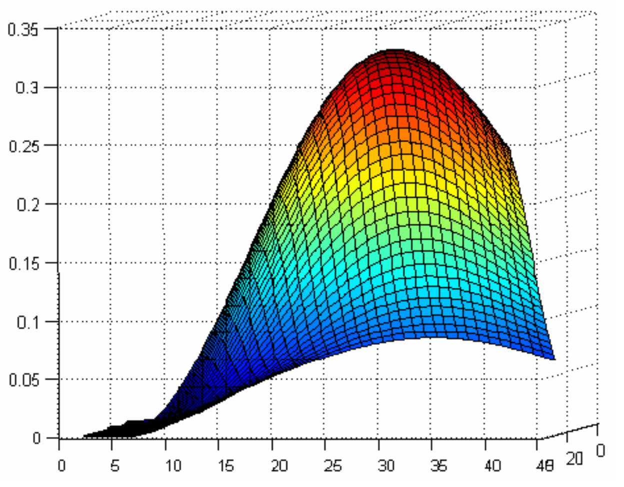


Figure 12: Link quality map for a sensor in the corner of the lab.

One would falsely expect that link qualities in the holes of the two rings (kitchen and the elevator) should be close to zero. This is not the case since we have no training or test data points inside those rings and the link quality just gets interpolated over that empty space.

Figure 12 shows the case when the sensor is positioned into a corner of the lab. We observe a similar bell like decay of link quality away from the sensor. Notice faster decrease in link quality towards the left part of the lab where the mobile phone base station was located.

4 Related Work

Power efficiency plays a central role in sensor networks. A lot of work has been done on estimating link quality, most of which focus on modeling reception rate over time. In [12] the authors derived analytical expressions for expected link lifetimes, rate of new link arrivals, and probability distributions for the above quantities, both of which are crucial for the understanding the underlying communication structure. Authors in [13] discuss different experiments to measure packet delivery performance. This work also models the spatial correlation between packet loss among individual receivers. A generic nonparametric statistical procedure for establishing a mapping between two characteristic properties of a sensor network is discussed in [14]. For example in this paper the authors model a probability density function of the reception rate and the distance between the two sensors, demonstrating the spatial correlation aspect of link qualities.

People also have investigated scalable and power-efficient protocols [5], power management [6], efficient routing [7] and querying in sensor networks [8]. The ones most related to our work are [9] and [10]. Our findings are in accordance to conclusions in [9] that the link characteristics are far from the theoretical models. However, most of these works survey the detailed link stability but not its effect on positioning, while our work concentrates in modeling link qualities in a natural environment and how they change with positions of the sensors.

The question we have not addressed here is obtaining the XY positions of the sensors. If sensors are deployed inside a building or some other controlled environment then obtaining coordinates of each sensor is realistic. On the other hand if sensors are scattered (electronic dust) then obtaining their positions is a nontrivial problem.

5 Conclusion

In this work, we showed exploratory results on the modeling of link qualities in a sensor network. Since link qualities are often invariant with respect to the time, we focused on the spatial aspect.

In our experiments simple regression techniques were quite effective. However, in our comparisons, support vector regression with radial kernel was the best performing approach. Intuitively, link qualities decay with distance, a property captured effectively by this model.

We also showed how dimensionality reduction techniques can be used to analyze link qualities, identifying critical nodes and sparsely connected parts of the sensor network.

References

[1] R. Sibson. *Studies in the robustness of multidimensional scaling: Perturbational analysis of classical scaling*. J. Royal Stat. Soc. B, Methodological, 41:217, 1979.

[2] G. H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996

[3] http://db.lcs.mit.edu/labdata/labdata.html

[4] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

[5] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. *Physical layer driven protocol and algorithm design for energy efficient wireless sensor networks*. In Mobile computing and networking, 2001.

[6] M. Bhardwaj, A. Chandrakasan, and T. Garnett. *Upper bounds on the life time of sensor networks*. In IEEE International Conference on Communications, 2001.

[7] W. Heinzelman, J. Kulik, and H. Balakrishnan. *Adaptive protocols for information dissemination in wireless sensor networks*. In Mobicom, 1999.

[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. *Model-Driven Data Acquisition in Sensor Networks*. In VLDB 2004.

[9] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. *Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks*. Technical Report UCLA/CSD-TR 02-0013, 2002.

[10] D. Son, B. Krishnamachari, and J. Heidemann. *Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks*. In Sensor and Adhoc Communication and Networks, 2004.

[11] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. *Wireless sensor networks for habitat monitoring*. Tech. R. IRB-TR-02-006, Intel Research, 2002.

[12] Prince Samar, Stephen B. Wicker. *On the behavior of communication links of a node in a multi-hop mobile environment.* In Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, 2004, Tokyo, Japan.

[13] J. Zhao and R. Govindan. U*nderstanding Packet Delivery Performance in Dense Wireless Sensor Networks.* In ACM Sensys 2003.

[14] A. Cerpa, J. L. Wong, et al. *Satistical Model of Lossy Links in Wireless Sensor Networks.* In the Fourth International Symposium on Information Processing in Sensor Networks.

Improving Study Planning with an Agent-based System

Aki Vainio and Kimmo Salmenjoki
Department of Computer Sciences
Faculty of Technology, University of Vaasa
Box 700, Vaasa, Finland
E-mail: aki.vainio@uwasa.fi, ksa@uwasa.fi

Keywords: autonomous agents, personalized study plans, collaborative planning

Received: May 8, 2005

This paper presents a system for designing and updating a personalized study plan in a collaborative environment. Unlike existing systems, which are mainly interested in storing the study plan, this system based on learning agents is able to suggest a study plan and if needed, identify potentially problematic choices in the future, thus bringing dynamics in to the system. By collaborating with other agents in a multi-agent environment, the chances of finding a mutually beneficial result is improved. A prototype of the system for creating study plans is available. Initial empirical results show that after a short learning period, the system is able to form a study plan which requires minimal attention from the students.

Povzetek: Predlagan je sistem učenja s pomočjo agentnega sistema.

1 Introduction

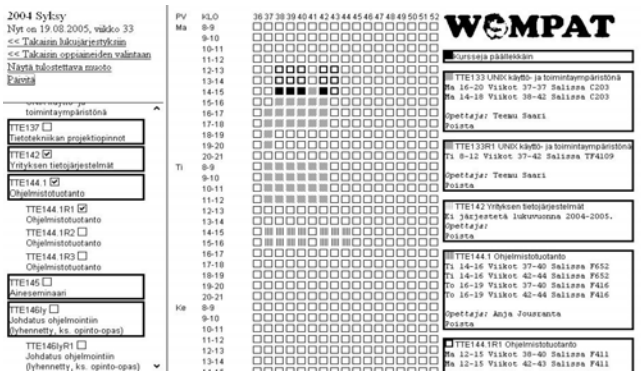
The Bologna process imposes many changes to the system of higher education in Europe [12]. In Finland, one of the more recent changes is the new limit to the amount of years a student can study. Currently, it is common to study for six or more years to obtain a master's degree. In the future, five years should be the norm. Since the students, university personnel or the ministry of education are willing to relinquish any of the requirements for a master level degree, better planning is needed.

According to [1], one tool for achieving this planning is the personalized study plan, which is a requirement for all students in Finland in the future. Since a first year student does not have enough knowledge to make decisions for the whole span of his studies, creating and keeping the personalized study plan up to date is important as the student becomes more knowledgeable. [1]

Current systems are not using the full potential of advanced information systems. The approaches OVI [2] and Oodi [3] are attempts at simplifying the process, but neither seems able to address the personal study planning as all they do is simply store the study plan. OVI even requires the student to make all the course choices, even though the mandatory courses could easily be selected beforehand. Oodi will include such functions and according to the current design, it can also check the study plan for correctness, but the timetables for the Oodi project are such that a usable version will not be available for at least a few years.

The new system proposed here will be a part of Wompat-system [4], which is designed as a tool for students to use when planning their schedules. Wompat has been used in University of Vaasa for two years with good feedback from both administration and students. Also, the consortium behind the Oodi system has expressed interest in it.

The student view of the web based Wompat system is shown in figure 1 with course options on the left, schedule organized by weeks, days and hours in the center and courses chosen by the student in the right. Using Wompat, the student obtains a weekly schedule for all the courses that he or she intends to study.



2004 Syksy
Nyt on 19.08.2005, viikko 33
<< Takaisin lukujärjestyksiin
<< Takaisin oppiaineiden valintaan
Näytä tulostettava muoto
Päivitä

toimintaympäristönä
TTE137 ☐ Tietotekniikan projektiopinnot
TTE142 ☑ Yrityksen tietojärjestelmät
TTE144.1 ☑ Ohjelmistotuotanto
TTE144.1R1 ☑ Ohjelmistotuotanto
TTE144.1R2 ☐ Ohjelmistotuotanto
TTE144.1R3 ☐ Ohjelmistotuotanto
TTE145 ☐ Aineseminaari
TTE146ly ☐ Johdatus ohjelmointiin (lyhennetty, ks. opinto-opas)
TTE146lyR1 ☐ Johdatus ohjelmointiin (lyhennetty, ks. opinto-opas)

Kursseja päällekkäin

TTE133 UNIX käyttö- ja toimintaympäristönä
Ma 16-20 Viikot 37-37 Salissa C203
Ma 14-18 Viikot 38-42 Salissa C203
Opettaja: Teemu Saari
Poista

TTE133R1 UNIX käyttö- ja toimintaympäristönä
Ti 8-12 Viikot 37-42 Salissa TF4109
Opettaja: Teemu Saari
Poista

TTE142 Yrityksen tietojärjestelmät
Ei järjestetä lukuvuonna 2004-2005.
Opettaja:
Poista

TTE144.1 Ohjelmistotuotanto
Ti 14-16 Viikot 37-40 Salissa F652
Ti 14-16 Viikot 42-44 Salissa F652
To 16-19 Viikot 37-40 Salissa F416
To 16-19 Viikot 42-44 Salissa F416
Opettaja: Anja Jousranta
Poista

TTE144.1R1 Ohjelmistotuotanto
Ma 12-15 Viikot 38-40 Salissa F411
Ma 12-15 Viikot 42-43 Salissa F411

Figure 1. Screenshot of the current Wompat-system

2 Automated Approaches for Study Plan Formulation

2.1 Agents

According to [7] a rational agent is described by its PAGE-description. The PAGE-description consists of percepts, actions, goal and environment. The creation of the study plan can be clearly divided into two parts: selection of courses and timing of the studies taken by the student. According to this, our system will consist of two agents, where the first one controls the selection of courses, storing the selection in a database and the second agent uses these selections in order to automate the preferred annual schedule of the selected courses. Communication between the agents is handled by changing the environment, which in this case means changes in the database. The core software of the personalized Wompat system is a combination of the two agents and their automated communication via the information content of the database. The details of the database will be given in section 3.

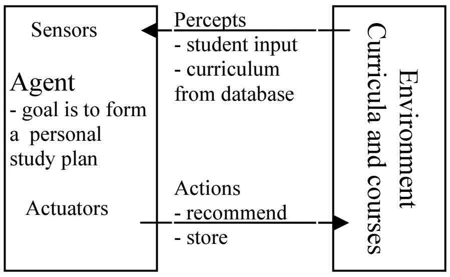


Figure 2: PAGE-description of the agents in Wompat

With this agent approach the use of study plans can be enhanced. One specific problem in small departments is the lack of resources for arranging courses annually. Often many courses, which are not popular, are only arranged every other year or even less frequently. In order to optimize the use of resources, the departments can use the information gained from the personalized curricula to arrange only the courses with enough interest.

The proposed system could also use this same information to examine beforehand which courses are likely to be arranged in a given year. If the system finds that a student has chosen a course with low common interest, it could make the student aware of the problem. The system could also try to find another course which might be of interest to the student, based on the student's earlier choices and the choices made in other curricula.

The system needs decision making when suggesting courses for a study plan and when finding the correct timing for a course. In many cases, the latter is not a real problem, but it is in some instances, where some courses might be suitable for two or more years.

Finding the courses, which are probably not arranged, is simple. The system can find them by simply checking how many people have chosen a course on a given year. The system can either be equipped with the knowledge of how many students are usually required for a course to be held. It can also be equipped with the knowledge of how many of the courses will be arranged in a given year and make an estimation based on that information.

The decision making in the agents of the system is based on knowing the prerequisite courses required for more advanced courses, the interests of the student and further learning of these issues. Through these methods, the system can find better suggestions to be presented to the student.

2.2 Prerequisite Sourse Utilization for Decision Making

The relationships of prerequisite courses and advanced courses form a directed acyclic graph (DAG). This graph can be used to find appropriate courses based on what the student has already taken or on the other hand, if the student or system sees fit to choose an advanced course, the system can easily find the prerequisite studies needed for that course and suggest them to the student. By using the DAG, the courses can be easily found by moving through the DAG recursively. On the other hand, if the student has not completed all the needed prerequisites and doesn't have enough time to complete them all, the system can suggest that the student stay away from those courses.

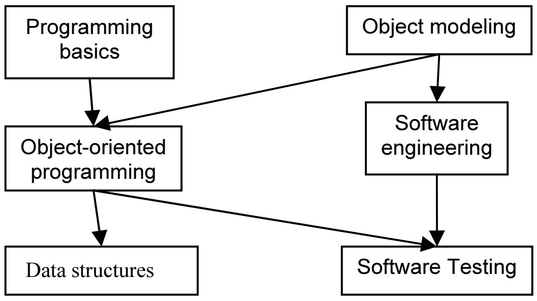
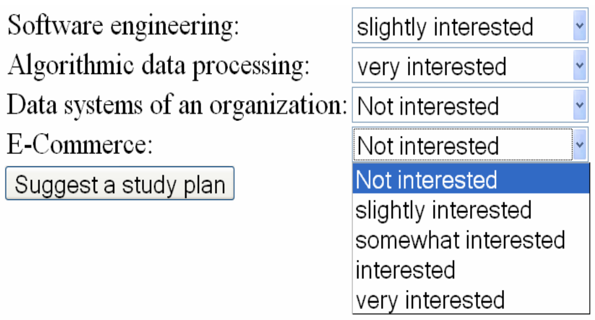


Figure 3: Example of sub-DAG of courses

2.3 Students Selection of Courses

As a student can focus his or her studies in a number of ways, certain prepared areas of orientation can be used to help the decision making. After these areas are defined, each course can be given a weighted relationship with these areas by an expert, for example a teacher.

To give the system information on which to base the decision making on, the student can give his or her interests in the orientation in a manner reminiscent of fuzziness, where the student can choose from descriptions acting as variables:



Software engineering:	slightly interested
Algorithmic data processing:	very interested
Data systems of an organization:	Not interested
E-Commerce:	Not interested

Figure 4: Form for gathering data on student's interests. See [5] for more details.

The agents receive the user's selections via the database. The database also contains the decisions made by the other students. All this information is used by the agents in the automated selection process.

2.4 Learning in the Agent System

Machine learning is change in the system which results in better performance [10]. Better performance in an agent means that the frequency of right or good decisions grows over time [10]. In this system, the agents can be said to learn if they choose the right courses more often and are able to time courses with better accuracy over time.

Since the system should be able to function as autonomously as possible, learning is required to give them some freedom from the subjective or uninformed views of the people who give the agent the preliminary information on the relationships between the orientations and the courses. Since the system has access to full feedback from all the decisions it has made, in the form of whether or not the students follow the agent's advice, learning can be very fast. This is called active and supervised learning [7].

Learning can be based on simple decision theoretical analysis using frequency as a basis for decision. However, with small group sizes, this isn't always possible or the data is not accurate enough. For this reason, a set of values based on the preset relationships can be used to give the system some basis to work on when there is not enough gathered data to draw conclusions from. This also stops the system from learning too much from the first few students. By using this method, the system does not work on probabilities, but rather on approximations of probabilities.

If needed, the system can be taught in a supervised manner. By giving the system some sample personalized curricula, it can use those as cases to learn from.

After the system has been in use for a long period of time, it might not be able to learn as quickly as before. The need to learn is still there as the curricula, tastes of the students and other matters change over time. This problem can be overcome by simply introducing limits on the number of students used in the learning phase.

Learning has another important function: Often the courses have no set year for completion. Even if the window is usually only years in these instances, this information might make a difference in the decision making process. This information can be learned from averages, with some tolerance for error. This learning also removes the need for telling the system when courses should be completed by the student. The system can make those decision based on the prerequisite courses as described in 2.2 and through learning.

2.5 The Process of Making the Study Plan Automatically

The process of making the study plan requires actions from both the student and the agent. The role of the student is mostly as a control measure to see that the study plan is to the students liking. The student has the power to change as many of the decisions made by the agents, but if the agents work correctly, not many changes should be needed.

Figure 5 presents the process and the messaging between the participants. The process is started by the student, who requires a study plan. Upon logging into the system, the student can be identified and the right curriculum can be chosen. Based on the curriculum, a simple form with the possible choices on orientation is presented to the student. The student than communicates his or her interests to the agent through the form.

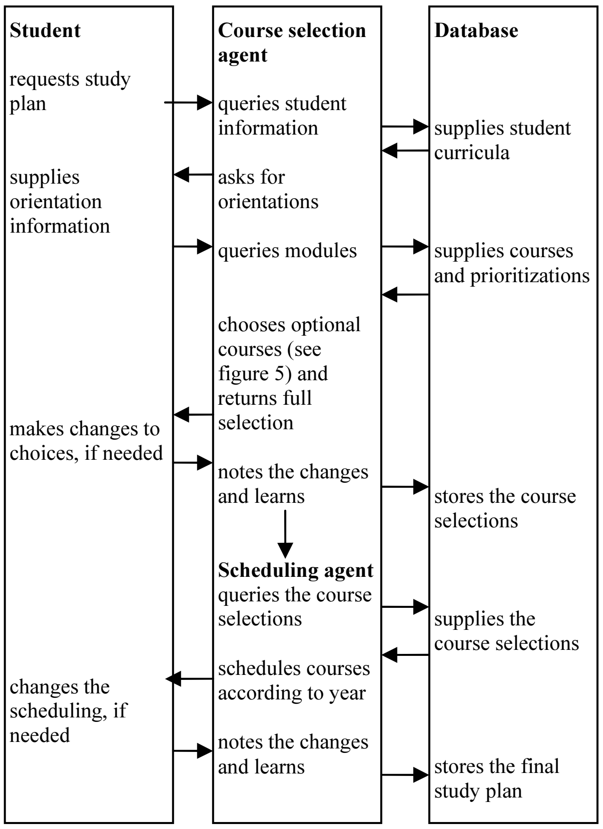

Student
requests study plan
supplies orientation information
makes changes to choices, if needed
changes the scheduling, if needed
Course selection agent
queries student information
asks for orientations
queries modules
chooses optional courses (see figure 5) and returns full selection
notes the changes and learns
Scheduling agent
queries the course selections
schedules courses according to year
notes the changes and learns
Database
supplies student curricula
supplies courses and prioritizations
stores the course selections
supplies the course selections
stores the final study plan

Figure 5: The process of the agent driven personal study plan

Based on that information, the agent queries the database for the information needed for choosing the courses. The courses chosen are then presented to the student, who can make changes as needed. The agent records the changes and learns from them. The agent also notes choices that were not changed so that it can base its future decisions on those.

After this, the agent stores the choices into the database and informs the scheduling agent so that it can begin its work. The scheduling agent queries the courses chosen by the student and also the learned information on which point of the students studies they fit best. Based on that information, the agent forms a full schedule on the courses, completing the study plan. The student can change any timing within the schedule. The agent will note those changes and learn accordingly. Finally, the study plan is stored in the database for further use.

2.6 Decision Making in the Agent System

The system has full information of the structure of the curriculum and from this information it can find the last modules which have optional courses in them. This is important as the approach used is that the student has a clear goal in his or her studies and the study plan is used to reach that goal.

Based on this approach, the choices on orientations made by the student are used to find the most suitable courses for the student based on learned suitability or values given by the expert.

After the courses have been chosen, the system uses the information on prerequisite studies and automatically adds them to its suggestion. After this, it moves down the curriculum to the next module with optional courses. At this point, the system may have already filled this module, if there were many prerequisites for the more advanced courses. If there is still a need for new courses, enough are chosen and the system checks for prerequisites again. This is repeated until the whole curriculum has been handled. In the end, all the mandatory courses are added.

With all the courses selected and after having given the student a chance to have his or her input on the choices by selecting or removing courses, the system can move onto arranging the courses by year.

A basic layout can be formed by using a topological sort. Many courses find their natural timing this way, but not all. The process continues by using the learned data on correct timings. The topological sort also gives the agent enough information to find a timing to suggest for new courses.

3 Data Storing Requirements for the Study Plan Environment

All of the information needed for the decision making and the study plans can be stored in a relational database. Figure 7 represents a possible structure for such database. This structure is used in the prototype version [5, 6].

The design is built around the student and the curricula. Curricula are divided into modules (basic studies, major, minor and so forth), which can have several different variants; although in many cases they don't have any. Modules are made up of courses and the student's study plan comprises of these with timing information.

As described in 2.4, the system requires some information on which courses are more advanced than others. This information is presented by forming a tree. The root is the curriculum itself and it usually has two children: bachelor level studies and master level studies. These are again divided into two or more sections and so forth. If a module has both mandatory and optional courses, these have been divided into different sections.

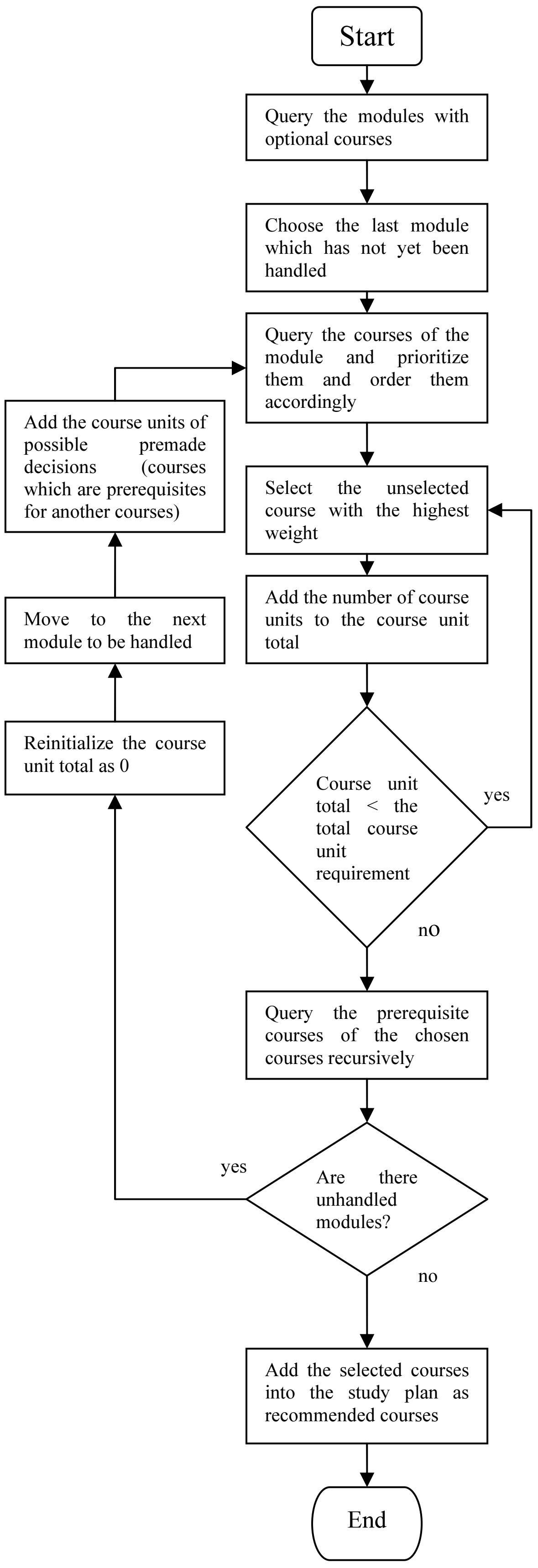

Start
Query the modules with optional courses
Choose the last module which has not yet been handled
Query the courses of the module and prioritize them and order them accordingly
Add the course units of possible premade decisions (courses which are prerequisites for another courses)
Select the unselected course with the highest weight
Add the number of course units to the course unit total
Move to the next module to be handled
Reinitialize the course unit total as 0
Course unit total < the total course unit requirement
yes
no
Query the prerequisite courses of the chosen courses recursively
yes
Are there unhandled modules?
no
Add the selected courses into the study plan as recommended courses
End

Figure 6. Algorithm for choosing the optional courses

The tree is represented in the database by giving each module the values 'left' and 'right'. Beginning with the root, each vertex is given the values depth first. On the first visit, the 'left' value is assigned and on the final visit, the 'right' value is assigned. Thus root will have values of left is 1 and right is double the total number of vertices. The children of each vertex can be identified by using these numbers as both of the values of all the children will be between 'left' and 'right' of the given vertex. The main advantage of this method is the ease of depth first searches. The whole tree and thus the curriculum can be easily represented by ordering it based on 'left' values and using indentation [8].

Prerequisite courses are represented by a table with two separate foreign keys linking it to the courses. This table functions as the basis for forming the DAG [11] (see 2.2). An example of the data contained in a DAG is given in figure 6.

The orientation options are in their own table, with another table connecting them to the courses and the choices made on them by the students. The table connecting the orientations and courses holds the key information for the system to base its decisions on selecting courses. The table holds the weight proposed by the expert, who is working on the orientation, as well as the learned weight from previous choices by the students. Also, the number of students, from whom the weight was learned, is presented. This figure does not have to be accurate. It can be used to control the learning process somewhat. In the prototype, each course has a default 2 on this value and it can never go higher than 10, except momentarily.

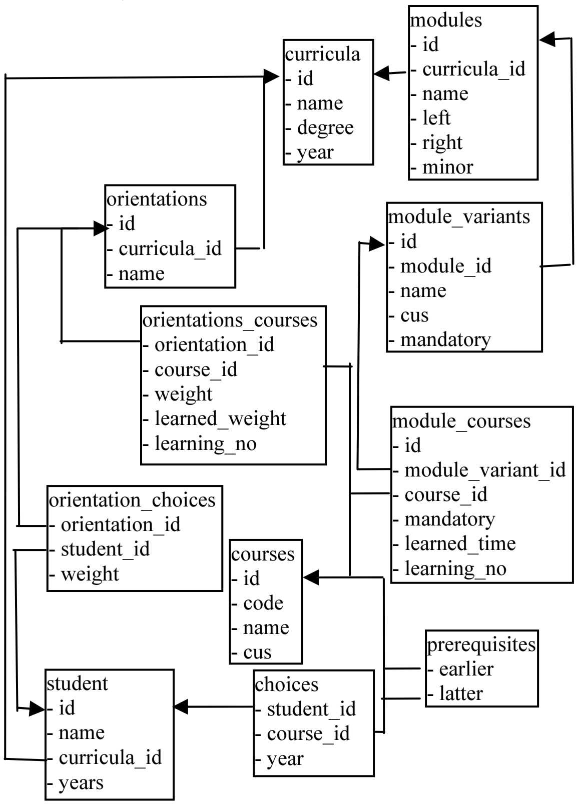


Figure 7: Structure of the database

The learned information on correct timing for courses is situated in the table 'module_courses' which represents which courses belong to which module.

