Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Methods for Development of Industrial Multi-Agent Systems

DOCTORAL THESIS

2015 Ing. Petr Kadera

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Methods for Development of Industrial Multi-Agent Systems

by

Ing. Petr Kadera

Supervisor: Doc. Ing. Pavel Vrba, Ph.D.

Dissertation submitted to the Faculty of Electrical Engineering of
Czech Technical University in Prague
in partial fulfillment of the requirements
for the degree of

Doctor

in the branch of study
Artificial Inteligence and Biocybernetics
of study program Electrical Engineering and Informatics

2015

Acknowledgements

First and foremost, I would like to thank my wife Jana who encouraged me and believed in me during the long path to my PhD.

I would like to thank Prof. Vladimír Mařík who guided and supported me during nearly all of my academic and professional life. His ability to immediately see the root of a problem and distinguish the promising approaches from fallacy was a priceless support as well as his willingness to find a slot for his students in his overbooked schedule.

I want to thank Associate Prof. Pavel Vrba, my supervisor, who supported me all the way during my PhD.

I want to thank Pavel Tichý for introducing me the world of industrial agents.

I want to thank Vašek Jirkovský and Petr Novák, my colleagues and friends, for the fruitful discussions we had.

Petr Kadera

Czech Technical University in Prague
Prague, 2015

Methods for Development of Industrial Multi-Agent Systems

Ing. Petr Kadera

Czech Technical University in Prague, Prague, 2015

Supervisor: Doc. Ing. Pavel Vrba, Ph.D.

From the industrial point of view, the beginning of the 21^{th} century is predominated by fast-changing demands of the global market. This phenomenon is referred to as a shift from mass production to mass customization and brings new challenges into the manufacturing domain. The new research and engineering issues have been reflected also by HMS initiative, which provided the basic concepts of holonic and agent-based manufacturing. However, there was still lack of complex methodology and supportive tools that would provide guidelines to developers of industrial large-scale agent-based systems.

The first part of this work proposes a set of methods integrated into a tool named Agent Development Environment (ADE) which is designed to fill this gap. The tool guides the developer through the entire development process starting with implementation of agent templates consisting of high-level parts written in an object-oriented language (currently JAVA and C++ are supported) and low-level parts (IEC 61131) that provide real-time responsiveness on local level. Consequently, the requested number of agents is instantiated. The approach proposed within this work guarantees the consistence of the high- and low-level parts. The tool also composes the hardware configuration from hardware components. Then the agents are simply assigned to selected computational units (Programmable Logic Controllers, PCs) and the ADE guarantees a correct setup of all communication links that represent an information backbone of the distributed solution.

Although the ADE ensures development of a syntactically error-free MAS,

meeting the performance requirements is still an open question, which is closely connected to the assignment of software agents to the hardware components.

The second part of the thesis proposes an approach that utilizes performance models to estimate the performance limits of various software/hardware configurations. The approach is based on observation of the emergent behavioral patterns among agents. Consequently, the observation is transformed into a Queueing Network performance model, which enables fast experiments and provides deep insight into the inner states of individual components. Therefore, it enables to speed up the configuration of a new MAS and to verify that the selected configuration meets the performance requirements.

The final part of the thesis introduces a congestion management mechanism that continuously observes the load of individual system components and, if necessary, postpones the newly arriving requests in order to guarantee the system responsiveness. Unless such an approach is implemented, the danger of decreased responsiveness makes setting of communication timeouts difficult.

Metody pro vývoj průmyslových multi-agentních systémů

Ing. Petr Kadera

České Vysoké Učení Technické v Praze, Praha, 2015

Školitel: Doc. Ing. Pavel Vrba, Ph.D.

Průmysl počátku 21. století je ovlivňován rychlými změnami požadavků na globalizovaném trhu. Tento jev bývá nazýván posunem od hromadné výroby k hromadné tvorbě zákaznických produktů, což přináší nové výzvy do oblasti výrobních systémů. Tyto nové výzvy byly reflektovány vznikem iniciativy zaměřené na výzkum holonických výrobních systémů. Výstupem této iniciativy bylo představení základních konceptů holonických a multi-agentních výrobních systémů. Navzdory již vynaloženému úsilí je zde stále nedostatek ucelených metodických doporučení a podpůrných nástrojů, které by usnadnily tvorbu holonických a multi-agentních výrobních systémů velkého rozsahu.

První část této práce popisuje sadu metod integrovaných do nástroje nazvaného Agent Development Environment (ADE), který byl navržen pro zaplnění této mezery. Tento nástroj vede tvůrce multi-agentního systému skrz celý vývojový proces začínající návrhem agentních šablon tvořených jednak popisem vysoko-úrovňové části agentní logiky vytvořené v objektově orientovaném jazyce (JAVA nebo C++) a jednak nízko-úrovňovými částmi vytvořenými pomocí standardu IEC 61131, který poskytuje schopnost řízení v reálném čase. Dalším krokem je vytvoření jednotlivých instancí agentů na základě šablon. Přístup navržený v této práci zajišťuje, že vysoko- a nízko-úrovňové části agentů jsou vytvořeny konzistentně. ADE slouží i ke konfiguraci hardwarové části daného systému z jednotlivých hardwarových komponent. V další fázi jsou jednotliví agenti přirazeni výpočetním zdrojům, což jsou např. programovatelné logické kontroléry nebo os-

obní počítače, pro které ADE vygeneruje výsledný kód, kde jsou mimo jiné automaticky vytvořené komunikační spoje mezi jednotlivými částmi výsledné aplikace.

Přestože tento nástroj umožňuje vytvořit multi-agentní systémy bez syntaktických a implementačních chyb, zajištění splnění výkonových požadavků, které jsou kladeny na výsledný multi-agentní systém, nijak zajištěno není. Proto druhá část této práce představuje metodu, která využívá výkonové modely (angl. performance models) pro odhad výkonových limitů různých konfigurací systému. Představená metoda je založena na pozorovaní komunikace mezi agenty, z čehož je následně vytvořen výkonový model založený na modelovacím přístupu nazvaném sítě front (z angl. Queueing Networks), který umožňuje rychlé provedení testů různých konfigurací systému a poskytuje vhled do dění na úrovni jednotlivých agentů (např. jejich zatížení nebo délky front příchozích požadavků).

Závěrečná část této práce popisuje systém chránící multi-agentní systémy před zahlcením požadavky. Tento systém nepřetržitě sleduje vznik nových úkolů řešených sledovaným multi-agentním systémem a pokud je to nutné, začátek řešení nových úkolů oddaluje, aby nedošlo k neúměrnému prodloužení času odezvy celého systému. V případě, že takový přístup není použit, prodlužování času odezvy, ke kterému dochází s růstem zátěže systému, značně ztěžuje správné nastavení časových limitů pro komunikaci mezi jednotlivými agenty.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The mass production area was predominated by big investments in customized equipment, in order to produce large quantities of identical products faster and cheaper. The changing environment of today's market raises the need for new production planning and production control models and requires new approaches for production lines and intelligent machines to provide stability, sustainability and economy under such production conditions. It is important to be agile and react fast with minimum risk to the sudden and unpredictable changes of requirements. Adaptive, reconfigurable and modular production systems address these requirements allowing machines and plants to flexibly adapt themselves to changing demands and interact with each other to fulfill the overall production goal, as stated in the Manufacture Strategic Research Agenda (Commission, 2007).

While physical components of such systems are available, the implementation of reconfigurable systems in the manufacturing industry is hindered by the lack of knowledge-based methods and intelligent tools for their optimal deployment and control of their operation. Currently available methods and tools are based mostly on traditional techniques applied in flexible manufacturing systems and are quite straightforward, addressing specific problems and lacking intelligence and learning capabilities. Moreover, their application takes place off-line, requiring significant down times as well as human interference. In order to reach the full potential of such systems, the adaptation of the system's performance to an optimal ma-

nufacturing solution for the appointed task needs to take place autonomously, in real-time and with as little human involvement as possible. Thus, the development of an autonomous intelligent governing system for adaptive and reactive manufacturing systems is of outmost importance (Commission, 2003), (Commission, 2007).

Holonic and Multi-Agent Systems (MASs) have been widely recognized as enabling technologies for designing and implementing next-generation of distributed and intelligent industrial automation systems (Bussmann et al., 2004). These systems are characterized by high complexity and requirements for dynamic reconfiguration capabilities to fulfill demands for mass customization, yet low-volume orders with reduced time-to-market. Self-diagnostics and robustness that allow efficient continuing in operation even if a part of the system is down are other important properties.

The trend of multi-agent systems applications is apparent at all levels of the manufacturing business. At the lowest, real-time control level, so called holons or holonic agents are usually tightly linked with the real-time control programs (implemented in IEC 61131-3 or IEC 61499 standards) through which they can directly observe and actuate the physical manufacturing equipment (Brennan et al., 2008). Intelligent agents are also used for production planning and scheduling tasks both on the workshop and factory levels (Pěchouček et al., 2007). More generic visions of intensive cooperation among enterprises connected via communication networks have led to the ideas of virtual enterprises (Camarinha-Matos, 2002).

Common principles in industrial deployment of the agent technology are the distribution of decision-making and control processes among a community of autonomously acting and mutually cooperating units – agents. At the shop floor level, for instance, an agent represents and independently controls a particular physical equipment, like a CNC machine, conveyor belt or docking station. The substantial characteristic is the cooperation among the agents as they pursue either their individual goals or the common goals of the overall control system. The inter-agent interactions vary from simple information exchanges, for example about the state of processing as the product moves from one machine to another, through requests to perform a particular operation, for example requesting an automated guided vehicle to transport a product to a particular work station, to complex

negotiations based on contract-net protocol or another auction mechanisms.

Distributed control systems provide many advantages such as efficient resource utilization, lower hardware investments and higher robustness. Despite these obvious advantages, the exploitation of the developed concepts in industrial practice is still very low. The decision makers are reluctant to take the risk of being the first adopters of this technology at large scale due to many negative factors, such as higher investments, anxiety over effects of emergent behavior, and lack of skilled maintenance personnel (Hall et al., 2005). Thus, classical centralized and hierarchical control architectures represented mainly by programmable logic controllers (PLCs) and their respective programming languages of the IEC 61131-3 Standard are still predominantly used in this area.

All methods introduced within this work have been designed with respect to the future transfer in the practice. The easiest way, at least up to now, is to preserve as much from current architectures, including mainly PLCs, as possible and build the agent layer on top of it. This keeps the most valuable feature of the control systems: the guaranteed real-time local responsiveness on the level of individual PLCs that directly control physical components via actuators using data from the local sensors. It is apparent that any degradation of the responsiveness on this level is not acceptable for safety reasons.

Although the responsiveness on the global level is not a critical parameter in terms of safety, it is a key non-functional parameter since large scale distributed systems suffer from performance problems, such as long response times or overloaded computational units. Frequently, the reason for the performance degradation is inappropriate architecture of the distributed system rather than the inefficient implementation of the components. Performance modeling and evaluation techniques provide ways to overcome these obstacles to benefit from the promising advantages. Unfortunately, the design of performance models is a complex and time consuming task. Therefore, this thesis proposes methods that enable to create the performance models from the logs of messages automatically.

Although performance models are capable tools for performance and capacity planning, they do not provide protection against temporal bursts of events that are hardly predictable and change the responsiveness of the global system significantly. This thesis proposes a method for online monitoring of the actual system load,

detecting possible bottlenecks and if necessary a slow-down of the arriving requests in order to prevent the system against entering a saturated operational regime.

1.1 Definition of Terms

The terms such as agent or multi-agent system are currently very popular among researchers and software engineers, but frequently the meaning of these terms lacks an explicit definition. The reason for various use and interpretation of these terms is the wide application area of agent-based approaches. Agents are either used for modeling and simulation of complex systems or for control of complex processes. In order to avoid the ambiguity, next subsections defines the meaning of these terms as used throughout this dissertation that is solely as agents designed for control of industrial processes.

1.1.1 Agent

As there are many application areas for agents, there are also many variations of the term agent, e.g., *software agent*, *intelligent agent*, *autonomous agent*, and *social agent*. Some of the proposed agent definitions follows:

- "An agent is a self-contained problem-solving system capable of autonomous, reactive, proactive, and social behavior" (Wooldridge and Jennings, 1999).
- "An agent is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives" (Jennings et al., 1998).
- "An agent is an object that can say go (dynamic autonomy) and no (deterministic autonomy) (Odell et al., 2001)"
- "An intelligent software agent is a program that acts on behalf of their human users in order to perform laborious information gathering tasks, such as locating and accessing information from various on-line information sources, resolving inconsistencies in the retrieved information, filtering away irrelevant or unwanted information, integrating information from heterogeneous

information sources and adapting over time to their human users' information needs and the shape of the Infosphere (Sycara et al., 1996)".

- "An agent is a software entity that has enough autonomy and intelligence to carry out various tasks with little or no human intervention" (Wong and Sycara, 1999).

In the context of this work, the best definition of an agent comes from Maturana (Maturana et al., 2003) and says:

"An agent is an autonomous unit that is able to interact with its environment in an intelligent manner."

The main advantage of this definition is that it does not perceive the agent only as a piece of software, but as a complex unit that consists of both software and hardware parts.

1.1.2 Multi-Agent System

A general definition of multi-agent system comes from Jennings, Sycara, and Wooldridge (Jennings et al., 1998). They simply define an agent-based system as environment where the agent abstraction is utilized. Beside the definition, they provide a set of characteristics that are essential for any multi-agent system:

- each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- there is no global system control;
- data is decentralized; and
- computation is asynchronous.

Systems considered within the framework of this thesis meet all these requirements.

1.2 From Flexible to Agent-Based manufacturing systems

Holonic Manufacturing Systems (HMS) are mainly results of work done within the framework of the Intelligent Manufacturing Systems (IMS) programme[1], founded on international cooperation in research and development in manufacturing technology. The programme started with several projects in 1993. The projects were launched as feasibility studies, which were lately followed by the full-scale research programme.

This study conceptualized the abstract of holons as well as formulated the holonic ideas quite soundly. The first phase was focused on the development of foundations for generic technologies and the second phase continued with the demonstration of the potential for distributed systems including physical equipment, manufacturing work cells, factories, and supply chains.

1.3 Main Characteristics of Industrial Multi-Agent Systems

Design and implementation of MASs and holonic systems are usually very complex tasks due to their distributed nature. Agents deal only with a partial knowledge of the whole system and are responsible for specific sections of this system. This can be both an advantage and a disadvantage at the same time. Let us briefly summarize the most important ones.

Major advantages are:

- An agent can act on its own when it is disconnected from the rest of the system (or from its part).
- An agent can be replaced or backed up by another agent; there can be multiple implementations even using different languages to enhance fault tolerance.

[1] http://www.ims.org

- Agents can flexibly enter or leave the system if necessary according to demand (i.e., the system can be scaled and adapted automatically if necessary).
- One implementation of an agent can be reused multiple times.

On the other hand, major disadvantages are:

- The development of reusable agent types (i.e., a library of agents) is more complex.
- The partitioning of the system into parts is often unclear.
- It is difficult to ensure even soft real-time capabilities of the system.
- Agent systems make extensive use of communication facilities to exchange information among distributed units that might cause undesirable network overload.
- It is more difficult to debug and test a massively parallel and distributed system, covering all possible interactions, than it is in the case of a sequential and centralized one.

Specific attributes of industrial MASs:

- Real-Time behavior;
- Robustness;
- Clearly definable scalability;
- Traceability;
- Physical more than Logical Distribution;
- Agentification of Legacy components.

1.4 Goals of this Thesis

This thesis is aimed at issues related to the design, development and deployment of industrial MASs in order to create robust industrial MASs in shorter time. In particular, our goal is to fulfill the following subgoals:

- Develop methodology and the corresponding supportive tool that guides system engineers through the entire development process;
- Develop a method in order to verify that the final application deployed on a specific hardware meets the performance requirements;
- Develop a method in order to avoid uncontrolled overloads of hardware resources caused by bursts of incoming requests.

1.5 A Bird's-Eye View of Chapters Contents

The thesis consists of the following chapters:

Chapter 1 briefly introduces the research field of MASs with the focus on the specifics of the industrial applications.

Chapter 2 provides an overview of industrial applications of MASs. It describes both the main application areas as well as concrete applications of the MAS technology.

Chapter 3 overviews the already existing methods and tools addressing similar goals as this thesis, i.e., to enhance development process and robustness of industrial MASs.

Chapter 4 describes the Agent Development Environment – a tool that guides a designer of a MAS through the entire development process.

Chapter 5 introduces common performance modeling notations used in software engineering, enumerates their positive and negative effects and gives reasons for the performed selection.

Chapter 6 provides a brief insight into the specifics of communication used in industrial MASs.

Chapter 7 introduces a process of Queueing Networks compilation using a log of timestamped messages that were sent among the agents as an input as well as presents a utilization of such models for identification of system performance indices, which cannot be observed directly, is presented.

Chapter 8 proposes an extension of a regular Directory Facilitator, which manages the social knowledge within a MAS, by a performance modeling mechanism that enables to identify a danger of entering a saturated operational regime. When

such a situation occurs, the execution of newly arrived requests is postponed by delaying the creation of social links between providers and consumers of the requested services.

Chapter 9 summarizes the achievements and proposes the possible future research directions for further development of the proposed methods.

Chapter 2

Industrial Applications of Agents

2.1 Introduction

The penetration of the agent-based technology in the industrial domain is a long-lasting process that started more than two decades ago. Agents are perceived as an alternative way of designing and implementing the industrial control systems. Traditionally, the control systems are organized in a hierarchical and centralized way as can be seen in the popular automation pyramid (see Fig. 2.1). Focusing on the two bottom levels of the pyramid – sensors and actuators on the lowest one and PLCs (Programmable Logic Controllers) in the level above – highly centralized and monolithic architectures are traditionally applied. Typically, one PLC administers large parts of the controlled system. It means that it receives signals from dozens, hundreds or even thousands of sensors, executes the control logic and then sends the control commands to the actuators. The control logic is usually implemented using the PLC programming languages of the IEC 61131-3 standard that follow rather procedural than component and object oriented approach. As a result the control application is monolithic and cumbersome, having the principal responsibility for processing a huge number of information coming from various

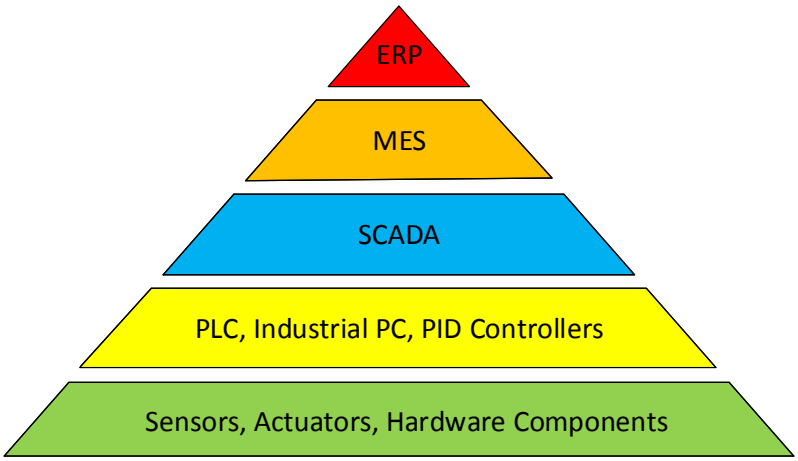


Figure 2.1: Automation Pyramid

parts of the controlled system to one place.

In this chapter we introduce several industrial domains that might benefit from adoption of agent-based technology and we also briefly introduce some of pilot deployments of multi-agent applications.

2.2 Networks

One of the industrial domains with the highest potential of adoption of agents is management of networks. Many networks are inherently distributed and reconfigurable. Frequently, the network is interconnected by different paths and therefore the selection of the best route has to be done dynamically at run-time. All these requirements are met by agents. In the remainder of this section, we will describe three different application fields. The first one is an example of a dynamic routing mechanism applied for the control of a Chilled Water System (CWS) at a battle ship. The second one is a general introduction of possible application in electric energy network named Smart Grids. The last one describes application of agent-based control for product transportation and material handling on shop-floor level.