The database structure is slightly redundant, as the information used for learning could be derived from the other tables, but the redundancy can be used for the aforementioned control and the structure can also make the system more efficient. The latter depends on how people use the system. If the study plan is constantly changed and not only looked over, the redundancy should probably be removed.

As the agent and decision approach proposed in section 2 is used, the classical Wompat-system of figure 1 is enhanced with an automated functionality on the personalized curriculum content selection.

4 Using Personal and Universal Information in Decisions of Agents

To keep the study plans up to date without the need for the student to check it regularly, the checking should be automated. This work can be done by an agent or several agents [9],

When the agent encounters a problematic choice, it notifies the student and begins to look for another course or courses which to suggest to the student. It can also leave a notification to a central message station that if others agents are having problems finding a suitable course they could perhaps make a common decision. This approach has the benefit of gathering a number of students to participate on a course so that the chance of that course being arranged is higher.

At this point, the agent must make a decision. Will it suggest another course, which is more interesting, but might not be arranged, if there aren't enough students, or will it suggest a course which is probably arranged, but is not as interesting to the student as the other possibilities?

The agent could also make an attempt to make a deal with the other agents. If the problematic course is a key course in the students study plan, the agent could offer to change another course in its study plan if other agents are willing to change a course in their study plans to the key course. Student input is crucial at many of the stages, since the student has to actually carry out the plan.

Figure 8 represents one case where two agents might be able to guide their students into mutually beneficial agreement. Since neither can make decisions without receiving input from the students, the process is slow. Also, the process might result in nothing, if there still aren't enough students who plan to take the course. The risk failure is increased by the fact that the students might get bored with the process.

In this case, the most important thing is to keep the student aware of the situation, so that the student can take action too if necessary or the agent can stop the negotiation if the course suggested is not to his or her liking.

Technically the decision making can be done with in the same way as before. The environment could be a ticket-like system built on the same database structure as the rest of the system. The agent can leave or read tickets from the database and use them for evaluation of its situation.

Agent 1 is looking at a study plan with four courses selected (**bold**) and four other possibilities. It also has access to the perceived suitability to the student in question (between 0 and 1).
Agent 1 sees that not enough students have chosen course A, which is according to the figures very central in the study plan.

Agent 2 is in a similar situation with course E.

Agent 1	Agent 2
- course A, 0.91	- course A, 0.45
- course B, 0.74	**- course B, 0.78**
- course C, 0.75	**- course C, 0.75**
- course D, 0.45	- course D, 0.44
- course E, 0.44	**- course E, 0.94**
- course F, 0.31	**- course F, 0.46**
- course G, 0.22	- course G, 0.24
- course H, 0.11	- course H, 0.31

Agent 1 finds that the next most suitable course would be E and agent 2 finds the course A (underlined). Just both making the change would help neither party. Also, if only one of the two makes a change, one is left without a key course. So, Agent 1 makes a suggestion. It is ready to give up course D in favor of E, if Agent 2 changes course F to A.

Agent 1	Agent 2
- course A, 0.91	- course A, 0.45
- course B, 0.74	**- course B, 0.78**
- course C, 0.75	**- course C, 0.75**
- course D, 0.45	- course D, 0.34
- course E, 0.44	**- course E, 0.94**
- course F, 0.31	**- course F, 0.46**
- course G, 0.22	- course G, 0.24
- course H, 0.11	- course H, 0.31

Since neither party has a strong bias between D and E or A and F, both might benefit if a change is made. The agents might need to persuade others to change their study plans too, but they have moved closer to their goal

Agent 1	Agent 2
- course A, 0.91	**- course A, 0.45**
- course B, 0.74	**- course B, 0.78**
- course C, 0.75	**- course C, 0.75**
- course D, 0.45	- course D, 0.34
- course E, 0.44	**- course E, 0.94**
- course F, 0.31	- course F, 0.46
- course G, 0.22	- course G, 0.24
- course H, 0.11	- course H, 0.31

Figure 8: Example of a negotiation between agents

Currently the problem of how to weight the risk of not being able to attend a course is unsolved. Perhaps this should be left to the student to decide or the system could learn to find the best way to go from experience. This learning might take years before it could reach a reasonable level of functionality and there isn't necessarily that much time.

5 Testing the Forming of the Study Plan

In the following tests the emphasis was on the learning capabilities of the system. As noted in section 2.4, learning is change in the system which results in better performance. Based on this, the test environment keeps records of changes made by the students. All changes can be regarded as wrong decisions by the system. If the number of those wrong decisions lowers over time, the system is able to learn. The tests were conducted with the software engineering students of University of Vaasa. First of the 20 cases was done by beforehand to give the system something to base its scheduling on. The other 19 were students ranging from first year to sixth year students.

In the beginning of the tests, it became obvious that the students were too willing to accept what the system chose as their courses. Only two of the students made any changes to the course selection and both only added courses to their selection. The small number of changes can probably be explained with the fact that the curriculum of these particular students has gone through very radical changes in the recent years. Because of this, the curriculum used for the tests might not have been the optimal choice.

student	course changes	schedule changes
1	0	39
2	0	21
3	0	21
4	4	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	26
10	0	14
11	0	15
12	0	8
13	2	7
14	0	8
15	0	11
16	0	9
17	0	9
18	0	4
19	0	7
20	0	5

Figure 9: Learning of the scheduling agent

The first two students both made 21 changes to their schedules. After that the system had adopted much of the information and next five students made no changes. Student number 9 chose a different amount of years for his study plan, which resulted in a need for 26 changes, which is the maximum of the series.

Because of student number 9, many of the learned values were radically changed, but the system seemed to be able to recuperate, although every student after that made changes. After long use, such radical changes should not happen. The progress of the learning can be seen in Figure 10:

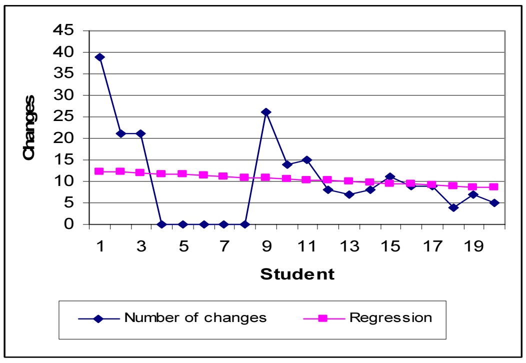


Figure 10: Learning of the scheduling agent represented as a graph

6 Conclusions

Finnish and EU students are going to need more guidance in the future. Obvious solution would be to hire more counselors, but as we have shown above, this is not the only option. In fact, our system could go beyond the capabilities of the counselor, if the whole system is used for decision making by the departments.

In this case, the system could be a usable tool for all parties: it can help the students plan their studies better which is also the goal of the public administration, and the departments can base their teaching plans on concrete and practical improvements.

The system is currently still under technical development [6]. The system can already identify the suitable courses and construct a timetable with good accuracy [5]. The solution is generic and can be used in a number of environments.

Acknowledgement

We wish to acknowledge all parties active in the development and testing of this system, all the people who have given their feedback on the system, the test subjects and especially the original developers of Wompat.

References

[1] Korkeakoulujen opintoaikojen lyhentämisen toimenpideohjelma (2003). Finnish Ministry of Education. In Finnish.

[2] OVI – ohjausta virtuaalisesti (2005). http://ovi.joensuu.fi. University of Joensuu. In Finnish.

[3] Oodi (2005). http://www.oodi.fi. University of Helsinki. In Finnish.

[4] Wompat-system. http://wompat.uwasa.fi. In Finnish.

[5] A. Vainio (2005). Opiskelijan opintojen suunnittelun avustaminen Wompat-järjestelmällä. In Finnish.

[6] Necora Systems Oy. http://www.necora.fi.

[7] S. Russell & P. Norvig (2003). Artificial Intelligence A Modern Approach, 2nd edition. Prentice Hall.

[8] J. Celko (2004). Trees and Hierarchies in SQL. Morgan Kaufmann Publishers.

[9] G. Weiss (ed.) (1999). Multiagent Systems A Modern Approach to Distributed Artificial Intelligence. The MIT Press.

[10] P. Mars, J.R. Chen & R. Nambiar (1996). Learning Algorithms – Theory and Applications in Signal Processing, Control and Communications. CRC Press.

[11] T. Cormen, C. Leiserson & R. Rivest (1990). Introduction to Algorithms. The MIT Press.

[12] The Bologna Process – Next stop Bergen. http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html. 2005.

From Basic Agent Behavior to Strategic Patterns in a Robotic Soccer Domain

Andraž Bežek and Matjaž Gams
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
Andraz.bezek@ijs.si

Keywords: MAS modeling, strategy learning

Received: June 17, 2005

The paper presents an algorithm for multi-agent strategic modeling (MASM). The method applies domain knowledge and transforms sequences of basic multi-agent actions into a set of strategic action descriptions in the form of graph paths, agent actions, roles and corresponding rules. The rules, constructed by machine learning, enrich the graphical strategic patterns, which are presented in the form of graph paths. The method was evaluated on the RoboCup Soccer Server Internet League data. Tests showed that the constructed rules successfully captured some decisive offensive moves and some major defense flaws, although the description itself was a bit awkward and needed interpretation by a human expert.
Povzetek: Predstavljen je sistem, ki si uči strateških vzorcev obnašanja iz enostavnega opazovanja gibanja agentov v domeni robotskega nogometa.

1 Introduction

Multi-agent game modeling is related to the following task: How can external observation of multi-agent systems be used to analyze, model, and direct agent behavior? Analysis of such systems must capture complex world state representation and asynchronous agent activities. From pure numerical data researchers tend to construct complex knowledge-level structures, typically in the form of rules or decision trees. These high-level structures are useful when characterizing state space, but lack the ability to clearly represent temporal state changes occurred by agent actions. Comprehending simultaneous agent actions and complex changes of state space represents another problem. To capture such qualitative information, most often a graphical representation performs better in terms of human understanding.

There were two major goals in the research presented in this paper. One was designing strategic patterns from basic agent behavior, and the second one was to present the constructed knowledge in a graphical and symbolic form. This paper therefore addresses the problem of graphical and symbolic representation of strategic patterns, describes an algorithm capable of discovering strategic agent behavior, and enabling humans to understand and study the underlying behavioral principles.

The presented MASM algorithm translates multi-agent action sequence and observations of a dynamic, complex and multivariate world state into a graph-based and rule-based strategic representation. By using hierarchically ordered domain knowledge the algorithm is able to generate strategic descriptions and corresponding rules at different levels of abstraction. The MASM scheme is presented in Figure 1.

Our approach is applied on a RoboCup Simulated League domain (Noda et. al 1997, RoboCup 2004), a multi-agent domain where two teams of 11 agents play simulated soccer games. The domain accurately simulates a physical 2D soccer but introduces uncertainty by adding noise when calculating forces on objects. Continuous time is approximated with discrete cycles. All agents can move and act independently as long as they comply with soccer rules. Agents communicate with each other, but their visual and hearing perception is distance-limited. The domain is quite complex and represents a challenging multi-agent modeling task for computers, but its soccer-related content makes it comprehensible by humans familiar with soccer.

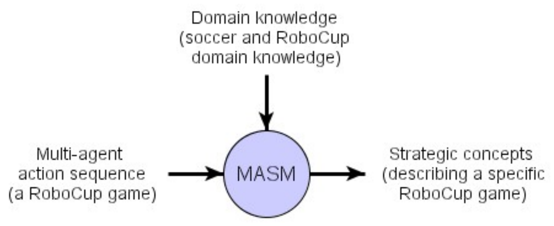

Domain knowledge (soccer and RoboCup domain knowledge)
Multi-agent action sequence (a RoboCup game)
MASM
Strategic concepts (describing a specific RoboCup game)

Figure 1: Multiagent System Modeling.

This paper is organized as follows. Section 2 thoroughly presents the MASM algorithm for creating graphical strategic paths. In Section 3, the learning algorithm is described for constructing symbolic strategic descriptions. An evaluation of the described method is presented in Section 4, and conclusion in Section 5.

2 Multi-Agent Strategic Modeling – the Graphical Part

Our multi-agent strategic modeling (MASM) algorithm transforms raw multi-agent action sequence into a set of discovered strategic action descriptions with corresponding rules. A strategic action description is a description of an agent behavior that exhibits some strategic activity. A strategic activity is a time-limited multi-agent activity that exhibits some important or unique domain-dependent characteristics.

Our approach is based on two basic processes:

1. Construction of graphical sequences of actions.
2. Learning symbolic rules.

The process in terms of creating increased higher-level structures is presented in Figure 2.

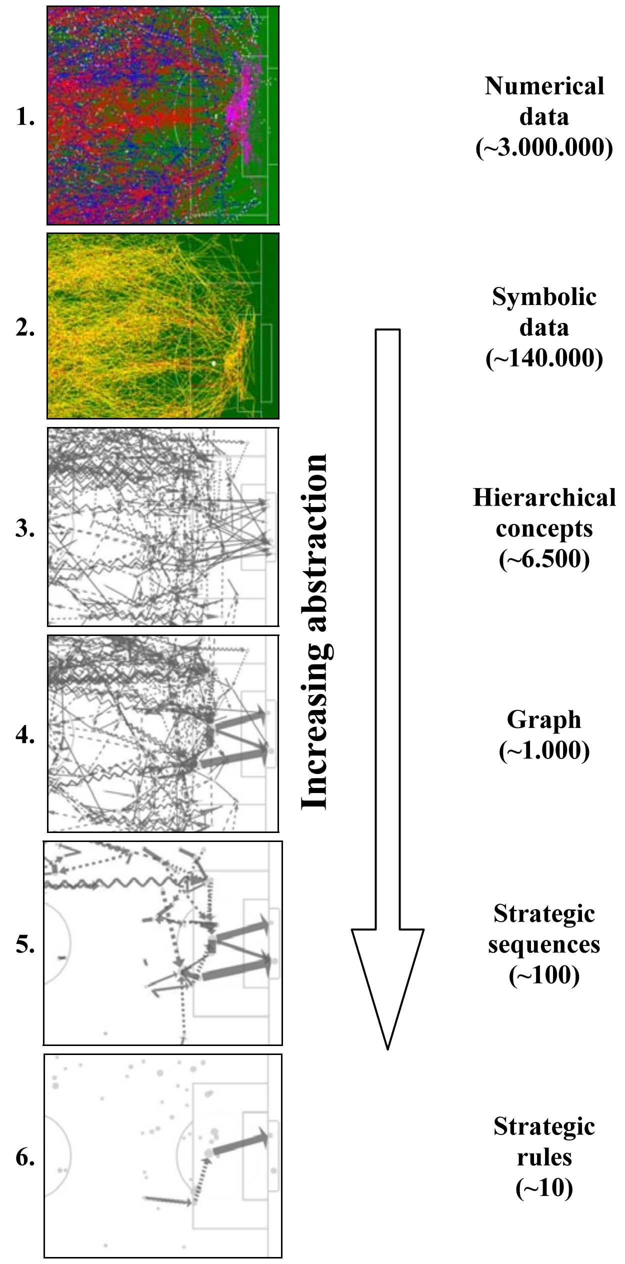
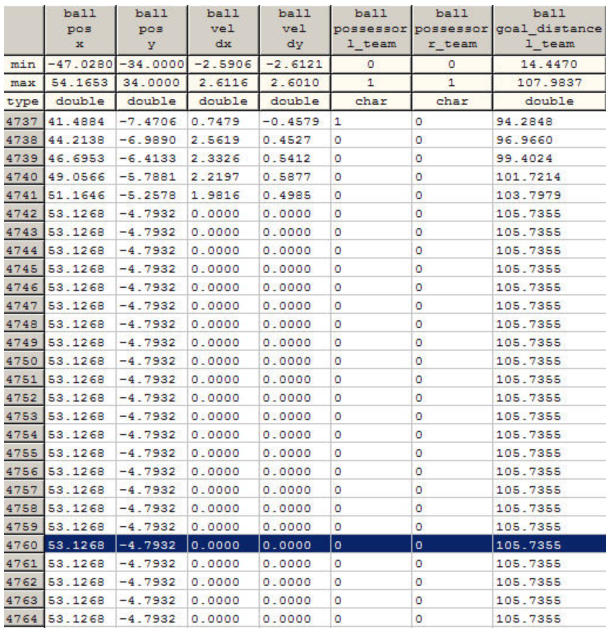


Figure 2: Construction of strategic patterns.

First 5 steps, presented in Figure 2, are going to be described in detail in this section and rule construction in the next section.

At the lowest level, RoboCup games are presented as time frames of agent and ball movements. For further information see (Cheny at al. 2003, RoboCup 2004).



	ball pos x	ball pos y	ball vel dx	ball vel dy	ball possessor l_team	ball possessor r_team	ball goal_distance l_team
min	-47.0280	-34.0000	-2.5906	-2.6121	0	0	14.4470
max	54.1653	34.0000	2.6116	2.6010	1	1	107.9837
type	double	double	double	double	char	char	double
4737	41.4884	-7.4706	0.7479	-0.4579	1	0	94.2848
4738	44.2138	-6.9890	2.5619	0.4527	0	0	96.9660
4739	46.6953	-6.4133	2.3326	0.5412	0	0	99.4024
4740	49.0566	-5.7881	2.2197	0.5877	0	0	101.7214
4741	51.1646	-5.2578	1.9816	0.4985	0	0	103.7979
4742	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4743	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4744	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4745	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4746	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4747	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4748	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4749	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4750	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4751	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4752	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4753	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4754	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4755	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4756	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4757	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4758	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4759	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4760	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4761	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4762	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4763	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355
4764	53.1268	-4.7932	0.0000	0.0000	0	0	105.7355

Figure 3: Raw data in numbers.

In a RoboCup game there are approximately 6000 equidistant time frames and 512 attributes (together around 3,000,000 values of types integer, Boolean, and real). An example of data is presented in Figure 3.

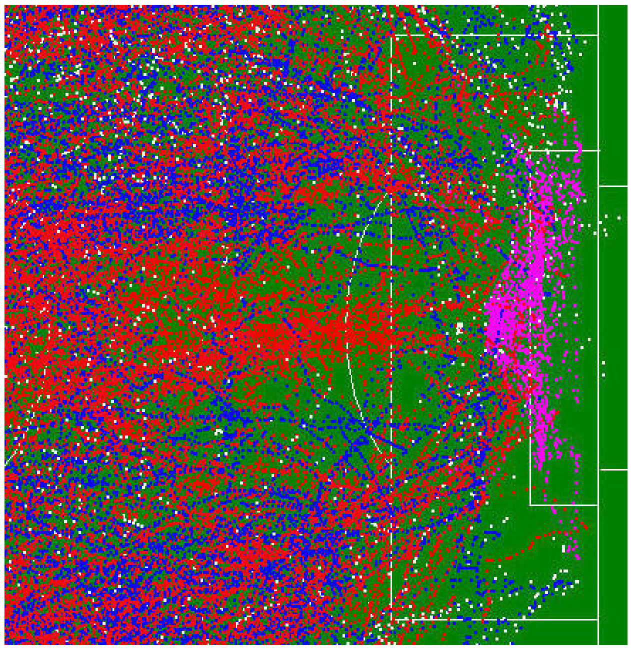


Figure 4: Visualization of raw data.

In Figure 4 there is a graphical presentation, i.e. visualization of the same data as presented in Figure 3. All game data was obtained from the Soccer Server Internet League (Robocup 2004).

In the next step, basic agent movement is transformed into basic agent actions using simple heuristic rules (Nair at al. 2002) such as: "An agent performs action "dash" if it increases speed." Each action lasts one time cycle. From a typical game, around 140.000 basic agent actions are obtained such as:

```
time player → action
--------------------
3192 LPlayer1 → catch
4012 LPlayer3 → kick
5400 RPlayer6 → dash
5900 RPlayer11 → turn
```

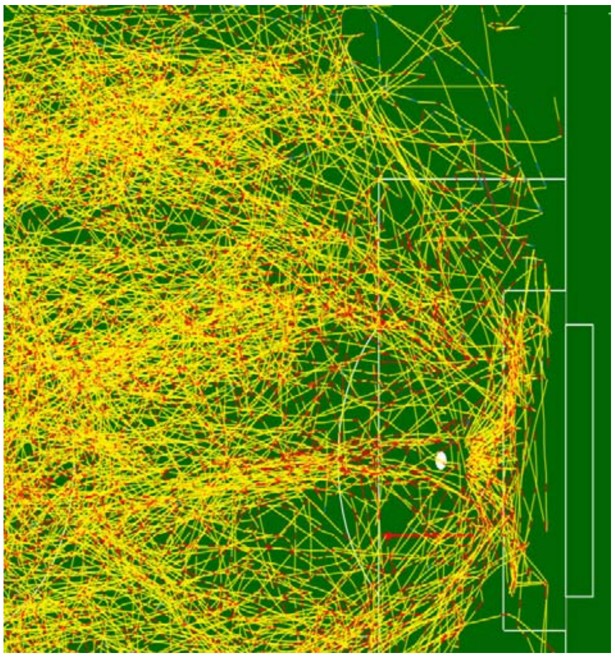


Figure 5: Basic agent actions.

In the next step, basic agent actions are transformed into higher-level actions using domain knowledge (Kaminka at al. 2002, Nair et al. 2003). The MASM algorithm exploits taxonomies, i.e. hierarchical representations of domain concepts. A concept x in a taxonomy is an ancestor of a concept y, $x \leftarrow y$, if it exhibits more general concept than the concept y. The rationale behind using hierarchically ordered domain knowledge is that this allows the MASM algorithm to travel up and down in the hierarchy to produce more or less abstract descriptions. Specifically, the MASM algorithm makes use of:

- a taxonomy of agent roles
- a taxonomy of agent actions
- taxonomies of binary domain features.

Taxonomies were created from the Internet, using Dictionary Of Soccer Terms, Concepts & Rules. Parts of these taxonomies are presented in Figure 6, determining agent roles and actions.

As agents in MAS can change roles and thus change their behavior (Nair et al. 2003), agent roles are assigned dynamically during agent activity. Each agent action is assigned with its corresponding hierarchical representation. Domain features are used to identify the truth of some particular domain feature but only with an association with another agent. For example, in a RoboCup domain the feature *HasBall* is true only for agent which controls the ball, and is false for all other agents. Agent's roles and actions are used to describe the activity of agents, while domain features are used to describe the domain state space. An example assignment of soccer role and action concepts is presented in Figure 6, some examples are presented below:

```
time player_role → action, action_duration
----------------------------------------------
3192 LTeam.Goalkeeper → catch, 1
4012 LTeam. LeftForward → pass_to_player, 10
5400 RTeamRightFullback → speed_dribble, 12
5900 RTeam.LeftMidfielder → intercept, 8
```

Around 1000 such high-level agent actions are constructed for a typical game.

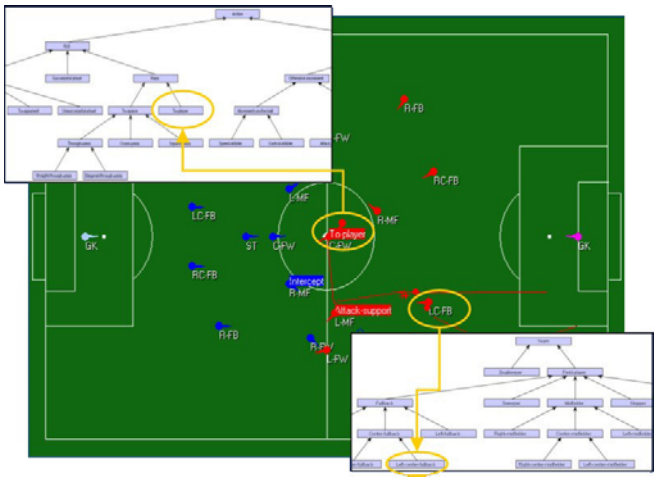


Figure 6: High-level agent actions.

After agent-role assignment, an *action graph* (AG) is constructed with the goal to create action patterns in a graphical form of paths. An action graph is a directed graph, where nodes represent state space at the start of agent action and connections correspond to agent actions. Nodes a and b are connected, $a \rightarrow b$, if an action, represented by node a, is followed by an action, represented by node b. Terminal actions (i.e. the last action in an action sequence) are connected to a terminal node. For example, an action sequence {a,b,c} is represented as an AG: $a \rightarrow b \rightarrow c \rightarrow c_{end}$. Node positions are calculated from agent positions in a domain space. An appropriate hierarchical action and role concepts are assigned to each node (Bezek 2005). This enables the MASM algorithm to generate more abstract descriptions of agent role and actions. Each node also keeps an original action instance (i.e. time cycle of an action in a soccer game) of the represented action. A more detailed description of action graphs and the construction process is given in (Bezek 2004). An example action graph, obtained from actions in a RoboCup game, is presented in Figure 7.

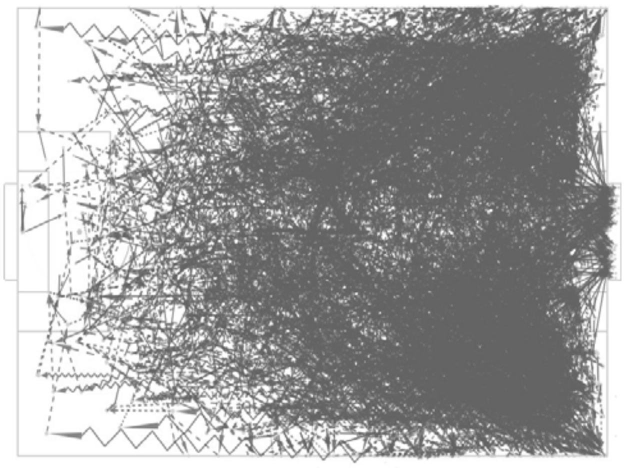


Figure 7: An action graph.

Complex action graphs with many nodes (see Figure 7) are difficult to present in a transparent manner and are thus difficult to comprehend by humans. A reasonable approach to overcome this problem is to reduce the number of nodes while at the same time preserving attained action concepts. This can be accomplished with hierarchical clustering of graph nodes. By merging the nearest two graph nodes, we generate a new node that represents common role and action concepts (i.e. first hierarchically common parent of both concepts). The rationale behind the merge process is that actions, frequently occurring near in a domain space, define strategic concepts. The distance between graph nodes is defined as a weighted sum of distances between node positions and conceptual distances between role and action concepts. The merging process is then iterated. A more in-depth description of the distance function and the whole abstraction process is described in (Bezek, Gams 2005).

The clustering process results in an *abstract action graph* (AAG), which is an action graph where graph nodes represent more than one agent action. It is expected that abstract action graph describes agent behavior in a more abstract way than the original action graph. An abstract action graph, where minimal distance between nodes is greater than *dist*, is labeled AAG_{dist}. Such graph can be achieved with repeated merging of nearest nodes until the minimal distance between nodes is grater than *dist*. An *action description* of a node in AAG_{dist} is a combination of a node position, corresponding action and role concepts, and a parameter *dist*. AAGs with greater value *dist* represent actions in a more abstract way that AAGs with a lower *dist* value. Therefore, the value of a *dist* parameter can be regarded as a value of abstraction of an AAG. An example of an abstract action graph A_{10} is presented in Figure 8.

In Figure 8 there are several connected arrows of different length, positions and thickness. One example of transformation from single actions into an aggregated one is shown in Figure 9. It represents a common and successful attack on the right side of the field, resulting in a successful shoot on a goal. It is an example of a desired graphical representation of a strategic pattern.