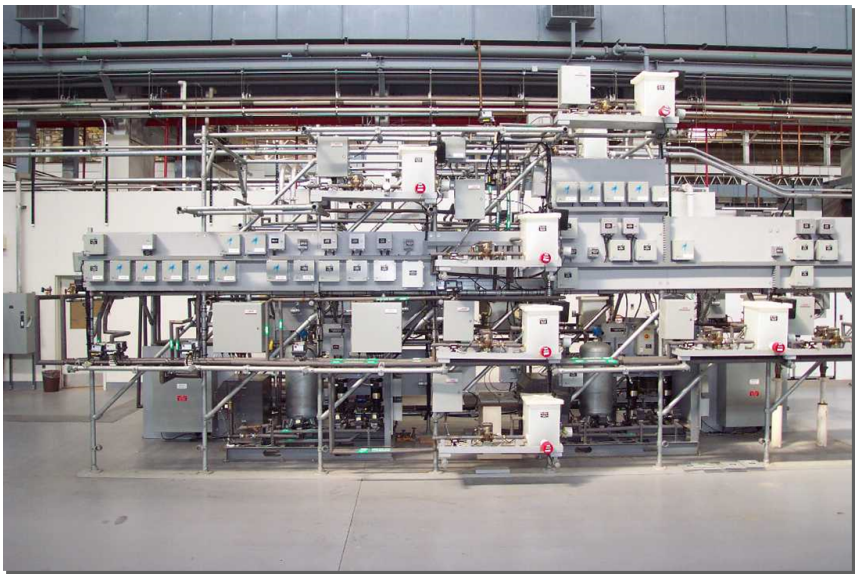


Figure 2.2: Reduced Scale Advanced Demonstrator

2.2.1 Chilled Water System

Chilled Water System (CWS) is a pilot system that is based on the Reduced Scale Advanced Demonstrator (RSAD) model (see Fig. 2.2), i.e., a reconfigurable fluid system test platform. The RSAD has an integrated control architecture, which includes RA technology for control and visualization. The physical layout of the RSAD chilled water system is a scaled-down version from a real U.S. Navy ship (see Fig. 2.3).

The first version of the CWS had agents implemented in C++ and LLC in relay ladder logic. This implementation has been successfully tested on the RSAD CWS. The second version of the CWS utilizes both C++ and Java versions of ACS and an Agent Development Environment (see Chapter 4) that has been developed as an Eclipse plug-in. It emphasizes fast reconfiguration of the system to speed up the development and debugging process. It is done by integrating design, control, visualization, and simulation in a single application that can be tested on a PC. The particular parts are kept independent and communicate via a data table. The table is a real-data table in a controller or a Java data table emulator developed by RA. This approach enables a flexible change from a simulation mode to deployment

Figure 2.3: U.S. Navy Battle Ship

on a real system.

The system configuration (see Fig. 2.4) that is considered in our application consists of two chillers (marked with 1), 11 services (marked with 2), and 32 valves (marked with 3) that are connected via the water piping system (pipes and pipe connectors called routers). This configuration is the same as in the RSAD CWS. Agents of type "chiller" offer cooling, agents of type "service" require cooling, and agents of type "valve" connect or split segments of the water piping system. Agents take actions to dynamically create cooling paths, which enable "service" agents to receive cold water from chillers. The system also provides the capability to find leakage, isolate broken parts, and find new cooling paths if they exist to keep other parts of the system sound.

Providing cooling in this application is reduced to creating paths that connect services and chillers. The components are connected when paths for cold and hot water exist. Hot water needs to be returned to the same chiller to keep an equal volume of produced and received water. Paths are composed of pipe sections connected by opening the relevant valves. Every valve has its own opening price and each chiller has its price for switching on. The total cost of a path is the sum

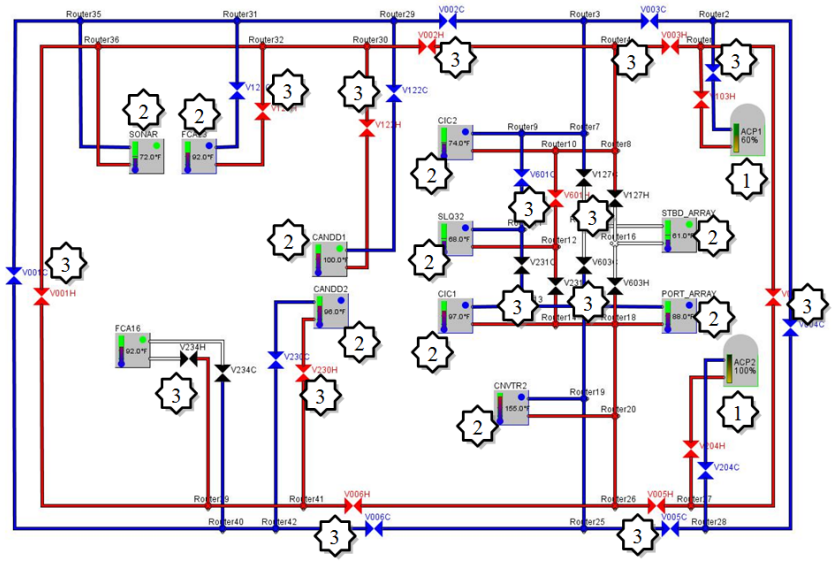


Figure 2.4: CWS configuration

of involved agent costs.

Since the cooling requirements of different services might differ, they are described with a number. The number is one of the parameters used for a service agent and is called “load”. All services have this parameter set to 20 in this application. In a similar way, chiller agents have the parameter “capacity” describing the maximum amount of cooling the chiller is able to produce. Both chillers contained in our application have set this parameter to 100.

Temperature of a service agent is its main parameter which influences its actions and that is why it was necessary to simulate it. Every service agent is accompanied by its own simulation component instance running in a separate thread. When the service agent has created a cooling contract, the actual temperature is decreased by one Fahrenheit degree every second to simulate the cooling; otherwise it is increased by one Fahrenheit degree to simulate heating up. This model represents only a rough simulation, which is, however, sufficient for the purposes of testing the MAS behavior. For more precise testing, it is possible to use the detailed MATLAB simulation (Maturana et al., 2005) that mimics in detail the functionality of the U.S. Navy’s Reduced Scale Advanced Demonstrator.

To test the water leakage isolation algorithms, a simulation of the water pressure measured by chillers was created. If the leakage identified by the pressure drop is detected, the agents collectively take appropriate action to locate the leak and isolate it by closing the appropriate valves. If there is an active water path used for cooling a device that goes through the isolated section, the agents find an alternative route for the water to continue in cooling.

2.2.2 Smart-Grids

The ongoing changes on the electricity market involving increasing penetration of Renewable Energy Resources (RESs) and Distributed Energy Resources (DERs) in the energy domain require new control approaches because the traditional hierarchical layout cannot meet the emerging requirements. The integration of DER in the European electricity network is strongly supported by the EU energy policy. It is expected that in 2020 the total contribution of DER-based generation could exceed 20% (*European SmartGrids technology platform: vision and strategy for Europe's electricity networks of the future*, 2006). Moreover, "Energy and climate change package" proposed by the European Commission in 2007 declares the energy targets for EU; 20% reduction of greenhouse gas emissions, 20% of renewable energy sources, and 20 % of the overall energy consumption reduction by year 2020. Some major players such as Germany set even more ambitious targets; 35% of renewable energy till 2030 and even 80% till 2050. However, the adoption of RESs and DERs remains slow, mainly due to the issues related to technology, market, and regulation (Mohd et al., 2008). The report of International Energy Agency (*World Energy Outlook 2013*, n.d.) illustrates the global nature of these issues since similar activities are observed all around the world.

The traditional electrical grid is a hierarchically organized system (see Fig. 2.5). The production of electric energy is provided by large-scale power plants – bulk generation, and is adjusted to the consumption. High voltage transmission lines (>100kV) transport the generated energy over long distances. Step-down transformers are used to reduce the high to medium voltage used on the level of the distribution grids (1 – 100 kV). Further voltage reduction is applied for the local distribution operated on low voltage (<1kV). This traditional organization

Figure 2.5: Organization of the traditional electric energy infrastructure

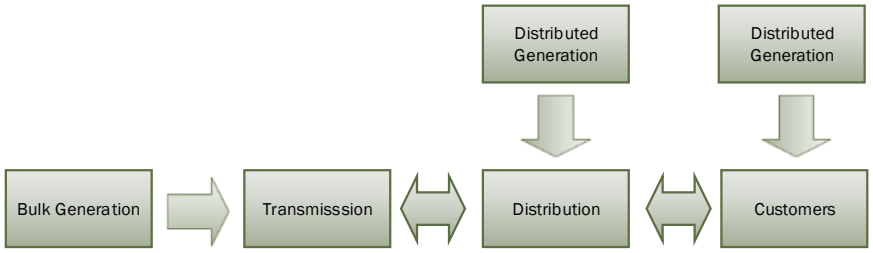


Figure 2.6: Organization of the future electric energy infrastructure

is purely hierarchical and characterized by unidirectional energy flow from the generation to the customer level.

The growing integration of RESs and DERs into electric energy infrastructure changes the traditional schema. Although the general architecture of the infrastructure preserves the hierarchical characteristics, the energy flow is at some levels bidirectional (see Fig. 2.6). The RES/DER generators are connected to the low voltage and medium voltage distribution grids causing fluctuations in the distribution network. Not only the advent of RESs/DERs but also the growing utilization of Electric Vehicles (EVs) and enhanced availability of controllable loads such as heat pumps or batteries bring new challenges that have to be tackled by flexible control approaches.

The Strasser (Strasser et al., 2013) provides the following list of the most important features of the automation systems for the future electric grid:

- Self-healing: Automatic restoration of grid operation in case of faults/errors;
- Self-optimization: Ability to optimize the grid operation due to fluctuating generation from RES and the availability of controllable loads/storages;
- Self-monitoring and diagnostics: Advanced monitoring and state estimation

capability; real-time or near real-time based condition monitoring of the grid components and devices;

- Condition dependent maintenance: Preventive maintenance according to component condition and remaining life-time;
- Automatic grid (topology) reconfiguration: Automatic adjustment of the grid topology for grid optimization (e.g., max. amount of DESs and EVs) or fault management and system restoration;
- Adaptive protection: Automatic adaption of protection equipment settings due to actual grid condition (e.g., adaptation of the protection system settings due to the bidirectional power flow);
- Demand response support: Advanced energy management taking distributed generation and controllable loads/storages into account;
- Distributed management: Distributed management and control with automatic decision finding process and proactive fault/error prevention;
- Distributed generators with ancillary services: Possibility to use ancillary services (e.g., voltage/frequency control) of DERs for grid optimization;
- Advanced forecasting support: Forecasting of generation and load profiles for optimized grid operation.

The fulfillment of the listed features requires a flexible approach that supports the dynamical reconfigurability. Thus agent-based systems are a promising technology to serve this task.

2.2.3 Product Transportation and Material Handling

The task of the discrete material flow control system is to deliver workpieces (products, parts, raw material, etc.) in a complex and redundant network of transportation routes from a location A to a location B. Usually, the transportation routes are in form of conveyor lanes with diverting and merging elements referred to as diverters and intersections, respectively. Classical approach used in factories

and warehouses today is the centralized global routing control. It means there is a single decision making component (PLC) that maintains the routing tables for each diverter and uses them to route the incoming workpieces according to their destination. The destination is determined on the basis of the workpiece's identity checked by a barcode or RFID reader located in front of each diverter. The routing tables are computed for given topology of the network using standardized path-planning algorithms, which involve some graph or tree-search algorithms, such as Dijkstra (Dijkstra, 1959). This is too complex task to be carried out in real-time by the PLC due to the limited capabilities of the IEC 61131 languages. Therefore, the path planning is done in advance and the resulting routing tables are set to the PLCs as fixed for given network topology. Obviously it is a cumbersome solution that fails in case the topology changes due to the faulty components (conveyors or diverters) or when the transportation system should be extended by adding or changing the routes.

As argued in (Sallez et al., 2009) there are only few works on the application of multi-agent systems to the distributed material-routing control. The agent-based solution for the airport baggage-handling system that relies on a single Route agent that holds the global view of the network topology and computes the optimal routes is presented in (Hallenborg and Demazeau, 2006). Another conveyor-based baggage handling is presented in (Black and Vyatkin, 2007). The routing paths are determined online by propagation of messages along the routes among particular conveyor agents. In the Production 2000+ project manufacturing system is composed of standardized modules consisting of a machine, three one-way conveyors, and two switches. An agent associated with each workpiece negotiates with the next switch agent about the routing along the shortest path. The switch agent can select an alternative route if a capacity bottleneck is detected (Bussmann and Schild, 2000). In the agent-based system for wafer fabrication production called FABMAS the agents ensure the routing of lots with wafers between different groups of parallel machines (Mönch et al., 2003).

Searching for optimal paths in the conveyor network can be generally viewed as a graph-search problem where work cells (points between which the products are transported) and diverters represent nodes of the graph and the conveyors are the valued edges of the graph. There is a well-known Dijkstra's algorithm (Dijkstra,

1959) and its extensions like the best first search (Pearl, 1984) or A* (Hart et al., 1968). There are numerous path-searching methods inspired by complex behaviors of biological societies, like ant colonies. The mobile agents move throughout the environment while leaving the pheromone signs at specific locations to mark the trail. This approach is popular, for instance, in telecommunication and ad hoc wireless-network routing (Di Caro and Dorigo, 1998), (Kumar and Cole, 2005) or mobile robotics (Wagner et al., 1999). The application of these principles in manufacturing domain is presented for instance in (Valckenaers et al., 2004) and (Sallez et al., 2009). In the latter case the virtual active products move in the virtual environment and leave the pheromone signs in form of routing tables at decision points. The physical active products then follow the best path in the real world using the signs left in the virtual world.

2.3 Planning and Scheduling for Industrial Enterprises

At present, almost all of the project-oriented enterprises are facing the following problem: How to set the optimal production plan to effectively exploit the company resources while reaching the highest turnover? To keep business running and to avoid such critical situations when the company production is not balanced (e.g. the company does not receive an optimal number of production orders to be able to manufacture), an extra attention to a production planning must be paid. Agent-based production planning tools could be a choice for enterprises where standard solution fails. The solution is suitable for enterprises using heterogeneous legacy systems. The main features and the most interesting properties of using the agent-based systems in contrast with the conventional software systems are scalability, modularity, and online reconfigurability. Agent approaches adopt principles and advantages of distributed algorithms (parallel computing). Automated workflow management systems have proved to be the driving force behind successful decision making in industry. Multi-Agent technology offers a convenient platform for workflow modeling. An environment where each agent represents a real information unit of the modeled enterprise is an appropriate model for optimization and

visualization of flows of material, work, and information.

The main requirements on an adaptive planner and scheduler are following:

- Intelligent reasoning about the enterprise resources with the aim to produce accurate estimation of project's deadline and costs
- System state update and maintenance
- Re-plan, i.e. maintenance of the plans created so far in such a way that they are dynamically updated with respect to eventual changes in co-operating agents (e.g. resource/agent breakdown).

Besides workflow management, the multi-agent technology is well suitable for manufacturing processes simulation, production control as well as scheduling, planning and re-planning.

2.3.1 Car Rental Scheduling

Rent-a-car business deals with one of the most difficult problems in the modern theory of optimization. In particular, when it is required to create schedules of many participants which are connected with each other, and changes in the schedule of one participant cause changes in the schedules of other participants. Company Magenta Technology has developed an agent-based scheduler to tackle this challenge (Andreev et al., 2009). Following the multi-agent paradigm, the global schedule is the result of distributed decision-making process that the process of finding of the best global schedule is a continuous negotiation of agents acting as demands and resources with conflicting interests. In 2009, the application was available as a commercial product that had proven its advantages in practice.

The application had to meet the following requirements:

1. The schedules of drivers and cars need to contain interconnected operations which they carry out at different stages of the business process taking into account logic sequence, a place, time and other attributes;
2. The application has to be able to balance multiple criteria such as car running costs, driver costs, penalty for late delivery, etc.;

3. The scheduling process has to be event-driven, i.e., an unexpected event (e.g. car accident) has to be immediately followed by a new round of negotiation to find the new optimal schedule with the additional constraint;

4. Integration of the scheduling and execution capabilities, i.e., the system has to be able to execute the found schedules;

5. The acceptable functioning of the system has to be guaranteed even in situations when not all possible combinations can be investigated, i.e., the system has to be aware of limited computational resources;

6. Take into account the general expectations (e.g. no driver will be left in the field without a car).

The solution of the problem is based on the Demand-Resource Networks conception to solve considered task of rent-a-car scheduling in real time (Rzevski et al., 2007). The scheduling process is represented by interaction of agents representing entities such as clients, rental orders, and cars. The application provides results that already outperform other tools for rent-a-car business management, although there is still great potential for further improvement.

2.3.2 Supplying International Space Station

Supplying the International Space Station (ISS) is a complex task dealing with lots of interrelated tasks to schedule unmanned cargo space flights (including starts, dockings and undockings) and piloted (manned) flights considering various requirements, support space crew life activity, deliver laboratory equipment, different material and instruments (Ivaschenko et al., 2011). Originally, the solution of the problem involved millions of iterations among lots of scientists, engineers and managers to come to a certain compromise solution to support ISS with all required stuff considering lots of limitations and constraints. The innovative approach proposed by Smart Solutions[1] adopts the agent-based principles to tackle the complexity of the task in order to assure intelligence and effectiveness of the decision making process.

The developed application solves the following tasks:

[1] http://smartsolutions-123.ru/en/

1. Overall flight program: the flight schedule has meet the basic requirements (e.g.: at least one piloted ship has to be docked to the station) and respect the essential constraints (e.g.: a necessary amount of time between operations of docking/undocking);

2. Cargo flow scheduling, that results in distribution of deliveries of units, blocks and systems across individual flights;

3. Schedule of fuel supply considering a forecast of ISS position changes, the Sun activity, and the ISS program;

4. Supply of basic human needs (e.g.: water and food) with respect to the ISS program;

5. Flight crew time scheduling.

The proposed agent-based approach treats the ISS supply chain as a complex network of continuously working and co-evolving specialized schedulers running in parallel on multiple engines. Scheduling in each engine is performed by agents of different types. The agents perform repetitive interactions involving other agent's communities supported by specialized subsystems and introduce fluctuating corrections of their schedules. Such a behavior mimics a scheduling process among real human beings, when decision makers, responsible for cargoes, fuel and water, negotiate about a suitable allocation altogether.

2.3.3 Adaptive Production Planning and Scheduling

The production and ramp-up of complex and highly customized products are exceptionally challenging for planning and control, especially in small lot sizes. Daily challenges like late requests for change, immature high technology products and processes create significant risks. The occurring risks are bigger than production of big series such as automotive. Thus, new ICT-based approaches are required. The aim is to develop mitigation strategies to respond faster to unexpected events. Therefore the knowledge base has to be enriched for real-time decision support, to detect early warning and to accelerate learning. Our approach is based on a new generation of service oriented enterprise information platforms, a service oriented

bus integrating service-based architecture and knowledge-based multi-agent systems (MAS). A holonic MAS combined with a service architecture will improve performance and scalability beyond the state of the art. The solution integrates multiple layers of sensors, legacy systems and agent-based tools for beneficial services like learning, quality, risk and cost management. Additionally the ecological footprints will be reduced. The ARUM solution will run in two modes: predictive and real time simulation. The predictive mode supports the planning phase whereas the real-time operations mode supports dynamic, time-, cost- and risk-oriented re-planning of operations. The provision of information for engineering to alter in case of immaturity or late requests for changes is supported equally. ARUM is strongly end-user driven and the results are tested on two industrial use cases with a focus on aircraft and aircraft interiors.

2.4 Manufacturing Control

The multi-agent systems are inherently dynamic. The new agents can be created on-line, existing agents can migrate to different computational units or can destroy themselves if they are not needed any more, coalitions of agents are dynamically created to solve a particular problem and when done the coalition is dissolved, etc. This unique set of features attracted attention of engineers developing the future generation of manufacturing systems that are focused on fast reconfigurability and adaptability of industrial control systems.