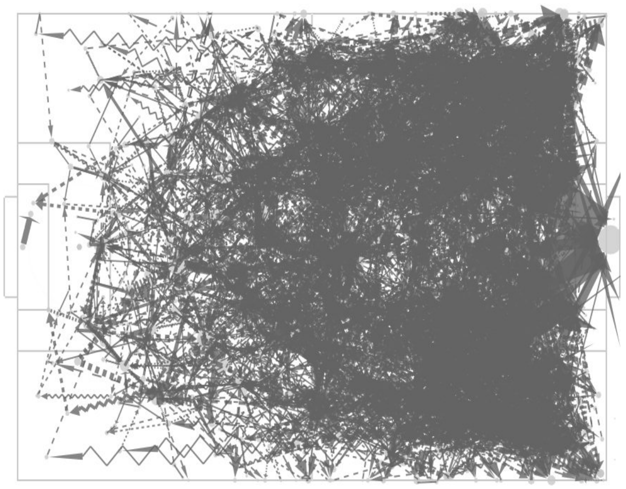


Figure 8: An abstract action graph (AAG_{10}).

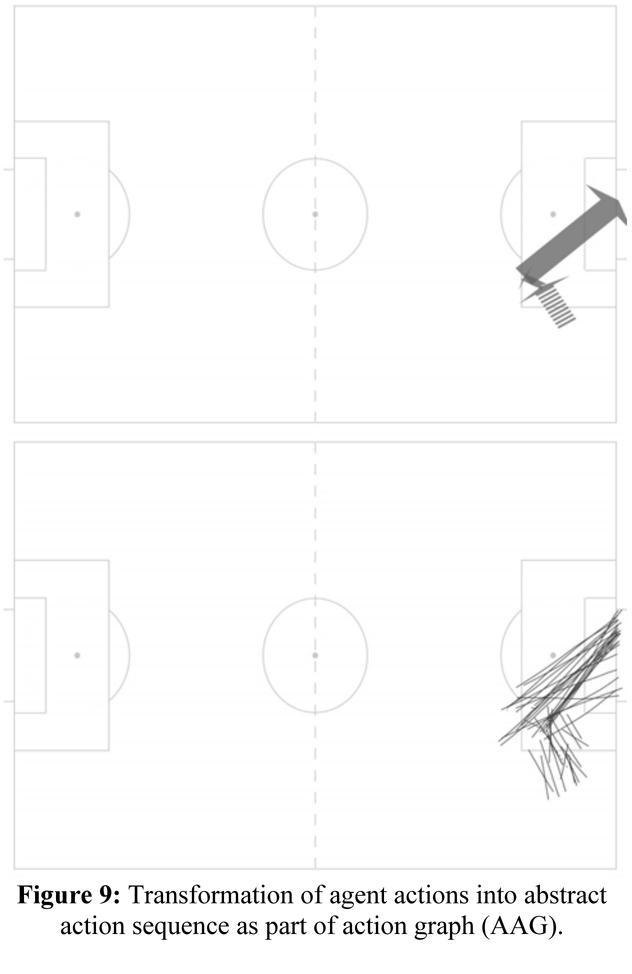


Figure 9: Transformation of agent actions into abstract action sequence as part of action graph (AAG).

This strategic abstraction of agent actions is based on clustering (Hirano et al. 2004, Riley at al. 2001), and on conceptual distance, based on the domain taxonomies. As a result, around 1000 of such structures are generated from a typical game:

```
role → action: {(action_start, duration)+ }
-----------------------------------
LTeam.Goalkeeper → catch:
{(412, 1), (501, 1), (3192,1)}
LTeam.Forward → pass:
{(1412, 5), (3401,12), (4012,10) , (5573,7)}
RTeam.Defender → speed_dribble:
{(1607,16), (2372,9), (5400,12), (5521,22)}
RTeam.Midfielder → intercept:
{(392, 4), (4509, 9), (5900, 8)}
```

A list of subsequent actions with corresponding symbolic description represents a *strategy,* i.e. a similar and frequent multi-agent activity that leads to a strategic situation. In an abstract action graph it is represented as a path. Path nodes thus represent a sequence of strategic actions.

What remains is construction of rules, as indicated by step 6 in Figure 2.

3 Construction of Rules

First it should be noticed that strategies vary in several parameters, such as number of actions (typical 2 to 4), abstractness of actions (corresponding to the number of single actions aggregated into one abstracted action), location, direction etc. In general, a strategy generated from AAG with a greater *dist* value is more abstract that the one generated from AAG with lower *dist* value. The strategy in Figure 9 was created using level of abstractness 8 (=*dist*). The strategic action sequence is presented in Table 1. From Table 1 and Figure 9 the strategic action sequence can be described as (no. of positive examples in parentheses):

Forward player **passes** a ball to a teammate (21 +),
who successfully **dribbles** (10 +), and
shoots towards a goal (23 +).
As a result, the ball ends in a goal (23 +)."

LTeam.FW:	LTeam.FW:	LTeam.FW:	LTeam.Field-player:
Pass-to-player	Control-dribble	Successful-shoot	Successful-shoot-(end)

Table 1: A strategic action sequence.

The action sequence in Table 1 is graphically presented as the path consisting of three connected arrows in Figures 9 and 10. Each action (an arrow) graphically starts from a circle (Figure 10) which corresponds to the neighborhood including aggregated actions. All circles in Figure 10 correspond to all aggregating neighborhoods.

As each node/circle defines a unique action concept it can be used to generate rules that describe this specific agent action. In particular, we generate data for rule inducing algorithms as follows: Positive examples are action instances in a target node and negative examples as instances in nearby nodes (i.e. near misses). For each instance we generate all pairs of agent role-domain feature and store the true ones.

We tested several approaches with association rules (Agrawal et al. 1994, Srikant et al. 1995), but due to complex representations we found the standard feature-value approach as not suitable. Namely, agents dynamically change roles and thus it is very difficult to generate feature values for all roles. Therefore, instead of feature-values we applied set-valued attributes that are attributes whose domains are sets instead of single values. In this way, each feature corresponds to one set-valued attribute where the value is a set of agent roles, whose corresponding agent-feature pair is true.

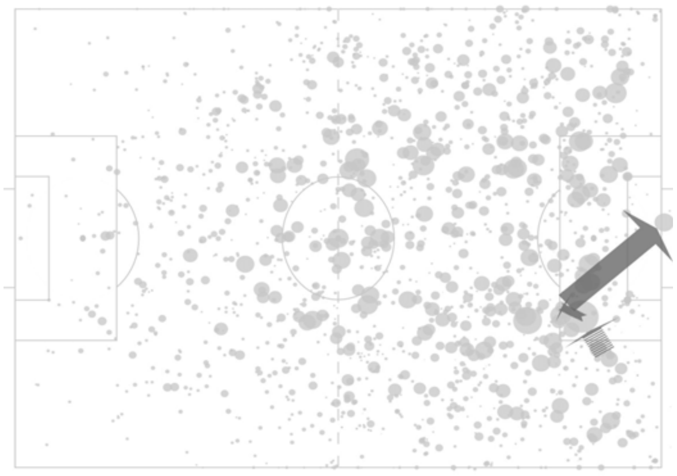


Figure 10: Strategy as a path in the abstract action graph, and all potential learning examples as circles.

By using set-valued rule inducer, such as SLIPPER (Cohen et al. 1999), the MASM algorithm is able to generate rules that describe actions in a strategy. In a typical experiment, 10 games of the same team were taken as input, and SLIPPER was applied on each node in a strategic path.

For example, a rule describing a node, which represents an action concept "Successful-shoot" performed by an agent with a role "left team center midfielder", is presented in Table 2.

ball:Penalty-box ∧ ball:Right-half ∧ ball:Fast ∧ LTeam.C-MF:Has-ball ∧ LTeam.R-FW:Moving-away ∧ LTeam.R-FW:Medium-dist ∧ RTeam.R-FB:Back ∧ RTeam.C-FB:Back ∧ RTeam.L-FB:Back.

Table 2: Symbolic description of a successful shoot by a left team's center midfielder.

There are several parameters that influence the learning algorithm, and the influence of distance is indicated in the following examples. The distance parameter corresponds to the number of learning examples. Since it seems reasonable to include all positive examples, because there are typically only around 10 or 20 of them (note that these are strategic patterns that actually occur in a game), the parameter varies the number of negative examples.

- All negative examples:

LTeam.FW:Pass-to-player (#+21 #-6987) <=
(LTeam.R-FW:Has-ball = 1) AND (LTeam.L-FB:Incoming-slow = 1) AND (RTeam.GK:Incoming = 1)
(*there are 21 positive examples and 6987 negative*)
LTeam.FW:Control-dribble (#+10 #-6998) <=
(LTeam.R-MF:Near = 1) AND (LTeam.L-MF:Attacking-third = 1) AND (LTeam.C-FW:Center-of-the-field = 1)
LTeam.FW:Successful-shoot (#+23 #-6985) <=
(LTeam.LC-FB:Moving-away-slow = 1) AND (RTeam.R-MF:Attacking-third = 1) AND (LTeam.R-MF:Fast = 1) AND (RTeam.R-FB:Far = 1) AND (RTeam.GK:Faster = 1) AND (RTeam.L-FW:Moving-away = 1) AND (LTeam.C-FW:Medium-distance = 1) AND (RTeam.C-MF:Fast = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-6787) <=
(Ball:Opponent-goal = 1)

- Only negative examples with distance <= 16

LTeam.FW: Pass-to-player (#+21 #-854) <=
(LTeam.C-FW:Incoming-fast = 1) (* no. of negative examples here is 845*)
LTeam.FW:Control-dribble (#+10 #-1425) <=
(LTeam.R-MF:Near = 1)
LTeam.FW:Successful-shoot (#+23 #-1447) <=
(LTeam.R-FB:Moving-away = 1) AND (LTeam.R-FW:Moving-away = 1) AND (RTeam.R-FB:Far = 1) AND (LTeam.R-MF:Attacking-third = 1) AND (RTeam.GK:Incoming = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-213) <=
(RTeam.GK:Right-half = 1) AND (RTeam.GK:Back = 1) AND (LTeam.L-MF:Center-of-the-field = 1) AND (LTeam.L-FW:Medium-distance = 1)

- Only negative examples with distance <= 8

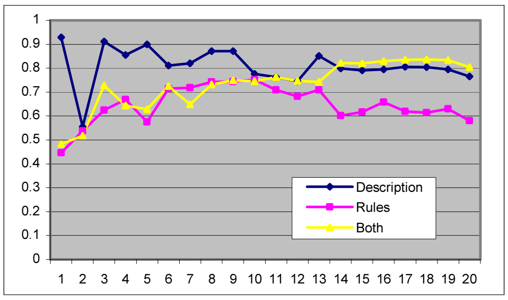
LTeam.FW: Pass-to-player (#+21 #-265) <=
(LTeam.C-FW:Incoming-fast = 1) (* no. of negative examples here is 845*)
LTeam.FW:Control-dribble (#+10 #-513) <=
(LTeam.RC-FB:Fast = 1)
LTeam.FW:Successful-shoot (#+23 #-573) <=
(RTeam.L-FW:Moving-away = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-113) <=
(RTeam.GK:Right-half = 1) AND (LTeam.LC-FB:Right = 1)

- Only negative examples with distance <= 4

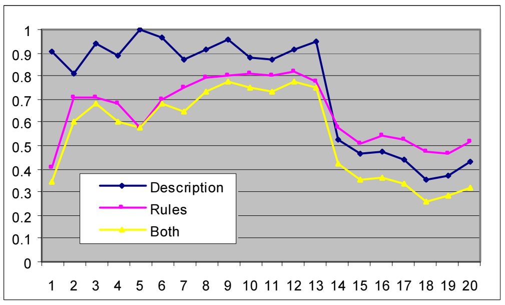
LTeam.FW: Pass-to-player (#+21 #-105) <=
(RTeam.R-FB:Incoming-fast = 1) (* no. of negative examples here is 105*)
LTeam.FW:Control-dribble (#+10 #-239) <=
(RTeam.GK:Right = 1)
LTeam.FW:Successful-shoot (#+23 #-200) <=
(LTeam.R-FB:Moving-away-slow = 1) AND (LTeam.R-FW:Moving-away = 1)
LTeam.Field-player:Successful-shoot-END (#+23 #-112) <=
(LTeam.R-MF:Right-wing = 1) AND (RTeam.GK:Right = 1) AND (LTeam.L-MF:Left-half = 1) AND (RTeam.GK:Short-distance = 1)

4 Measurements

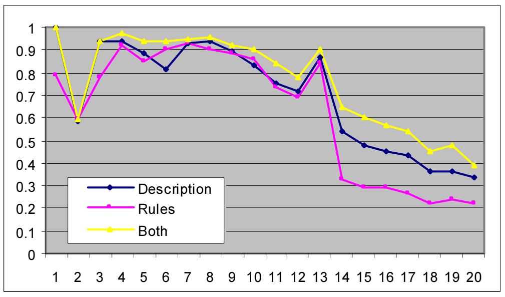
We evaluated the MASM approach on 10 RoboCup games played during SSIL (Robocup 2004). A leave-one-out strategy was used to generate 10 learning tasks. A pre-determined strategy, shown in Table 1 and in Figure 10, was used as a reference and was generated on all 10 games, for AAG_1 to AAG_{20}. For each learning task, a strategy was generated on 9 games and tested on the remaining game, again for AAG_1 to AAG_{20}. Tests measured the quality of action descriptions, the quality of an average rule and the quality of joint use of rules and action descriptions. Figure 11 presents averaged results obtained during 10 tests where x-axis presents the value of a parameter *dist*. These results indicate that a) the accuracy of action descriptions is approximately constant regarding abstraction. However, the accuracy of rules increases until it peaks at dist=10 and then slowly decreases. This is expected because for lower abstractions, nodes represent only a few action instances consequently prohibiting rule inducer to generate good rules.



a) Accuracy



b) TP rate



c) Precision

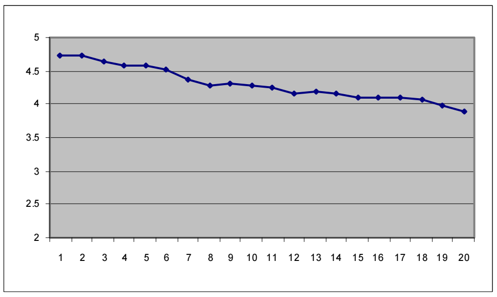

d) Abstractness

Figure 11: Accuracy a), true-positive rate b) and precision c) measured in relation to the abstractness level presented on x-axis. Abstractness of attributes is in d).

With high *dist* values, nodes represent different action concepts, thus producing more abstract and less accurate rules. But using rules and action descriptions together gives the best results with higher dist values.

When measuring true-positive rate, i.e. the percentage of correctly classified true cases, all test scenarios give similar results as shown in Figure 11 b): the quality of classifying true cases increases until about dist=12, and then quickly drops. The similar phenomenon is observed when measuring precision, that is the rate of correctly classifying the true cases, shown in Figure 11, c). This can be explained by generating too abstract strategies that represent the agent behavior in a too abstract way.

The last test was performed to verify if the abstraction process generates more abstract descriptions. For this test we measured the abstraction of generated rules, defined as an average feature depth in the feature taxonomy for features used in rules. The results, presented in Figure 11 d), clearly show that the average feature depth is negatively correlated with the parameter *dist*. This proves that as *dist* value increases, the rules contain more abstract features.

The constructed strategic patterns were also examined by the research team and a human expert. We studied the games on the screen in real time and the constructed strategic patterns. Firstly, we realized that the direct computer output was unintelligible for a non-computer specialist. Second, the constructed computer output had to be studied also by the research team since quite often the meaning of constructed features had to be figured out. For example, instead of a meaningful "fast ball" the actually constructed feature was "distance between a ball and a player is growing fast". Another annoying property of the learning algorithm was that sometimes quite irrelevant features were constructed, at least from the point of human understanding. But in our joint overall opinion, the algorithm finds some significant features (moves), which is quite a success since the algorithm has no knowledge whatsoever about rules of soccer or any predefined knowledge about strategies, i.e. a list of potential soccer strategies.

5 Conclusion

We have designed and implemented the MASM algorithm as a general domain-independent framework for discovering strategic behavior of multi-agent systems. The only domain-specific knowledge was introduced in the form of role, action and domain feature taxonomies. We assume that changing a domain should be a straightforward task that would require changing specific domain-knowledge in a similar form. We believe that there is a wide range of possible domains that can be exploited by the MASM since its essence is a stepwise abstraction in the domain-space.

The tests show that the system with 30.000 source code lines achieves reasonable results in terms of accuracy, true-positive rate and precision. Our tests also confirm that the increased abstraction process generates more abstract descriptions of agent activities.

However, there are some open questions that need to be addressed. First, the MASM system was evaluated only on the RoboCup domain with a limited number of tests. Although authors believe that no major problem should emerge when introducing another domain, this should be verified in practice. Second, while the output of the MASM system seemed promising to the research team and the soccer coach performing preliminary evaluation, this should be systematically verified by a number of unrelated humans and soccer experts. The third open question is how to objectively specify relevant strategic situations.

Overall, the MASM algorithm was able to create human comprehendible strategic descriptions in the form of graphical arrows and related strategic rules with reasonable accuracy from basic agent observations in a RoboCup games. This seems quite promising since the system had only limited domain knowledge.

References

[1] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules. Proc. 20th Int. Conf. Very Large Data Bases, VLDB, 1994.

[2] A. Bezek: Modeling Multiagent Games Using Action Graphs. Proceedings of Modeling Other Agents from Observations (MOO 2004), New York, 2004.

[3] A. Bezek: Discovering Strategic Multi-Agent Behavior in a Robotic Soccer Domain, Proceedings of AAMAS 05, Utrecht, 2005.

[4] A. Bezek, and M. Gams: Discovering strategic multi-agent behavior in a robotic soccer domain. Proceedings of Information Society 2005, Ljubljana, 2005

[5] M. Cheny et al.: Users Manual for RoboCup Soccer Server, 2003.

[6] W. W. Cohen and Y. Singer: A simple, fast, and effective rule learner. Proceedings of the sixteenth national conference on Artificial intelligence, pp.: 335 - 342, Orlando, United States, 1999.

[7] Dictionary Of Soccer Terms, Concepts & Rules, http://www.soccerhelp.com/Soccer_Tips_Dictionary_Terms.shtml.

[8] S. Hirano and S. Tsumoto: Finding Interesting Pass Patterns from Soccer Game Records. The Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004).

[9] I. Noda et. al: Overview of RoboCup-97. In Hiroaki Kitano, editor, RoboCup-97: Robot Soccer World Cup I, pp. 20-41. Springer Verlag, 1997.

[10] G. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso: Learning the sequential coordinated behavior of teams from observations. Proceedings of the RoboCup-2002 Symposium, June, 2002.

[11] R. Nair, M. Tambe, S. Marsella and R. Raines: Automated assistants for analyzing team team behaviors Journal of Autonomous Agents and Multiagent Systems. JAAMAS, 2002.

[12] R. Nair, M. Tambe, and S. Marsella: Role allocation and reallocation in multiagent teams: Towards a practical analysis. Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS), 2003.

[13] P. Riley and M. Veloso: Coaching a Simulated Soccer Team by Opponent Model Recognition. Proceedings of the Fifth International Conference on Autonomous Agents, 2001.

[14] RoboCup 2004: RoboCup Simulation League. RoboCup04 world cup game repository, http://carol.science.uva.nl/~jellekok/robocup/rc04/, 2004.

[15] R. Srikant and R. Agrawal: Mining Generalized Association Rules. Future Generation Computer Systems, 1995.

A Composite Design-Pattern Identification Technique

Marjan Heričko and Simon Beloglavec
University of Maribor, Institute of Informatics,
Smetanova ulica 17, 2000 Maribor, Slovenia
E-mail: marjan.hericko@uni-mb.si, simon.beloglavec@uni-mb.si

Keywords: Design patterns, design metrics, design patterns identification, composite design patterns

Received: June 17, 2005

This paper introduces a new technique for identifying composite design patterns from existing pattern-based designs. We propose two pattern metrics: pattern coverage and overlapping that can help detect a composite pattern. The effective composite patterns reflect quality properties that are considered desirable in the solution for a given problem domain and selected programming paradigm. To identify appropriate candidates, we propose an assessment with a set of design metrics in addition to pattern metrics. The calibration of value intervals for metric scores is proposed with the intention of offering the designer the possibility of adjusting the technique for each individual type of software. In this paper, we present the steps required for detecting and identifying the suitable composite pattern candidates through pattern and design metric assessment.

Povzetek: Prispevek predstavlja nov pristop z novimi metrikami vzorcev k identifikaciji sestavljenih načrtovalskih vzorcev v obstoječih načrtih informacijskih sistemov.

1 Introduction

The typical software design rarely includes an independent pattern; increasingly, applied patterns are interconnected. A design pattern (henceforth "pattern") can be applied to various structural forms. A set of applied patterns, in selected forms, can promote in the existing designs desired quality characteristics. What qualifies as an appropriate design quality depends on the type of software that has been developed (e.g. local component, distributed component, programming library, etc.). Therefore, in some cases a set of patterns proves to be an efficient solution while in other cases it results in unwanted design complexity. The designer's goal in a pattern-based design is the application of an effective pattern combination. The proven solutions of pattern applications can be identified from existing designs.

We propose a composite pattern identification technique that consists of three main steps. The first step towards the identification of suitable composite patterns is the construction of the pattern coverage matrix for the selected design. The matrix holds information over the selected pattern instantiation form. The instantiated form is one of the allowable forms of a pattern that includes all allowed structural and behavioural variations for the selected pattern. The information over the instantiated pattern form captured in the pattern coverage matrix contains a detailed description over the selected structural and behavioural variations that are applied in a design. The constructed matrix is then assessed with the pattern coverage metric that is defined in this paper. The goal of the assessment is the identification of design fragments that are covered with patterns. During the second step, we construct a pattern overlapping matrix based on the pattern coverage matrix. In this paper, we define a pattern overlapping metric that is intended for detecting various levels of overlapping. This step extracts the set of composite patterns candidates that is assessed with design metrics in the final step. The final assessment uses a set of design metrics that exposes flaws in the design when considering quality attributes valid for a given solution domain and the selected programming paradigm. The result of the final stage is a small subset of new composite patterns or an individual composite pattern. A possible outcome is also an empty acceptable set from the set of extracted pattern candidates. Identified composite patterns act in future applications equal as atomic patterns.

The application of the technique is presented in two design cases where composite patterns are identified. The paper demonstrates how the proposed technique applied on the first simplified design detects the well-known composite pattern (MVC- Model-View-Controller pattern) from an existing design. The second example demonstrates the technique’s application through a complex design where the calibration of value intervals for metric scores is presented in detail and a new composite pattern is extracted.

The rest of the paper is structured as follows: In Section 2, relevant background and related works are discussed. Section 3 contains the steps of the technique and defines the proposed pattern metrics for coverage and overlapping. Section 4 demonstrates the application of the technique through the identification of the MVC pattern from a design. An approach to the calibration of value intervals for design metric scores is discussed in Section 5. Section 6 gives some conclusions and ends with a discussion of the research findings.

2 Background and Related Works

To avoid ambiguity when discussing patterns, it is important that we define the term composite pattern and also define the types of patterns that are suitable for the application of the technique. The composite pattern [19] refers to a composition of patterns that have a common solution space and is not to be mistaken for the design pattern from a fundamental catalogue [1]. Patterns can be classified in many ways: lifecycle stage (requirement, analysis and design patterns) and level of abstraction (idioms, design and architectural patterns). This research focuses on design patterns and makes a clear distinction between an atomic and a composite pattern. Atomic patterns are considered to be the fundamental patterns, which build a pattern language and cannot be broken down into a set of sub-patterns. Composite patterns are a product of pattern integration that go beyond a simple composition that groups patterns without any synergy [16]. The existing research defines composite patterns in various ways. Some researchers consider a composite pattern to be a set of patterns from various architectural levels (analysis, design, implementation) [18], others focus on the dependencies between applied parts of the patterns in the design [16], [17] or on compositions that are discussed in a pattern catalogue level without considering the target design [20]. The fundamental pattern catalogue [6] also defines a set of relations that can be treated as connections in a composition. The presented technique focuses on the patterns applied into a design and considers the pattern overlapping that can be present in specific design parts. Overlapping occurs when an individual design part has a role in two different patterns. Composite patterns that are identifiable with the presented techniques all have constituents as overlapped patterns. The identification technique starts the analysis from the instantiated pattern variant in a design. The specific treatment of pattern overlapping distinguishes the presented approach from other existing attempts at composite pattern identification.

Some of the early attempts at identifying patterns from an existing solution were built exclusively from the structural information that was constructed from a source code. The fundamental presumption in such research has been that pattern extraction is possible without additional information. Many authors ([12], [14], [16] and [22]) use object-oriented software metrics for the purpose of identifying structural GoF patterns [6]. In the case of other pattern types, false positives can occur ([12] and [16]). False positives must be detected and inspected by the user alone. Single class metrics are used to reduce the search space in a structure. In previous research, metrics such as NOA (Number of Attributes) and NOO (Number of Operations) have appeared in various configurations. Metric scores are used for the detection of candidate classes for structural patterns. The similar usage of metrics, for detecting the structure of fundamental patterns, has been tracked by various authors [11], [16].

Patterns can be detected with the help of basic metrics on the class structure. A question has arisen in the past: does the application of patterns have an influence on software quality metric scores? In many cases, patterns promote weak coupling between classes and a greater abstraction if the impact is observed on the level of an individual pattern [7], [21], [9]. A comparison has to be carefully made while also considering various influences (other patterns, external non-pattern classes). The process of detecting composite patterns can return different results, and should be assessed on adjusted score intervals, as shown later in the paper. Design metrics, if applied properly, have proven effective as indicators of flaws and the inappropriate use of patterns in existing designs [23].

The domain and language-independent discovery of patterns is possible with the use of formal specifications, which serve as an independent meta-layer between a specific design and conceptual artefacts. A formal specification language enables the formal definition of the patterns themselves and their application [1][5]. The independence from a design paradigm is not pursued in all research [4]. While in most cases, the analysis of a source code is the leading source of data, some researchers also decided to include the data over behaviour during system run-time [7]. A demanding construction procedure with such specifications prevents researchers from utilizing other approaches. The presented technique does not require such specifications.

3 Proposed Technique

New editions of pattern catalogues have motivated the quest for discovering new design patterns. The expression discovery process can be ambiguous. Some research uses the expression discovery, when actually a recovery of well-known patterns is done. The identification of patterns using the proposed technique results in new composite patterns. In the presented case, we analysed existing solutions where we presumed that proven composite patterns are present. The technique is meant to be applied in cases that have already proven to be successful in the real world. We use the term identification instead of discovery, in order to stress the fact that in presented cases, composite patterns are already present and only need to be identified. Applying the technique enables the designer to select candidates from a design and identify the appropriate ones, considering the positive properties for the selected programming paradigm. The pattern-based design preserves the information on applied patterns (instantiated pattern variants and their locations in a design). The goal of the identification process in all cases is to detect the patterns that can be atomic or composite. Atomic patterns are not a result of composing existing patterns. Early research dealt with the discovery of atomic patterns, which are included in existing catalogues. Finding a new extracted pattern that can be used in future designs, like any other pattern, justifies the invested effort. The application of a composite pattern increases the pattern's usage and protects a designer from the inappropriate application of several patterns. The set of patterns can be applied in a design with many variations, while the composite pattern consists of a proven solution for their application.

A single pattern can appear in different designs in many variations. Pattern catalogues suggest basic forms of a pattern while possible variations are rarely discussed in detail. Some parts of a pattern can be omitted without compromising the mission of a pattern. For example, the pattern Lightweight [1] can in some cases includes the classes that represent the unshared concrete flyweight, while in other cases these classes are omitted. In some cases, the same building element appears in different shapes. For example, the Flyweight pattern itself can be described with an abstract class or with an interface. It is to be expected that the same patterns will have a different cardinality and types of elements. This fact does not directly interfere with the presented technique. This fact should be considered during the construction of the input data for the technique. The use of a standardized template, with fixed elements for each individual pattern, is not adequate in our approach.

The variety of formats tilts many reengineering and assessment projects away from specifying patterns in their design [3], [8], [10], [15]. The information in the applied patterns is a valuable base for further analysis. The purpose of the presented method is not to identify pattern candidates through the structural information that is constructed from the program's source code. A base consists of information on a pattern's variants that are applied in a design. If existing designs preserve information over the applied patterns, we can extract the necessary data to apply the technique. In order to automate the whole procedure, a mapping facility must be constructed that translates the pattern information into the form required by the proposed technique. We avoided building a meta-level specification (formal or informal) in this research. Existing designs, known to authors, use a variety of semi-formal and formal notations for describing applied patterns. The motivation that drove this research was establishing the minimal denominator of the pattern information, where construction is feasible in all known cases.

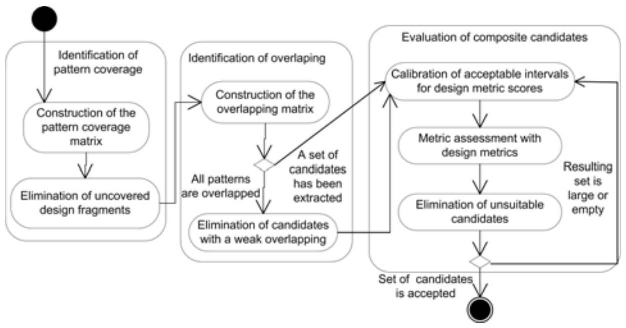

Identification of pattern coverage
Construction of the pattern coverage matrix
Elimination of uncovered design fragments
Identification of overlaping
Construction of the overlapping matrix
All patterns are overlapped
A set of candidates has been extracted
Elimination of candidates with a weak overlapping
Evaluation of composite candidates
Calibration of acceptable intervals for design metric scores
Metric assessment with design metrics
Elimination of unsuitable candidates
Resulting set is large or empty
Set of candidates is accepted

Figure 1: Activities for a composite pattern identification

In order to perform the technique presented in Figure 1, the input data must be prepared in a prescribed manner. For all the patterns used in an observed design, the distinct variants of the pattern application should be identified with all the corresponding parts. We presume that the existing specifications of an analysed design will allow us to identify the pattern parts in the design at the detailed level of methods and attributes. The pattern coverage matrix needs to be constructed in order to perform the remaining steps. The matrix values are calculated as pattern metric scores. The pattern coverage metric is defined in the following chapter. The values in the matrix enable the elimination of uncovered design parts from further analyses. They also constitute the base for detecting the overlapping of applied patterns, as calculated and assessed in the overlapping matrix. Only the parts of the design that are actually covered with patterns should be considered. Other parts are not important in the further identification process. The matrix data on pattern coverage serves for the detection of pattern overlapping. The reasoning behind treating overlapping as a key data in the discovery process is explained in Section 4. In some cases, analyses of the pattern overlapping matrix produces only one composite pattern candidate that includes all patterns, which appear in the design. To avoid the extreme case of accepting a whole design as a pattern, the strength of overlapping should also be inspected. Later in the paper, we define the strength levels for overlapping. Patterns with weak overlapping can be eliminated from the candidate pool. If all patterns are connected with the same strength of overlapping, this combination becomes the only composite pattern candidate. The type of software that is being developed dictates the attributes, which can be expressed through design metric scores. When multiple candidates are present in a set of detected composite patterns, the design metric assessment eliminates the unsuitable candidates. The assessment is also reasonable in cases when there is only one candidate for a composite pattern. The purpose of the assessment is to examine the candidates' suitability with regard to the quality characteristics implied by a solution domain. The technique does not behave as a decision function that result in one candidate only. The number of final candidates depends on the calibration of allowed value intervals for metric scores. The designer's decision is to accept all the positive candidates or only the most appropriate ones considering the metric scores.

Designers try to avoid the realization of the following statement: "Patterns usually lead to an increased number of software artefacts, which normally increases the static complexity of a software system considerably" [23]. A high level of overlap in a pattern prevents the undesired increase of artefacts. Upholding this level forces the designer, with each new pattern application, to integrate a new pattern well into the design.

There is no standardized definition for the "glue" between patterns in a composition. If the connecting glue is presented by the interaction-dependency between the pattern parts of various patterns there are as many candidates to be considered as the composites [16]. If the analysis encompasses the abstraction level of an interface (all public patterns are taken into consideration) or an implementation (all detailed structures are considered), an excessive amount of interaction is to be expected. Observing patterns as a whole in a design, it appears that all the patterns are connected through some interactions. An alternative presents the relationships that are defined by the pattern catalogue. Using these relationships between patterns, like glue in a composite, significantly reduces the possible combinations. However, no standardized set of pattern relations is defined when considering multiple catalogues. This reduces the space for pattern detection on an individual pattern language with the presumption that there is an appropriate set of relationships available. There is also another downside to this approach – instantiated pattern variants are not considered. We followed the idea in the statement: "Integrated patterns should show synergy that makes the composition more than just the sum of its parts" [16]. Our interpretation of a pattern synergy concept is as follows: The individual pattern parts in a composite should provide more pattern functionality than they provide when applied separately. The guideline for good synergy between patterns, in a composition, can be found in the level of pattern overlapping. The patterns in a composite can overlap. An individual part of such a design has various roles in used patterns. Overlapping can be observed on all the building parts of a pattern that are suggested in a pattern definition. A high level of overlapping indicates strong integration between individual patterns. Henceforth, we will define composite patterns as a set of patterns that are connected with the overlapping parts. When overlapping between patterns is detected, the candidates for composites can be extracted.

The data needed for pattern coverage and pattern overlapping presentation requires that patterns applied in a design be conceived as sets of the connected building elements, which include classes, interfaces, methods and attributes. The methods and attributes, which are prescribed by a pattern, present the building parts for pattern classes and pattern interfaces. Classes and interfaces are referred to as the main elements of a pattern or a design, while the methods and attributes of a class or interface are referred to as sub-elements of a pattern or a design. The pattern coverage matrix precisely defines the form of instantiated patterns in individual design fragments. The matrix can be presented on a whole pattern, an element or a sub-element level of detail. For reasons of clarity, we will present only a small fragment of the sample design on a detailed level.

Let $p^s = <e^s_1, ..., e^s_i>$ be a pattern p^s where e^s_x is an element of the pattern p^s. For each e^s_x there are an array of sub-elements $e^s_x = <s_1, ..., s_j>$ where e^s_x is a sub-element of the pattern p^s. The design can be presented in a similar way. Let $d=<e^d_1, ...e^d_m>$ be a design or a design fragment. For each e^d_x there are an array of design sub-elements as in the pattern $e^d_x=<s^d_1, ..., s^d_n>$. A main element of a design (class or interface) can be covered with the multiple pattern elements that belong to various patterns. In the overlapping matrix, the columns represent pattern parts, while the rows represent design parts. The matrix can be presented through various detail levels, which reveal the pattern coverage on a level that is appropriate to perform analyses. On a sub-elemental level, the matrix values can only be presented with the values of 0 or 1. The value 1 means that a sub-element of the pattern is instantiated in the sub-element that is presented in a matrix row. On the main elemental level, the idea is to determine how many pattern sub-elements (attributes and methods) cover the main element of a design. The value is the sum of the coverage. On the whole pattern level, the values as expected represent the sum of all main element coverage. The previously described coverage values are defined by the following formulas:

$$(1)\ \mathrm{cov}_{sub-sub}(s^d_x, s_y) = \begin{cases} 1 & s_y \to s^d_x; s^d_x \in d \wedge s_y \in p \\ 0 & otherwise \end{cases}$$

$$(2)\ \mathrm{cov}_{main-sub}(e^d_x, s_y) = 1 + \sum_i \mathrm{cov}_{sub-sub}(s^d_i, s_y)$$

$$(3)\ \mathrm{cov}_{main-main}(e^d_x, e_y) = \sum_i \mathrm{cov}_{main-sub}(e^d_x, s_i)$$

$$(4)\ \mathrm{cov}_{main-pattern}(e^d_x, e_y) = \sum_i \mathrm{cov}_{main-main}(e^d_x, e_i)$$

Formula 1 is used to determine coverage between the sub-elements of a particular pattern and the sub-elements of a design. The value 1 in formula (2) is added because a class or interface should also be counted as an element. For representing the matrix in all coverage details, the following formulas are also necessary:

$$(5)\ \mathrm{cov}_{sub-main}(s^d_x, e_y) = 1 + \sum_i \mathrm{cov}_{sub-sub}(s^d_x, s_i)$$

$$(6)\ \mathrm{cov}_{sub-pattern}(s^d_x, p) = \sum_i \mathrm{cov}_{sub-main}(s^d_x, e_i)$$

The coverage on the whole design is not important because it results in the number of all pattern parts in a pattern. Thus, it is meaningless, since we are interested in those parts of a design that are strongly related to applied patterns. Pattern coverage (cov) is the first of the two pattern-based metrics we proposed in this paper. To demonstrate the use of the defined coverage metric, we will use a sample design, presented in Figure 2. As we can see, the well-known MVC [13] composite pattern has been applied. The MVC pattern integrates three atomic patterns: Observer, Strategy and Composite [1].

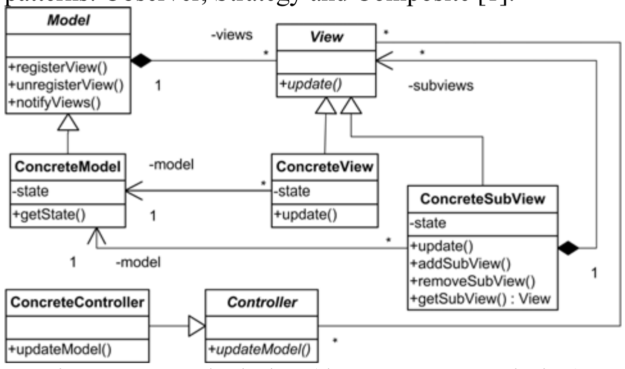


Figure 2: Sample design (the MVC pattern design)

Design / Pattern (*cov*)	**Composite**	**Observer**	**Strategy**	*Context*	*Strategy*	*Concrete Strategy*
Model	**2**	**5**	**0**	*0*	*0*	*0*
ConcreteModel	**0**	**3**	**0**	*0*	*0*	*0*
View	**2**	**2**	**2**	*1*	*0*	*0*
ConcreteView	**2**	**4**	**0**	*0*	*0*	*0*
state	*0*	*1*	*0*	*0*	*0*	*0*
model	*0*	*1*	*0*	*0*	*0*	*0*
update	*1*	*1*	*0*	*0*	*0*	*0*
ConcreteCompositeView	**6**	**4**	**0**	*0*	*0*	*0*
Controller	**0**	**0**	**1**	*0*	*1*	*0*
ConcreteController	**0**	**0**	**1**	*0*	*0*	*1*

Table 1: Pattern coverage for sample design

Table 1 contains pattern coverage values with various level of details for a sample design (Figure 2). With the previously defined formulas (1-6) we calculated table-cell values only. The main design element ConcreteView is presented on a sub-elemental level of details. The pattern Strategy is presented on the main-element level. We propose that the level of detail be adjusted by the designer, as regards the desired clarity level of the presentation. The pattern coverage matrix that is presented in an appropriate level of detail proves useful when presenting how parts of a pattern are instantiated in a particular design.

From the main-element level of details for the pattern Strategy, we can notice that the design class View represents the context in the Strategy pattern, for which different strategies can be available. In the sample design, only one concrete strategy is present and is instantiated in the design class "ConcreteController". The basic behavior of the strategy is defined in the pattern with the class Strategy that is instantiated in the design class "Controller". The inspection of the sub-elemental level of detail for the design class ConcreteView shows that the attributes "state" and "model" have a role in the pattern Observer, while the update() method appears to have a role in both the Composite and Observer patterns.

As presented, some design elements are covered with multiple patterns. We use the term pattern overlapping in cases where an individual design part is covered with multiple patterns. Pattern overlapping can be observed, in a similar way as pattern coverage, in various levels of detail. Pattern overlapping is meaningful when observed in different patterns. Let s_x, s_y be a sub-elements and e_x, e_y a main-elements of distinct pattern applications p^x, p^y for a same pattern:

$$\forall x, y; s_x, e_x \in p^x \wedge s_y, e_y \in p^y \wedge x \neq y.$$

If in a design there are two applications of the same pattern, these patterns are considered as different and an overlapping value can be calculated. We applied the following formulas:

$$(7)\quad ovl_{sub-sub}(s_x, s_y) = \begin{Bmatrix} 1 & s_x \to s^d \wedge s_y \to s^d ; \exists s^d \in d \\ 0 & otherwise \end{Bmatrix}$$

$$(8)\quad ovl_{sub-main}(s_x, e_y) = 1 + \sum_i ovl_{sub-sub}(s_x, s_i)$$

$$(9)\quad ovl_{main-main}(e_x, e_y) = \sum_i ovl_{sub-main}(s_i, e_y)$$

$$(10)\quad ovl_{main-pattern}(e_x, p^y) = \sum_i ovl_{main-main}(e_x, e_y)$$

$$(11)\quad ovl_{pattern-pattern}(p^x, p^y) = \sum_i ovl_{main-pattern}(e_x, p^y)$$

Formula (7) defines overlapping on its basic sub-elemental level. In formula (9) we provided a joint formula for the overlapping of the two main pattern elements. Overlapping is also assessed on a whole pattern level in formula (11). The remaining formulas (8) and (10) enable the calculation of a presentation on various detail levels.

Pattern (*ovl*)	**Composite**	**Observer**	**Strategy**
Composite	**-**	**10**	**1**
Observer	**10**	**-**	**1**
ObserverPart	*2*	-	*0*
ConcreteObservedPart	*0*	-	*0*
Observer	*2*	-	*1*
ConcreteObserver	*6*	-	*0*
Strategy	**1**	**1**	**-**

Table 2: Pattern overlapping matrix for the sample design

Table 2 lists scores for the overlapping metric. The pattern Observer is shown on a main-element level of detail. To express how strong the overlapping is between two patterns we define a pattern metric, the overlapping factor. Let n_{px} and n_{py} be the number of all the pattern parts (main and sub-elements) for the patterns p^x and p^y. The overlapping factor fovl between these patterns can be expressed as:

$$(12)\quad fovl_{pattern-pattern}(p^x, p^y) = \frac{ovl_{pattern-pattern}(p^x, p^y)}{n_{px} + n_{py}}$$

Pattern (*fovl*)	**Composition(15)**	**Observer(14)**	**Strategy(4)**
Composition	-	0,34	0,05
Observer	-	-	0,06
Strategy	-	-	-

Table 3: Pattern overlapping factors

Table 3 shows values for the factor of overlapping that is calculated on the base of results from the Table 2. The scores show that if the pattern p^x overlaps with the pattern p^y it is also true that p^y overlaps with p^x. For this reason, we omit a redundant calculation of these elements if the table is observed as a matrix. The numbers of pattern parts are stated in brackets near the pattern name. The results show that in the MVC pattern all elements are connected through overlapping. The overlapped patterns are the appropriate candidates for new composites.

4 The Overlapping Detection

In the previously presented sample design, the MVC pattern has been detected. The calculated values for the overlapping factor show different strengths between used patterns. These strength levels can serve for the extraction of smaller pattern candidates that show high integration, if overlapping factor is considered. The following example is a design with five applied patterns. The intention is to demonstrate a possible reduction of a pattern candidate's size in the situation where all pattern parts appear to build a single composite pattern. From the patterns applied in a design, the designer should identify the suitable composite pattern candidate that appears to have the strongest overlapping between involved patterns.

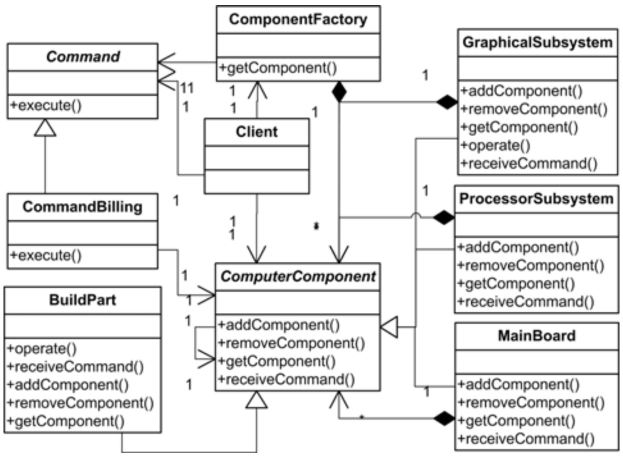


Figure 3: Sample design (the MVC pattern design)

Figure 3 shows a design for the bill of material component (BoF). The following patterns are applied: Decorator, Command, Composite, Visitor and Flyweight. The Composite pattern enables the building of a composite BoF. There are the three possible compositions that can appear in the BoF: "GraphicalSubsystem", "ProcessorSubsystem" and "MainBoard". A final leaf component is represented by the instances of the class "BuildPart". The client façade is presented by the class "Client". The Façade pattern is not explicitly exposed explicitly in the further analysis. The Flyweight pattern introduces a pool of instances for the building parts. This prevents the redundancy of objects that construct a large BoF. The Decorator pattern is introduced to later enable a dynamic adding of functionality to the class "ComputerComponent". The Visitor, in combination with the Command, enables the execution of individual calculations of the individual building parts for the BoF. To extract the most suitable composite pattern it is recommended to isolate parts with a high level of overlapping.

Overlapping / Patterns	Dekorator	Command	Composite	Visitor	Flyweight
CommandBilling	0	2	0	2	0
Command	0	2	0	2	0
Client	1	5	2	0	4
ComponentFactory	0	0	0	3	3
ComputerComponent	3	0	4	1	1
BuildPart	0	1	2	3	2
MainBoard	1	1	5	5	4
ProcessorSubsystem	1	1	5	5	4
GraphicSubsystem	1	1	5	5	2

Table 4: Pattern overlapping matrix

Table 4 shows the pattern coverage in the given component design. A brief analysis of the calculated values indicates a strong overlapping in some cases. To distinguish between different overlapping levels, we propose following value intervals for pattern overlapping factors that can present a base for the classification. We have defined three levels of overlapping: weak {0<*fovl*<0,3}, medium {0,3<*fovl*<0,5} and strong {*x*>0,5}. The intervals were defined based on our experiences and an analysis of various designs. The scores for the detected MVC pattern in the previous example indicate a weak overlap between the pattern Strategy and the other two patterns. A medium overlap exists between the patterns Composition and Observer. Reduction should be considered in cases where the candidate pattern appears to be over-specialized. The trash point should be determined by the designers, based on their experience.

Pattern (*fovl*)	Dekorator	Command	Composition	Visitor	Flyweight
Dekorator	-	0,16	0,21	0,26	0,23
Command	-	-	0,24	0,38	0,43
Composition	-	-	-	0,52	0,79
Visitor	-	-	-	-	0,89
Flyweight	-	-	-	-	-

Table 5: Pattern overlapping factors

Table 5 shows pattern overlapping factors for the BoF components. In some designs, such a table can become large and unclear. To achieve a clearer overview we propose a graphical representation of the overlapping levels.

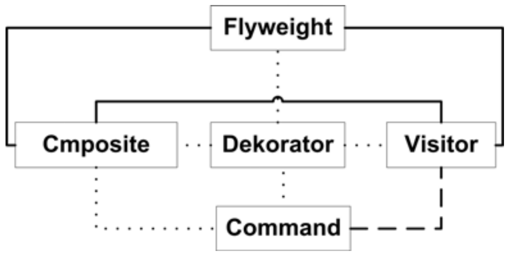


Figure 4: Graphical representation of overlapping levels

The lines that connect the patterns show the strength of the pattern overlap. In Figure 4, weak overlapping is indicated with a dotted line, medium with a dashed one, and strong overlap with a solid line. In the presented case, the Decorator pattern can be omitted from the composite pattern candidate if weak overlapping is not considered. To confirm the suitability of the composite pattern candidate, a design metric assessment should be performed.

In some cases, multiple existing designs have to be reviewed and analysed and the designer has to select suitable composite pattern candidates. If various levels of strength in overlapping are detected, then only the patterns connected with a medium or strong overlap should be considered in the further analysis.

5 Assessment of Candidates

According to the proposed technique, composite pattern candidates should be validated in the final stage. Validation is performed in the form of an assessment with the selected design metrics. The acceptance criteria should be defined based on the design metric scores that are specific for the solution space and the targeted type of software. The metric assessment eliminates unsuitable candidates in the final stage of the composite pattern identification procedure. The interval for the individual metric has to be calibrated to meet the expected property values for the given solution space and design paradigm. The sets of metrics are specific for the individual programming paradigm. Selected metrics in a set vary regarding the type of software that is developed.

The metric assessments used in an appropriate design stage help detect weaknesses in a design. Their application in the re-engineering phase helps to analyse the suitability of the design fragments. Only metrics that are influenced by the pattern application are stressed. Patterns, if assessed individually, promote a weak coupling and higher abstraction levels, which reflects on metric scores. Expected scores should reflect the desired qualities for the type of software (for a given problem domain and/or solution space). We propose a calibration of the targeted acceptance intervals for the each particular case. Defined intervals should reflect the properties that are expected to be met. For example: patterns that help build individual components should allow inherent coupling, and promote re-usability of the whole structure instead of re-usability on an individual class level. To prevent the influence of non-pattern elements, the design metric assessment is performed on isolated design fragments. Those that are influenced by a pattern application in the design phase of software development.

6 Conclusions

This paper presented the technique for identifying composite patterns in existing pattern-designs. The identification process encompasses various metric assessments. We have introduced two pattern-based metrics that enabled us to assess design fragments. While other existing researchers propose pattern identification through source code metrics, the presented technique performs assessments on the pattern level. With a sample design, we have demonstrated that the technique is also able to identify well-known composite patterns such as MVC. The identification of composite patterns, i based on pattern metrics, can result in multiple pattern candidates. To confirm if the given candidates are suitable, an additional assessment with design metrics was proposed. The goal of this assessment was to identify the most suitable candidate. A designer specifies acceptable intervals for selected metric scores that reflect the properties of a design fragment. The final result of performing the steps of the technique is composite candidates with metric scores within acceptable intervals. We have demonstrated a sample calibration of intervals for metric scores with the sample design of a component.

Through the application of the presented technique, new composite patterns can be identified in existing designs. Identified patterns can enhance the existing fundamental catalogues and provide good practice for how to apply a group of atomic patterns in similar solution spaces. This technique distinguishes itself from existing approaches of pattern identification through the use of combined assessment with pattern and design metrics. The technique can also be modified for the identification of composite anti-patterns. An additional repository of anti-patterns could prove useful in forward engineering, when the composition of patterns is required.

References

[1] J. Bazan, J. F. Peters, A. Skowron, N. Hung Son, M. Szczuka, Rough set approach to pattern extraction from classifiers, Electronic Notes in Theoretical Computer Science, Volume 82, Issue 4, March, 2003, p. 1-10.

[2] S. Chidamber, C. Kemerer, A metric suite for object-oriented design, IEEE Transactions on Software Engineering 20(6), 1994, p. 476-493.

[3] J. Dong, Adding pattern related information in structural and behavioral diagrams, Information and Software Technology, Volume 46, Issue 5, 15 April 2004, p. 293-300.

[4] A. H. Eden, A. Yehudai, J. Y. Gil, Precise specification and automatic application of design patterns, Proceedings of the 1997 International Conference on Automated Software Engineering ASE'97, 1997.

[5] J. Fabry, T. Mens, Language-independent detection of object-oriented design patterns, Computer Languages, Systems & Structures, Volume 30, Issues 1-2, April-July 2004, p. 21-33.

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns – elements of reusable object-oriented software, Addison-Wesley, Reading, MA, 1995.

[7] B. Henderson-Sellers, Object-oriented metrics: Measures of complexity, Prentice-Hall, 1996.

[8] H. Huang, S. Zhang, J. Cao, Y. Duan, A practical pattern recovery approach based on both structural and behavioral analysis, Journal of Systems and Software, Volume 75, Issues 1-2, 15 February 2005, p. 69-87.

[9] B. Huston, The effects of design pattern application on metric scores, Journal of Systems and Software, Volume 58, Issue 3, 15 September 2001, p. 261-269.

[10] D. K. Kim, R. France, S. Ghosh, A UML-based language for specifying domain-specific patterns, Journal of Visual Languages & Computing, Volume 15, Issues 3-4, June-August 2004, p. 265-289.

[11] H. Kim, C. Boldyre, A method to recover design patterns using software product metrics, 6th International Conference, ICSR-6, Austria, Lecture Notes in Computer Science 1844, 2000, p. 318-335.

[12] K. A. Kontogiannis, R. DeMori, E. Merlo, M. Galler, M. Bernstein, Pattern matching for clone and concept detection, Automated Software Engineering vol. 3, no. 1-2, July 1996, p. 77-108.

[13] G. E. Kramer, S. T. Pope, A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80, Journal of Object-Oriented Programming 1, August/September 1988, p. 26-49.

[14] C. Krämer, L. Prechelt, Design recovery by automated search for structural design patterns in object-oriented software, Proceedings of Working Conference on Reverse Engineering, Monterey, USA, IEEE CSPress, 1996.

[15] A. Lauder, S. Kent, Precise visual specification of design patterns, ECOOP'98, Lecture Notes in Computer Science 1445, 1998, p. 114-134.

[16] J. Mayrand, C. Leblanc, E. M. Merlo, Experiment on the automatic detection of function clones in a software system using metrics, Proceedings of the 1996 International Conference on Software Maintenance, 1996, p. 244.

[17] W. B. McNatt, J. M. Bieman, Coupling of design patterns: Common practices and their benefits, Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC'01), 2001.

[18] D. J. Ram, M. Sreekanth, Reusable integrated components of inter-related patterns for software development, Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC'00), p. 364-371.

[19] D. Riehle, Composite design patterns, Proceedings of OOPSLA'97, ACM, 1997, p.218-228.

[20] F. Shull, W. L. Melo, V. R. Basili, An inductive method for discovering design patterns from object-oriented software systems, Technical report, University of Maryland, Computer Science Department, College Park, MD, 20742 USA, 1996.

[21] L. Tahvildari, K. Kontogiannis, J. Mylopoulos , Quality-driven software re-engineering, Journal of Systems and Software, Volume 66, Issue 3, 15 June 2003, p. 225-239.

[22] L. Tahvildari, K. Kontogiannis, On the role of design patterns in quality-driven re-engineering, Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMR'02).