The term reconfigurability refers to the capability of the control system to modify its behavior in order to:

1. Cope with the contingencies such as equipment failures,
2. allow for change of the shop floor layout by adding, removing or changing the equipment, and
3. deal with the modification of the production process as innovated or new products are introduced.

It is generally acknowledged that the key attributes of agents – the autonomy, local intelligence, and loose coupling provide effective means for dealing with the

reconfiguration issues. As there is no central decision-making component, the system can withstand local failures more effectively and thus minimize the possible impact on the system's overall functionality. The agent that is in charge of the failed resource notifies other agents about its unavailability and in subsequent negotiations and an alternative resource is found. When the new hardware equipment is installed in the factory or existing one is removed, new agent instances are created and plugged-in, or existing agents, are removed. When an upgraded or completely new product should be made on the existing production line, the behavior of the corresponding product agents is modified online without a need to stop the production line or do changes or reprogramming of other parts of the system.

2.4.1 Manufacturing Agent Simulation Tool

Manufacturing Agent Simulation Tool (MAST) is an application designed for both simulation and control of material handling systems. It is built on top of a publicly available agent platform - JADE (Java Agent DEvelopment Framework)[2]. The platform enables to create and run simulation of hardware equipment as one Java application together with agents. This approach significantly reduces time needed for development and debugging of new agent-based applications.

MAST tool has been extended to be able to simulate the holonic packing cell of the Center for Distributed Automation and Control (CDAC) at the Cambridge University's Institute for Manufacturing. This lab provides a physical testbed for experiments with the agile and intelligent manufacturing focusing particularly on the Automatic Identification (Auto-ID/RFID) systems. This emerging standard for automatic product tracking introduces the Electronic Product Code (EPC) as an alternative to the bar code label. The unique EPC number is embedded in an RFID (Radio Frequency IDentification) tag comprised of small silicon chip and antenna. Reading and writing is done wirelessly via specialized devices - RFID readers - using high or ultra high frequency radio waves.

In the Cambridge packing cell (see Fig. 2.7) the RFID technology is used in controlling the packing of Gillette gift boxes. The user can select from two types of

[2] http://jade.tilab.com/

boxes that can be filled by any combination of three out of four Gillette grooming items (gel, razor, deodorant and shaving foam). The lab physically consists of the following components (numbering corresponds with labels in Fig. 2.7):

1. Conveyor loops (Montech track) to transport the shuttles that carry boxes. There is one main feeding loop and two subsidiary loops leading to robots.
2. Gates (diverters) that navigate the shuttles out of the main loop to the subsidiary loops and vice versa.
3. RFID readers that read the EPCs of passing boxes - the data are provided by the readers to the gates to be able to properly navigate the shuttle.
4. Docking stations at which the shuttles are held while box is being packed.
5. Fanuc M6i robots that pack the boxes by the items picked up from the storage units.
6. Storage units for temporary holding of the items in four vertical slots (each for a particular type of the Gillette item).
7. Rack storages that hold shuttle trays with both the empty and packed boxes as well as with the raw items that can be used to feed the temporary storage areas.
8. Gantry robot agent that is able to pick the box out of the rack storage and drop it to the shuttle waiting in the docking station in the main loop.

2.4.2 Actor-Based Assembly System

Another example of an agent-based approach adopted by assembly domain is Actor-based Assembly Systems (ABAS). According to (Lastra et al., 2005): "ABAS are reconfigurable systems built by autonomous mechatronic devices that deploy auction and negotiation-based multi-agent control in order to collaborate towards a common goal, the accomplishment of assembly tasks." This system was developed by the Tampere University of Technology in cooperation with Schneider Electric and is recognized as the first agent-based simulation tool aimed at visualization

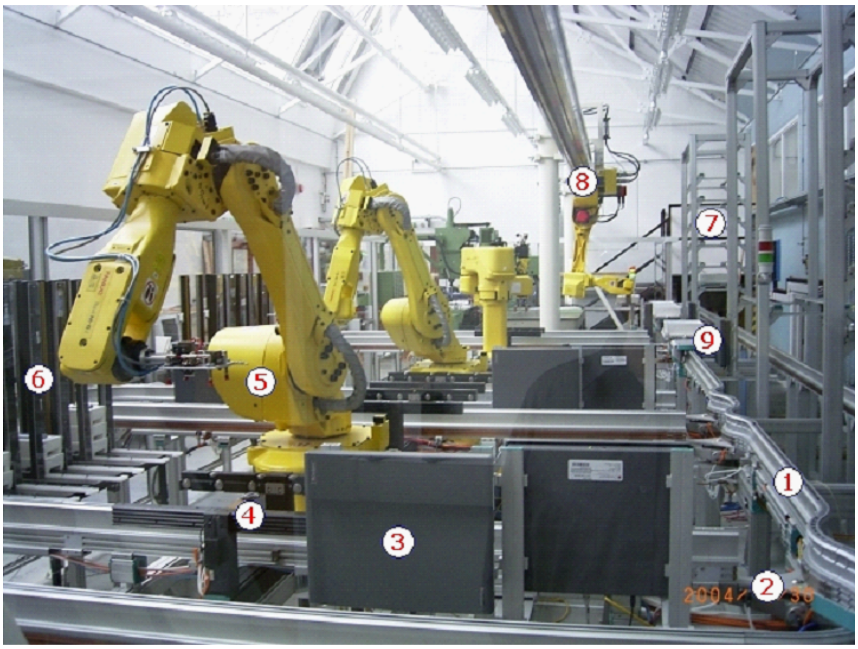


Figure 2.7: CDAC laboratory

and simulation of the operation of robot in the 3D manufacturing space. The authors of the system introduced a term "Actor" which refers to a newly developed components extending the traditional software agents with a mechatronical counterpart.

The results of the projects were not limited only to the definition of new concepts, but also provided software tools to emulate actors, societies (groups of actors) and their behavior, to provide support in the design of a new system. The developed environment consisted of two independent tools. The first tool addresses both the modeling and the emulation of physical intelligent agents – actors. The second one serves as a runtime platform where actor societies can be deployed and visualized in a 3D environment, and which performs executive control of assembly processes. The platform can serve as a runtime environment for physical and/or emulated actors indistinguishably.

2.5 Conclusion

This chapter illustrated how agent-based technology contributes to development of complex manufacturing systems. We have shown that there are several levels on which the industry can benefit from the agent paradigm including:

1. Control of various types of networks;
2. Planning and scheduling; and
3. Assembly and material handling.

From the point of view of this thesis, it is important that all the mentioned projects were developed and implemented by highly specialized engineers and researchers. The dominant reason of their slow adoption by the industrial enterprises is the lack of methods and supportive tools that would simplify the development and testing process.

Chapter 3

Related Work

3.1 Introduction

This chapter provides an overview of existing supportive tools that ease the development of agent-based systems. The availability of such tools is necessary for successful transfer of the agent-based technology from laboratories to factories.

3.2 FIPA

The Foundation for Intelligent Physical Agents[1] (FIPA) is a non-profit association registered in Geneva, Switzerland, founded in 1995. The main goal of the FIPA is to maximize interoperability across agent-based applications, services, and equipment, accomplished through FIPA specifications. Especially, FIPA provides standards for agent communication languages (see section 6.2.2). FIPA specifies the set of interfaces which the agent uses for interaction with various components in the agent's environment, i.e., humans, other agents, non-agent software, and the physical world. It focuses on specifying external communication among agents rather than the internal processing of the communication at the receiver.

FIPA produces mainly two kinds of specifications:

[1] http://fipa.org/

1. Normative specifications that mandate the external behavior of an agent and ensure interoperability with other FIPA compliant subsystems.

2. Informative specifications of applications for guidance to industry on the use of FIPA technologies. FIPA standards attempt to be high level, neutral abstractions.

The core FIPA specifications are neutral with respect to both (i) the application area and (ii) the hardware and software platforms to be used. Of course, the existing software infrastructures should be considered when designing the standards and leveraging the advantages of the existing, non-agent software technology.

The FIPA specification of the message transport protocol (MTP) defines how the messages should be delivered among agents within the same agent community and particularly between different communities. For the latter case, the protocol based on IIOP or HTTP ensures the full interoperability between different agent platform implementations. It means that the agent running e.g. on the JADE agent platform can easily communicate with agent hosted by the FIPA-OS platform etc.

FIPA defines an Agent Platform (AP) as a system that provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components and agents. The defined agent management components are as follows:

- Agent Management System (AMS) controls access and use of the agent platform and provides services like maintaining a directory of agent names. It provides white page services to other agents. Each agent must be registered with an AMS.

- Message Transport Service (MTS) supports the transportation of FIPA ACL messages between agents on any given AP and between agents on different APs.

- Directory Facilitator (DF) is optional and provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are offered by other agents, including the discovery of agents and their offered services in ad hoc networks.

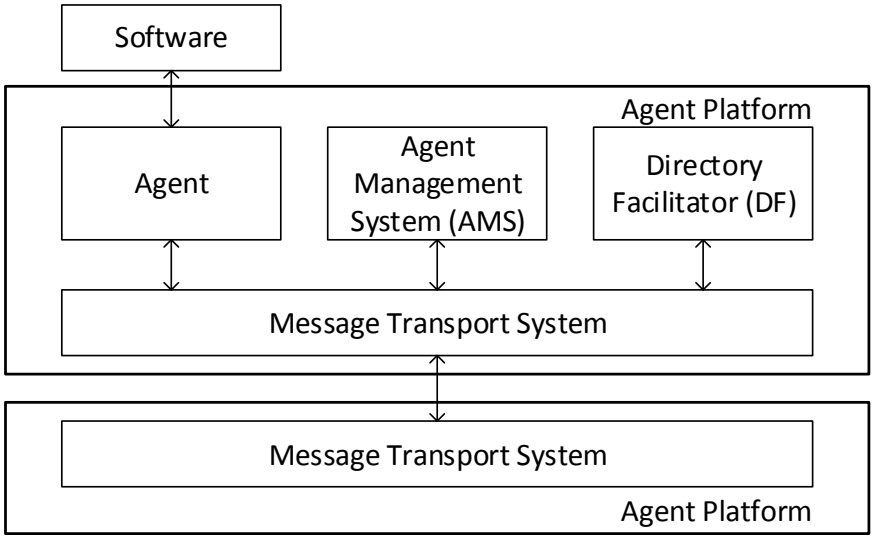


Figure 3.1: FIPA Agent Management Reference Model.

All these components form the FIPA Agent Management Reference Model that describes the architecture of a FIPA compliant multi-agent system (see Fig. 3.1).

3.3 Platforms for Industrial Multi-Agent Systems

A multi-agent platform is a tool which provides developers with a set of code libraries for specification of user agent classes with specific attributes and behaviors. Compliancy with the FIPA standards has been recognized as a crucial property ensuring the interoperability of agents not only at the lowest real-time control level (allowing e.g. communication of different kinds of agents hosted by PLC controllers from different vendors) but also the interoperability among agents from different layers of information processing within the company, e.g. data-mining agents, ERP agents, supply chain management agents and so on.

3.3.1 ACS

The Autonomous Cooperative System (ACS) was originally developed as the C++ based agent platform dedicated to the Logix family of PLC controllers. The agent platform enables to run the agents directly inside the PLCs (ControlLogix, FlexLogix, etc.), supports the agent management (registration, deregistration, services look up, etc.) and ensures the transport of messages among agents. Later, the ACS was implemented in JAVA, but this version was not compatible with the most of the PLCs, since it required availability of a JAVA virtual machine.

The ACS platform is designed with respect to minimizing the use of memory and CPU resources of the PLC so as to not impact the performance of real-time control programs that run in parallel with the agents. The agents use a specially designed communication language – JDL (Job Description Language) for the message exchange in the RT-tasks as well as for the planning purposes. The JDL messages can be converted into FIPA-compliant messages by adding an appropriate FIPA wrapper to allow the ACS agents to communicate with other FIPA-compliant agent platforms (for example JADE or FIPA-OS).

3.3.2 JADE

Java Agent DEvelopment framework (JADE) (Bellifemine et al., 2007) is probably the most widespread agent platform in use today. JADE is a completely distributed middleware system with a flexible infrastructure allowing easy extension with add-on modules. The framework facilitates the development of complete agent-based applications by means of a run-time environment implementing the life-cycle support features required by agents, the core logic of agents themselves, and a rich suite of graphical tools. As JADE is written completely in Java, it benefits from the huge set of language features and third-party libraries, and thus offers a rich set of programming abstractions allowing developers to construct JADE multi-agent systems with relatively minimal expertise in agent theory. JADE was initially developed by the Research & Development department of Telecom Italia, but is now a community project and distributed as open source under the LGPL licence.

3.4 Multi-Agent Social Models

Agents can cooperate only if they can contact each other. Therefore, maintenance of the social knowledge is one of the key capabilities of any MAS. Many architectures for maintaining social knowledge have been proposed and their comprehensive overview provides Tichý (Tichý, 2003). The basic categorization of architectures distinguishes static and dynamic approaches. The static architectures are characterized by the creation of the social knowledge at design time, i.e., the social knowledge is established and distributed once and remains constant at runtime. On the contrary, the dynamic architectures are characteristic for the continuous development at runtime and changes during agents' life-cycles. A short introduction of the most used architectures follows.

Static architectures are either hierarchical or flat. The former are inspired by a common hierarchical organization in an enterprise. Its advantage is the efficiency on the other hand its disadvantage is low resilience, since an outage of any component disconnects all its successors from the rest of the system. In contrary, the flat knowledge base architecture is free of a superior nodes and therefore these approaches are also called "point2point" or "peer2peer".

Dynamic architectures are equipped with a mechanism that enables to maintain the social knowledge at runtime. This is necessary for the realization of the Plug&Play concept which means that a new agent can be easily added to an already running system. The focal point of any dynamic architecture is the process of **matchmaking** during which an specialized agent (frequently denoted as meta-agent) creates a communication link between two agents. Sycara defines the matchmaking as "process of finding an appropriate provider for a requester through a middle-agent" (Sycara et al., 2002).

The most used dynamical architectures are as follows:

- Broadcasting: a simple approach that generates a lot of communication. Anytime an agent finds a provider of a particular service, it sends the request to all agents. Only agents that provide the service handle the request and send a response to the initiator.

- Federated Architectures: more advanced approaches when the agents repre-

senting service providers and consumers are accompanied with meta-agents that enable to create cooperation links between agents dynamically according to the current conditions. There exist various variants such as: Matchmaker, Broker, Mediator, Blackboard, Monitor, Facilitator, Embassy, Anonymizer, and Job Agency. The detailed description explaining specifics of individual approaches can be found in (Tichý, 2003).

The proposed method is applicable for the federated architectures, because the meta-agents can influence the carrying out of new tasks.

3.4.1 Fault Tolerant Structure of DF agents

One of the main advantages in using MASs is fault tolerance. When an agent fails a multi-agent system could offer the services of another agent that can be used instead. However, if a MAS uses a middle-agent to search for the capabilities of providers, i.e., to search for an alternative agent with the same capability, then this middle-agent can become a single point of failure, i.e., social knowledge is centralized in this case. It is not possible to search for capabilities of other agents and to form virtual organizations any more if the system loses the middle-agent. The fault tolerance issue is tightly coupled with load sharing. When only one middle-agent is used it becomes a communication bottleneck and can be easily overloaded.

Tichý (Tichý, 2003) designed and implemented a special structure of Directory Facilitators called dynamic hierarchical teams (DHT) (see Fig. 3.2) that has a user-defined level of fault tolerance and is moreover fixed scalable, i.e., the structure can be extended by a fixed known cost. This structure is not only usable for Directory Facilitators, but also for other types of agents, mainly for so called middle-agents that stand between providers and requesters of capabilities

3.5 Development Tools

Currently, there exist multiple development environments that are used to develop agents for given MAS platforms, such as the JACK Development Environment described in (Evertsz et al., 2004), (Systems, 2011), and Cybele in (Manikonda

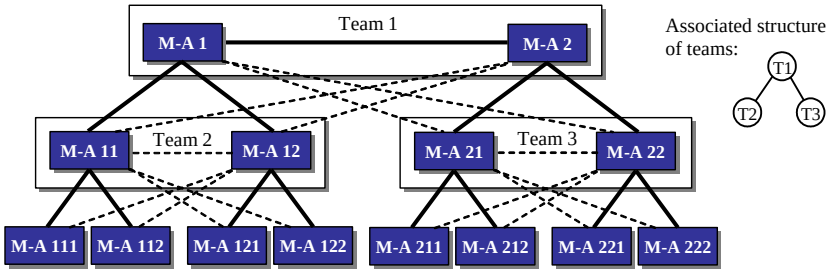


Figure 3.2: Example of 3-level DHT architecture (Tichý, 2003)

et al., 2004). Very detailed reviews of these systems can be found, for example, in (Bitting et al., 2003). These development environments either only guide the user during the development of agents or provide more elaborate support when the user programs the agents as Belief-Desire-Intention (BDI) models developed by (Bratman, 1999). The resulting runtime code is either interpreted or automatically generated and then directly executed.

3.6 Debugging and Visualization

Development of a multi-agent system relies not only on the agent development environment and agent platform, but also on monitoring tools. To create a fully functional multi-agent system, there is a need for a visualization and debugging tool to observe the internal communication and behavior of the system and to discover potential problems. This need led to the design and implementation of a tool for the visualization and debugging of multi-agent system communication, the *Sniffer*.

3.6.1 JADE Sniffer

While all the other tools are for the most part used for debugging a single agent, this tool is extensively used for debugging, or simply documenting conversations among agents. It is implemented by the class "jade.tools.sniffer.Sniffer". The "sniffer" subscribes to a platform AMS to be notified of all platform events and of all message exchanges within a set of specified agents.

When the user decides to sniff an agent or a group of agents, every message directed to, or coming from this agent/group is tracked and displayed in the sniffer GUI. The user can select and view the details of every individual message, save the message to a disk as a text file or serialize an entire conversation as a binary file (e.g. useful for documentation).

3.6.2 Java Sniffer

This tool allows receiving messages from all agents in the system, reasoning about this information, and visualizing it from different levels of view. It is also possible to configure running agents by sending service messages to them from the Sniffer. Full documentation can be obtained with the tool. Nevertheless, we will briefly describe its main parts.

The Sniffer visualization screen is divided into four main sections (see Fig. 3.3). Each section provides the observer with information at a different level of detail.

The window marked with number 1 located at the bottom left corner displays a temporally organized list of messages as standard sequential UML diagram. The header of this view contains names of individual agents and each row represents one message sent from one agent to another agent, displayed as an arrow pointing from the sender to the receiver of the message. The text assigned to the arrow shows an overall description of the message and the very left column displays message numbers and time-stamps.

The content of a selected message is displayed in the partition marked with number 2. It provides various tabs such as:

1. Tree tab to show the overall structure of the message;

2. Envelope tab shows message parameters as defined by FIPA;

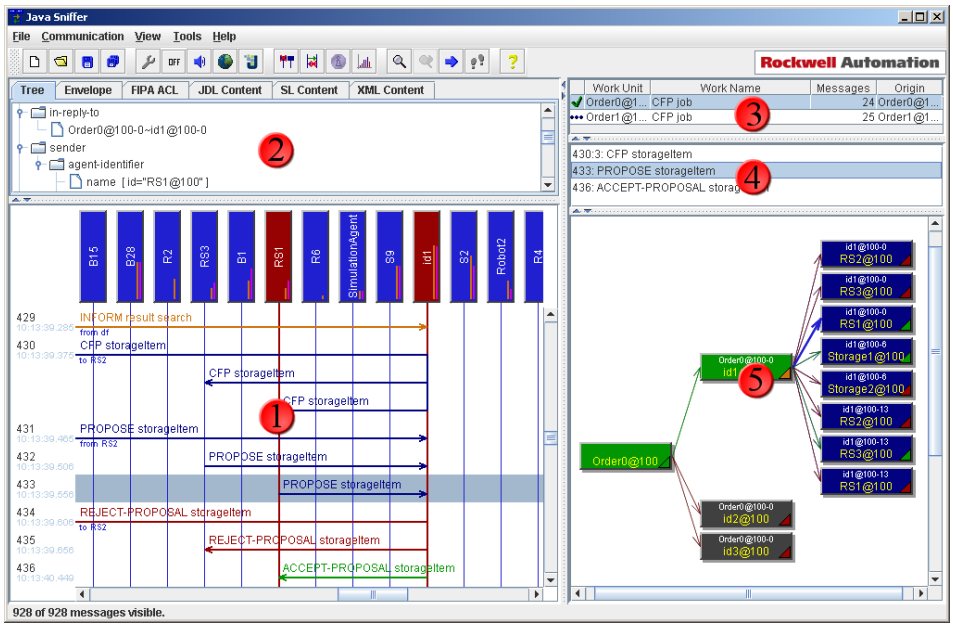


Figure 3.3: Java Sniffer – Main Window

3. FIPA ACL tab shows Agent Communication Language (ACL) slots of the selected message;

4. JDL Content Tab provides content of the Job Description Language (JDL) message part;

5. SL Content shows content of the message in FIPA Semantic Language (SL); and

6. XML Content tab visualizes the content of the selected message as a XML tree.

The window marked with 3 shows a list of workflows – plans communicated among agents. Each description of a workflow consists of a workflow identification, work name, number of subsequent messages belonging to this workflow, and the original requester of the work. Also, the status of the workflow is pictographically visualized showing “in progress” or “successfully” or “unsuccessfully” finished.

Window 4 shows details of either selected rectangle or selected arrow in Workflow view. If the list is filled with messages that are associated with selected arrow

the user may select one message from the list. The selected message will be shown in UML View and Message Detail View will be also updated.

Window 5 displays a dynamically created tree of a workflow that belongs to the work unit selected in the list of work units. Any request for and reply to planning, commitment, or execution of the work is visualized as an arrow pointing from the parent (creator of the work) to child (solver) agent. The arrow represents all messages that belong to a particular part of the work. The workflow view shows the whole tree constructed from all parts of the work unit.

3.7 Agents and low-level control with IEC 61499

The call for reconfigurable control system brought many proposals. One of the most accepted and developed is a set of standards IEC 61499 proposed and standardized by the International Electrotechnical Commission (IEC) for Industrial Process Measurement and Control Systems (IPMCS). It introduces concept of a Function Block-oriented (see Fig. 3.4) paradigm for IPMCS. It is built on top of the widely used industrial programming languages standardized in the IEC 61131-3.

The most of the ongoing activities related to IEC 61499 are organized by community around the Framework for Distributed Industrial Automation and Control – 4DIAC[2]. It is an open source project enabling the further development of IEC 61499 for its use in distributed IPMCS. The project started in 2007 and from the beginning provided everything necessary to program and execute distributed IPMCS. Currently, 4DIAC is the dominant tool for IEC 61499-based projects and its further development is supported by entering the community of Eclipse projects. 4DIAC supports the other research activities such as Eclipse Internet of Things Industry Working Group by addressing the lower level real-time layers which are not currently targeted by any other project. The framework consists of two projects: integrated development environment – 4DIAC IDE and runtime environment – 4DIAC RTE (FORTE).

[2] http://www.fordiac.org/

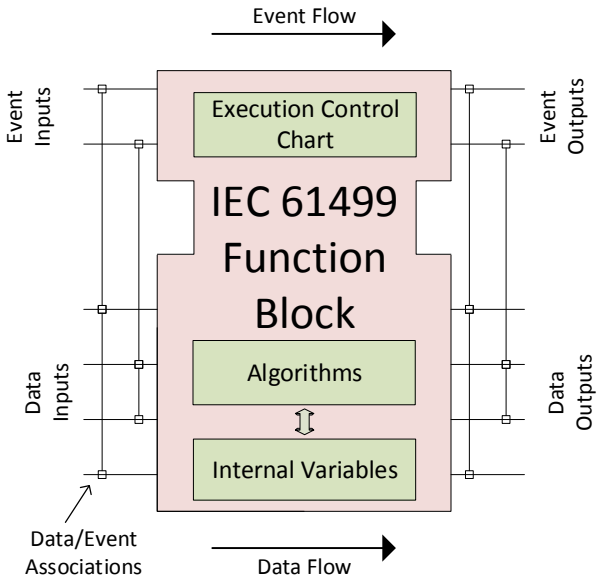


Figure 3.4: IEC 61499 Function Block

3.7.1 4DIAC IDE

4DIAC IDE is an IEC 61499 compliant engineering environment implemented in Java as an Eclipse plug-in. The environment was designed with accent on the user-friendliness of the whole development process. Many configurations can be done in a graphical way via drag&drop operations. Figure 3.5 depicts development of an IEC 61499 application. System configuration window provides view on the global organization of systems and application within a workspace. Application editor is the central part of the development process. All application components and their interconnections are here created, configured, and monitored. The function blocks are created by drag&drop mechanism from the Function Block library. The bottom editor enables to configure parameters of selected components, e.g.: set values of data inputs.

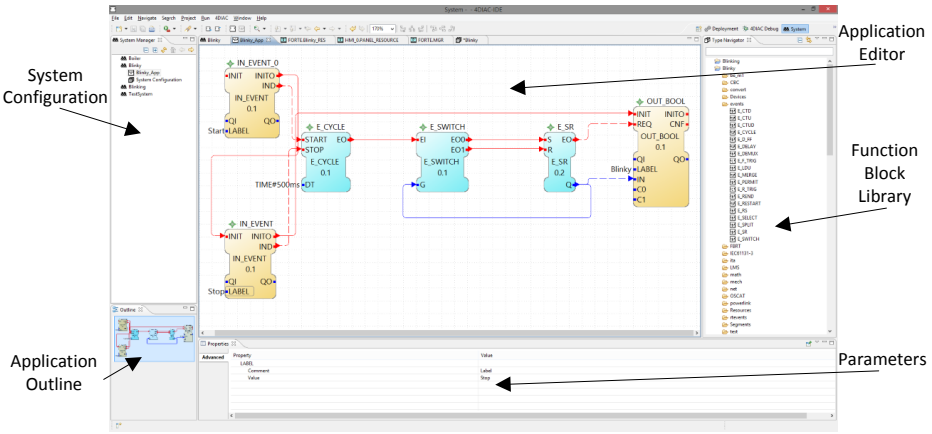


Figure 3.5: 4DIAC – IDE: Main Window

3.7.2 4DIAC RTE

4DIAC RTE – FORTE is a IEC 61499 compliant runtime environment. It is implemented in C++ and it is available in several versions for various computational platforms (PC, Embedded). The runtime is directly runnable from the 4DIAC IDE and also the code created in the IDE can be directly deployed on the running FORTE. Both FORTE and the IDE support monitoring to provide the engineer the view of a running system.

3.7.3 Verification of IEC 61499 Non-Functional Parameters

IEC 61499 provides a concept how to improve flexibility and reconfigurability of industrial automation and control systems. The key part of the concept is the replacement of the scan based approach used in traditional control systems based on IEC 61131-3 with event driven solution. This is crucial for distributed systems where the flow of events eliminates the need for synchronization between individual computational units. However, developers of these systems lack efficient methods to verify the non-functional reliability of such systems. This methodological gap complicates the design of complex commercial systems with strict performance requirements.

The author of this thesis was involved in research conducted within the frame-

work of the AKTION[3] project number 69p18 focused on applying performance models for verification of control systems based on IEC 61499 standard. In order to show how the performance models can help to identify problems in EBIAS a real-world scenario has been selected. The chosen application is an event-based automation system for the AIT Smart Grid laboratory SmartEST (see Fig. 3.6). This laboratory is a highly flexible and configurable 1 MW research and test/validation facility for Smart Grid systems. It allows to test power system and ICT components in a Smart Grid configuration but also the validation of the component's behavior.

Figure 3.6: AIT Smart Grid SmartEST laboratory.

A brief overview of the software architecture of the laboratory is shown in Fig. 3.7 whereas mainly open source tools are being used to control the laboratory (Zoitl et al., 2013). The control layer – implemented in the IEC 61499 4DIAC environment – is responsible for automation tasks, security issues and data processing. It interacts with the process/hardware layer (i.e., sensors and actuators) using mainly Modbus/TCP and Ethernet POWERLINK fieldbus systems. The supervisory control and visualization layer is implemented with the open source SCADA ScadaBR. The interaction of the SCADA with the control layer is based on a TCP/IP communication as defined in the IEC 61499 standard using Abstract Syntax Notation 1 (ASN.1) encoding of the data.

[3] http://www.dzs.cz/en/aktion-czech-republic-austria/

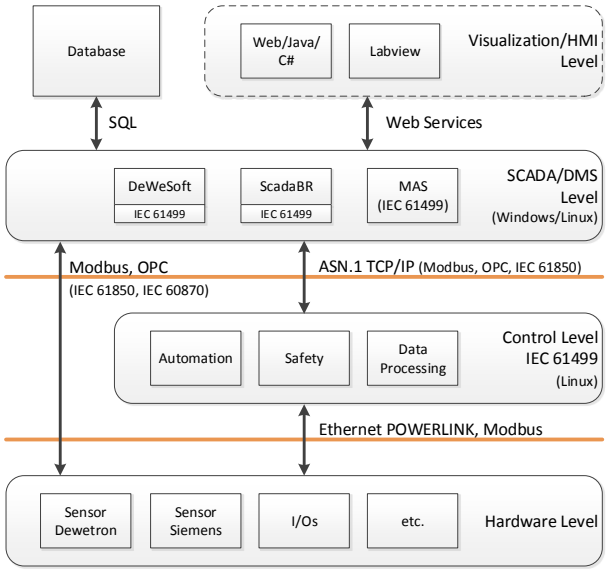


Figure 3.7: Brief overview of the laboratory software layer architecture.

The concrete issue that was experienced within the work on control and visualization system of the SmartEST laboratory was the connection of the control and the SCADA layers. If the frequency of events coming from the control layer exceeded certain level, i.e., the messages notifying the SCADA system about a data change came close to each other, then the SCADA system was not able to distinguish individual notifications and incorrectly joined two or more. The accidental notification mixing up caused a data type mismatch error.

Originally, this problem was solved by a specific interconnection of the used IEC 61499 function blocks which is designed with respect to the detected problem. This design was possible only because of the thorough understanding to the controlled system. In general, this solution is not acceptable from the engineering point of view, because it puts inadequate additional load on the system engineers. Moreover, this load steeply increases with the size of the controlled system and therefore it was highly advisable to develop an alternative solution.

The proposed solution was based on identification of characteristics of individual system components and implementation of two new function blocks. The first is named Event Aggregator and its task is to aggregate all events coming within

a time interval which length is specified by a function block parameter. The second functional block preserves the overall number of events, but the overall delays continuously accumulates.

3.8 Diagnostics of MASs

3.8.1 Model-Based Diagnostics

An example of a model-based verification approach for MAS is based on MABLE (Wooldridge et al., 2002) that is a conventional imperative programming language and it is extended by constructs from MAS. The agents designed in MABLE maintain their social knowledge using linear temporal belief-desire-intention logic. The major advantage of this approach is the ability to logically prove that any interaction of agents will not lead to a fault state. On the other hand, the fundamental disadvantage and perhaps the stopper for the wider spread of this technique is the limited set of constructs that an agent can use. In other words, the agent has to be designed in the way suitable for MABEL from the very beginning.

3.8.2 Formal Time Analysis for Embedded Systems

The domain of embedded systems is facing a dramatic increase of the network complexity. For example, modern automotive control systems contain more than fifty electronic control units (ECUs) that are produced by various suppliers (Schliecker et al., 2009). The units are inter-connected via a communication network that is a representation of a shared resource. There is the need to assure that a potential conflict in usage of the shared resource would not lead to a dangerous situation. It means that for example function of ABS in a car must not be harmed by increased communication of other systems. Similar problem and requirements come from the aircraft industry and also from the designers of multiprocessor systems calculus (Richter et al., 2003). In general, networked or distributed systems can be characterized by observing a high amount of data flows within the network. To address the challenges posed by the increased network complexity the network calculus (Cruz, 1991) and its extension – real-time calculus (Thiele et al., 2000) were introduced. Network Calculus enables to evaluate timing properties of data

flows in communication networks. Real-Time Calculus extends this concept to make it suitable for real-time embedded systems. To summarize it, the basic idea behind these two approaches is to substitute individual events by data flows called event streams. The validation problem is then translated into the examination of flows, which is a well-handled task.

3.8.3 Qualitative and Quantitative Analysis of Industrial Multi-Agent Systems

The investigation of methods for validation of industrial agent-based applications was addressed by the EU FP7 research project GRACE (Leitão and Rodrigues, 2012). One of the project outputs is the methodology for qualitative analysis based on Petri net modeling notation. The behavior of each agent is represented as a single Petri net which can be verified by the regular methods to find out whether the model is bounded (the resource can only execute one operation at a time), reversible (the agent can reinitiate by itself) or out of deadlocks (the agent can make at any state an action). The extension of these models with concept of time provides methods for quantitative analysis of MASs. The transitions are extended with the time parameters to capture the times of transition activations. Such a simulation shows the evolution of the tokens over places and over time. The complete information about the progress of the agent behavior is summarized with a Gantt chart.

Chapter 4

Agent Development Environment

4.1 Introduction

In this chapter the Agent Development Environment (ADE) is introduced. In the past, the software part of MASs used to be typically developed ad hoc in a target object-oriented programming language such as Java or C++. This manual process was error-prone, time consuming, and required highly specialized skills. Thus, the utilization of a development environment that guides the development process was essential for acceptance of agent-based technology by the industrial enterprises.

One of the biggest challenges of the MAS development is the requirement to synchronize and integrate the development of the high-level control (HLC) and low-level control (LLC) application parts. Therefore, the ADE does not only support the development of agents that can be used for high-level decision-making, but also the development of distributed LLC. The LLC guarantees real-time execution (both the soft and the hard real-time) at local level. This is important especially in industrial applications described, for example, in (Mařík et al., 2005) and (Pěchouček and Mařík, 2008), or where fault-tolerance is one of the most important aspects as described in (Tichý et al., 2006).

This thesis proposes several concepts and methods how to enrich the LLC with concepts of object-oriented programing, which is necessary to enable automatic interoperability between the HLC and LLC application parts. Namely:

1. Indirect references to mimic static variables;
2. Containment to encapsulate a component within another one;
3. Macro-instructions to proceed an operation over a collection of components,
4. Inheritance to reuse an existing class for definition of a new one.

The detailed description of these concepts can be found in section 4.2.

4.1.1 Integration of HLC and LLC

The main characteristic of the architecture that we call a holonic agent as described, for example, in (Christensen, 1994) is to deploy agents at the lowest control levels and still preserve the ability of the control system to work under hard real-time constraints. This is achieved by an encapsulation of the HLC and LLC. This is followed by establishing a control interface for interactions between the HLC and LLC (see Fig. 4.1).

The LLC application part is implemented as a regular control module deployed on a programmable logical controller (PLC) that executes the code in a common scan-based manner in order to meet the requirements on the real-time behavior. The PLC reads the input values from the sensors in a controlled process through the analog or digital I/O cards; the control programs perform a computation including calculation of new output values; and finally the output values are propagated again via the I/O cards to the actuators in the controlled system.

Currently, the ADE supports the IEC 61131-3 standard of PLC programming languages for implementation of the LLC. In the future, the support for the IEC 61499 standard (see section 3.7) is possible. Utilization of the standard known as "function blocks" would enable to benefit from its inherent characteristics aiming at distributed and modular applications.

The HLC module is represented by the agent as it is defined in the agent-based technology: an autonomous software unit capable of intelligent decision-

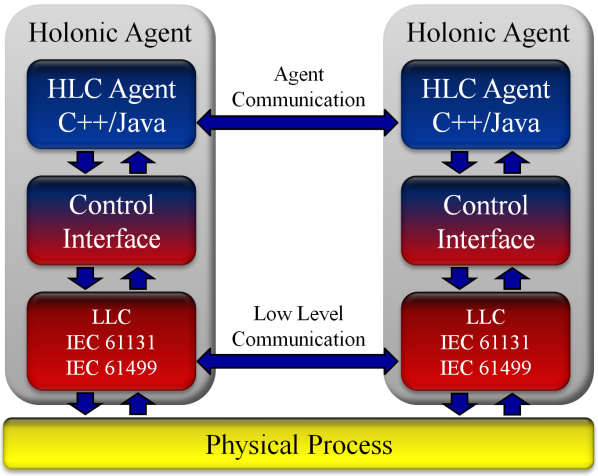


Figure 4.1: Interaction of high-level control (HLC) and low-level control (LLC).

making, communication, and cooperation with other agents. Hence, "HLC" and "agent" will be used in this chapter as synonyms. The complexity of the agent behavior requires the employment of some high-level programming language like C++ or Java. Therefore, a PLC has been augmented to allow direct execution of agents, i.e., a PLC is able to host a multi-agent system developed in ADE in an environment called the Autonomous Cooperative System as described, for example, in (Tichý et al., 2002). For testing purposes it is also possible to execute this system on a PC.

4.1.2 ADE Characteristics

One of the main characteristics of MASs mentioned earlier is the re-usability of agent templates. An agent template is designed once and reused multiple times for as many agent instances as needed. They are then interconnected via social knowledge and are able to coordinate their actions and cooperate. Thus, the goal is to develop an agent library, essentially a collection of class definitions called templates that describe the behavior of both the agents and any non-agent components. Agents are part of HLC and components reside in LLC.

There can be multiple libraries, each targeted to a specific application domain,

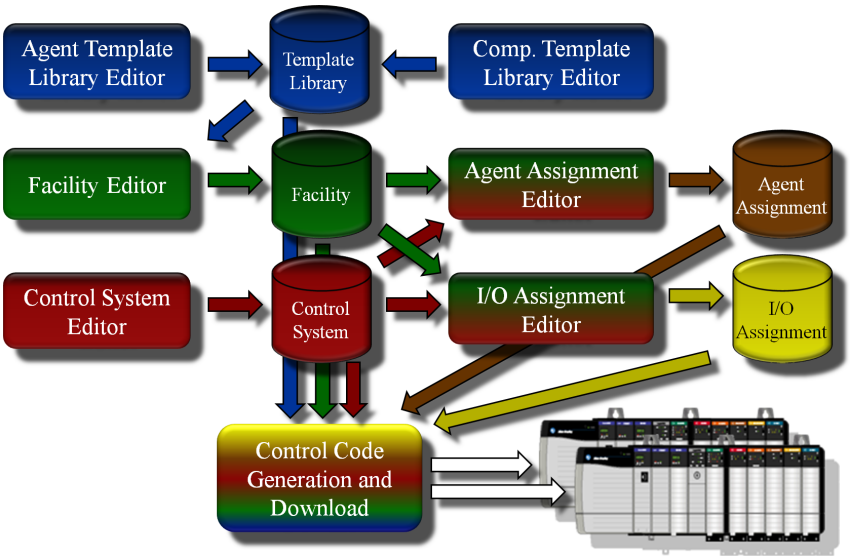


Figure 4.2: Development flow in the Agent Development Environment.

e.g., material handling systems, assembly, logistics, or batch processing, and thus can be used to create control systems for multiple similar facilities in the same domain.

The control code for the components or the low-level part of agents is generated automatically from the same templates. In this way, the effort to build the library is amortized over all the facilities (control applications) built with the library. Moreover, each facility has the benefits of having control system software that is well structured, predictable, optimized, and tested. This approach leads to fault-tolerant architecture as described in (Tichý et al., 2010).

Moreover, any additions or modifications to the library subsequently appear in all affected instances after the code re-generation. In other words, instead of changing all affected instances individually, the user applies the change only in the template in the library, thereby not only speeding up the process but also avoiding any inconsistencies and mistakes in programming.

The process of development within ADE is shown in Fig. 4.2. Arrows represent information flow and thus also dependencies between various parts of the environment.

- First, the user creates components and agents in the template library (TL)

or imports them from some existing library and thus reuses previously tested and optimized code.

- Then the user creates a facility (F) from components and agents of the TL and customizes their parameters.
- Next, the user establishes a control system (CS) that describes all the PLCs, I/O cards, and networks, plus their interconnections.
- After the TL, F, and CS parts are completed, the user assigns agents and components to the execution units (usually PCs and PLCs) via agent assignment.
- The user also assigns inputs and outputs of agents and components to real I/O points that are connected to real hardware or to the simulation of this hardware via I/O assignment.
- With all this information available, it is now possible to generate and compile the control code.
- The last step is to download the code to the execution units and execute it.