[23] P. Wendorff, Assessment of design patterns during software reengineering: Lessons learned from a large commercial project, Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR'01), 2001.

Comparative Analysis of Educational Networks

Alenka Žibert
The National Education Institute of the Republic of Slovenia,
Poljanska 28, 1000 Ljubljana, Slovenia.
E-mail: alenka.zibert@zrss.si

Vladimir Batagelj
University of Ljubljana, FMF, dept. of Mathematics,
Jadranska 19, 1000 Ljubljana, Slovenia.
E-mail: vladimir.batagelj@fmf.uni-lj.si

Vladislav Rajkovič
University of Maribor, Faculty of organizational sciences,
Kidričeva 55a, 4000 Kranj, Slovenia.
E-mail: vladislav.rajkovic@fov.uni-mb.si

Keywords: networking, educational networks, portals, comparative analysis

Received: May 30, 2005

Abstract: Educational networks and portals are a formation of thematically gathered data on the web. Structure of national educational networks and portals depends on environment of their origin. In the paper, selected educational networks are analysed according to the following criteria: content-services, navigation, search, user interface, help, credibility, validity and target groups. The criteria were identified by a group of experts and final users (students and teachers) on the basis of web survey. The importance of criteria was articulated by using the Analytical Hierarchical Method and program Saaty. For the evaluation was selected Slovene National Educational Network (SIO) as well as educational networks of Canada, Ireland, United Kingdom, Europe, Germany, Africa, Australia and America. Based on the results of the comparative analysis a concept and guidelines for improvements of our national educational network SIO were prepared.

Povzetek: Podana je primerjava izobraževalnih mrež v več državah.

1 Introduction

Recently we decided to evaluate the current status and position of SIO - the Slovenian Education Network (http://sio.edus.si/), and to prepare some guidelines for its improvements and future development. For this purpose we first made and overview and comparisons of selected educational networks (EN). In this paper we present the main results of this analysis.

SIO was founded in 1995 with the aim of providing access to individual educational servers and the material they offer. Educational users need a safe online environment they can trust and we strive to create one.

Its main goals were:

- to connect educational servers in Slovenia;
- to collect and organize the information about educational resources and events in Slovenia and world-wide;
- to support collaboration among students, teachers and parents;
- to facilitate the distribution of educational materials and products;
- to provide support for solving common problems (FAQs, recommendations, manuals, dictionaries, libraries of templates, ...);
- to support distance learning;
- to provide access to official documents on education (curricula, projects, announcements, ...).

To automatize most of the SIO's functions we developed our own support system Trubar - a system of programs for Windows to build, search and maintain the catalogs - collections of units described by list of properties (dictionaries, directories, lexicons, catalogues, inventories, glossaries ...). It is freely available at: http://www.educa.fmf.uni-lj.si/trubar/. Tools, like Trubar, support the idea that every user should also contribute to the growth of a network. At the very heart of SIO are its catalogues of information – different collections of data: interesting websites, educational resources, educational institutions, educational events and more.

Besides this SIO offers some additional services such as: Ask the experts - services that help users to solve any problem related to teaching and learning with ICT; Bulletin board; Forum; Distance learning support– a collection of educational materials: articles, online textbooks and manuals; Electronic journal *List SIO.*

SIO is a member of EUN Schoolnet (www.eun.org) and we are collaborating on different projects on national and international level. Schools and individuals are encouraged to take part in several actions and projects.

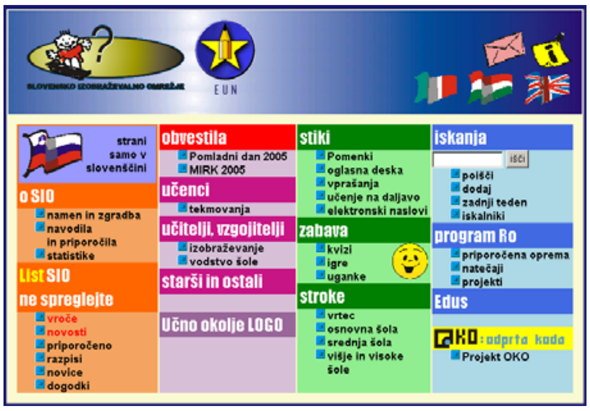


Figure 1: SIO – Slovenian Educational Network entry page

2 Overview and Comparisons

Educational networks selected for the overview and comparisons are listed in Table 1:

Table 1: Selected educational networks

Name	URL	Type
SIO - Slovensko izobraževalno omrežje, *Slovenia*	http://sio.edus.si	EN
EUN Schoolnet	http://www.eun.org	EN
Schoolnet Africa	http://www.schoolnetafrica .net/	EN
Canada's Schoolnet	http://www.schoolnet.ca/	EN
Scoilnet, Irland	http://www.scoilnet.ie/Sco ilnet/	EN
EDNA Education Network Australia	http://www.edna.edu.au	EN
NGfL - National Grid of Learning, *UK*	http://www.ngfl.gov.uk/	EN
Ask ERIC - Educational Resource Information Center	http://www.askeric.org/	portal
SAN - Schulen ans Netz, *Germany*	http://www.schulen-ans-netz.de/	EN

First we identified their basic characteristics – criteria for comparison and possible directions of improvements of SIO. They are presented in Table 2.

Table 2: Basic characteristics

	SIO	EUN Schoolnet	Canada's Schoolnet	Schoolnet Africa	Scoilnet	EDNA	NGfL	Ask ERIC	SAN
BASIC DATA	I	C	C	C	I	C	C	I	C
CREDIBILITY	founder missing	✓	✓	✓	✓	✓	✓	✓	✓
VALIDITY OF LINKS	not up to date	✓	✓	✓	✓	✓	✓	✓	✓
	D –d	D d	D –d	d	d l	d l	d l	d	d
NAVIGATION	✓– P	✓– P	✓	✓	✓– P	✓+ P	✓+ P	✓	✓– P
USER INTERFACE	✓	✓	✓	✓	✓	✓	✓	✓	✓
SUPPORT	?	?	✓	✓	NA	✓	✓	NA	NA
SEARCHING	BASE	BASE	BASE	BASE	BASE	BASE+S	BASE	BASE+S	BASE
TARGET GROUPS	✓	no parents	✓	✓	✓	✓	✓	✓	✓

C – contact addresses; I – user instructions; D – dead links; d – description; l – label; P – personalization support; NA – not available; BASE – database; BASE+S – database and servers;

According to Table 2 the most complete EN is EDNA, followed by NGfL, Schoolnet Africa and Canada's Schoolnet. Our SIO is at the end of the list with several options to be considered for implementation or improvement.

3 Criteria

Analysis was performed according to the criteria presented in Figure 2.

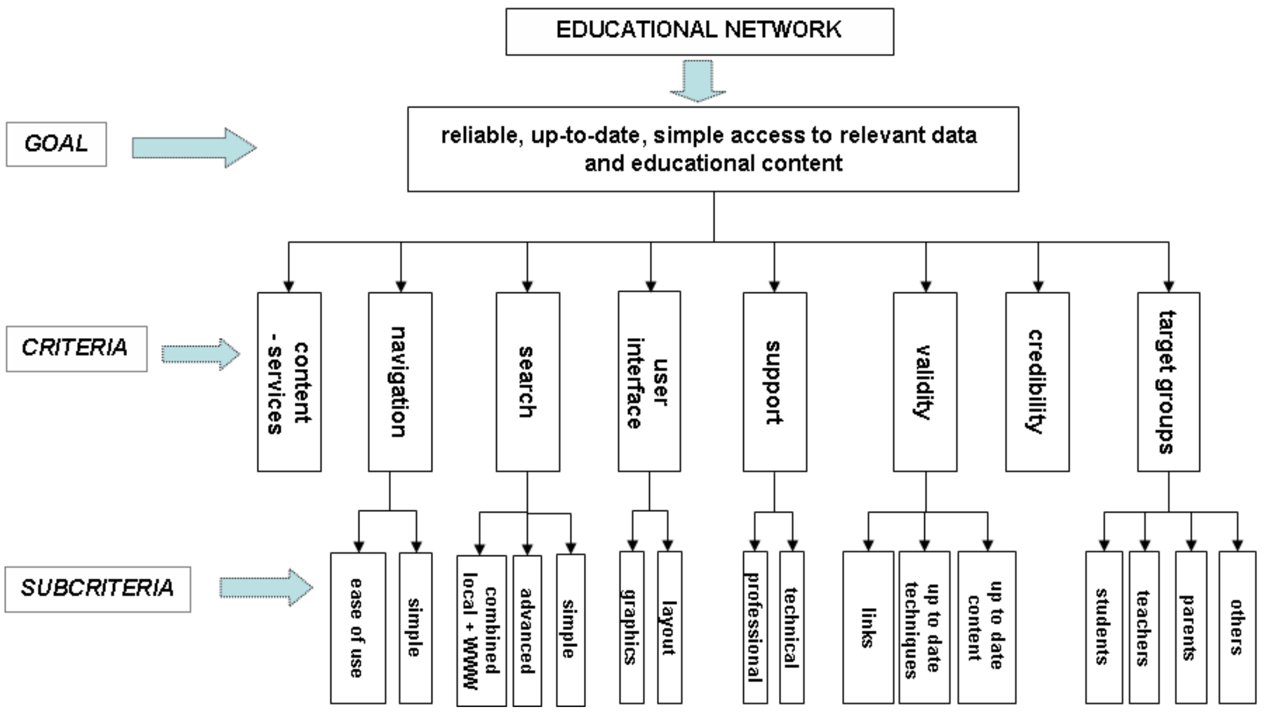


Figure 2: Characteristics of educational network

Table 3 summarizes pair wise comparison scores for selected criteria assigned by a group of experts. According to Saaty's AHP method these scores are integers from 1 to 9 and their inverses. The interpretation of main values are: 1 –criteria ***i*** and ***j*** are of equal importance; 3 – criterion ***i*** is weakly more important than criterion ***j*** ; 5 – criterion ***i*** is strongly more important than criterion ***j*** ; 7 – … The product of symmetric entries equals to 1. In ideal case a kind of 'transitivity' should hold for the scores: $a_{ik} . a_{kj} = a_{ij}$ - we say that the scores are consistent. The real life comparison matrices are usually inconsistent. A special consistency coefficient K was introduced. It is assumed that a comparison matrix is consistent enough if K < 0.10.

Table 3*: Pairwise comparisons*

j / i	cont/serv	navigation	searching	user I	support	credibility	validity	target G
cont/serv	1	3	3	3	5	5	1/2	4
Navigation	1/3	1	2	1	2	3	1/4	2
Searching	1/3	1/2	1	1	3	3	1/4	2
user I	1/3	1	1	1	3	2	1/3	1
Support	1/5	1/2	1/3	1/3	1	2	1/5	1/2
Credibility	1/5	1/3	1/3	1/2	1/2	1	1/5	1/2
Validity	2	4	4	3	5	5	1	3
target G	1/4	1/2	1/2	1	2	2	1/3	1

From the comparison matrix we get a vector of relative importance of criteria as the eigen-vector corresponding to its largest eigen-value. In our case we get λ = 8.30594, K = 0.0311 and from the eigen-vector the criteria ranking presented in Table 4.

Table 4: Criteria ranking

CRITERION	RANK	EIGEN VECTOR
validity	1	0.30416494
content/services	2	0.24085495
navigation	3	0.10880157
searching	4	0.09626298
user interface	5	0.09251457
target groups	6	0.07215293
support	7	0.04651902
credibility	8	0.03872904

After we obtained the experts' opinion about the importance of criteria we tried to get the users' opinion. Therefore, we conducted a survey.

4 Survey

To get an insight to usability of networks and their comparison from the point of view of final users a web poll was constructed. In the construction the findings of J. Spool were considered. In his research about Web Site Usability he found out that web site users usually do not make use of it like web designers have planned for them. The skills and knowledge ob web designer usually do not ensure a useful web page.
Web poll was planned to be filled in by various groups of users. Unfortunately only 59 users answered the questions, 61% of them were teachers. Due to a very small budget allocated to this research and due to a very small number of users who took part in the survey, we got only the teachers' point of view on usability of web sites.
Users had to visit 3 foreign education networks and answered 2 questions about each of them. Then an opinion about Slovenian Education network had to be written together with the comparison of chosen networks.
The chosen education networks are rich on various learning resources and interactive activities. The networks had been chosen because of: English language, a long time of existence (Canadian Network) and because SIO is a member of EUN Schoolnet. Majority of activities of both educational networks are closely linked. NGfL is one of most extensive European EN.
The users had to visit networks first and using them answer to very simple questions. The answers were placed on 1st or 2nd level. The users were very successful with searching the answers at English education network - NGfL and the least successful with EUSchoolnet.
The answers to question about their latest visit on Network were as follows. Users wished to explore what was offered (47%), a quarter of them did that because of poll award, others have been searching for new learning resources and information for their work at school, some of them (13%) were not successful. The last group mostly made a remark like "I was unable to find anything" or "Not useful".
This was their profile: almost half (48%) of users check the SIO portal a few times per year, for a quarter of them it was their first visit, almost the same size of group check the SIO portal once a month. They had five levels to express their opinion (1 – very poor, 2 – poor, 3 - medium 4 - good, 5 – very good) on content and users experience. The majority (68%) evaluate the content with 3-medium and the user-experience with 3-medium too (63%). They express their opinion as follows: the content has to be changed (44%), the design has to be changed (29%), name and title has to be changed (11%). The navigation is simple (58%).
The answers to question " What should be changed about SIO in order to achieve the portal to become a valuable information source for education?" were as follows: more programs, more teacher education, up to date and solid information, promotion, better design and more activities to convince additional teachers to use SIO.
As you can see in the table below CSF was evaluated as least useful (4 out of 6 criteria are low), NGfL was evaluated as most useful (4 out of 6 were marked high), EUN Schoolnet was evaluated as useful (2 out of 6 were marked high).

Table 5: Percentage answers above level 3

	searching	navigation	layout	content	user interface	graphics
CSN	77%	67%	72%	81%	56%	53%
NFGL	77%	78%	81%	83%	71%	65%
EU	79%	74%	92%	81%	67%	62%

According to users' evaluation the NGfL is the best choice limited to 3 chosen networks. The layout and user interface are best evaluated for NGfL.
The connection of the type of users and their efficiency was not explored. In order to carry out such analysis a larger number of users from all target groups have to take part in the poll.

5 Conclusions - Improvements to SIO and Future Development

SIO was renewed for the first time in 1999. In the survey, described in the previous section, the users were satisfied with the design and content of its pages. SIO has a good position in the Slovenian web – according to the internet research study SIO was in 2002 on the 56th place among all Slovenian web sites (Petrič, 2003) with respect to betweenness (Freeman, 1979). It contains also a critical mass of contents. Their downsides are the problems with maintenance of the content. The main reason for this is that SIO is not institutionally appropriately integrated and supported by the Slovenian educational system.

From our overview of ENs and results of the survey we can conclude that SIO plays an important role in Slovenian education, but it should be renewed in **technological** (new tools), **functional** (additional services) and **organizational** (collection of materials, maintenance) sense.

In the development of an EN there are two basic options: to establish a central institution that provides most of educational web services; or to establish only an ‘index’ to educational services distributed across several institutions. We believe that the limit situation is the second option. For this reason the primary function of SIO is to collect, maintain and provide information about educational resources and services. As we already explained in the first section, SIO is based on catalogue system Trubar in which also different materials (photos, drawings, texts, maps, programs, data, sounds …) are collected. Because of SIO’s financial and man-power limitations these materials are mainly contributed by users. Systematic approaches have to be developed to provide ‘complete’ collections of materials.

During the last months we have checked several content management systems to find an appropriate replacement for Trubar. We will probably base the new SIO on the open source Zope connected with Python and MySql on the Apache server. These tools provide an up-to-date and platform independent development environment. We do not expect special problems in transferring the services from Trubar to the new environment. In the new solutions we intend to provide several new options: active link control, access statistics, voting evaluation system, editing, personalization, … They will support semantic web (RDF, OWL) and educational (SCORM) standards.

Since an EN is used by several types of users with different needs, we decided to develop the new SIO as a **multi-entry** site – each entry providing different, user/goal-oriented view of the content stored in catalogues: portal (entry **dveri**), e-journal (entry **list**), entertainment (entry **zabava**), for non-Slovenian guests (entry **english**), … For example, the portal entry will provide fresh information (last contributions in catalogues, news, surveys, events, …) and information sources (addresses and basic data about schools and other educational institutions, manuals, dictionaries, templates, …).

A special challenge is the kindergarten entry. Here we will try to produce an environment adapted to the capabilities of kids – use of picture language, sound (audio) output …

A big problem on the web is non permanent contents – they are appearing, changing and disappearing. An additional service of SIO could be an **Archive** of selected educational materials.

6 References

[1] Batagelj V & Brodnik A & Lokar M1996. Slovensko izobraževalno omrežje, Kongres Ro/2, Ljubljana, 26. – 27.9.

[2] Batagelj V& Žibert A & Rajkovič V & Čampelj B 1999. Educational Networks Vision and Reality, *IFIP WG 3.1 and 3.5 Open Conference Aulanko – Hämeenlinna – Finland*, 20-23.

[3] Butcher N: Best Practice in Education Portals. Research document prepared for the Commonwealth of Learning and School Net Africa., October 2002. http://www.col.org/Consultancies/02EducationPortals_Report.pdf

[4] Commission of the European Communities 2000. Designing tomorrow's education promoting innovation with new technologies, *Report from the commission to the council and the european parliament, Brussels*.

[5] Freeman LC 1979. Centrality in social networks: *Conceptual clarification. Social Networks, 215-239*.

[6] Petrič G 2003. Družbeno delovanje v omrežju svetovnega spleta: individualni in strukturni vidik. Phd thesis, FDV, Ljubljana.

[7] Saaty TL 1980. The Analytic Hierarchy Process. *McGraw Hill,* New York

[8] Spool J 1999. Web Site Usability: A Designer's Guide, *M. Kaufmann Publishers, Inc*.

[9] Žibert A & Rajkovič V (mentor) 1998. Kritična analiza šolskih računalniških omrežij: *diplomsko delo univerzitetnega študija*, IMPRESUM: Kranj.

[10] Žibert A & Rajkovič V (mentor) & Batagelj V (komentor) 2004. Critical analysis of usability on educational networks (in Slovene), *Master thesis*, IMPRESUM: Kranj.

[11] "Complete e-learning introduction on the national scale in Slovenia", *project within National Target Research Program "Slovenian Competitiveness 2001 - 2006",* funded by Ministry for Education.

Open Source Software Usage Implications in the Context of Software Development

Gregor Polančič, Marjan Heričko and Romana Vajde Horvat
Institute of Informatics,
University of Maribor,
FERI, Smetanova 17, SI-2000 Maribor,
Slovenia
E-mail: gregor.polancic@uni-mb.si, marjan.hericko@uni-mb.si, romana.vajde@uni-mb.si

Keywords: open source software, open source projects, business process diagrams, risk benefit analysis.

Received: May 6, 2005

Open source software (OSS) is becoming increasingly popular in several aspects of software engineering activities, ranging from using OSS for development or execution environments to incorporating OSS directly into developed products. OSS and its development projects differ from proprietary software and closed source projects in several aspects. Therefore, these aspects should be known and analyzed, before making a decision for using OSS in a software development project. This paper analyses various OSS usage strategies in the context of software development projects. Dependent on cases of usage, different open source project collaboration models, based on business process models, are analyzed from several relevant aspects.

Povzetek: Na osnovi procesov sodelovanja in definiranih atributov so analizirane prednosti in tveganja različnih modelov uporabe odprtega programja v kontekstu projektov razvoja programske opreme.

1 Introduction

Software development projects are often timely and financial ineffective, while on the other hand producing low qualitative and vulnerably artefacts (software). Lack of quality and productivity in software development projects has raised several strategies capable of confronting with this problem.
According to Boehm (Boehm 1999), there are three major strategies for improving software development productivity and software quality:
- working faster (usually with better tools),
- working smarter (usually with more optimized processes) and
- work avoidance (usually with software reuse).

Two strategies presented above (working faster and work avoidance) are realized with software. Such software can be developed "in-house", obtained from another company (for free or purchased) or open source based.
In this article we analyse implications of incorporating open source software into software development strategy. Open source software (OSS), which is becoming increasingly popular and important (Brown & Booch 2002; Ruffin & Ebert 2004), is computer software that has its source code made available under an open source definition (OSD) based license (Open Source Initiative 2005). OSD based license implicates that the source code of software is released with binary, allowing users and developers to use and to modify the software and to distribute any improvements they make. Consequently, most of OSS is being developed in public accessible projects where everyone capable of contributing knowledge, ideas or code is welcome to join in. Such projects are called open source projects – OSP (see also Figure 1).

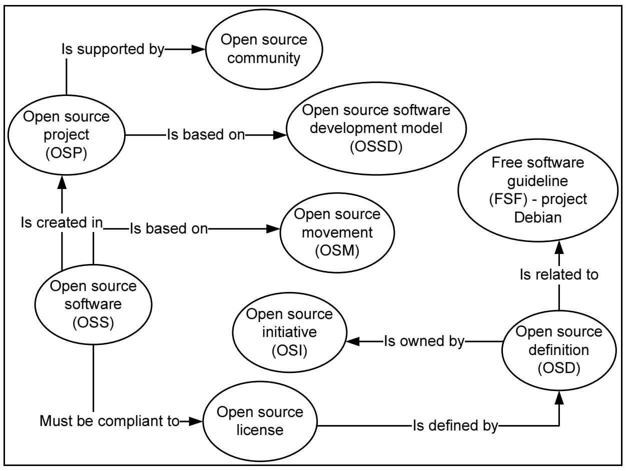

Is supported by
Open source community
Open source project (OSP)
Is based on
Open source software development model (OSSD)
Free software guideline (FSF) - project Debian
Is created in
Is based on
Open source movement (OSM)
Is related to
Open source software (OSS)
Open source initiative (OSI)
Is owned by
Open source definition (OSD)
Must be compliant to
Open source license
Is defined by

Figure 1: Relations between common open source terms

According to open source advocates, such development model leads directly to more robust software and more diverse business models (Wu & Lin 2001).
Software development companies are looking toward OSS as a way to provide greater flexibility on their development practises, jump-start their development efforts by reusing existing code and provide access to a much broader market of users (Brown & Booch 2002; Kasper Edwards 2004).
On the other hand, there are several risks and limitations concerned with using open source software, which should be properly addressed. Low code quality, non-

existing project plan and non-deterministic stability of the project are some of them (Fitzgerald 2004).
Related to open source software (potential) benefits and risks, which were mentioned above, the research question can be stated as "What are the implications of a specific open source software usage strategy in a software development project?"
Based on the research question, we identify and analyse different open source software usage strategies, for the purpose of determine benefits and risks of each strategy, with respect to software license, development processes and software, from the point of view of closed source software developer and in the context of business process models.

1.1 Scope of the Paper

Section two of the paper connects this research to the existing body of knowledge. In the section three, open source projects, their development model, its common design and characteristics are introduced. Additionally, a comparative study is performed, comparing open source and closed source (proprietary) projects.
Based on open source development model, its unique characteristics and related work (concerned with open source software usage in commercial environment), different usage strategies are presented and evaluated accordingly to predefined attributes.
The research has the following limitations. Closed source projects are defined as projects which are based on a well established development model. In the context of software collaboration processes between open source and closed source projects, only technical activities are analysed. Additionally, because of parsimony, the open source and closed source software development processes are presented on a high level view.

2. Related Work

Several descriptive studies exist in the field of using open source software (OSS) in commercial context.
In the article "Using open source software in product development: A primer", Ruffin and Ebert (Ruffin & Ebert 2004) state, that the use of OSS in industrial products is growing. They discuss major legal aspects and risks in using OSS and how to mitigate them in product development. Additionally, OSS must meet several criteria, required to reduce risks of technical and legal exposure during deployment.
Madanmohan and De in the article, titled "Open source reuse in commercial firms" (Madanmoban & De 2004) state, that using OSS components raises many issues, from requirements negotiation to product selection and integration. They define a model of the stages involved in locating and using an OSS component. Five critical issues for reusable OSS components are identified: cost, customization requirements, component characteristics, licensing, maintenance and support. They state that if the OSS component offers the best solution and reliability for the price, then it is the most appropriate.
In the article titled "Reusing open-source software and practices: The impact of open-source on commercial vendors", authors Brown and Booch (Brown & Booch 2002) find out that as a result of the open-source movement there is a great deal of reusable software available in the public domain, which can be used in commercial projects. Open source movement is described as a diverse collection of ideas, knowledge, techniques and solutions. Additionally, the authors state, that there are several questions concerned with applying OSS ideas into commercial environment.
The paper, titled: "Towards a Product Model of Open Source Software in a Commercial Environment", from Deng, Seifert and Vogel (Jianjun Deng, Tilman Seifert, & Sascha Vogel 2003) state that there are many reasons for commercial organisations to be interested in using OSS. Aspects of OSS development for commercial use are analysed in the paper. Second, different categories of OSP are identified together with typical requirements, which have to be realized by instances of OSS. Third, an open source process model, based on the concept of work products and product networks is defined.
Another type of research has published Edwards in the article titled "An economic perspective on software licenses—open source, maintainers and user-developers" (Kasper Edwards 2004). Based on economic theory, he defined several models, which illustrate the possible choices available to users and developers once a program has been distributed under a specific type of software license. The basics premise of the research is that users are prepared to contribute to projects if there is a net benefit. Based on two different open source (GPL and BSD) and a proprietary (Microsoft EULA) software license, three different models are developed by deducting the behaviour (activities) possible for software developers and users. Based on developed models, the incentives for developers and users together with their relationships are analysed. Individuals and organisations related to open source software are treated differently, because of different incentives for contributing to open source projects.

3. Open Source Projects

Open source projects (OSP) are software projects, which are based on open source software development model (OSSD), a recent phenomenon, which became available with the existence of the global communication infrastructure – internet. Because of open source license, OSP have different project structure, compared to "traditional" software projects.

3.1 Open Source Software Development Model

Most of commercial or proprietary software projects are based on closed source software development model (CSSD) (Vidyasagar Potdar & Elizabeth Chang 2004). Such development model follows strictly defined activities and their relationships. Several CSSD models exist, for example: cascade, spiral, iterative-incremental

(Figure 2), V-model and RUP (Rational Unified Process).

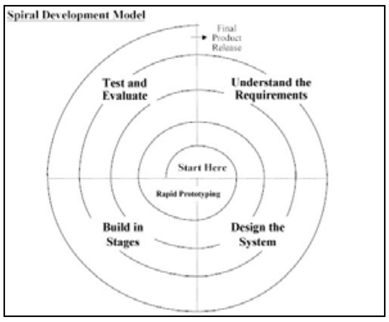


Figure 2: Spiral software development model

On the other hand, open source project are based on open source software development model (OSSD) (Vidyasagar Potdar & Elizabeth Chang 2004). OSSD is an evolutionary development model (Figure 3), where software is permanently evolving according to user needs (Vidyasagar Potdar & Elizabeth Chang 2004). OSS never reaches its final state, because it keeps evolving as long as there is an active user community available. Consequently, such development model emphasizes frequent minor point releases and as much feedback on these releases as possible.