Since the TL, F, and CS editors are independent, it is possible to change only the necessary parts when the system needs to incorporate changes. For example, a new PLC can be added by the CS editor and subsequently can have some agents assigned to it. The system is regenerated in a manner consistent with all modifications. In a similar way the Agent Assignment and I/O Assignment are semi-independent from other parts of the system and thus if there are no structural changes (e.g., removal of an execution unit or addition of I/O) then it is not necessary to modify or enter this information again. Moreover, it is possible to easily change the distribution of agents and components. This distribution to execution units strongly influences the non-functional parameters of the final application such as the maximal throughput of tasks the system can fulfill or the latency of the system responses. Moreover, it is a complex optimization process that requires special supportive methods. These methods are proposed within chapter 7.

The ADE software itself is designed using the Model-View-Controller (MVC) design pattern described, for example, in (Buschmann et al., 2007). Using this

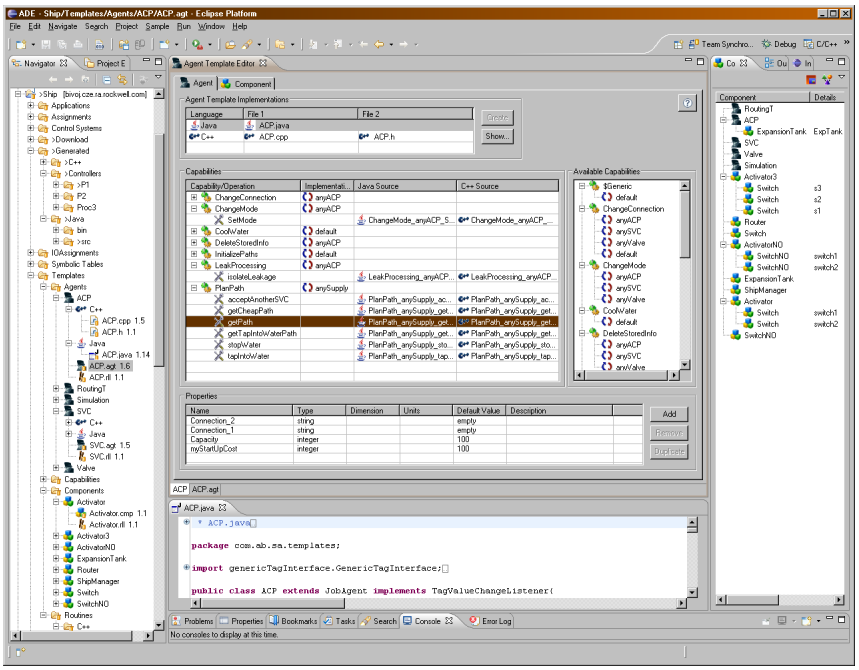


Figure 4.3: Eclipse-based Agent Development Environment GUI.

pattern allows us not only to distinguish and separate the inherent model from the user interface (UI) during the development so that sections of the code can be worked on by multiple development teams, but also to easily replace the UI in the future. Another advantage is that different parts of the application can notify themselves about changes that the user made to ensure the data and UI consistency throughout the whole environment. MVC allows both UI components (such as editors and views) and parts of the model to subscribe and be notified about changes in the model.

The ADE (see Fig. 4.3) has been implemented in Java as a plug-in to Eclipse development platform[1]. The ADE contains a built-in full Relay Ladder Logic (part of IEC 61131-3 standard of PLC programming languages) editor (see Fig. 4.4) . Besides standard functionality it allows implementing special ladder logic templates, indirect references, and macros that will be further explained in the following section.

[1] http://www.eclipse.org/

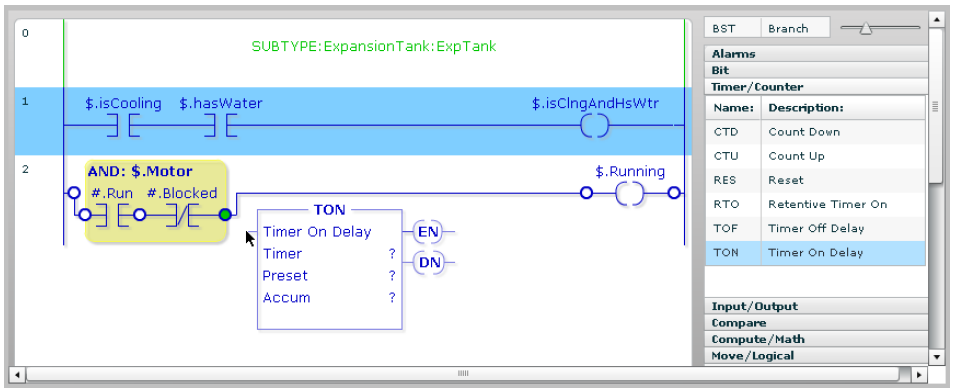


Figure 4.4: Relay Ladder Logic template GUI example during the adding of a new TON instruction from the instruction catalogue.

The utilization of the Eclipse environment allows us to reuse many of its parts related to Java and C++ programming such as the project navigation tree, perspectives suited for different parts of the development, windows that describe and manage problems and errors, user defined settings, persistence of open window positions, and the integrated help system.

4.1.3 MAS Architecture

The agent code generated from ADE is executable by the Autonomous Cooperative System described in (Tichý et al., 2002). This agent runtime environment is available in both a C++ version, which allows running the C++ agents on either a PC or a ControlLogix PLC, and the Java version available only for a PC. Additionally, it is possible to extend the agent code generation engine to support other target multi-agent systems such as JADE without changing the way that templates of agents and their capabilities are defined within ADE. The inter-agent communication is based on sending messages compliant with the FIPA-ACL standard, described in (FIPA, 2002). Other important attributes of a message is information about the conversation protocol that applies to the current message (e.g., FIPA-Request) and the conversation ID that uniquely identifies which context (conversation thread) this message belongs to. This is useful for keeping the track of multiple conversations, which might be happening between two agents following the same protocol at the same time. For the content language of

ACL we use our proprietary Job Description Language (JDL), described in (Mařík et al., 2005), with Bit-Efficient encoding. The FIPA messages are encapsulated inside Common Industrial Protocol (CIP) packets standardized by ODVA[2]. This encapsulation enables communication on different types of industrial networks, e.g., EtherNet/IP, ControlNet, and DeviceNet. The agents are grouped into high-level organizational structures called agent platforms. Members of each platform are usually geographically "close": for example, they may be run on one computer or PLC, or be located on a local area network. Each platform must provide its agents with the two following FIPA mandatory services: the Agent Management System (AMS) and Directory Facilitator (DF); see (FIPA, 2002). Negotiation among agents in our application is based on the Contract Net protocol described in (Smith, 1980) and its extension Plan-Commit-Execute protocol (Kadera and Tichý, 2009*b*). Any member of an agent group can temporarily become a manager that initiates a negotiation process and contacts other agents. When the contacted agent realizes that it cannot satisfy the order with its own resources, it can become an initiator of another contract to distribute subtasks among other agents. The manager chooses the best bid from the obtained bids and delegates the contract to the chosen agent(s).

4.2 Low-Level Code Generation

To support object-oriented features like instantiation, inheritance or virtuality, all of which are useful for the definition of agents, it is required to enrich the low-level control languages accordingly to enable the definition of the LLC part of the agent template. A major issue is that the LLC code cannot be dynamically "created" at runtime in the PLC as are the instances of the HLC. All the LLC instances have to be generated in advance before deployment in the PLC. This complicates the creation of the LLC template, where the system designer needs, for example, to iterate over a collection of LLC instances although he/she does not know how many of them will be really created. Additionally, the attribute values of all LLC instances are stored in PLC tags, each of which must have a unique name. For example, a LLC for a valve contains a generic I/O point called close. If there are

[2] http://www.odva.org

two valves $V1$ and $V2$, their attributes have to be stored in two different tags, e.g. $V1\_close$ and $V2\_close$. Again, it is not known at design time what the actual name of the LLC instance will be; however, there must be support for working with the attributes in the LLC template. To solve these issues we have developed unique enhancements of the IEC 61131-3 that are summarized in the following sections.

4.2.1 Indirect References

Indirect references represent a technique to access attributes of the LLC templates. A special character $ is introduced with a notion of this from typical object-oriented (OO) languages. For example, the construct $ .close in the valve LLC template is used to make a reference to the close attribute value of this class. If this special character is not used then the user references the close attribute (tag) of common attribute space. This technique is similar to public static variables that are unique and can be accessed by all components.

4.2.2 Containment

Another technique supported in the LLC definition, which is particularly applicable for hierarchical systems, is the use of containment rules. They allow a designer to specify that a component (e.g., conveyor) contains subcomponent(s) (e.g., motor). The construct $.motor1.run can be used in the LLC of the conveyor to access the run attribute of its subcomponent motor1. References in the opposite direction, i.e., to a component's parent, is also possible using the ^ character. A notation ^.ready in the LLC of the motor means the reference to my parent component's (conveyor) ready attribute. A complementary feature is the specification of the default attribute value that is set when the LLC instance is created.

4.2.3 Macro Instructions

Yet another technique is the use of macro instructions that gives the system developer the ability to specify basic operations over a collection of components. A macro instruction consists of:

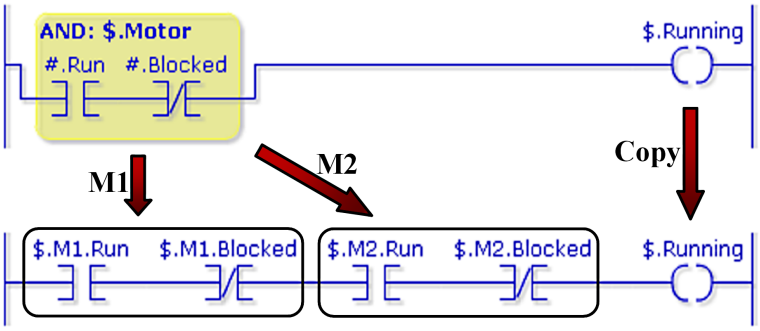


Figure 4.5: Macro instruction expansion example.

1. the definition of the operation type (e.g., AND, OR),
2. the definition of the collection (such as subcomponents of a given type or list of subcomponents separated by commas) and
3. subinstructions that can contain the special character # that is, step by step, substituted for by each element of the collection.

Using the OR operation type will in general generate a ladder logic branching construct, whereas using AND will generate an instruction series. For example, it is possible to test the value of the variable run of all subcomponents of type Motor although the instances are not known in advance. Assume that a component contains two subcomponents of type Motor, namely $M1$ and $M2$ and there is one AND macro instruction (see Fig. 4.5). This macro instruction is, during code generation, expanded two times by substituting # with $\$.M1$ and $\$.M2$ respectively.

4.2.4 Inheritance

The next technique described is inheritance. As in most OO languages, inheritance allows reuse of an existing class to define a new, more specific class that extends the original one. We have introduced the same technique also for the LLC parts of agent templates to provide full object-oriented features in the definition of control components. The major advantage of applying the object-oriented design methodology for both the HLC and LLC is the re-usability of the agent templates. Existing templates, grouped into agent libraries for specific application domains,

can be directly reused as building blocks in the development of another application. They can also be extended or adapted using the inheritance feature to the specifics of the developed system.

4.2.5 Low-Level and High-Level Integration

Interoperation between LLC code and HLC code is carried out by both asynchronous and synchronous information exchange, i.e., LLC informs HLC in two ways:

- By sending messages that are readable by HLC code and are asynchronously processed by agents. Priorities can be used here to make sure that important messages are processed before others.

- By altering values in the datatable of PLC. These values are either periodically processed by agents or, a change notification mechanism can be applied to automatically generate notifications to agents only about changed values.

The HLC affects the LLC code by changing values in the datatable of the PLC and does not require any special notification mechanism since the code in the PLC runs periodically and synchronously tests all required values. The code for component communication is generated automatically according to the information already contained in the development environment.

4.3 Conclusion

The proposed approach of a semi-automated development of industrial MASs that guides the system developer through the entire process contributes in many ways.

The concept of agent and component template libraries increases the potential of agent and component re-usability. For commercial engineering applications, it is expected that the development of the templates will be done only once by an expert with specialized skills on the agent-based technology, while the implementation of the individual instances of the MASs will be performed by system integrators, who will not need to have deep knowledge of agents.

The ADE tool significantly speeds up the development of the MASs. This is done on multiple levels. First, much of the code is generated automatically, second, the deployment of the code is also performed automatically according to the description of the control system topology.

The ADE decreased the number of errors in the code, because every piece of information is entered into the system only once and then it is distributed to other places, where it is needed, automatically.

The seamless integration of the High-Level and Low-Level control significantly eased the development of large scale MASs. The proposed approach treats both parts as integral units preserving the advantages of object-oriented programming, while the automatic code generations guaranties error-free generation of a low-level code for local real-time control.

Chapter 5

Performance Models for Agents

5.1 Introduction

This chapter introduces four performance modeling notations and is a prerequisite for chapters 7 and 8. It is necessary to point out, that there is no universal method suitable for all performance related issues of MASs. The investigation of the available performance modeling notations has identified four main candidates for the deeper analysis – Bounded Analysis, Queueing Networks, Queueing Petri-Nets, and Stochastic Process Algebras.

5.2 Bounded Analysis

The Bounded Analysis (Denning and Buzen, 1978), (Di Marco and Inverardi, 2011) based on operational laws represents the basic tool of performance analysis. It is the most straightforward modeling technique for performance considerations. It outperforms all other approaches in the speed, is simple for the implementation and does not require any complex skills from the user. It is well suitable for bounding estimations, i.e. estimation of the system performance under the best and the

worst conditions. This notation can be used only for systems with homogeneous workload, i.e. the behavior of system components (jobs, resources) does not evolve during the time. Operational laws cannot include any notion of synchronization or exclusive access. The description of the individual laws follows.

Utilization Law: During a time interval T, tasks arrive at a service station, the station adds them into the input queue and processes them task by task. The finished tasks leave the station. The activity of the server with the period T can be characterized by following attributes:

1. The number of tasks arriving at the server during T, denoted A.
2. The overall amount of time within interval T, during that the service station processes the incoming tasks B.
3. The number of tasks that were finished during T is C.

Based on this parameters, we can form following equations.

The mean service time:

$$S = \frac{B}{C} \tag{5.1}$$

.

The output rate:

$$X = \frac{C}{T} \tag{5.2}$$

.

The utilization rate:

$$U = \frac{B}{T} \tag{5.3}$$

.

Utilization law is derived from the utilization rate, which is expressed as

$$U = \frac{B}{C} \cdot \frac{C}{T} \tag{5.4}$$

that can be written as

$$U_i = S_i \cdot X_i \tag{5.5}$$

, where i states for the i-th service station.

Little's Law: The total number of tasks processed by a server at each time interval t during an observation period T provides the total amount of waiting and

processing time for all tasks during T. The measurement of completed tasks C, an observation period T, as well as accumulated waiting time W during T lead to the following calculations.

The mean response time is:

$$S = \frac{B}{C} \tag{5.6}$$

.

The mean number of tasks that are being processed by a server

$$n = \frac{W}{T} \tag{5.7}$$

.

The mean throughput of a server

$$X = \frac{C}{T} \tag{5.8}$$

.

Little's law is derived from the mean number of tasks at a service station, which is expressed as

$$n = \frac{W}{C} \cdot \frac{C}{T} \tag{5.9}$$

and can be then defined as

$$n_i = R_i \cdot X_i \tag{5.10}$$

, where i states for the i-th service station.

Forced Flow Law: This is a very important law, because it calculates the system throughput using the visit ratio and throughput of any of the service stations. Consequently, the knowledge of the visit ratio of all service stations and the throughput of any of them is sufficient to calculate the throughput of the all remaining service stations.

The law is based on following observations that are done during the period T.

- C_0 is the number of transactions done during the period T by the whole system.
- C_i is the number of transactions done during the period T by the i-th service station.

The visit ratio is defined as the average number of tasks per transaction for the i-th service station:

$$V_i = \frac{C_i}{C_0} \tag{5.11}$$

The system throughput is defined as the average number of transactions done during the time interval T:

$$X_0 = \frac{C_0}{T} \tag{5.12}$$

The service station throughput is the average number of tasks completed within T:

$$X_i = \frac{C_i}{T} \tag{5.13}$$

Simply by multiplying the system throughput by the visit ratio of a concrete service station we get its throughput:

$$\frac{C_i}{T} = \frac{C_i}{C_0} \cdot \frac{C_0}{T} \tag{5.14}$$

5.3 Queuing Networks

The basic building block of QNs (Jackson, 1963), (Baskett et al., 1975) is a single queue. A single queue (also called Service Center (SC) or Service Station (SS)) consists of a buffer, where arriving jobs wait, and a server (in general more than one server can be assigned to each queue), which serves the jobs (some authors use customers instead). The queue is characterized by:

- Arrival rate of the jobs
- Buffer Capacity
- Queuing strategy (e.g.: *first-come-first-serve*, *last-come-first-serve*, *random selection*)
- Number of servers ($n : n \geq 1$)
- Service time distributions

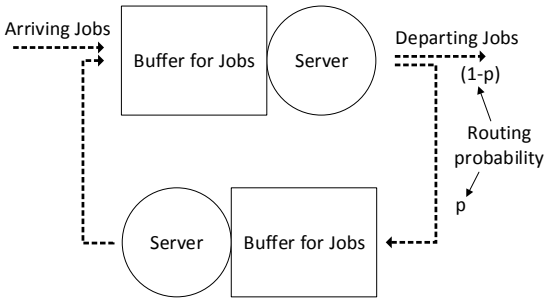


Figure 5.1: Simple Queueing Network

Queueing Network is a system of connected single queues that jobs visit sequentially. The characteristics inherited from the single queue system are extended with routing probabilities (see Fig. 5.1).

For many applications, the QNs are the best compromise between performance and expressiveness. In comparison to Bounded analysis, QNs are able to capture the ordering of jobs, both within a queue of a SC, or between particular SCs. This enables not only to identify the system bottleneck but also to identify the utilization and the queue lengths at all other SCs. On the other hand, QNs do not provide any support for modeling synchronization. In other words, this notation is not able to consider the join probabilities of parameters between various SCs. However, this can be also seen as a benefit, since this means that the computational complexity grows linearly with the number of SCs.

5.4 Queuing Petri-Nets

Queueing Petri Nets (Bause, 1993) stands on the top of the family of Petri Nets. They are time-augmented extension of Generalized Stochastic Petri Nets (GSPN) (Kartson et al., 1994). GSPN are based on Stochastic Petri nets that extend classical Petri Nets that are defined as sets of places, tokens, transitions and arcs.

Informally, Queueing Petri Nets extend the conventional ones with a new type of places - queuing places and ordinary places. A queuing place is composed of two parts. Any token, it enters to a queuing place, is added into a queue with respect to the scheduling strategy of this queue (First Come First Serve, Last Come First

Serve, etc.).

Formally, Petri Net is a 5-tuple $PN = (P, T, I^-, I^+, M_0)$ where

- P is a non-empty finite set of places
- T is a non-empty finite set of transitions
- $I^-, I^+ : P \times T \rightarrow$ are output and input incidence matrices that specify the connection between places and transitions.
- M_0 is the initial marking

GSPN is a 4-tuple $GSPN = (PN, T_1, T_2, W)$ where

- PN is the underlying Petri Net
- $T_1 \subset T$ is a set of timed transitions
- $T_2 \subset T$ is a set of ordinary (immediate) transitions, $T_1 \bigcap T_2 = \emptyset, T_1 \bigcup T_2 = T$

Colored Generalized Stochastic Petri Net (CGSPN) (Li and Georganas, 1990) is a double $CGSPN = (GSPN, C)$ where

- $GSPN$ is the underlying Generalized Stochastic Petri Net
- C is a colour set specifying the type of tokens which may reside on the place

Finally, a Queueing Petri Net (Bause, 1993) is a triple $QPN = (CGSPN, P_1, P_2)$ where

- $CGSPN$ is the underlying Colored Generalized Stochastic Petri Net
- $P_1 \subset T$ is a set of queueing places
- $P_2 \subset T$ is a set of ordinary places, $P_1 \bigcap P_2 = \emptyset, P_1 \bigcup P_2 = P$

Queueing Petri Nets provide rich vocabulary to describe various aspects of MASs. For example, they enable to describe synchronization among multiple events (necessary to express that an agent needs input from two or more other agents before it can perform an action), or they can model a mutually exclusive access to a shared resource. Their significant disadvantages are the complexity of the model definition as well as high computational demands of their solvers.