Because no strict sequence of phases is defined in OSSD (Figure 3), OSP cannot be tracked according to phases. Instead, the progress is usually tracked with file versioning system, for example CVS (Concurrent Versioning System).

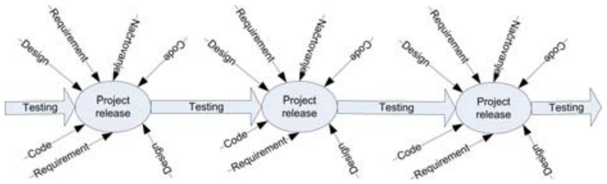


Figure 3: Evolutionary software development model (Michael Nash 2003)

3.2 Open Source Community Structure

Open source projects (OSP) are based on virtual community concepts. Because the available project resources are proportional to the user community size, they support open standards and standard development and collaboration tools. Because OSP usually lack of finances, they are trying to minimise project costs with using public available information infrastructure (for example "Sourceforge.net" repository).

OSS software communities are virtual work groups consisting of members with skills in software development. They work in temporary, cultural diverse, geographically dispersed, electronically communicating work groups (Wolfgang Maass 2004). Based on user roles, open source communities, are generally organized as presented below (Jen-Fang Lee & Tzu-Ying Chan 2004; Richard P.Gabriel & Ron Goldman 2002).

In the centre of the community is a small group of core developers (see also Figure 4). Core developers have most rights and also responsibilities in OSP. They have write access to source code's baseline. They make decisions concerned with code merging, quality assurance and releases.

Beside code developers, there is usually a larger group of code developers, which are developing new functions and performing other, less responsible tasks, for example: improving user interface, fixing bugs and writing documentation.

The largest group is represented by active and passive users. Active users participate in OSP in form of identifying bugs, proposing new features, creating documentation and offering user support. Passive users only use OSS and other project artefacts.

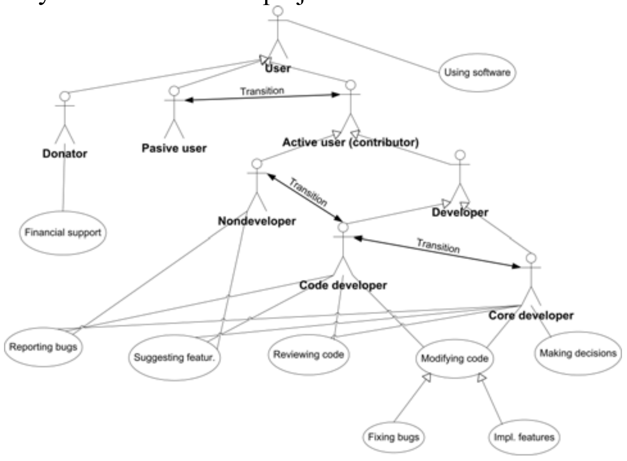


Figure 4: High level use case diagram of open source community

3.3 Open Source Project Characteristics

Open source projects have in common following characteristics (Gacek & Arief 2004):

a. Adherence to OSD (Open Source Definition), which acts as an open source accordance guideline.
b. Open source software developers represent a subset of open source user community (see also Figure 4). Consequently OSS developers are also OSS users.

Despite of commonalities presented above, OSP differ in several aspects (Gacek & Arief 2004):

a. Project starting point. OSP can start from scratch or from existing proprietary or research (closed source) project.
b. Motivation. A lot of open source research is related to motivational aspect of willing to freely participate in OSP (Andrea Bonaccorsi & Cristina Rossi 2005; Wolfgang Maass 2004). Individuals usually participate from personal believes or because they require functions which might be provided by OSS. Corporations usually get involved to gain market share, to lower their software infrastructure costs or to be less dependent from commercial software vendors.
c. Community. Two basic types of open source communities exist: centralized and decentralized. Central organized communities have a strict hierarchy of active users, which allows a more centralized power structure. Their opposites are

decentralized communities, which have looser organisational structures with most of developers on the same level. One-level organisational structure requires more sophisticated decision making processes.
The basic idea, underlying open source projects, is that knowledge, shown through contributions, increases the contributor's perceived merit, which in turn leads to power (this is called meritocratic culture).

d. Software development support. OSP differ in their modularity (high modularity is prerequisite for effective remote collaboration), visibility of software architecture (system architecture might be available or not), documentation, testing, submission acceptance (involves choosing the work area, decision making and disseminating the submission information), tools and collaboration support.
e. Licensing. Several types of licenses conform to OSD. From the user point of view the most important license characteristics are its impact on derived works and possibility to "close" the licensed software (Table 1).

Table 1: Implications of main OSD licenses (Gacek & Arief 2004)

OSD based license	Impact on derived works?	Can be closed?
GPL (GNU General Public License)	Yes	No
LGPL (GNU Lesser GPL)	No	No
BSD (Berkley Software Distribution)	No	Yes
IBM Public License	No	Yes
MPL (Mozilla Public License)	No	Yes

3.4 Open Source Project Compared to Closed Source Projects

Beside different development models, open source projects differ from closed source (proprietary) projects in several other aspects. Some of them are briefly presented below (Vidyasagar Potdar & Elizabeth Chang 2004):

a. Documentation. Within CSP, the process of writing documentation is defined in project plan or requirements. On the other side, OSP participants usually prefer writing code. Consequently, there is usually lack of qualitative and updated documentation.
b. Testing. In OSP software users act as software testers. This is called "many eyeballs" principle (Eric S.Raymond 2000). They either try to solve problems or to notice the community. CSP are tested by specified number of software testers.
c. Security. In CSP the security of software is achieved through obscurity, while in the OSP the security is achieved through openness of the code. Both strategies have their strengths and risks. However in highly secure systems, openness is preferred.
d. Release and delivery. In CSP, software might be released because of market pressures or defined project milestones. OSS is released when it meets release criteria. OSP releases are usually frequent but not scheduled.
e. Development environment. CSP are usually centralized on a single physical location. OSP development occurs in virtual communities which offer decentralized and distributed development.

4. Modelling Open Source Software Usage Strategies

Despite of differences between open source and closed source projects a lot of different collaboration opportunities exist between them (Brown & Booch 2002; Kasper Edwards 2004). Such OSS usage models depend on several factors, for example: business strategy, software license and software type.

4.1 Identification of Usage Strategies

Several OSS usage classifications exist. According to Gacek and Arief (Gacek & Arief 2004) following OSS business models are viable:

- using OSS for personal use,
- packaging and selling OSS,
- using OSS as a platform or foundation for commercial or research software development.

On the other hand, Edwards classifies software use, according to software licenses (Kasper Edwards 2004) into:

- commercial or proprietary license,
- BSD based open source license and
- GPL based open source license.

Ruffin and Ebert (Ruffin & Ebert 2004) classify OSS usage, dependent on the licensee role, into:

- end user OSS and
- OSS that is embedded into in a product that is further distributed. This is called software reuse.

Based on classifications presented above, their differences and commonalities, a use case model of common open source software usage strategies in closed source projects can be defined (Figure 5):

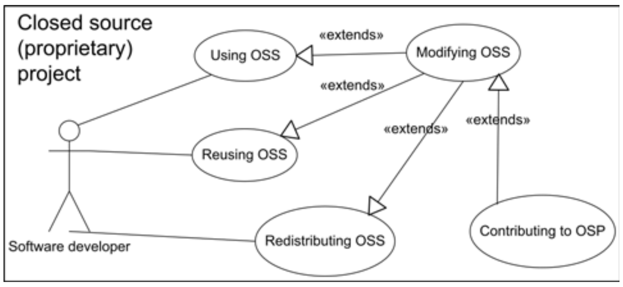


Figure 5: Use case model of common open source software usage strategies in proprietary projects

The identified strategies of using OSS in proprietary software projects are following (Figure 5):

a. Using OSS. OSS is used for project or product infrastructure, which includes: development tools, collaboration tools, software testing environment and software execution environment.
b. Reusing OSS. Reused OSS (for example: software snippet, software component or software framework) is embedded into developed product.
c. Redistributing OSS. Added value, based on additional artefacts (commercial software, documentation, plug-ins, etc.) and services is included and distributed with OSS. Distribution can be proprietary or open source based.

Each of the main strategies presented above, can be additionally divided accordingly to (Figure 5):

- using OSS as-is or suiting it to specific needs;
- treating modifications as intellectual property or committing them to the open source community.

Based on use case model presented on Figure 5, twelve (3x2x2) different OSS usage scenarios might occur, each with its strengths and risks.

4.2 Notation Used for Modelling Usage Strategies

Models of OSS usage strategies, resulting from use case model presented in section 4.1 (Figure 5), are based on business process modelling notation – BPMN (BPMI 2004). BPMN is developed by business process management initiative (BPMI). The current specification of BPMN, which is 1.0, was released to the public in May, 2004. BPMN defines a business process diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations. A business process model is a network of graphical objects, which are activities and the flow controls that define their order of performance. Four basic categories of elements in BPMN are (Stephen A.White 2004):

- flow objects (events, activities, gateways),
- connecting objects (sequence flows, message flows, associations),
- swimlanes (pools and lanes) and
- artefacts (data objects, groups and annotations).

We decided to use BPMN because it is easily understandable, supported by OMG (Object Management Group) and highly expressive.

We used Microsoft Visio as a software modelling tool. Additional, an open source based BPMN stencil was used. The stencil is available on Sourceforge.net repository (https://sourceforge.net/projects/bpmnpop).

4.3 Analysis of Usage Strategies

Based on resulting business process models, we performed two types of analyses.

First, we performed a high level risk-benefit analysis for each resulting model. Risk is the potential harm that may arise from some present process or from some future event. Risk-benefit analysis is the comparison of the risk of a situation to its related benefits. Risk-benefit analysis was performed on activities and relevant events that occur in resulting business process models.

Second, we performed a comparative study of all three usage strategies. Several attributes were defined for comparative study, ranging from user types, major benefits and desirable OSS characteristics. These attributes are presented in section 5.5.

5. Resulting Models

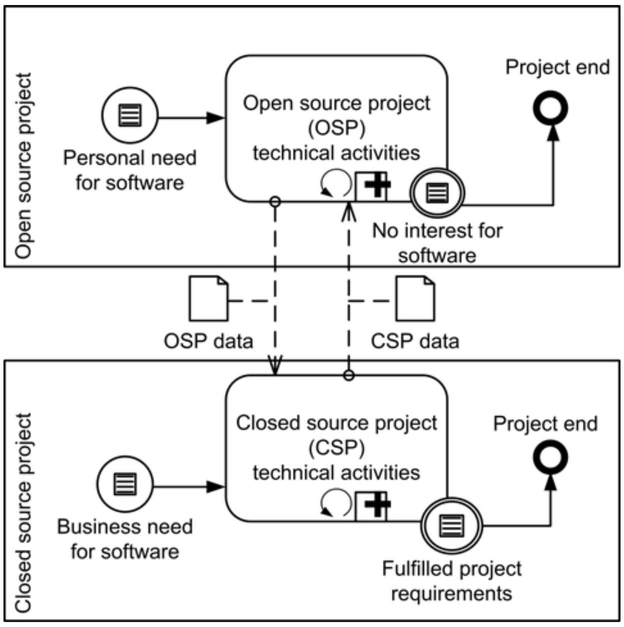
Based on OSS usage strategies, defined in section 4.1 we modelled and descriptively presented one generic and three special business models. They are presented and analysed in following subsections.

5.1 Generic Model

All special OSS usage models are derived from the top level usage model which is presented on Figure 6. Therefore the special models include same BPMN constructs (pools, events, messages, processes) as presented on generic model.

The generic model consists of two pools (rectangles), representing independent processes of OSP (Open Source Project) and CSP (Close Source Project), which differ in the underlying software development model. CSP development and OSP development are modelled with repeatable sub-processes (rounded rectangles with curved arrow and "+" sign).

The collaboration between projects is modelled with bi-directional data exchange using BPMN messages mechanism (dotted arrows) exchanging data objects (documents).



Open source project
Personal need for software
Open source project (OSP) technical activities
Project end
No interest for software
OSP data
CSP data
Closed source project
Business need for software
Closed source project (CSP) technical activities
Project end
Fulfilled project requirements

Figure 6: Generic OSS usage model

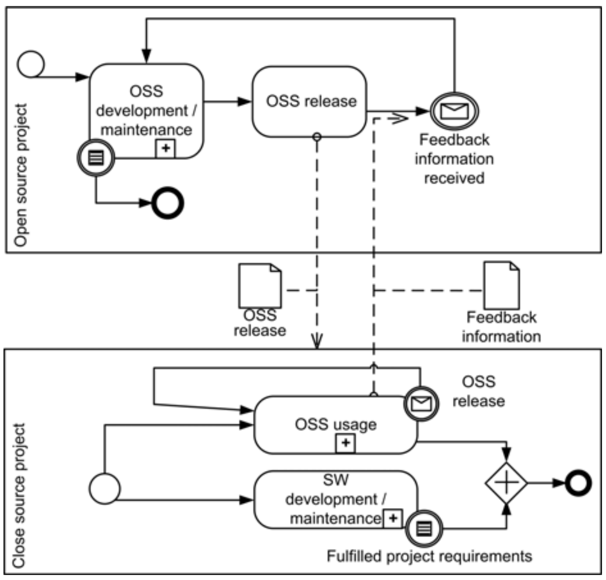
Additionally, different events (presented as rules in circles) initiate, direct flow and finish OSP and CSP.

There is usually a business need for starting a CSP, requiring sufficient human and financial resources. On

the other hand, OSP start, because there is a personal need for some functionality (software).
CSP usually have a predictive end, consisting of documented list of functional and non-functional requirements, which have to be fulfilled. OSP usually do not have a predictive end. Non predictive end might present a risk to CSP. Because OSP are "organic projects" they are finished if there is no interest for software being developed.

5.2 Using of OSS

Based on the business model on Figure 7, using OSS is comparable to using proprietary software. Because OSS is (in most cases) used as provided by OSP, modification activities are not modelled. However, OSS can be suited to specific needs, if necessary. As a new version of OSS is released, software developer (if necessary) installs new release and uses it as infrastructure software (development, maintenance, execution or collaboration software). When OSS is used, feedback information can be sent to OSP, for example: modification proposals, new feature requests and identified bugs. Using OSS might end with fulfilled CSP project requirements.


Open source project
OSS development / maintenance
OSS release
Feedback information received
OSS release
Feedback information
Close source project
OSS usage
OSS release
SW development / maintenance
Fulfilled project requirements
Figure 7: Model of using OSS

a. Benefits. The main objective of using OSS in CSP is to lower the cost of project infrastructure or decrease dependency from specific commercial software vendors. Additional, a benefit of using OSS can be free support and add-ons which are available from open source community. Beside, OSS can be influenced with sending feedback to CSP. In such way, OSS can be better suited to CSP needs.
b. Risks. There are several risks concerned with using OSS. First, releases are usually not determined. Therefore, planning the OSP on some future OSS releases is risky. Second, there are no legal guaranties for using OSS. For example, if there is a bug in OSS or a defined release date was postponed, nobody is responsible for potential damage. Third, there is no guarantee that feedback information will be considered by OSP. Feedback is usually considered if there is a community size interest for them.

5.3 Redistributing OSS

Commercial vendor can decide to redistribute OSS. Based on the model on Figure 8, an OSS redistribution project is restarted each time new stable version of OSS is released. Additional, CSP can make some modifications or additions to OSS, which can be sent back to community (for example: identified bugs or functions which can be further developed by user community) or (if the OSS license allows), treated as intellectual property of commercial vendor. Finally commercial vendor releases software (SW) package. Final users might send feedback information to CSP, which can further be mediated to OSP.

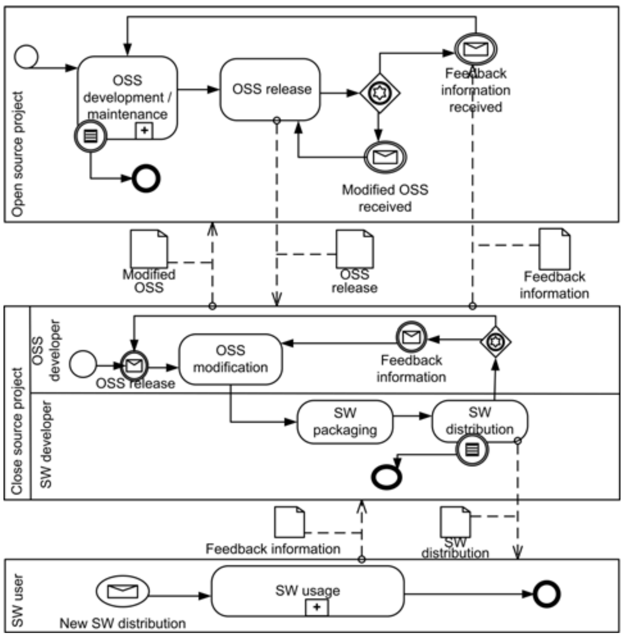


Figure 8: Model of redistributing OSS

a. Benefits. The main objective of redistributing OSS is to gain market share or to make profit from selling software, supporting services or distributions. Second OSP can be directly influenced by CSP with sending modified OSS code back to the open source community. In such way open source community can further develop or maintain code, which was primary developed by CSP. Consequently CSP costs are lowered.
b. Risks. Most of the risks, concerned with redistributing OSS, are related to non determined OSS releases and potentially unstable open source community. Therefore, planning release dated might be risky. Second there might be legal problems concerned with viral OSS licenses which prohibit that OSS changes licensing model. For example we cannot make binary distributions of GPL based software. Third, future directions of OSS might change unpredictably. For example, if CSP is distributing OSS with a proprietary plug-in,

problems could be caused with changed plug-in interface.

5.4 Reusing OSS

When reusing OSS in CSP, following activities occur (Figure 9). First, if a specific OSS component is suitable for software development, it can be adapted (if necessary) and afterwards included into developing software. Modified OSS can be sent back to OSP or it can be treated as intellectual property of CSP. Finally software is released together with reused OSS. End users use released software (SW) and if necessary, send feedback information to CSP. CSP can react to feedbacks with direct software changes or mediate feedbacks to open source community. OSS modification and integration activities are usually performed, when there is a new version of OSS available.

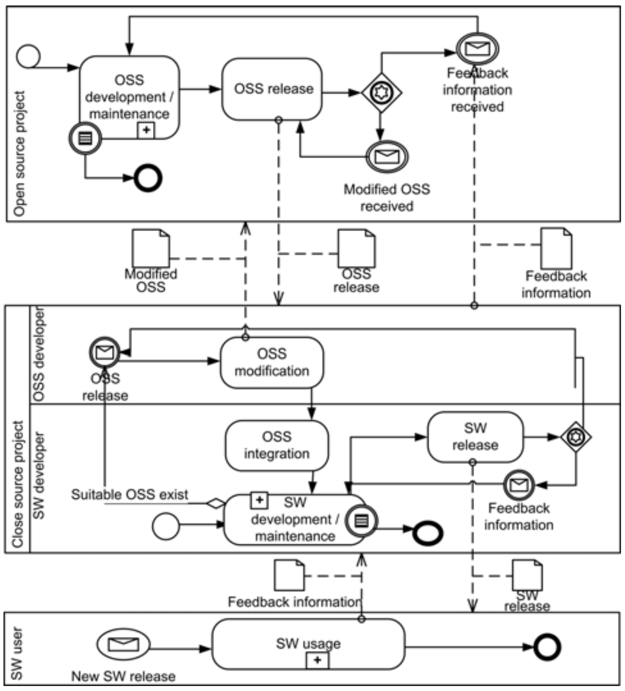


Figure 9: Model of reusing OSS

a. Benefits. The main objectives of reusing OSS in proprietary projects are simultaneously increasing productivity and software quality through OSP developed and maintained reusable software artefacts. Productivity is increased, because parts of software (reused OSS) are developed and maintained by OSP. Second software quality is increased because reused OSS is tested and improved by open source community.
b. Risks. Several risks are present in such reuse strategy. First, rarely or delayed OSS releases might influence (expand) CSP project plan. Second if, there are to frequent releases and unstable OSS architecture, a lot of effort is spent for OSS integration. Third, OSS license might prohibit reusing OSS in proprietary software (for example GPL or LGPL license).

5.5 Comparing Three Usage Models

Models, defined in previous section differ in complexity, benefits and risks. Additional, there are several other factors that should be considered before making a decision for a specific usage strategy. Following factors and sub-factors were considered in the comparative study:

a. Open source software (suitable software licenses according to Table 1, desirable software characteristics and most suitable software types).
b. Open source software user (OSS user roles, closed source developer activities when using OSS, most frequent collaboration artefacts between OSP and CSP).
c. Open source project major desirable characteristics.
d. Closed source project (major benefits, major investments and major risks).

Results of the comparative study are summarized below in Table 2.

Table 2: Results of comparing different OSS usage models

OSS usage strategy	**Using OSS**	**Redistribute OSS**	**Reusing OSS**
Suitable software license	All	Non viral licenses	Non viral licenses (BSD, IBM, MPL)
Desirable OSS characteristics	Quality in-use	Quality in-use, software quality, process quality	Software quality, process quality, reusability
Suitable OSS types	Infrastructure software	Infrastructure software and office tools	Reusable components and frameworks
OSS user roles in CSP	Active user	Developer	
Closed source developer activities related to OSS	Usage	Usage, modifications, packaging	Reuse, modifications, integration
Collaboration artefacts between CSP and OSP	Identified bugs, feature requests	Identified bugs, feature requests, code	
OSP major desirable characteristics	Good support	Stable releases	Stable architecture
Major benefits	Lower direct and indirect cost	Commercial distributions, market penetration	Increased productivity and software quality
Major OSS cost factors	Learning OSS	Learning OSS modifying OSS, collaborating with OSP	Learning OSS modifying OSS, integrating OSS, collaborating with OSP
Major risk	Low OSS quality, lack of support	Unsuitable OSS license, undetermined OSP stability	Unsuitable OSS license, unstable OSS architecture, week OSS reusability.

6. Conclusion

In this study we analysed open source projects from the closed source software development point of view. We presented open source project structure its characteristics, and specialities compared to traditional software projects. Because of increasing interest in using open source software in commercial projects, following basic open source software usage strategies were identified: using, redistributing and reusing open source software. All strategies were presented in business process models, based on business process modelling notation - BPMN. Additionally risk-benefit analysis was performed on activities and events of each business model. Finally a comparative study, comparing all three models was performed, based on predefined attributes.

Future research might be directed into specifying cost models of specific usage strategies. Additional, empirically testable success factors should be defined for OSS that is commonly used in a specific usage strategy.

To summarize, open source software has a huge usage potential in commercial software development environment, where open source community acts as a resource of software developers and testers. Open source can supply commercial projects with software infrastructure, reusable components or products, which can be further commercially redistributed. However technical, managerial and legal aspects should be properly studied before deciding for a specific usage strategy.

References

[1] Andrea Bonaccorsi & Cristina Rossi "Contributing to OS Projects. A Comparison between Individual and Firms", in *Collaboration, Conflict and Control*, pp. 18-22.

[2] Boehm, B. 1999, "Managing software productivity and reuse", *Computer*, vol. 32, no. 9, pp. 111-113.

[3] BPMI. Busines Proces Modelling Notation ver 1.0. 2004. Busines Process Management Initiative (BPMI).
Ref Type: Generic

[4] Brown, A. W. & Booch, G. 2002, "Reusing open-source software and practices: The impact of open-source on commercial vendors", *Software Reuse: Methods, Techniques, and Tools, Proceedings*, vol. 2319, pp. 123-136.

[5] Eric S.Raymond 2000, "The cathedral and the bazaar", *Computers & Mathematics with Applications*, vol. 39, no. 3-4, p. 263.

[6] Fitzgerald, B. 2004, "A critical look at open source", *Computer*, vol. 37, no. 7, pp. 92-94.

[7] Gacek, C. & Arief, B. 2004, "The many meanings of open source", *IEEE Software*, vol. 21, no. 1, p. 34-+.

[8] Jen-Fang Lee & Tzu-Ying Chan 2004, "Organisational Structure of "User Collaboration Community": Insights from the Case of an Open Source Software Project", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[9] Jianjun Deng, Tilman Seifert, & Sascha Vogel "Towards a Product Model of Open Source Software in a Commercial Environment", in *3rd Workshop on Open Source Software Engineering*, pp. 31-38.

[10] Kasper Edwards 2004, "An economic perspective on software licenses—open source, maintainers and user-developers", *Telematics and Informatics*.

[11] Madanmoban, T. R. & De, R. 2004, "Open source reuse in commercial firms", *Ieee Software*, vol. 21, no. 6, p. 62-+.

[12] Michael Nash 2003, *Java Frameworks and Components: Accelerate Your Web Application Development* Cambridge University Press.

[13] Open Source Initiative. Open Source Initiative - OSI - The Open Source Definition. 2005. 20-8-2005.
Ref Type: Generic

[14] Richard P.Gabriel & Ron Goldman 2002, "Open Source: beyond the Fairytales", *Perspectives on Business Innovation* no. 8.

[15] Ruffin, M. & Ebert, C. 2004, "Using open source software in product development: A primer", *Ieee Software*, vol. 21, no. 1, p. 82-+.

[16] Stephen A.White. Introduction to BPMN. July 2004. 2004. BPTrends.
Ref Type: Unpublished Work

[17] Vidyasagar Potdar & Elizabeth Chang 2004, "Open Source and Closed Source Development Methodologies", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[18] Wolfgang Maass 2004, "Inside an Open Source Software Community: Empirical Analysis on Individual and Group Level", in *4th Workshop on Open Source Software Engineering*, pp. 105-109.

[19] Wu, M. W. & Lin, Y. D. 2001, "Open source software development: An overview", *Computer*, vol. 34, no. 6, p. 33-+.

Model-Based Tuning of Process Parameters for Steady-State Steel Casting

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
E-mail: bogdan.filipic@ijs.si

Erkki Laitinen
Department of Mathematical Sciences
University of Oulu
P.O. Box 3000, FIN-90014 Oulu, Finland
E-mail: erkki.laitinen@oulu.fi

Keywords: steel production, continuous casting, process parameters, coolant flows, stochastic optimization, evolutionary algorithm, next descent

Received: October 28, 2005

We present an empirical study of process parameter tuning in industrial continuous casting of steel where the goal is to assure the highest possible quality of the cast steel through proper parameter setting. The process is assumed to be under steady-state conditions and the considered optimization task is to set 18 coolant flows in the caster secondary cooling zone to achieve the target surface temperatures along the slab. A numerical model of the casting process was employed to first investigate the properties of the parameter search space, and then iteratively improve parameter settings. For this purpose, two stochastic optimization algorithms were used: a steady-state evolutionary algorithm and next-descent local optimization. The results indicate the difficulty of the optimization task arises not from a complicated fitness landscape but rather from high dimensionality of the problem.

Povzetek: V članku predstavljamo uglaševanje procesnih parametrov za industrijsko kontinuirano ulivanje jekla na osnovi numeričnega modela procesa in z uporabo stohastičnih optimizacijskih metod.

1 Introduction

Manufacturing and processing of materials are nowadays largely based on numerical analysis and computer support. Material scientists and engineers rely on computational approximation both in process design and control. Numerical simulators enable insight into process evolution, allow for execution of numerical experiments and facilitate manual process optimization by trial and error. In addition, reliable process simulators and efficient optimization techniques allow for automated optimization of process parameters and improvement of material properties. These goals can be achieved by interconnecting a process simulator with an optimization algorithm through a cost function which allows for automatic assessment of the simulation results. This framework has recently been extensively studied and applied to a number of material processes under the project COST 526: *Automatic Process Optimization in Materials Technology* (APOMAT) [5].

Continuous casting is a predominant technology of steel production in modern steel plants. It is a complex metallurgical process in which liquid steel is cooled and shaped into semi-manufactures of desired dimensions. To achieve proper quality of cast steel, it is essential to control the metal flow and heat transfer during the casting process. They depend on numerous parameters, such as the casting temperature, casting speed and coolant flows. Finding optimal values of process parameters is difficult since different, often conflicting criteria may be applied, the number of possible parameter settings is high, and parameter tuning through real-world experimentation is not feasible because of costs and safety risk. Over the last years, however, several computational techniques have been used to enhance the process performance and product characteristics, including knowledge-based heuristic search [4], genetic algorithms [10, 2], and evolutionary multiobjective optimization [3].

In this paper we report on preliminary numerical experiments in optimizing secondary coolant flows on a casting machine of the Rautaruukki steel plant in Finland. Calculations were done for a selected steel grade under the assumption of steady-state caster operation. Their objective was to get better insight into the properties of this optimization task and tune the coolant flows with respect to the given temperature distribution requirements. The paper describes the optimization problem, the applied mathematical model of the casting process and the experimental setup, and reports on numerical experiments and results.

2 The Optimization Problem

Figure 1 shows a schematic view of a continuous casting machine. In the continuous casting process molten steel is poured into a bottomless mold which is cooled with internal water flow. The cooling in the mold extracts heat from the molten steel and initiates the formation of the solid shell. The shell formation is essential for the support of the slab after mold exit. After the mold the slab enters into the secondary cooling area in which it is cooled by water sprays. The secondary cooling region is divided into cooling zones where the amount of the cooling water can be controlled separately.

The secondary cooling area of the considered casting device is divided into nine zones. In each zone, cooling water is dispersed to the slab at the center and corner positions. Target temperatures are specified for the slab center and corner in every zone. Water flows should be tuned in such a way that the resulting slab surface temperatures match the target temperatures. Formally, a cost function is introduced to measure the differences between the actual and target temperatures. It is defined as

$$\begin{aligned} c(T) &= \frac{1}{2}(\sum_{i=1}^{N_Z} l_i(T_i^{\text{center}} - T_i^{\text{center}*})^2 + \\ &\quad + \sum_{i=1}^{N_Z} l_i(T_i^{\text{corner}} - T_i^{\text{corner}*})^2), \qquad (1) \end{aligned}$$

where N_z denotes the number of zones, l_i the length of the i-th zone, T_i^{center} and T_i^{corner} the slab center and corner temperatures, while $T_i^{\text{center}*}$ and $T_i^{\text{corner}*}$ the respective target temperatures in zone i. The optimization task is to minimize the cost function over possible cooling patterns (water flow settings). Water flows cannot be set arbitrarily, but according to the technological constraints. For each water flow, minimum and maximum values are prescribed.

Table 1 shows an example of the prescribed target temperatures and water flow intervals for continuous casting of the steel grade analyzed in this study. The slab cross-section in this case was 1.70 m × 0.21 m and the casting speed 1.4 m/min.

Table 1: Target temperatures and water flow intervals for continuous casting of steel considered in the empirical study

Position	Zone number	Target [°C]	Flow number	Min. [m^3/h]	Max. [m^3/h]
	1	1050	1	7.1	26.1
C	2	1040	2	22.8	57.5
e	3	980	3	13.3	39.9
n	4	970	4	1.5	7.9
t	5	960	5	2.7	10.0
e	6	950	6	0.8	6.5
r	7	940	7	0.7	5.9
	8	930	8	1.0	5.8
	9	920	9	1.2	6.2
	1	880	10	7.1	26.1
C	2	870	11	22.8	57.5
o	3	810	12	13.3	39.9
r	4	800	13	1.2	3.5
n	5	790	14	2.4	4.4
e	6	780	15	2.4	2.9
r	7	770	16	0.7	5.9
	8	760	17	1.0	5.8
	9	750	18	1.2	6.2

3 Mathematical Model of the Casting Process

The simulation model calculates the temperature field of the steel slab as a function of the casting parameters. We consider steady-state casting conditions, i.e. the parameters are constants in time. We denote the 3D geometry of the slab by $\mathcal{V} = \Omega \times [0, L_Z]$, where $\Omega = [0, L_X] \times [0, L_Y]$ is a 2D cross-section of the slab and L_Z is the length of the strand. Moreover, we denote by L_M the length of the mould. We divide the boundary $\Gamma = \partial\mathcal{V}$ into four parts:

$$\begin{aligned} &\Gamma_0 = \Omega \times \{0\}, \\ &\Gamma_N = \{(x,y) \in \partial\Omega : x = 0 \vee y = 0\} \times [L_M, L_Z], \\ &\Gamma_S = \{(x,y) \in \partial\Omega : x \neq 0 \wedge y \neq 0\} \times [0, L_Z) \cup \Omega \times \{L_Z\}, \\ &\Gamma_M = \{(x,y) \in \partial\Omega : x = 0 \vee y = 0\} \times [0, L_M]. \end{aligned} \qquad (2)$$

The mathematical model for the temperature field $T = T(x, y, z, t)$ of the slab can be written as

$$\begin{cases} \frac{\partial H(T)}{\partial t} + v\frac{\partial H(T)}{\partial z} - \Delta K(T) = 0 & \text{in } \mathcal{V} \times (0, t_f], \\ T = T_0 & \text{on } \Gamma_0 \times (0, t_f], \\ \frac{\partial K(T)}{\partial n} + h(T - T_w) + \\ +\sigma\epsilon(T^4 - T_{ext}^4) = 0 & \text{on } \Gamma_N \times (0, t_f], \\ \frac{\partial K(T)}{\partial n} = 0 & \text{on } \Gamma_S \times (0, t_f], \\ \frac{\partial K(T)}{\partial n} = Q & \text{on } \Gamma_M \times (0, t_f], \\ T(x, y, z, 0) = T^0 & \text{in } \mathcal{V}. \end{cases} \qquad (3)$$

Here n is the unit vector of outward normal on $\partial\mathcal{V}$, h is the heat transfer coefficient, v is the casting speed, T_w and T_{ext} are known temperatures, σ is the Stefan-Boltzmann constant and ϵ is the emissivity. The cooling efficiency Q in the mould is a known constant and t_f is the simulation time. $H(T)$ and $K(T)$ are the temperature dependent enthalpy and Kirchoff functions (see [13] for details).

Equations 3 are discretized using the finite element method (FEM) and the corresponding nonlinear equations solved with relaxation iterative methods [7]. A more detailed description of discretization and construction of

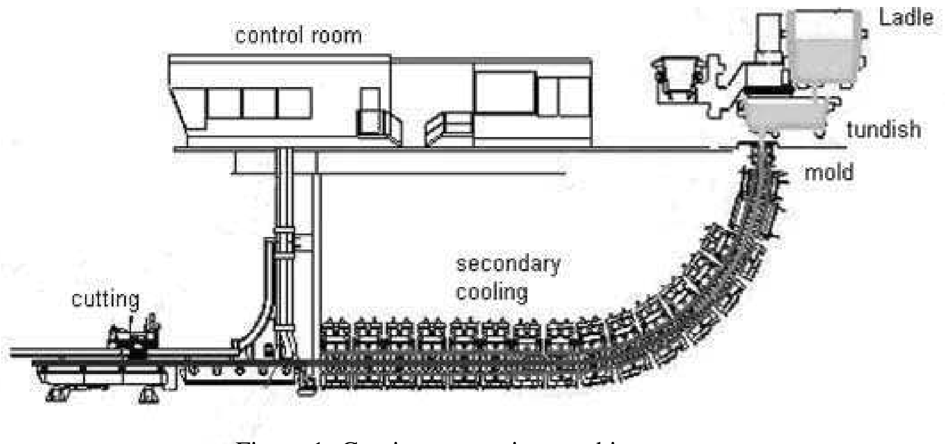


Figure 1: Continuous casting machine

FEM matrices is presented in [6]. We note that in our method it is sufficient to construct only 2D- and 1D-matrices. Therefore, it is obvious that the model is computationally much more efficient than in the case of using the ordinary 3D-brick elements.

4 Experimental Setup

Evaluation of cooling patterns and their assessment with respect to cost function (1) was done using the described mathematical model implemented in the form of a computer simulator. Its principal task is to dynamically track the temperature field in the slab as a function of process parameters. In this study it was applied under the assumption of steady-state caster operation, and the search for optimal cooling patterns performed in the off-line manner. A single simulator run takes about 40 seconds on a 1.8 GHz Pentium IV computer.

Before the integration of the simulator with the optimization algorithms, a number of simulator runs were performed to get an initial insight into the properties of the fitness landscape associated with the optimization problem. Specifically, the cost was analyzed as a function of individual parameters and pairs of parameters, while keeping the remaining parameters fixed at the values from the middle of their intervals.

The resulting plots show simple dependencies between the parameters and cost function in the form of monotonic or at most U-shaped curves and surfaces (see examples in Figures 2 and 3). They are much simpler than usual artificial test functions for numerical optimization, which is understandable because of the underlying physical process. Similar properties were found in the analysis of the fitness landscapes in parameter tuning for a continuous casting machine at the Acroni steel plant in Jesenice, Slovenia [11]. However, one should bear in mind that such analyses offer a very limited view of the problem characteristics. Nevertheless, the real difficulty comes with high dimensionality of the problem, as there are 18 independent process parameters subject to optimization.

Before the application of optimization procedures one has to decide whether to search for optimal solutions in continuous or discretized parameter space. In analogy to previous studies performed on similar task from the Acroni steel plant [9, 14, 8], the discrete version was considered. The rationale behind it is in the engineering approach to coolant flow tuning where it is meaningless to consider changes below certain amount as they do not reflect in changing the cost value. For the purpose of numerical experiments three discretizations were defined, a very rough one for initial tests of the optimization algorithms, another one with medium step sizes to refine the results, and the one with the uniform step size of 0.1 m^3/h which is the minimum change considered in practice for all coolant flows (see Table 2).

Given these dicretizations, one can to calculate the number of possible parameter settings. For a parameter from the interval $[p_i^{\min}, p_i^{\max}]$ with step size p_i^{step}, there are $v_i = \lfloor (p_i^{\max} - p_i^{\min})/p_i^{\text{step}} \rfloor + 1$ values possible, and the total number of settings is $v = \prod_{i=1}^{N_p} v_i$, where N_p is the number of parameters. This results in $4.6 \cdot 10^{12}$ possible setting for discretization 1, $4.9 \cdot 10^{23}$ for discretization 2, and $4.7 \cdot 10^{33}$ for discretization 3.

5 Numerical Experiments and Results

Two stochastic optimization techniques were applied to the coolant flow optimization problem, the steady-state evolutionary algorithm [1] and the next-descent local optimization algorithm. They were selected as they performed well in solving similar optimization problems for the Acroni steel plant [9, 14]. Both methods iteratively improved candidate solutions represented as real vectors of coolant flow values. The evolutionary algorithm was run with the

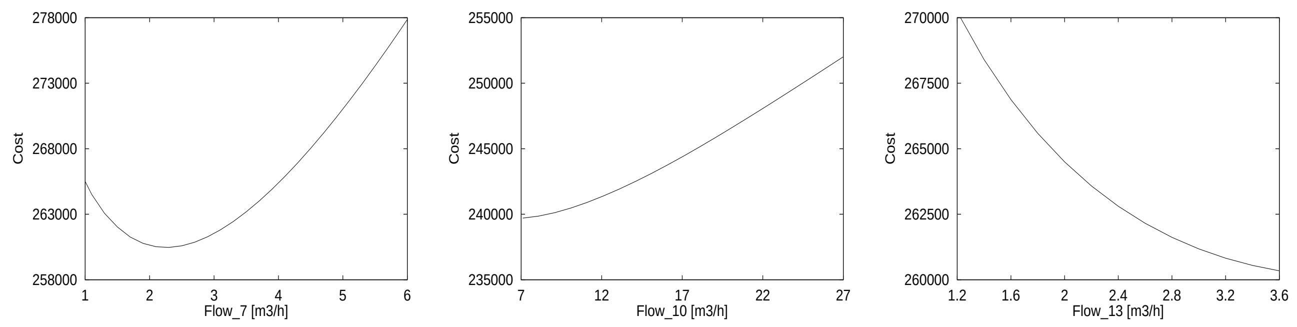


Figure 2: Examples of cost function dependencies on individual process parameters

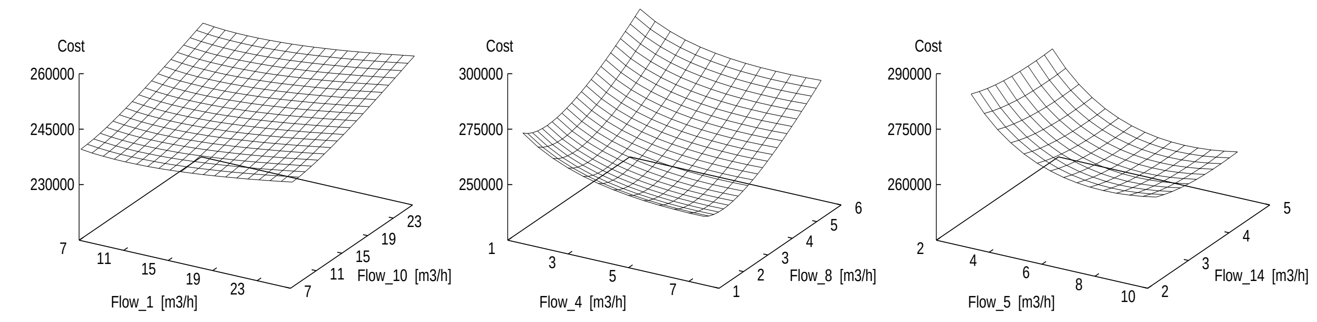


Figure 3: Examples of cost function dependencies on pairs of process parameters

Table 2: Parameter discretizations used in the optimization process; #val denotes the number of values possible for each parameter

Flow no.	Discretization 1		Discretization 2		Discretization 3	
	Step [m^3/h]	#val	Step [m^3/h]	#val	Step [m^3/h]	#val
1	4.7	5	1.0	20	0.1	191
2	8.6	5	1.0	35	0.1	348
3	6.6	5	1.0	27	0.1	267
4	1.6	5	0.5	13	0.1	65
5	1.8	5	0.5	15	0.1	74
6	1.4	5	0.2	29	0.1	58
7	1.3	5	0.2	27	0.1	53
8	1.2	5	0.2	25	0.1	49
9	1.2	5	0.2	26	0.1	51
10	4.7	5	1.0	20	0.1	191
11	8.6	5	1.0	35	0.1	348
12	6.6	5	1.0	27	0.1	267
13	0.5	5	0.2	12	0.1	24
14	0.5	5	0.2	11	0.1	21
15	0.1	6	0.1	6	0.1	6
16	1.3	5	0.2	27	0.1	53
17	1.2	5	0.2	25	0.1	49
18	1.2	5	0.2	26	0.1	51

population of 20 solutions, applying arithmetic crossover and Gaussian mutation adjusted to perform vector variation with prescribed discretization. The local optimization algorithm relied on the neigborhod relationship among candidate solutions. Two solutions were considered neighbors if differing in the i-th vector component for $\pm p_i^{\mathrm{step}}$. In this way each solution, with the exception of those on the edge of the search space, had $2N_p = 36$ neighbors. The algorithm started from a randomly selected point and was restarted after reaching a local minimum.

For each of the three search space discretizations the algorithms were run five times and their results evaluated statistically. The number of solutions checked (parameter settings evaluated) in each algorithm run was 200 for discretization 1, 500 for discretization 2, and 2000 for discretization 3. No other parameter adjusting was involved as this empirical study was a preliminary one.

The performance of the algorithms under different search space discretizations is illustrated in Figure 4 and the results in terms of cost summarized in Table 3. For discretization 1, the performance of random search is also shown to provide an empirical upper bound for the results. In this case, the local optimization algorithm clearly outperforms the evolutionary algorithm, but the cost values produced are still high which indicates the discretization is too rough to allow for detection of the near-optimal solution. With the refinement of discretization better results are found by both methods and their performance compares differently. The finer the discretization, the closer the final results, while in the initial stage of the search the evolutionary algorithm outperforms the local optimization algorithm. The solutions found with local optimization are however not dispersed as with the evolutionary algorithm. It turns out that the more complex the search space the more obvious the efficiency of the evolutionary algorithm in identifying the promising regions which suggests an appropriate hybrid of the two algorithms would reduce the number process simulations needed in the optimization procedure.

Certainly, the key result for material engineers at the plant are the optimized coolant flows. Their values will

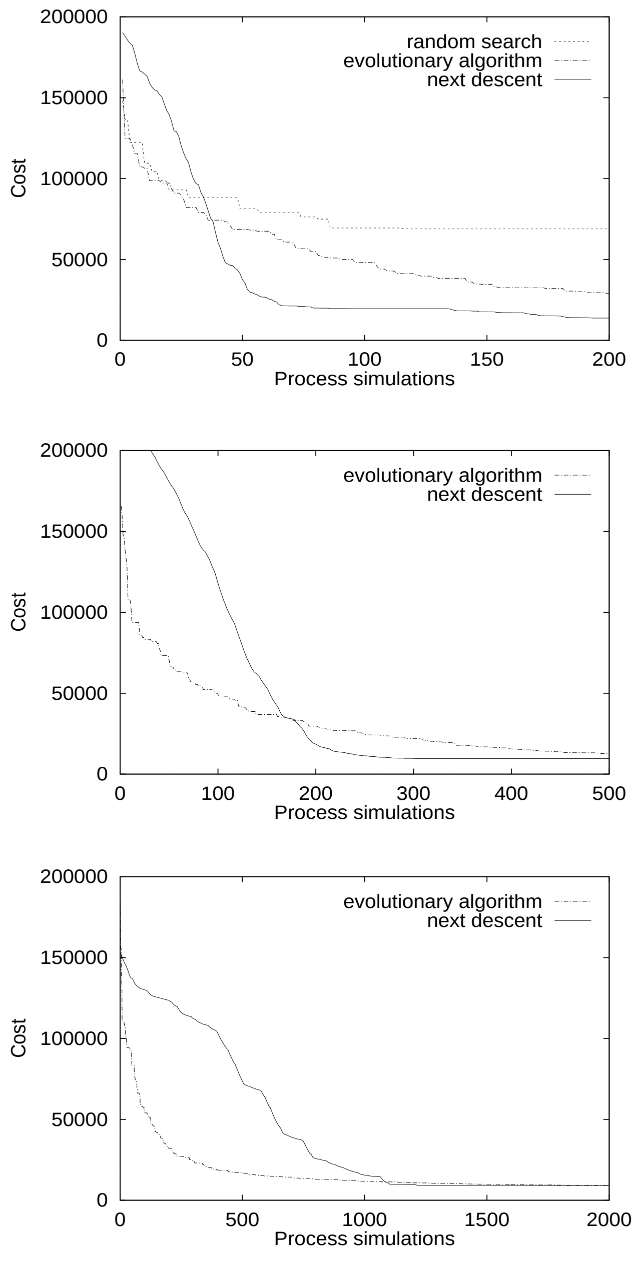


Figure 4: Performance of the optimization algorithms averaged over five runs of each algorithm for parameter discretizations 1 (top), 2 (center), and 3 (bottom)

Table 3: Summary of the optimized cost values found for three parameter discretizations; EA denotes the steady-state evolutionary algorithm, and ND next descent local optimization

Discr.	Method	Best	Average	Worst	St. dev.
1	EA	24988.8	28965.9	32842.5	2800.8
	ND	13417.9	13794.9	15062.7	716.3
2	EA	10371.3	12466.6	14092.0	1790.4
	ND	9592.9	9592.9	9592.9	0.0
3	EA	9078.5	9194.0	9247.2	73.7
	ND	9070.4	9070.4	9070.4	0.0

be compared with the empirical settings used in practice, and checked for possible contribution to the improvement of steel quality.

6 Conclusion

Optimization of coolant flow settings in continuous casting of steel is a key to higher product quality. It is nowadays to a high degree performed through virtual experimentation involving numerical process simulators and advanced optimization techniques. In this preliminary study of optimizing 18 cooling water flows for a Rautaruukki casting machine under steady-state conditions, an empirical investigation of the problem properties was done, two stochastic algorithm applied and their performance compared.

The results indicate the importance of the applied search space discretization and suggest the construction of a hybrid algorithm to find near-optimal solutions in smaller number of solution evaluations. With the same objective in mind, the algorithms will be systematically tuned and enhanced with the mechanisms of gradual refinement of the search focus, such as dynamic parameter encoding [15] or the multilevel technique [12]. On the practical side, the optimized coolant flows will be evaluated with respect to the settings used on the caster machine and checked for potential further improvements of the casting process.

Acknowledgement

This work was supported by the Slovenian Research Agency and the Academy of Finland under the Slovenian-Finnish project BI-FI/04-05-009 *Numerical Optimization of Continuous Casting of Steel*, by the European Science Foundation under COST 526: *Automatic Process Optimization in Materials Technology* (*APOMAT*), and by the Slovenian Research Agency under the Research Programme P2-0209-0106 *Artificial Intelligence and Intelligent Systems*.

References

[1] T. Bäck, D. B. Fogel, Z. Michalewicz (Eds.). *Handbook of Evolutionary Computing*, Institute of Physics Publishing, Bristol, Philadelphia, and Oxford University Press, New York, Oxford, 1997.

[2] N. Chakraborti, R. Kumar, D. Jain. A study of the continuous casting mold using a Pareto-converging genetic algorithm. *Applied Mathematical Modelling*, 25 (4): 287–297, 2001.

[3] N. Chakraborti, R. S. P. Gupta, T. K. Tiwari. Optimisation of continuous casting process using genetic algorithms: studies of spray and radiation cooling regions. *Ironmaking and Steelmaking*, 30 (4): 273–278, 2003.

[4] N. Cheung, A. Garcia. The use of a heuristic search technique for the optimization of quality of steel billets produced by continuous casting. *Engineering Applications of Artificial Intelligence*, 14 (2): 229–238, 2001.

[5] COST 526, Automatic Process Optimization in Materials Technology (APOMAT), http://www.cost526.de.

[6] R. Dautov, R. Kadyrov, E. Laitinen, A. Lapin, J. Pieskä, V. Toivonen. On 3D Dynamic Control of Secondary Cooling in Continuous Casting Process. *Lobachevskii Journal of Mathematics*, 13: 3–13, 2003.

[7] C. M. Elliot, J. R. Ockendon. Weak and Variational Methods for Moving Boundary Problems. Pitman Publishing, Boston, 1982.

[8] B. Filipič. Efficient simulation-based optimization of process parameters in continuous casting of steel. In: D. Büche, N. Hofmann (Eds.), *COST 526: Automatic Process Optimization in Materials Technology: First Invited Conference*, pp. 193–198, Morschach, Switzerland, 2005.

[9] B. Filipič, T. Robič. A comparative study of coolant flow optimization on a steel casting machine. *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, OR, USA, Vol. 1, pp. 569-573. IEEE, Piscataway, 2004.

[10] B. Filipič, B. Šarler. Evolving parameter settings for continuous casting of steel. *Proceedings of the 6th European Conference on Intelligent Techniques and Soft Computing EUFIT'98*, Vol. 1, pp. 444–449. Verlag Mainz, Aachen, Germany, 1998.

[11] B. Filipič, B. Šarler. An empirical investigation into the Properties of Coolant Flow optimization in the Steel Production Process. In: B. Zajc, A. Trost (Eds.), *Proceedings of the Fourteenth International Electrotechnical and Computer Science Conference ERK 2005*, Portorož, Slovenia, Vol. B, 59–62. Slovenia Section IEEE, Ljubljana, 2005.

[12] P. Korošec, J. Šilc. The multilevel ant stigmergy algorithm: an industrial case study. *Proceedings of the 8th Joint Confeence on Information Sciences JCIS 2005*, pp. 475–478, Salt Lake City, Utah, USA.

[13] E. Laitinen. Online control of secondary cooling in steel continuous casting process. In: D. Büche, N. Hofmann (Eds.), *COST 526: Automatic Process Optimization in Materials Technology: First Invited Conference*, pp. 174–182, Morschach, Switzerland, 2005.

[14] T. Robič, B. Filipič. In search for an efficient parameter tuning method for steel casting. In: B. Filipič, J. Šilc (Eds.), *Bioinspired Optimization Methods and their Applications, Proceedings of the International Conference BIOMA 2004*, pp. 83-94. Jožef Stefan Institute, Ljubljana, Slovenia, 2004.

[15] N. N. Schraudolph, R. K. Belew. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9 (1): 9–21, 1992.

Visualization of Text Document Corpus

Blaž Fortuna, Marko Grobelnik and Dunja Mladenić
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: {blaz.fortuna, marko.grobelnik, dunja.mladenic}@ijs.si

Keywords: text visualization, latent semantic indexing, mutlidimensional scaling

Received: June 2, 2005

Visualization is commonly used in data analysis to help the user in getting an initial idea about the raw data as well as visual representation of the regularities obtained in the analysis. In similar way, when we talk about automated text processing and the data consists of text documents, visualization of text document corpus can be very useful. From the automated text processing point of view, natural language is very redundant in the sense that many different words share a common or similar meaning. For computer this can be hard to understand without some background knowledge. We describe an approach to visualization of text document collection based on methods from linear algebra. We apply Latent Semantic Indexing (LSI) as a technique that helps in extracting some of the background knowledge from corpus of text documents. This can be also viewed as extraction of hidden semantic concepts from text documents. In this way visualization can be very helpful in data analysis, for instance, for finding main topics that appear in larger sets of documents. Extraction of main concepts from documents using techniques such as LSI, can make the results of visualizations more useful. For example, given a set of descriptions of European Research projects (6FP) one can find main areas that these projects cover including semantic web, e-learning, security, etc. In this paper we describe a method for visualization of document corpus based on LSI, the system implementing it and give results of using the system on several datasets.

Povzetek: Predstavljena je vizualizacija korpusa besedil.

1 Introduction

Automated text processing is commonly used when dealing with text data written in a natural language. However, when processing the data using computers, we should be aware of the fact that many words having different form share a common or similar meaning. For a computer this can be difficult to handle without some additional information -- background knowledge. Latent Semantic Indexing (LSI) is a technique for extracting this background knowledge from text documents. It employs a technique from linear algebra called Singular Value Decomposition (SVD) and the bag-of-words representation of text documents for extracting words with similar meanings. This can also be viewed as the extraction of hidden semantic concepts from text documents.

Visualization of a document corpus is a very useful tool for finding the main topics that the documents from this corpus talk about. Different methods were proposed for visualizing a large document collection using different underlying methods. For instance, visualization of large document collection based on document clustering [3] , or visualization of news collection based on visualizing relationships between named entities extracted from the text [4] . Another example used in our work is visualization of European research space [5] . Given a set of descriptions of European research projects in IT (6th Framework IST), using document visualization one can find main areas that these projects cover, such as *semantic web, e-learning, security*, etc.