5.5 Stochastic Process Algebras

Stochastic Process Algebras (SPAs) were introduced in the 90's as a formalism for modeling performance and dependencies (Harrison and Strulo, 1995). It is currently used for modeling of complex biological systems (Hillston, 2005), (Ciocchetta et al., 2009). They are build on conventional Process Algebras, but they provide capability to represent stochastic and time characteristics of system dynamics. In comparison to all other notations introduced within this paper, SPAs explicitly supports compositionality. This is important for systems, where it is important to model the emergent behavior patterns. The main issue related to SPAs is that they are prone to exponential state explosion.

5.6 Conclusion

Performance modeling involves multiple approaches. We have considered some of them for modeling interactions between agents with focus on the time aspects to identify the system bottlenecks and the maximal system performance. Based on the both positive and negative aspects of these different notations (see table 5.1), we have focused our attention on the Queueing Networks (design-time modeling of MAS behaviour presented in chapter 7) and the Bounded Analysis (run-time congestion management presented in chapter 8), which provide a sound compromise between the expressivity, usability and solvability.

Notation	*Advantages*	*Disadvantages*
Bounded Analysis	+ Straightforward to use; + No special skills required; + Low computational demands.	– Very limited expressivity (i.e., no interactions between components, no synchronization).
Queueing Networks	+ Transparent modeling notation; + Clearly separated characteristics of individual components.	– Neither support for modeling synchronization nor joint probabilities.
Queueing Petri Nets	+ Transparent graphical modeling notation.	– Non-trivial applications generates more states than is feasible to handle.
Stochastic Process Algebras	+ High expressivity (i.e., synchronization, exclusive access, explicit connecting components); + Methods for the state aggregation to face state-space explosion.	– Requires special skills; – The aggregation of states cannot be easily automated.

Table 5.1: Comparison of performance modeling notations.

Chapter 6

Analyzing Communication

"Communication is something so simple and difficult that we can never put it in simple words."
- T. S. Matthews

6.1 Introduction

Since MASs are distributed systems of loosely coupled components (agents), the communication among the agents is essential. As stated in (Weiss, 2013): "Engineering a multi-agent system means rigorously specifying the communications among the agents by way of interaction protocols." Usually, the communication is based on message passing techniques.

Debugging and tuning MASs is a challenging process that cannot be tackled by methods and tools originally designed for development of monolithic applications. In order to fill this methodological gap which hinders a faster transfer of the agent-based technology to industrial enterprises, we proposed a set of methods based on observation and backward analysis of the agent communication. Because communication analysis is essential for the methods introduced in the following chapters, we will cover this topis in a standalone chapter describing the following:

1. The general principles of the communication used in MAS;
2. The traceability of Contract-Net Protocol-based protocols;

3. The concept of workflows and their retrieval; and

4. The construction of a loading matrix holding information of the computational load issued from a particular type of request.

6.2 Languages for Agents

Besides many ad hoc developed agent communication languages, there are two of them with the broadest uptake, namely "KQML" and "FIPA ACL".

6.2.1 KQML

Knowledge Query and Manipulation Language (KQML) is a language and protocol for exchanging information and knowledge, which was developed within the framework of the project Knowledge Sharing Efforts supported by organization DARPA. It was intended to be an adjunct to the other knowledge management methods such as ontologies. KQML defines format of messages and a message-handling protocol that supports run-time knowledge sharing among agents. The language supports agent activities as identification agents suitable for collaboration, creating connections, and information sharing.

KQML introduced a set of communication primitives such as "query" and "tell". The primitives represented specific semantics. For example, an agent using the communication act "tell" informs another agent/agents about a fact that is stored in its knowledge base. Similarly, when an agent receives a communication act "tell", it inserts the included fact into its knowledge base. This closely corresponds with an import assumption: the agents are cooperative.

The KQML97 specification contains 36 performatives split into three groups:

- Discourse performatives: These performatives are used for standard discussions

- Intervention performatives: These performatives are used for influence standard conversation process or abortion of communication.

- Networking and facilitation performatives. This group of performatives is used for handling social knowledge such as registering and unregistering agent capabilities and binding communication among agents.

6.2.2 FIPA ACL

FIPA Agent Communications Language (FIPA ACL) (FIPA, 2002) is a communication standard proposed by FIPA. The counterparts of speech acts used in KQML are in FIPA-ACL called communicative acts. However, the syntax of the ACL is very close to the KQML communication language. Although the principles are similar, performatives used in FIPA-ACL are mostly new. The total number of performatives was decreased to 22 that are split within 5 groups:

- Inform
  - Confirm: The sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition;
  - Disconfirm: The sender informs the receiver that a given proposition is false, where the receiver is known to believe, or likely believes that the proposition is true;
  - Inform: The sender informs the receiver that a given proposition is true;
  - Inform-if: A macro action for the agent of the action to inform the recipient whether or not a proposition is true; and
  - Inform-ref: A macro action for sender to inform the receiver the object which corresponds to a descriptor,for example, a name.
- Inform about a request
  - Agree: The action of agreeing to perform some action, possibly in the future;
  - Cancel: The action of one agent informing another agent that the first agent no longer has the intention that the second agent perform some action;
- Negotiate

  - Accept-proposal: The action of accepting a previously submitted proposal to perform an action ;
  - Call-for-proposal: The action of calling for proposals to perform a given action;
  - Propose: The action of submitting a proposal to perform a certain action, given certain preconditions; and
  - Reject-proposal: The action of rejecting a proposal to perform some action during a negotiation;

- Request an action
  - Propagate: The sender intends that the receiver treat the embedded message as sent directly to the receiver, and wants the receiver to identify the agents denoted by the given descriptor and send the received propagate message to them;
  - Proxy: The sender wants the receiver to select target agents denoted by a given description and to send an embedded message to them;
  - Request: The sender requests the receiver to perform some action;
  - Request-when: The sender wants the receiver to perform some action when some given proposition becomes true; and
  - Request-whenever: The sender wants the receiver to perform some action as soon as some proposition becomes true and thereafter each time the proposition becomes true again.

- Error handling
  - Failure: The action of telling another agent that an action was attempted but the attempt failed; and
  - Not-understood: The sender of the act (for example, i) informs the receiver (for example, j) that it perceived that j performed some action, but that i did not understand what j just did. A particular common case is that i tells j that i did not understand the message that j has just sent to i.

6.3 Communication Protocols

The cooperation among agents is usually realized according to a jointly accepted protocol. The protocol defines both the format and the correct sequence of messages. In this section, we describe the essential auction and Contract-Net-Protocol based communication protocols.

6.3.1 Auctions

Auction protocols provide a solution to the problem of allocating resources among agents in a MAS. We provide a short description of the most used auction types:

- English auction: The auctioneer starts English auction with a price bellow the expected market price and waits for successive bids of buyers. Each bid has to be higher than the previous price and the minimal increment is set by the auctioneer. There are various stopping conditions, e.g. (i) the auction ends at fixed time, (ii) the auction ends if there is no bid for a predefined amount of time. The deal is closed with the issuer of the last accepted bid.

- Japanese auction: It is similar to the English auction because it is also based on an iterative price growth. In contrast to the English auction, where the increases were issued by the potential buyers, in Japanese auction is the price managed by the auctioneer who starts the auction with a price bellow the expected market price and waits if any buyer accepts the price. As soon as a buyer accepts the proposed price, the auctioneer issues a new offer with a higher price. This process is repeated until no buyer accepts the price. The contract is closed with the issuer of the latest proposal, but only if the price is higher than the reserved price of the auctioneer.

- Dutch auction: The auctioneer starts Dutch auction with a price above the expected market price and decreases the proposed price repeatedly. As soon as any buyer accepts the price, the deal is closed, unless the price is below the auctioneer reserved price.

- Sealed auction: The auctioneer calls all buyers for proposals, who make their proposals privately. Afterwards, the contract is made with the issuer of the

highest proposal.

- Vickrey auction: Vickrey auction is similar to the sealed tender, but the winner of the auction does not pay the price he proposed, but the second highest proposal.

6.3.2 Contract-Net and Plan-Commit-Execute Protocols

Contract Net Protocol (CNP) is a typical example of a communication scheme used for agent negotiation. It was introduced by Smith (Smith, 1980). Later, the protocol became part of FIPA standards and is frequently used for multi-agent planning. The protocol is based on a straightforward mechanism: any member of the agent group can temporarily become a manager that initiates a negotiation process and contacts other agents. If a contacted agent cannot satisfy the request by himself, he can divide it into suborders and initiate a new conversation in order to distribute the work among other agents. The contacted agents sent their proposals, from which the manager selects the best one(s). In other words, the CNP is used to split the original complex task into a set of simpler sub-tasks that can be accomplished by multiple agents in parallel.

PCE protocol (Kadera and Tichý, 2009*b*) was developed in a reaction to demand of developers of industrial multi-agent applications, where the clear separation of the individual cooperation phases was necessary, because in the distributed industrial systems holds: "Before everything is agreed, nothing can be done". To be more specific, the optimal cooperation plan is sought in the **plan** phase. If a sufficient plan is found, the **commit** phase is used for resource allocation and if the allocation is possible, the **execution** phase starts to control the hardware outputs (see Fig. 6.1). The multiphase process prevents the agents from doing impetuous actions on the hardware level, where a rollback of an action might be a very expensive or eventually an impossible task. The negotiation in the all three phases has to be limited in time by communication timeouts to prevent the system from infinity waiting for responses (e.g. responses from a broken agent will never come). On the other hand, the timeout limit cannot be too short, otherwise some responses needed for the overall solution might be missed. The setting of the optimal timeout length is a challenging task, because the value depends on multi-

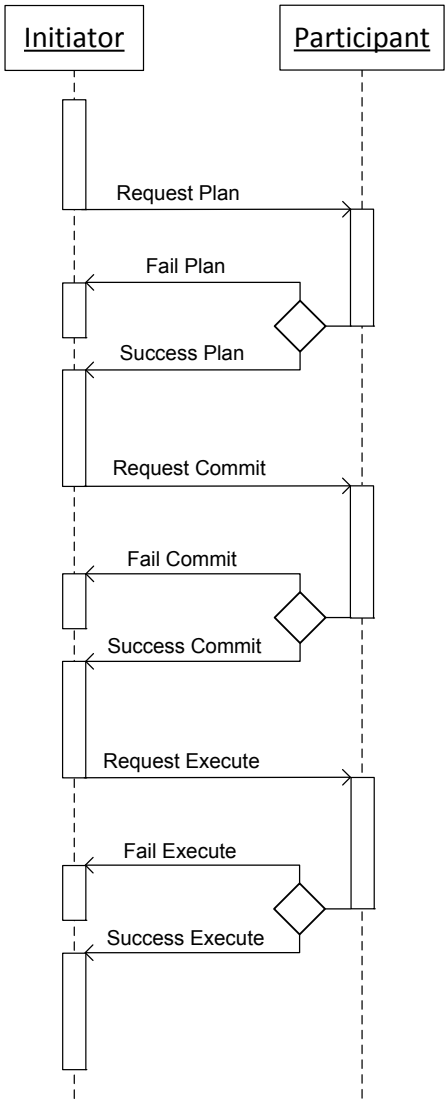


Figure 6.1: PCE Protocol

ple parameters (e.g. agent assignment to computational resources or frequency of coming requests).

6.4 Workflows

A workflow is a tree graph, whose nodes are the cooperating agents and the edges represent the communication links. The root of the tree is the initiator of the communication. The intermediate nodes represent agents that are not capable of satisfying the received request on their own, but are able to inquire other agents. On the contrary, the workflow leafs represent agents, who (i) can fully satisfy the request or (ii) cannot and even are not able to involve any other participants into the negotiation process. Java Sniffer supports visualization of multiple communication protocols such as various action protocols (English, Dutch, First Price Sealed Bid, and Second Price Sealed Bid Vickery) as well as visualization of 3-phase extension of Contract-Net Protocol named Plan-Commit-Execute (PCE) Protocol.

Messages sent among agents are composed according to the FIPA ACL specification. This specification defines a set of mandatory parameters that each message has to contain. These parameters are used to identify who the sender is and which conversation the message belongs to. Such an identification is necessary to enable agents to work on multiple tasks in parallel without mixing up messages coming from different conversations. In order to enable the backward communication traceability, agents use parameter "Reply With" to add additional pieces of information into the messages. The content of this field is composed iteratively by all agents participating in the particular negotiation. The seed of the parameter creates the initiator of the communication and stores in it its name and the number of the conversation, in which this agent takes part (e.g.: svc1@100:0). The next participant extends the content of this field by the "∼" character and its own identifier (e.g. svc1@100:0∼acp@100:310) and so on. This annotation is sufficient for the reconstruction of the workflows. A workflow visualized in Java Sniffer is depicted in Figure 6.2.

Frequently, the workflows are composed of hundreds of messages. An example of a complex worklflow is depicted in Fig. 6.3.

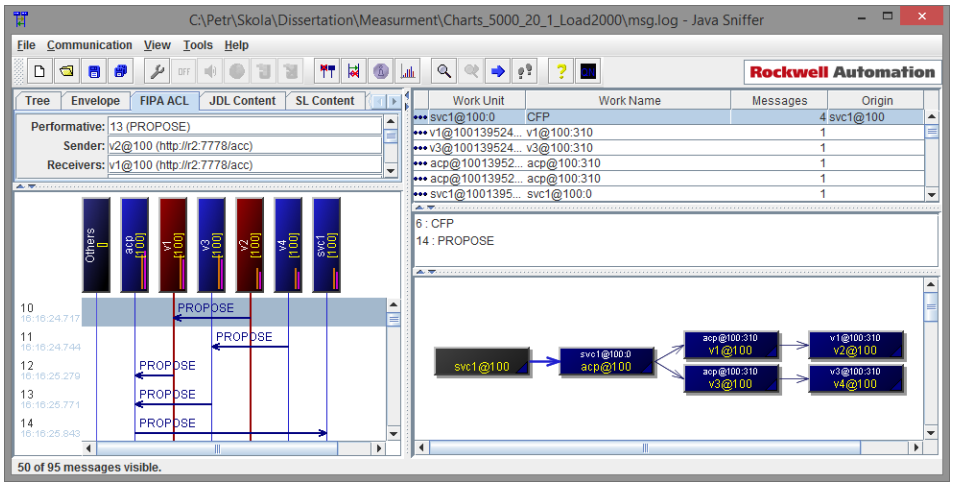


Figure 6.2: The main screen of the Java Sniffer visualizing a part of a communication log.

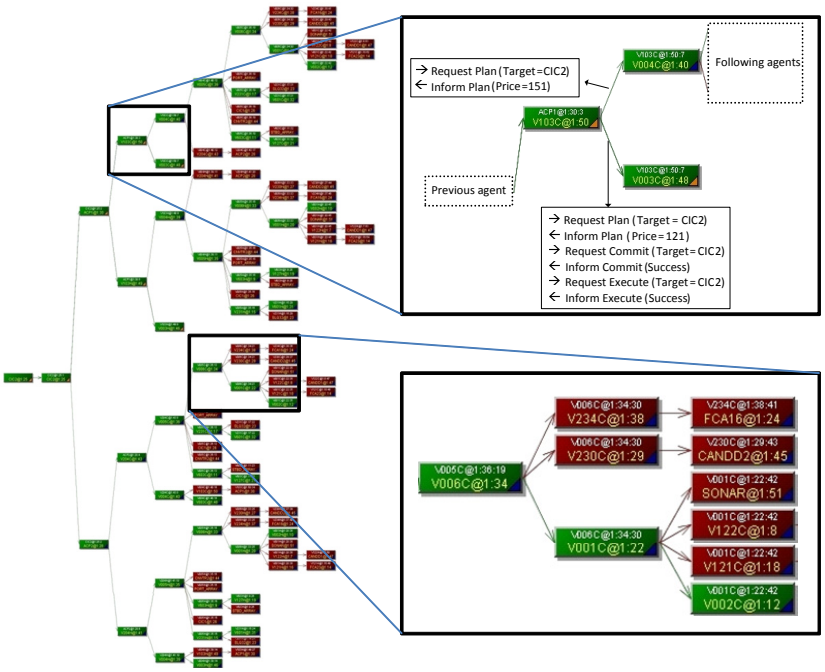


Figure 6.3: Workflow in CWS

6.5 Conclusion

In this chapter, we have briefly described the prevalent principles of the MAS communication. The chapter has introduced two dominant languages (KQML and FIPA ACL) as well as two important negotiation protocols – CNP and its extension PCE protocol. Based on the knowledge of the cooperation protocol, it is possible to organize the messages sent within an agent community into a schema named workflow, which captures the causality of the individual cooperation steps. This is an important prerequisite for further performance analysis.

Chapter 7

Verification of MAS Design

"All models are wrong, some are useful."

- George E.P.Box

7.1 Introduction

The chapter Agent Development Environment has introduced methods and tools for a foolproof design of MASs. Unfortunately, the syntactically correct design of a MAS is only necessary but insufficient condition for development of an agent-based application that also meets the performance requirements.

Industrial MASs are usually designed to control complex systems such as water distribution networks or supply chains and the practical experiments have demonstrated that the large scale MASs suffer from performance problems, for instance, long response times and overloaded computational resources. Usually, the reason for the performance degradation is an incorrect architecture of the distributed system rather than an inefficient implementation of the individual agents.

The need for specifications is aptly captured by the following quotation, paraphrased from (Lee and Seshia, 2011):

"A design without specifications cannot be right or wrong, it can only be surprising!'

Performance models enable to figure out the optimal system configuration that

meets the performance requirements, yet avoids a resource profligacy. Since the construction of performance models is a challenging process requiring specialized skills and deep knowledge of the modeled system, an integral part of the method is an automatic creation of Queueing Network performance models from logs of messages sent among agents. These models are used to estimate the performance indices (e.g. maximal system throughput, system response time or lengths of individual queues) of a MAS under various loads. This means that an initial system configuration provides information that can be used to efficiently estimate the behavior of the system under different conditions. This kind of the performance simulation assists the developer of the system to select the optimal configuration for the final MAS in order to meet the performance requirements without wasting system resources. If the experiments with the models indicate that the system cannot meet the performance requirements, the developers of the MAS have to redesign the system, i.e., optimize the behavior of individual agents or use more powerful hardware. The method is applicable to MASs utilizing the Contract-Net Protocol (CNP) (Smith, 1980) or its extension Plan Commit Execute (PCE) protocol (Kadera and Tichý, 2009*b*), because the messages produced in such systems create chains which offer to identify initiators and to trace the successors.

It is natural to expect that every MAS has its performance limits, i.e., the number of tasks that a system can solve within a time unit is limited. Moreover, it was observed, that if the load posed for the MAS exceeds a certain level, the performance does not remain on the maximal level, but decreases. Therefore, the behavior of MAS is sensitive to the current load. The reason for the performance degradation is the sharing of resources between the forward (e.g., requests) and backward (e.g., responses) messages. We illustrate this behavior on two simple models. The first represents a system with only forward propagation of requests (see Fig. 7.1 and Fig. 7.2). In this case, the system has its maximal performance limit – maximal system throughput, which cannot be exceeded, but if the load increases beyond the limit, the throughput remains on the maximal level. On the contrary, the second example (see Fig. 7.3 and Fig. 7.4) represents a system with not only forward propagation of requests but also backward propagation of responses. In this case, beyond certain point, the throughput of the system decreases.