In automated text processing document are usually represented using the bag-of-words document representation, where each word from the document vocabulary stands for one dimension of the multidimensional space of documents. Consequently, in automated text processing we are dealing with very high dimensionality of up to hundreds of thousands dimensions. Dimensionality reduction [6] is important for different aspects of automated text processing including document visualization.

We propose to use dimensionality reduction for document visualization by first extracting main concepts from documents using LSI and than using this information to position documents on a two dimensional plane via multidimensional scaling [1]. The final output is graphical presentation of a document set that can be plotted on a computer screen. The proposed approach is implemented as a part of *Text Garden* software tools for text mining [7] [1] in a component providing different kinds of document corpus visualization based on LSI and multidimensional scaling.

[1] `http://www.textmining.net/`

This paper is organized as follows. Section 2 provides a short description of LSI and multidimensional scaling, while its application to document visualization is given in Section 3. Description of the developed system implementing the method is given in Section 4. Section 5 provides conclusions and discussion.

2 Building Blocks

First step of our approach to visualization of a document corpus is mapping all the documents into two dimensional space so we can plot them on a computer screen. Ideally they would be positioned in such a way that the distance between two documents would correspond to the content similarity between them.

We obtain this mapping by sending the document corpora trough the pipeline for reducing dimensionality, consisting from building blocks presented in this Section. The whole pipeline will be outlined in the Section 3.

2.1 Representation of Text Documents

The first step in our approach is to represent text documents as vectors. We use the standard Bag-of-Words (BOW) representation together with TFIDF weighting [9]. In the BOW representation there is a dimension for each word; a document is encoded as a feature vector with word frequencies as elements. Elements of vectors are weighted, in our case using the standard TFIDF weights as follows. The *i*-th element of the vector containing frequency of the *i*-th word is multiplied with $IDF_i = \log(N/df_i)$, where N is total number of documents and df_i is document frequency of the *i*-th word (the number of documents from the whole corpus in which the *i*-th word appears).

2.2 Latent Semantic Indexing

A well known and used approach for extracting latent semantics (or topics) from text documents is Latent Semantic Indexing [2]. In this approach we first construct term-document matrix A from a given corpus of text documents. This is a matrix with vectors of documents from a given corpus as columns. The term-document matrix A is then decomposed using singular value decomposition, so that $A = USV^T$; here matrices U and V are orthogonal and S is a diagonal matrix with ordered singular values on the diagonal. Columns of matrix U form an orthogonal basis of a subspace in the bag-of-words space where vectors with higher singular values carry more information -- this follows from the basic theorem about SVD, which tells that by setting all but the largest k singular values to 0 we get the best approximation for matrix A with matrix of rank k). Vectors that form the basis can be also viewed as concepts and the space spanned by these vectors is called the *Semantic Space*.

Each concept is a vector in the bag-of-words space, so the elements of this vector are weights assigned to the words coming from our documents. The words with the highest positive or negative values form a set of words that are found most suitable to describe the corresponding concept.

A related approach (not used here) that also aims at extracting latent semantics from text documents is Probabilistic Latent Semantic Analysis (PLSA) introduced in [8] . Compared to standard Latent Semantic Analysis which comes from linear algebra and performs a Singular Value Decomposition of co-occurrence tables, this method is based on a mixture decomposition derived from a latent class model. This method assigns each word a probability to be in a concept, where the number of concepts is predefined.

2.3 Dimensionality Reduction

We are using a sequential combination of linear subspace methods and multidimensional scaling for reducing document space dimensionality. Both methods can be independently applied to any data set that is represented as a set of vectors in some higher dimensional space. Our goal is to lower the number of dimensions to two so that the whole corpus of documents can be shown on a computer screen.

Linear subspace methods [10] , like Principal Component Analysis (PCA) or Latent Semantic Indexing, focus on finding direction in original vector space, so they capture the most variance (as is the case for PCA) or are the best approximation for original document-term matrix (as is the case for LSI). By projecting data (text documents) only on the first two directions we can get the points that live in the two dimensional space. The problem with linear subspace methods is that only the information from the first two directions is preserved. In case of LSI it would mean that all documents are described using only the two main concepts.

Multidimensional scaling [1] enables dimensionality reduction by mapping original multidimensional vectors onto two dimensions. Here the points representing documents are positioned into two dimensions so they minimize some energy function. The basic and most common form of this function is

$$E = \sum_{i\neq j}\delta_{ij} - d(x_i, x_j))^2,$$

where x_i are two dimensional points and δ_{ij} represents the similarity between two vectors (in our case documents i and j). An intuitive description of this optimization problem is: the better the distances between points on the plane approximate real similarity between documents, the lower the value of the energy function. Function E is nonnegative and equals zero only when distances between points match exactly with similarity between documents.

3 Visualization Using Dimensionality Reduction

We propose combining the two methods (linear subspace and multidimensional scaling) in order to take advantage of the nice properties they both have. What follows is description of the proposed algorithm:

Input: Corpus of documents to visualize in form of TFIDF vectors.
Output: Set of two dimensional points representing documents.
Procedure:

1. Calculate k dimensional semantic space generated by input corpus of documents,
2. Project documents into the semantic space,
3. Apply multidimensional scaling using energy function on documents with Euclidian distance in semantic space as similarity measure.

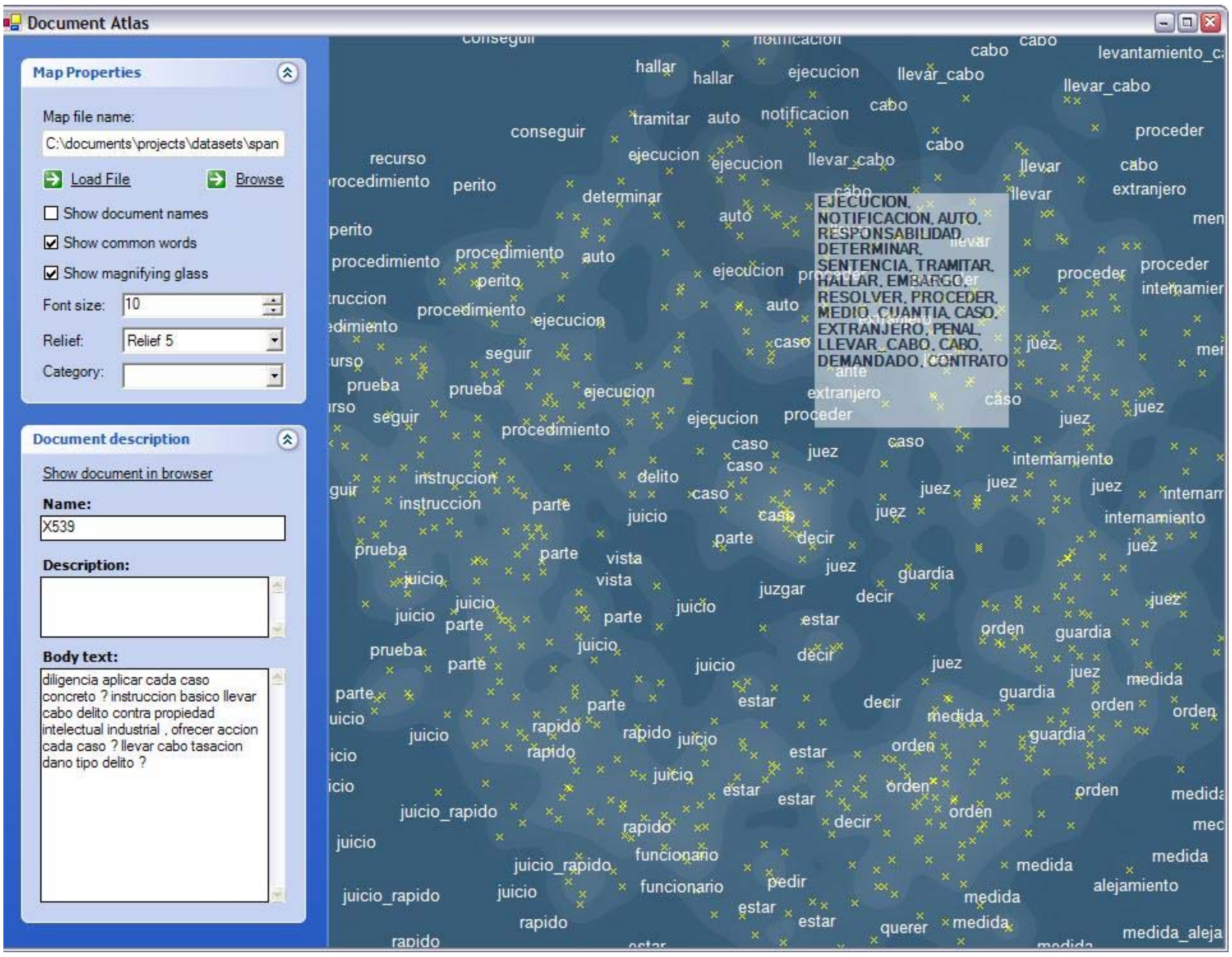
There are two main problems to be solved to make the above algorithm work efficiently. First problem is how to determine the value of k. The way we choose is by checking the singular values. Let $\Sigma_k = S_1^2 + S_2^2 + \ldots + S_k^2$, where S_i is i-th singular value. We know that .$\Sigma_n = \mathrm{Trace}(A^TA)$, where n is the number of the documents in the corpus and A is the term-document matrix. From this we can guess k by prescribing the ratio Σ_k / Σ_n to some fixed value, eg., 50%.

A more difficult problem is how to perform multidimensional scaling efficiently. Common way is to use gradient descent. The problem with this approach is that the energy function is not convex: it usually has many local minima which are not that interesting for us. One could start this method more times with different initial state and than choose the results with the lowest energy.

We choose a slightly different way which is based on reformulation of the energy function. Given a placement of points, we calculate for each point how to move it so we minimize energy function. Lets denote the current positions of points with (x_i,y_i) and the desired position with

$(x_i',y_i') = (x_i + \delta x_i, y_i + \delta y_i)$. Than we have

$$d_{ij}'^2 - d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 -$$
$$(x_i + \delta x_i - x_j - \delta x_j)^2 + (y_i + \delta y_i - y_j - \delta y_j)^2 \approx$$


Document Atlas

Map Properties
Map file name: C:\documents\projects\datasets\span
Load File | Browse
☐ Show document names
☑ Show common words
☑ Show magnifying glass
Font size: 10
Relief: Relief 5
Category:

Document description
Show document in browser
Name: X539
Description:
Body text:
diligencia aplicar cada caso concreto ? instruccion basico llevar cabo delito contra propiedad intelectual industrial , ofrecer accion cada caso ? llevar cabo tasacion dano tipo delito ?

EJECUCION, NOTIFICACION, AUTO, RESPONSABILIDAD, DETERMINAR, SENTENCIA, TRAMITAR, HALLAR, EMBARGO, RESOLVER, PROCEDER, MEDIO, CUANTIA, CASO, EXTRANJERO, PENAL, LLEVAR_CABO, CABO, DEMANDADO, CONTRATO

Figure 1 Visualization of questions. The dataset is a collection of around 800 most frequent questions asked by Spanish judges regarding the law and trials. Each question is treated as one document. Dataset is taken from a case study of the EU SEKT project.

$$= \approx (x_i - x_j)\,\delta x_i + (x_j - x_i)\,\delta x_j + (y_i - y_j)\,\delta y_i + (y_j - y_i)\,\delta y_j$$

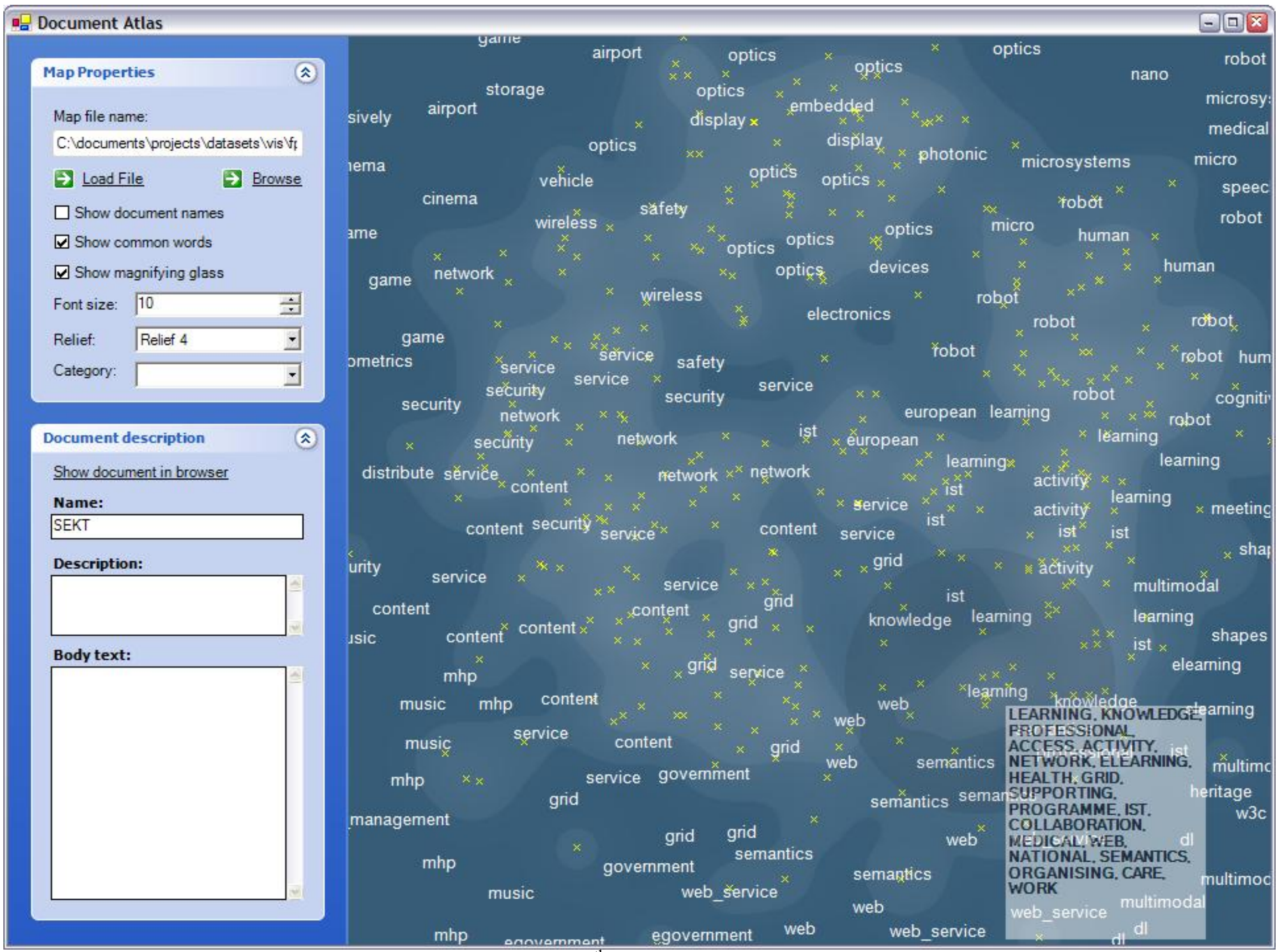
$$= [(x_i - x_j),\ (x_j - x_i),\ (y_i - y_j),\ (y_j - y_i)][\ \delta x_i,\ \delta x_j,\ \delta y_i,\ \delta y_j]^T.$$

By writing this for each pair *(i,j)* and substituting d_{ij}' with the original distance between *i*-th and *j*-th document we get a system of linear equations which has a vector of moves (δx and δy) for a solutions. This is an iteration which finds a step towards minimizing energy function and is more successful at avoiding local minima. Each iteration involves solving a linear system of equations with a very sparse matrix. This can be done very efficiently using Conjugate Gradient (CG) method. Finally, the points are normalized to lie in the square $K = [0,1]^2$.

4 Visualization Beyond Dimensionality Reduction

After the documents from corpus are mapped onto a two dimensional plane, some other techniques can be used to make the structure of documents more explicit for the users:

- ***Landscape generation***: landscape can be generated by using the density of points. Each point from square *K* is assigned height using the formula
  $h(x, y) = \sum_i \exp(-\sigma\, ||(x,y) - (x_i, y_i)||^2)$.
- ***Keywords***: each point from square *K* can be assigned a set of keywords by averaging TFIDF vectors of documents which appear within a circle with centre in this point and radius *R*.



Map Properties

Map file name: C:\documents\projects\datasets\vis\f

Load File Browse

☐ Show document names

☑ Show common words

☑ Show magnifying glass

Font size: 10

Relief: Relief 4

Category:

Document description

Show document in browser

Name: SEKT

Description:

Body text:

Figure 2 Visualization of European IST projects from 6th framework. Dataset consists of descriptions of EU IST projects taken from CORDIS web site. One can see in the visualization the main areas covered by the projects. The lower right side consists of projects about semantic web. In counter-clockwise direction the topics change to multimodal integration, e-learning, robotics, optics, safety, networking, grid computing, and than back to web related projects. These topics can be easily read from the map by checking the keywords. We can notice that besids putting similar documents together, the visualizations also puts similar topics more close on the map. Each document from the dataset corresponds to a description of one research project.

We use these features when showing visualizations to make them more descriptive and to improve the overall user experience.

In our system, called *Text Garden Document Atlas*, the documents are presented as yellow crosses on a map and the density is shown as a texture in the background of the map (the lighter the color, the higher the density). The most common keywords are shown for the areas around the map. Positions, for which keywords are computed, are selected randomly. Keywords are displayed using white color font. When the user moves the mouse around the map a set of the most common keywords is computed in real-time for the area around the mouse (the area, from which these keywords are extracted, is marked darker on the map and the list of keywords is shown in the semi-transparent window next to the mouse). The user can also zoom-in to see specific areas in more details. By clicking on a document (yellow crosses on the map), more information about it is shown on the left bottom side side of the screen.

Two examples of the visualizations can be seen in Figure 1 and 2. Figure 1 shows visualization of a corpus with questions from Spanish judges [8] . A map of the European IST projects from 6th framework is shown in Figure 2 and zoom-in on the part showing projects related to web and semantic web is shown in Figure 3..

5 Conclusions and Future Work

We have proposed a approach to efficient visualization of large data collections and describe the developed system implementing the proposed approach. The system was successfully used for visualizing different kinds of document corpora – from project descriptions, scientific articles to short questions from legal domain and even clients of an Internet grocery store. We found that the system is very helpful for data analysis offering quick insight into the structure of the visualized corpus. However providing a more systematic evaluation of the system including users questionnaire remains for the future work.

We will continue to use the user feedback as a guide for adding new features, which would make this tool even more informative and useful. One area not fully explored yet is the use of background relief in visual representation of the document corpus. Currently we use the relief to show the density of documents but it can also be used for showing some other attributes. Another direction we are considering for future work is to improve scalability by making the multi-dimensional step more scalable with the number of documents.

Acknowledgement

This work was supported by the Slovenian Research Agency and the IST Programme of the European Community under SEKT Semantically Enabled Knowledge Technologies (IST-1-506826-IP) and PASCAL Network of Excellence (IST-2002-506778). This publication only reflects the authors' views.

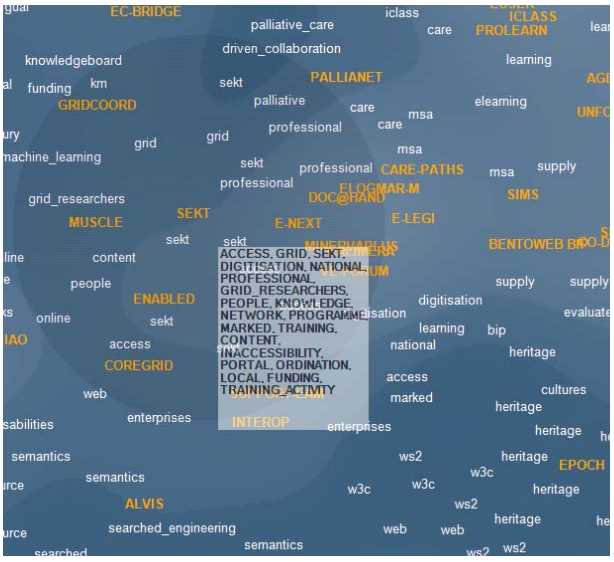


Figure 3 Zoom-in to projects related to web such as 6FP projects SEKT, MUSCLE, ALVIS, etc. Uppercase orange words are the project names (acronyms).

References

[1] Carroll, J.D., Arabie, P. Multidimensional scaling. In M.R. Rosenzweig and L.W. Porter (Eds.), Annual Review of Psychology, 1980, 31, 607-649.

[2] Deerwester, S., Dumais, S., Furnas, G., Landuer, T., Harshman, R. Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science, 1990.

[3] Grobelnik, M., and Mladenic, D.: Efficient visualization of large text corpora. Proceedings of the Seventh TELRI seminar. Dubrovnik, Croatia, 2002.

[4] Grobelnik, M., Mladenic, D. Visualization of news articles. Informatica journal, 2004, vol. 28, no. 4.

[5] Grobelnik, M., Mladenic, D. Analysis of a database of research projects using text mining and link analysis. In Data mining and decision support : integration and collaboration, Kluwer Academic Publishers, Boston; Dordrecht; London, 2003, pp. 157-166.

[6] Grobelnik, M., Mladenic, D. Text Mining Recipes, Springer-Verlag, Berlin; Heidelberg; New York (to appear), 2006, (accompanying software available at `<http://www.textmining.net>`.

[7] Mladenic, D. Feature Selection for Dimensionality Reduction. In Subspace, Latent Structure and Feature Selection techniques: Statistical and Optimisation perspectives, Springer Lecture Notes in Computer Science (to appear), 2006.

[8] Hoffman, T., Probabilistic Latent Semantic Analysis, Proc. of Uncertainty in Artificial Intelligence, UAI'99, 1999

[9] Salton, G. Developments in Automatic Text Retrieval, Science, Vol 253, pages 974-979, 1991.

[10] Shawe-Taylor, J., Cristianini, N. Kernel Methods for Pattern Analysis, Cambridge University Press, 2004, 143-150

[11] Vallbe, J.J., Marti, M.A., Fortuna, B., Jakulin, A., Mladenic, D., Casanovas, P. Stemming and lemmatisation, Springer Lecture Notes, (to appear), 2006.

CONTENTS OF *Informatica* Volume 29 (2005) pp. 1–505

Papers

NEDJAH, N. & L. DE M. MOURELLE. 2005. Multi-Objective CMOS-Targeted Evolutionary Hardware for Combinational Digital Circuits. Informatica 29:309–320.

NEDJAH, N. & L. DE M. MOURELLE. 2005. Efficient Pre-Processing for Large Window-Based Modular Exponentiation Using Ant Colony. Informatica 29:155–161.

OMRAN, M.G. & A.P. ENGELBRECHT, A. SALMAN. 2005. A Color Image Quantization Algorithm Based on Particle Swarm Optimization. Informatica 29:261–269.

ONG, S.L. & W.K. LAI, T.S.Y. TAI, C.H. OOI, K.M. HOE. 2005. Application of Ant-based Template Matching for Web Documents Categorization. Informatica 29:173–182.

PODGORELEC, V. & . 2005. Complexity-driven Evolution of Decision Graphs for Classification of Medical Data. Informatica 29:41–51.

POLANČIČ, G. & , M.HERIČKO, R.V. HORVAT. 2005. Open Source Software Usage Implications in the Context of Software Development. Informatica 29:483–490.

POTOČNIK, B. & D. HERIC, D. ZAZULA, B. CIGALE, D. BERNAD, T. TOMAŽIČ. 2005. Construction of Patient Specific Virtual Models of Medical Phenomena. Informatica 29:209–218.

RICCI, A. & , M. VIROLI. 2005. Coordination Artifacts: A Unifying Abstraction for Engineering Environment-Mediated Coordination in MAS. Informatica 29:433–443.

SUSI, A. & , A. PERINI, J. MYLOPOULOS, P. GIORGINI. 2005. The Tropos Metamodel and its Use. Informatica 29:401–408.

TOSHNIWAL, D. & R.C. JOSHI. 2005. Similarity Search in Time Series Data Using Time Weighted Slopes. Informatica 29:79–88.

TRENCANSKY, I. & , R. CERVENKA. 2005. Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS. Informatica 29:391–400.

TSENG, H.-W. & C.-C. CHANG. 2005. A Very Low Bit Rate Image Compressor Using Transformed Classified Vector Quantization. Informatica 29:335–341.

VAINIO, A. & , K. SALMENJOKI. 2005. Improving Study Planning with an Agent-based System. Informatica 29:453–459.

VIZINE, A.L. & L.N. DE CASTRO, E.R. HRUSCHKA, R.R. GUDWIN. 2005. Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm. Informatica 29:143–153.

WANG, D. & X. MA. 2005. A Hybird Image Retrieval System with User's Relevance Feedback Using Neurocomputing. Informatica 29:271–279.

WANG, X.-Y. & J.M. GARIBALDI. 2005. Simulated Annealing Fuzzy Clustering in Cancer Diagnosis. Informatica 29:61–70.

WEYNS, D. & , T. HOLVOET. 2005. On the Role of Environments in Multiagent Systems. Informatica 29:409–421.

ZHANG, J. & Q. WU, Y. WANG. 2005. A New Efficient Group Signature With Forward Security. Informatica 29:321–325.

ZHANG, J. & W. ZOU, D. CHEN, Y. WANG. 2005. On the Security of a Digital Signature with Message Recovery Using Self-certified Public Key. Informatica 29:343–346.

ZITAR, R.A. & A. AL-JABALI. 2005. Towards Neural Network Model for Insulin/Glucose in Diabetics-II. Informatica 29:227–232.

ŽIBERT, A. & , V. BATAGELJ, V. RAJKOVIČ. 2005. Comparative Analysis of Educational Networks. Informatica 29:477–481.

DE FRANÇA, F.O. & F.J. VON ZUBEN, L.N. DE CASTRO. 2005. Max Min Ant System and Capacitated p-Medians: Extensions and Improved Solutions. Informatica 29:163–172.

Editorials

ABONYI, J. & A. ABRAHAM. 2005. Introduction. Informatica 29:1–2.

NEDJAH, N. & L. DE M. MOURELLE. 2005. Introduction. Informatica 29:123–124.

EGIAZARIAN, K. & A.E. HASSANIEN. 2005. Introduction. Informatica 29:251–252.

OMICINI, A. & , P. PETTA, M. GAMS. 2005. Introduction. Informatica 29:377–378.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slove**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in `.eps` format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Informatica WWW:

http://www.informatica.si/

Informatica

An International Journal of Computing and Informatics

Archive of abstracts may be accessed at America: http://ocean.ocean.cs.siu.edu/informatica/index.html,
Europe: http://www.informatica.si/, Asia: http://www3.it.deakin.edu.au/ hdai/Informatica/.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2005 (Volume 29) is
– 60 EUR (80 USD) for institutions,
– 30 EUR (40 USD) for individuals, and
– 15 EUR (20 USD) for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printed by Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. Drago Torkar, Jožef Stefan Institute: Tel (+386) 1 4773 900, Fax (+386) 1 219 385, or send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova Ljubljanska Banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, ACM Digital Library, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Cybernetica Newsletter, DBLP Computer Science Bibliography, Engineering Index, INSPEC, Linguistics and Language Behaviour Abstracts, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 29 Number 4 November 2005 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **29** (2005) Number 4, pp. 377–505