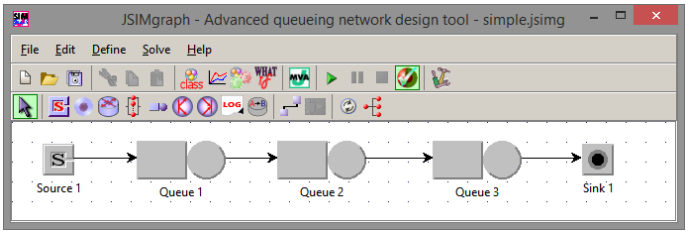


Figure 7.1: One-Way System.

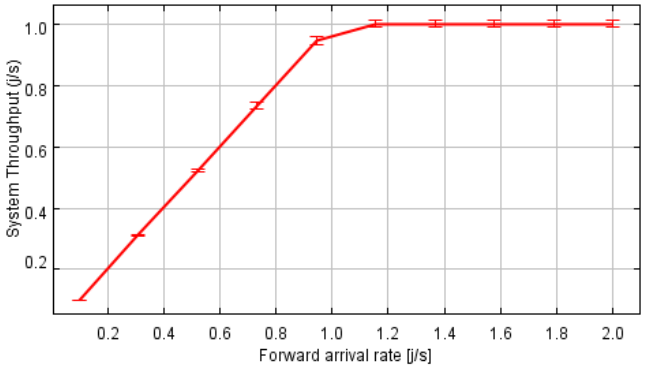


Figure 7.2: One-Way system characteristic.

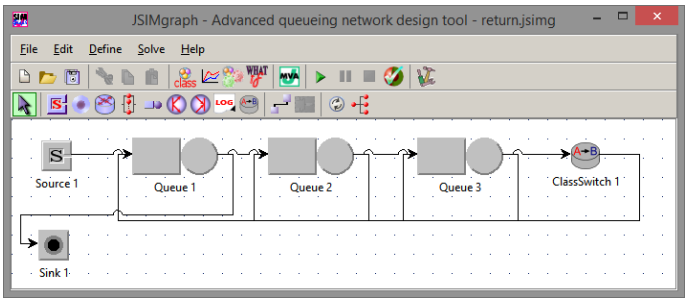


Figure 7.3: Two-Way System.

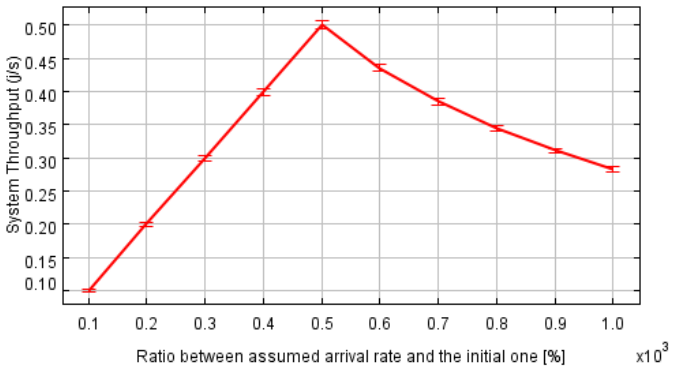


Figure 7.4: Two-Way system characteristic.

7.2 Modeling MASs using Queueing Networks

Based on our investigation of available performance modeling notations, we selected Queueing Networks as the most suitable approach. The representation of the MASs in terms of the QN notation is introduced in this section.

7.2.1 Agents

Agents are represented by service centers. For the sake of simplicity, we assume that each agent is hosted by a dedicated computational resource, i.e., the agents do not share the computational resources. Thus, every agent is modeled by a service center with properties describing the behavior of the agent. The most characteristic parameters and those that are derived from the analysis of the communication logs describe the service and routing sections of the service center. The former defines the service times of the particular agent for all types of tasks (customer types) the agent participates in. The latter describes routing of customers leaving the service center, i.e., who the next participating agents are.

7.2.2 Workflows

As described in the previous chapter, a workflow represents all communication activities related to fulfilling a task by a group of agents. Each workflow type is represented by a single customer class. The processing of a customer by the SC

depends on the class the customer belongs to.

7.2.3 Workload and Service Times

Workloads describe the service demands of individual tasks on the particular agents, i.e., the SC in QN terminology. Because the demands might fluctuate over time, they are expressed using statistical distributions. A brief description of the most used distributions follows.

Exponential distribution: This is one of the most used probability distributions in the field of QNs mainly due to its simplicity. The probability density function of this distribution is:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{7.1}$$

Normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{7.2}$$

Gama distribution: This is a two parameter probability distribution.

$$f(x) = \frac{1}{\Gamma\sqrt{2\pi}} x^{k-1} e^{-\frac{x}{\Theta}} \tag{7.3}$$

Student's T-distribution (studT(ν)):

$$f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}} \tag{7.4}$$

The listed probability distributions share a disadvantage: their parameters can be estimated only from a significant amount of data. However, the amount of available data might be limited and then simpler probability distribution comes on the scene.

Deterministic distribution: This distribution is used to describe constant demands on resources.

$$f(x) = \begin{cases} 1 & x = k \\ 0 & x \neq k \end{cases} \tag{7.5}$$

Uniform distribution: This distribution is useful when only range data $\langle a, b \rangle$ of the service time is known.

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in \langle a, b \rangle \\ 0 & x \notin \langle a, b \rangle \end{cases} \tag{7.6}$$

Triangular distribution: If the mode c is also given in addition to the range data $\langle a, b \rangle$

$$f(x) = \begin{cases} 0 & x < a \\ \frac{2(x-a)}{(b-a)(c-a)} & a \leq x < c \\ \frac{2}{b-a} & x = c \\ \frac{2(b-x)}{(b-a)(c-a)} & c < x \leq b \\ 0 & b < x \end{cases} \tag{7.7}$$

For the sake of simplicity, the experiments conducted within the framework of this thesis were limited to the deterministic distribution.

7.2.4 Request – Response Distinction

A specific aspect of the agent cooperation based on CNP is the diffusion of requests heading from the initiator to participants and backwards going responses. Although both the requests and the responses belong to the same workflow, the processing is different and therefore it is necessary to model these types of communication acts separately. In order to achieve this, a component of QNs called Class Switch is utilized at agents that act as workflow leafs, because at this point the communication is reversed and requests are replaced by responses.

7.3 Performance Indices

The most common indices used in the field of the performance of systems are as follows:

- Number of customers: This parameter indicates how many customers are present at a station or in the system as a whole;
- Queue time: Time spent in the station buffer, i.e., the parameter does not include the service time;
- Residence time: Total time that a customer spends at a station (Queue time + Service time), if the customer visits the service station during the execution multiple times, this parameter sums all visits;
- Response time: Total time that a customer spends at a station (Queue time + Service time) during a single visit;
- Utilization: Index from interval $\langle 0, 1 \rangle$ represents the utilization rate, i.e, busy time/total time;
- Throughput: Rate at which customers leave a station (station throughput) or the system (system throughput);
- System Power: This index is computed as the rate between System throughput X and System Response time R, i.e., X/R. In the optimal operational point is this index maximized.

7.4 Construction of Queueing Networks

In general, construction of QNs requires deep knowledge of the system, i.e., it is necessary to know performance characteristics of individual components, but also the interconnection of the components, which defines the routing of the jobs through the system. Since one of the essential characteristics of MASs is that their behavior emerges from the interactions of individual agents in runtime, the knowledge on the routing is not available in design time. Therefore, we propose a two-step method for construction of QNs from communication logs. First, the

parameters of individual JCs are set up according to the loading matrix. Second, the routing probabilities defining interconnections between JCs are deduced from the captured workflows that are integrated into a single structure.

7.4.1 Loading Matrix

A loading matrix is a table (see Table 7.1) containing information on how each initial request (Customer class in terminology of performance models) $C_r, r \in \mathbf{R}$, $\mathbf{R} = \{1, 2, \ldots, R\}$ loads a resource (Job Center in terminology of performance models) $J_i, i \in \mathbf{M}$, $\mathbf{M} = \{1, 2, \ldots, M\}$.

	C_1	C_2	$\cdots$	C_R
J_1	L_{11}	L_{12}	$\cdots$	L_{1R}
J_2	L_{21}	L_{22}	$\cdots$	L_{2R}
$\vdots$		$\ddots$		
J_M	L_{M1}	L_{M2}	$\cdots$	L_{MR}

Table 7.1: Loading Matrix.

The computation of the loading matrix is based on an assumption that the effort dedicated to work on a particular request is bounded by an input and an output message. How it works is described in Algorithm 1 and illustrated with a concrete example (see Fig. 7.5 and Table 7.2). For instance, the matrix cell [1,2] holds value 172, which means that the agent "v1" was one participant (with others) for 172 ms in serving the request triggered by the agent "svc1". This was computed from the time distance between the input and output message (17:05:55.093 - 17:05:54.921) at the agent "v1".

	ACP	V1	V2	V3	V4
svc1 request	177	172	330	173	333
svc1 response	99	207	0	202	0

Table 7.2: Loading matrix for the request "SVC cooling". The values represent time in milliseconds.

Because the communication among agents is bi-directional (requests flow from initiators to participants and responses vice versa) and the key part of the method is a model of the job flow between individual job centers, it was necessary to

Algorithm 1 Agent workload computation

```
for all workNode in agent.worknode list do
  for all
  inputMessage in worknode.inputEdge.message list
   do
    for all
    outputMessage in worknode.outputEdge.message list
     do
      if (inputMessage.phase == outputMessage.phase) then
        agent.load = agent.load + (outputMessage.time - inputMessage.time)
      end if
    end for
  end for
end for
```

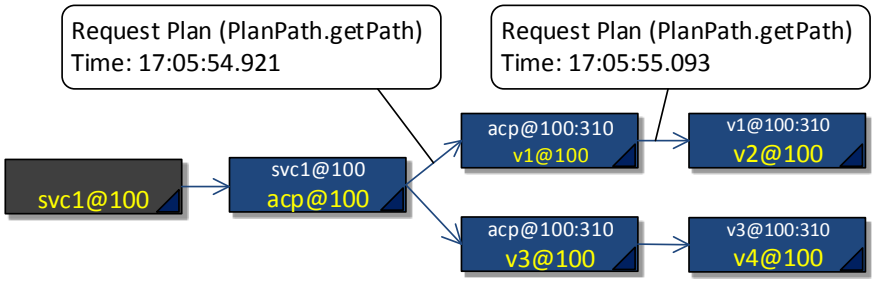


Figure 7.5: Example of a workflow that illustrates a piece of negotiation in CWS application.

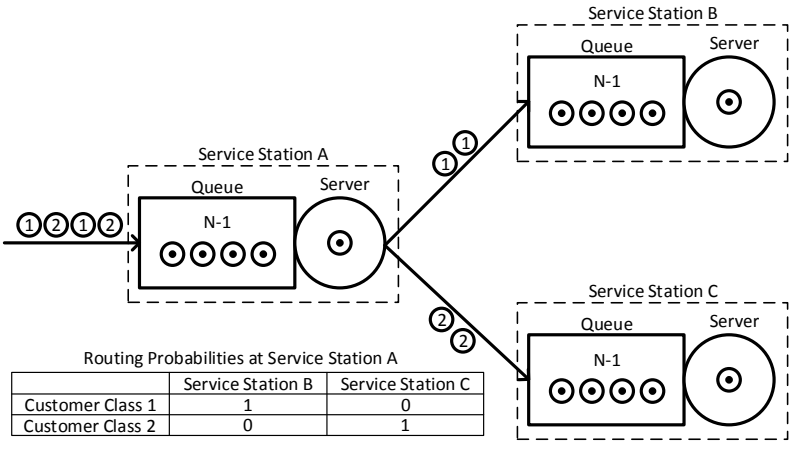
reflect this in the construction of the loading matrix. Therefore, each workflow is represented by two rows. The first row represents the load caused by the forward propagation of requests ("svc1 request"), while the second row corresponds to the load caused by the returning responses ("svc1 response").

This method assumes that the processed tasks do not influence each other, i.e., at each instance only one task is solved by the MAS. In practice, this can be accomplished by doing a test run during which the tasks are triggered one by one.

A brief introduction of the frequently used probability distributions in the field of the Queueing theory has been introduced above. The simplest approach represents the service times by the averages and then uses these values as parameters for the deterministic probability distribution. However, this approach is suitable only for systems with low fluctuations of the service times. If the variation grows, a probability distribution with higher expressivity has to be utilized in order to model also the reliability intervals of the simulation results.

7.4.2 Routing Probabilities

The flow of jobs through a Queueing Network is defined by routing probabilities. For each job center and job class the routing probabilities determine how the jobs are routed from the job center to the other job centers. If we assume that the formation of workflows is time invariant, then the probabilities can be only 0 or 1 and in fact represent a routing table 7.6. Because an agent can fork the communication, i.e., for a request start multiple parallel communication branches, a job in a QN can be also forked to multiple jobs (see Fig. 7.13 where the jobs leaving job center "ACP1" are forked to job centers "V103C" and "V103H").



Routing Probabilities at Service Station A

	Service Station B	Service Station C
Customer Class 1	1	0
Customer Class 2	0	1

Figure 7.6: Routing Probabilities

7.5 Experiments

The experimental part of this work was conducted in order to determine the accuracy of the results of the simulation and illustrate how the model can provide insight into the system by modeling performance indices such as utilization of agents. For this purpose, a part of the CWS application was implemented and deployed on a hardware testbed. The communication produced by agents was logged by the Java Sniffer and processed according to the proposed method. The obtained performance model was loaded into the JSIMgraph tool and graphically arranged within this tool (see Fig. 7.13).

7.5.1 Software Platform for Experiments

Simulation runs of the composed QNs are performed by Java Modeling Tools (JMT) (Bertoli et al., 2009). It is a bundle of six tools for performance experiments and capacity planning of computer systems. Two of them – JSIMwiz and JSIMgraph are directly compatible with the output of the previous step. The former is a wizard-based tool while the later provides a graphical user interface (GUI) for manual construction of QNs. Both tools share the same simulation engine that performs an on-line statistical analysis of measured performance indices, plots the collected values, and computes the confidence intervals.

Both JSIMwiz and JSIMgraph tools can be used to simulate the behavior of the automatically constructed QNs. Because the QNs are stored in XML files respecting the structure defined by JMT, these files can be opened as if they were created manually. The only thing, which has to be set up manually and defined via a wizard dialog of a GUI is a set of simulation parameters such as (i) performance indices to be computed, (ii) size of the simulation step, and (iii) requested accuracy are defined via a wizard dialog or a GUI. An example of a performance model imported and then modified in JMT is depicted in Fig. 7.13.

The tools provide simulation of various performance indices. System throughput is the key indicator for consideration of what the highest arrival frequency of tasks is. Its usage is described in the following subsection. Beside this, the tools provide information on how many jobs are present at each SC (important for correct settings of buffers), what the response times are (necessary for the set-

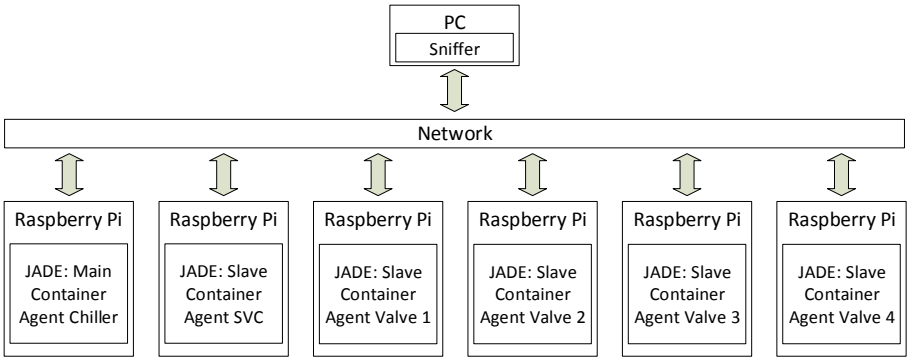


Figure 7.7: Schema of the testbed.

ting of negotiation timeouts) or utilization of individual SCs (to detect resource profligacy).

7.5.2 Hardware Platform for Experiments

The hardware testbed consists of six minicomputers Raspberry Pi model B connected to a dedicated Local Area Network (LAN) (see Fig. 7.7 and Fig. 7.8). The minicomputers host JADE platform and mimics the behavior of a part of the CWS.

7.5.3 Simulation Accuracy

We have conducted a set of experiments to investigate how accurately the estimated performance models represent the original system. In order to achieve this, we have developed a set of testing agents, that contained a method with an only purpose - to simulate a load. Moreover, the load was adjustable by an input parameter of the method.

The experiments consisted of two parts: QN simulation conducted in the JSIMgraph tool and measurements of the real system performance.

In both cases we observed, how the system throughput depends on the frequency of arriving jobs. This performance index was selected because it can be easily observed at the real system. The procedure of the experiment was identical for the model and the real system. It started with a low input frequency, which

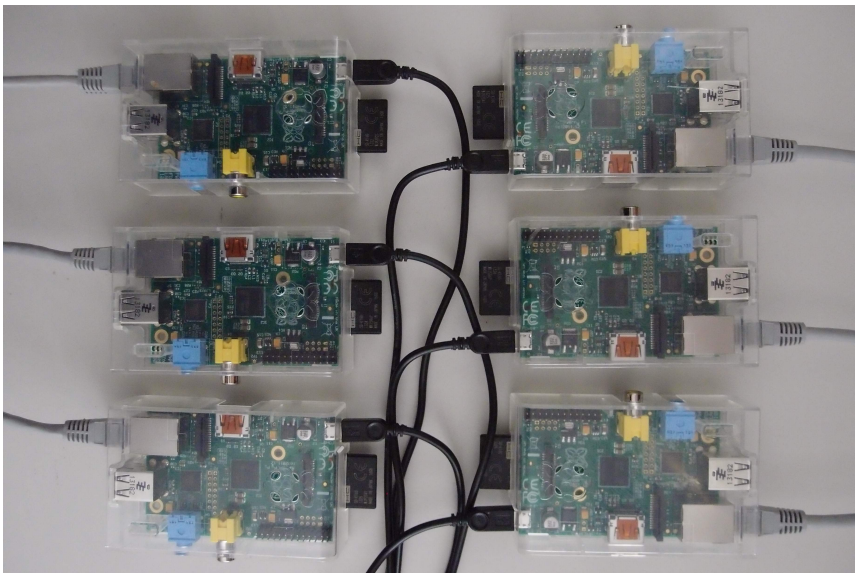


Figure 7.8: Photo of the testbed.

was then increased step by step. In all steps the corresponding output frequency was identified.

It is important to be able to quantify the model accuracy. In order to do so, we have focused our attention on the identification of the maximal frequency of arriving jobs that the system can handle from the long term perspective. The identification of this frequency is based on the following assumption:

The frequency of arriving requests (Input frequency) and the frequency of finished tasks (Output frequency) have to be equal in a sustainable MAS.

If the input frequency outperforms the output frequency, the difference denotes how intensively the MAS accumulates the unfinished requests. This accumulation initially decreases the system responsiveness, but if the imbalance exceeds a certain limit, the overall cooperation among agents breaks apart (either the buffers are completely filled, or the processing of a particular request takes so long that it is stopped by a communication timeout).

The following tables and figures show, how the model accuracy depends on the load.

Experiment 1: Load = 500, Table 7.3, Fig. 7.9

	ACP	V1	V2	V3	V4
Request	103	93	198	108	214
Response	62	54	0	68	0

Table 7.3: Loading matrix for the load 500. The values represent time in milliseconds.

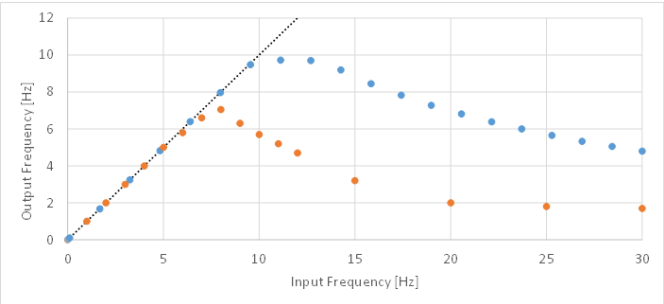


Figure 7.9: Throughput of the real system and the model - Load = 500.

Experiment 2: Load = 1000, Table 7.4, Fig. 7.10

	ACP	V1	V2	V3	V4
Request	136	127	287	132	311
Response	89	159	0	164	0

Table 7.4: Loading matrix for the load 1000. The values represent time in milliseconds.

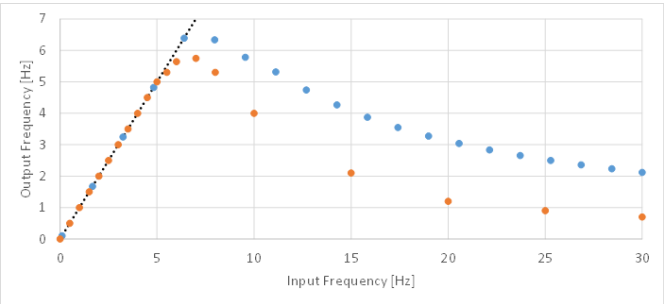


Figure 7.10: Throughput of the real system and the model - Load = 1000.

Experiment 3: Load = 1500, Table 7.5, Fig. 7.11

	ACP	V1	V2	V3	V4
Request	282	268	466	280	452
Response	214	311	0	307	0

Table 7.5: Loading matrix for the load 1500. The values represent time in milliseconds.

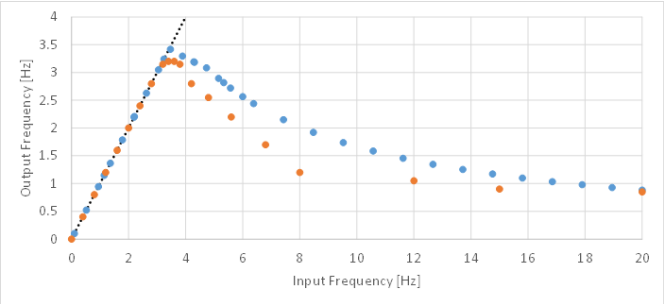


Figure 7.11: Throughput of the real system and the model - Load = 1500.

Experiment 4: Load = 2000, Table 7.6, Fig. 7.12

	ACP	V1	V2	V3	V4
Request	559	543	709	548	725
Response	480	627	0	643	0

Table 7.6: Loading matrix for the load 2000. The values represent time in milliseconds.

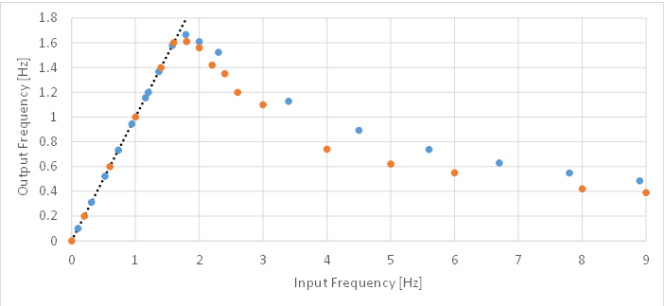


Figure 7.12: Throughput of the real system and the model - Load = 2000.

The results of the experiments confirmed our initial hypothesis that the simulation accuracy increases with the size of service times. This is caused by the fact that we do not take into account the load of the communication infrastructure. Thus, if the service times are short, the effect of the communication is more significant and the results of the simulation significantly differ from the reality. As the load of agents grows, the behavior of the models is getting more realistic.

7.5.4 Modeled Characteristics

The next experiment was focused on modeling of a part of the CWS in order to provide insight into the performance indices that are not directly observable (utilization, number of jobs in a queue, response time). The procedure consisted of the following steps:

1. The communication log was analyzed and the performance model was compiled.

2. The throughputs of the model and the real system were compared.

3. The other performance indices were modeled in order to provide insight into the internal states of the agents.

The comparison of the throughputs is depicted in Fig. 7.14. It can be read from the plot, that in the case of the real system, the input/output frequencies start to differ at approx. 2.4 Hz and in the case of the model, at approx. 2.6 Hz. Thus, the relative failure is approx. 8 %.

The obtained QN model can be further exploited to derive performance indices that cannot be observed directly. This provides the MAS developer with priceless information about utilization of individual resources (see Fig. 7.15), number of task waiting to be served, which is important for the correct setting of buffer sizes for input messages (see Fig. 7.16) and response times depending on the frequency of incoming request (see Fig. 7.17).

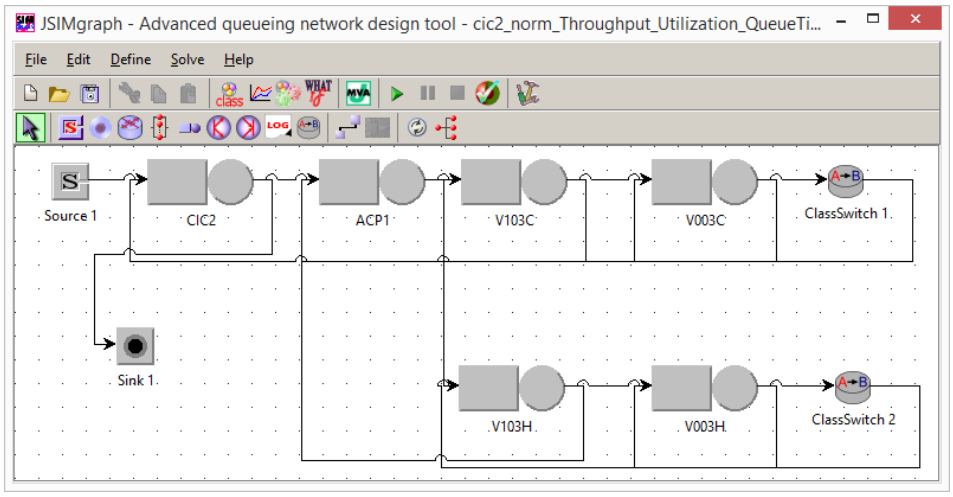


Figure 7.13: Queueing Network Model loaded into JMT representing a part of the CWS application.

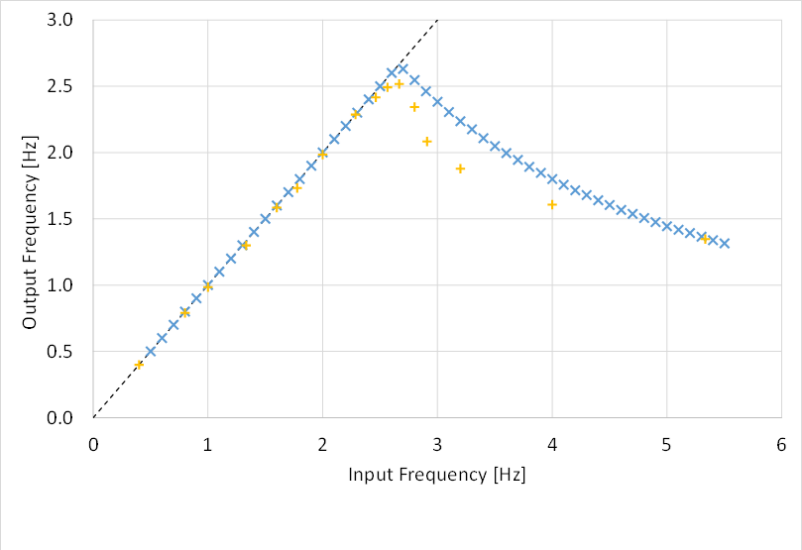


Figure 7.14: Comparison of the simulated and the real MAS behavior.

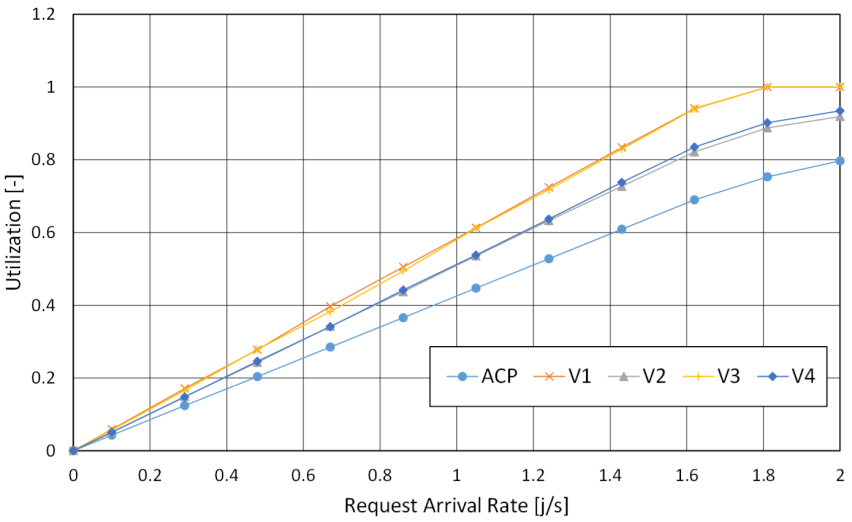


Figure 7.15: Utilization of agents.

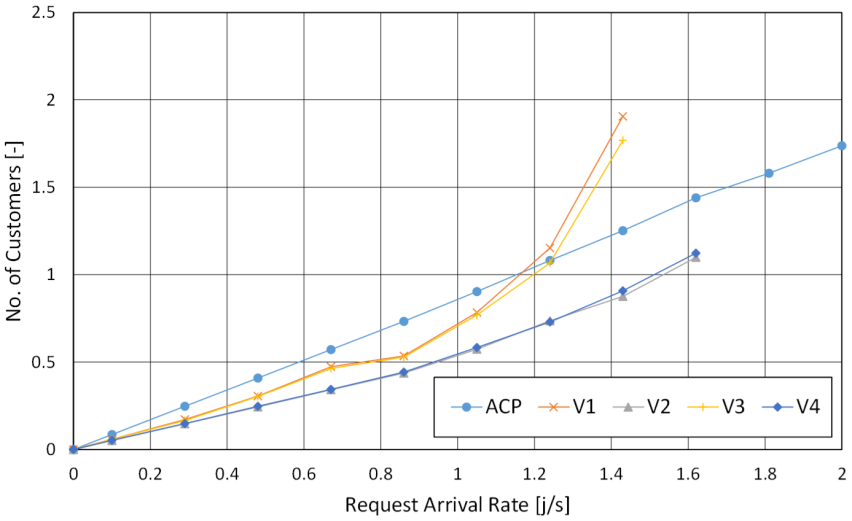


Figure 7.16: Number of Customers.

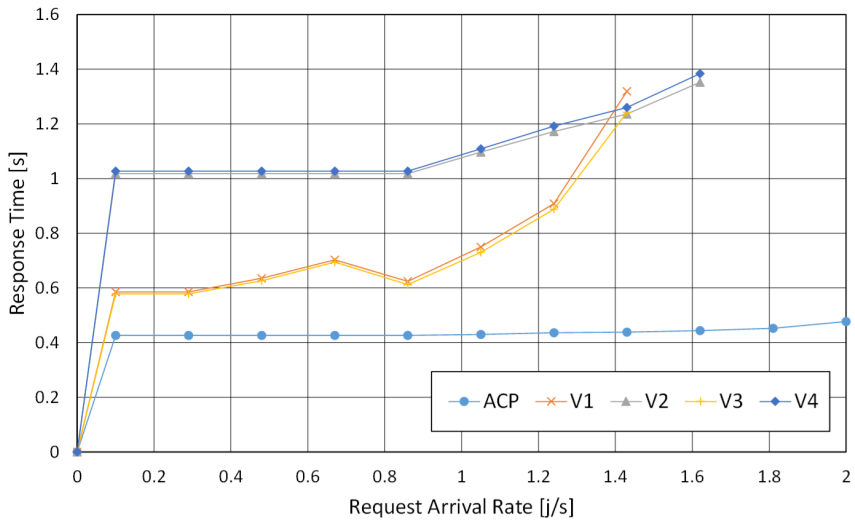


Figure 7.17: Response time.

7.6 Conclusion

The suitability of the proposed method was tested on a set of various system configurations which differed in the service time values. It was observed, that the method provides good results only if the service times of agents significantly outperform the influence of other factors, e.g. the influence of the communication network.

Furthermore, this method was used to model a part of the CWS application. This part consisted of one service agent SVC (requesting cooling), four valve agents (V1, V2, V3, V4) and one chiller agent ACP (providing cooling). Each of these agents was deployed on a dedicated mini computer Raspberry Pi. The communication was logged by Java Sniffer, which integrated the individual messages into workflows that were analyzed to derive the loading matrix. In this case only a single type of a request occurs (cooling requested by SVC), thus the integration of workflows was not applicable and the workflow was directly transformed to the queueing network model using the parameters from the loading matrix. It was tested that the model can predict the maximal frequency with a failure 8%. The obtained model was used to provide insight into other performance indicators: utilization, number of waiting customers and response times.

Chapter 8

Load-Aware Directory Facilitator

8.1 Introduction

This chapter describes a method that protects MASs based on Contract-Net Protocol (CNP) against overloading caused by bursts of requests. Providing such a protection is of the highest importance because even a temporal burst of requests can transfer a MAS to a state from which the system is not able to recover. In general, these problems are caused by saturation of computational resources. Highly utilized resources decrease the agent responsiveness, which might end up with exceeding the communication timeouts and, consequently, stop the particular negotiation attempt. If so, agents usually invoke new attempts to cooperate, but it repetitively ends up with not passing the timeouts due to the reoccurring overloads. This forms a never-ending loop, from which the agents cannot escape and the system never recovers the correct operational regime.

This closely corresponds with one of the key tasks that developers of a MAS have to solve – to find a correct setup of communication timeouts, which the agents use to bound their waiting for responses. The communication timeouts are an integral part of the most interaction protocols including the Contract-Net Protocol

and their fine-tuning is essential for reaching the optimal compromise between the communication efficiency and the system robustness. Too long timeouts decrease the performance by waiting for messages that never come (e.g. from a broken agent). On the other hand, the communication timeout cannot be too short either, because it might disallow the system to converge to the best solution due to messages that come, but come after the deadline and are thus discarded as we have many times experienced during work on the Chilled Water System (CWS) application (Kadera and Tichý, 2009*b*). Moreover, the optimal setup is specific for each system configuration (number of agents, computational performance of the used hardware, and system load). Thus, any hardware or software change conducted on a well tuned system has to be followed by a new timeout setup. This significantly limits direct usability of the component-based approach provided by the ADE, because the communication timeouts have to be tuned for every instance of the MAS individually.

This work introduces an approach (see 8.6) that replaces the fine tuning of timeouts with a congestion management mechanism that prevents the system from entering overloaded operational regimes. Performance analysis based on observation of the agent communication is utilized to select from all resources the potential bottlenecks, i.e., such resources that under certain load saturate as first ones. The impact of the new requests entering the system on the bottleneck candidates is computed and the requests are delayed if their immediate carrying out would saturate a resource. The low computational complexity makes this method applicable at runtime, where it extends functions of the regular Directory Facilitator (DF). Consequently, the DF can spread the possible burst of requests into a longer time period, to prevent any part of the system to become saturated. The method is applicable to MASs utilizing the Contract-Net Protocol (CNP) (Smith, 1980) or its extension Plan Commit Execute (PCE) protocol (Kadera and Tichý, 2009*b*), because the cooperation in such systems forms chains, which offer to identify initiators and trace the successors in order to estimate the overall impact of the initial requests.

The remainder of this chapter is organized as follows. A brief introduction of the most used agent social models is provided. Next, a bottleneck identification using convex polytopes is introduced and followed by description of a regular

Directory Facilitator extension with a load monitor. Discussion on issues related to the setting of communication timeouts and the contribution of the proposed approach follows. Finally, the results of conducted experiments are presented and a conclusion is provided.

8.2 Multi-Agent Social Models

Agents can cooperate only if they can contact each other. Therefore, maintenance of the social knowledge is one of the key capabilities of any MAS. Many architectures for maintaining social knowledge have been proposed and Tichý (Tichý, 2003) has also provided their comprehensive overview. The basic categorization of architectures distinguishes static and dynamic approaches. The static architectures are characterized by the creation of the social knowledge at design time, i.e., the social knowledge is established and distributed once and remains constant at runtime. On the contrary, the dynamic architectures are characteristic for the continuous development at runtime and changes during agents' life-cycles. A short introduction of the most used architectures follows.

Static architectures are either hierarchical or flat. The former are inspired by a common hierarchical organization in an enterprise. On one hand, its advantage is the efficiency, while on the other hand its disadvantage is low resilience, since an outage of any component disconnects all its successors from the rest of the system. In contrary, the flat knowledge base architecture is free of a superior nodes and therefore these approaches are also called "point2point" or "peer2peer".

Dynamic architectures are equipped with a mechanism that enables to maintain the social knowledge at runtime. This is necessary for the realization of the Plug&Play concept which means that a new agent can be easily added to an already running system. The focal point of any dynamic architecture is the process of **matchmaking** during which a specialized agent (frequently denoted as meta-agent) creates a communication link between two agents. Sycara (Sycara et al., 2002) defines the matchmaking as "process of finding an appropriate provider for a requester through a middle-agent".

The most used dynamical architectures are as follows:

- Broadcasting: a simple approach that generates a lot of communication. Anytime an agent finds a provider of a particular service, it sends the request to all agents. Only agents that provide such service handle the request and send a response to the initiator.
- Federated Architectures: more advanced approaches when the agents representing service providers and consumers are accompanied with meta-agents that enable creating cooperation links between agents dynamically according to the current conditions. There exist multiple variants such as: Matchmaker, Broker, Mediator, Blackboard, Monitor, Facilitator, Embassy, Anonymizer, and Job Agency. The detailed description explaining specifics of individual approaches can be found in (Tichý, 2003).

The proposed method is applicable for the federated architectures, because the meta-agents can influence the carrying out of new tasks.

8.3 Identification of Possible System Bottlenecks

As one of the key requirements on the online congestion management is the computational speed, it is of the highest importance to simplify the approach as much as possible. Thus, in this section we describe a method proposed in (Casale and Serazzi, 2004) for identification of possible bottlenecks of very large systems. The method analyzes the loading matrix of the system and identifies such resource that can be under certain circumstances a system bottleneck. Consequently, the other resources can be excluded from the performance-related considerations which decreases the computational complexity of the proposed method.

In case of systems with a single type of customers, the identification of the bottleneck is a trivial task, since it is the resource with the highest demands L_{max}. For instance, if a customer of class c_1 is served by resources R_1 for 1 seconds, by R_2 for 2 seconds, and by R_3 for 3 seconds, the L_{max} is the biggest of these values, i.e., 3 seconds. Consequently, the computation of the saturating workload expressed as the arrival rate λ_s is also a straightforward task based on the following relation:

$$\lambda_s = \frac{1}{L_{max}} \tag{8.1}$$

In the terms of the recent example, the $\lambda_s = \frac{1}{3}$ [Hz].

The identification of the saturating workload is more challenging in systems with various classes of customer. The global workload is then represented as a vector $\overline{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$, where λ_i denotes the arrival rate of customer class c_i.

Unfortunately, in case of the MAS we have to deal also with system with many types of customers that load the resources differently. In such a case, the particular load of the individual resources depends on the mix of the arriving customers, which fluctuates and cannot be easily predicted. However, the bottleneck analysis utilizes an approach coming from convex optimization in order to enable to focus the attention only to such resources that can really be bottlenecks.

The resources are divided into two basic categories: Possible Bottlenecks and Impossible Bottlenecks with further subcategories:

- Possible Bottlenecks
  - Natural bottlenecks: A resource with the highest load coming from a certain customer type.
  - Composed bottlenecks: A resource that is not a natural bottleneck, but a certain combination of customer types can make this resource to be the most demanding.
- Impossible bottlenecks
  - Dominated resources
  - Masked-Off resources

We illustrate the types of the resources in the Fig. 8.1, which is a grafical representation of a loading matrix with 5 resources and 2 customer classes. Points $R1$, $R2$, $R3$, $R4$, and $R5$ represent individual resources (in our case CPUs hosting agents). Their x resp. y-coordinates represent the load imposed by a single customer of the class 1 resp. 2. Each customer class has its natural bottleneck. It is the resource which is loaded with the particular customer class the most. For example, resource $R1$ is the natural bottleneck for customers of class 2, similarly, $R2$ is the natural bottleneck for customers of class 1. The last possible bottleneck is $R3$ that can be a bottleneck if a particular mixture of load coming from both

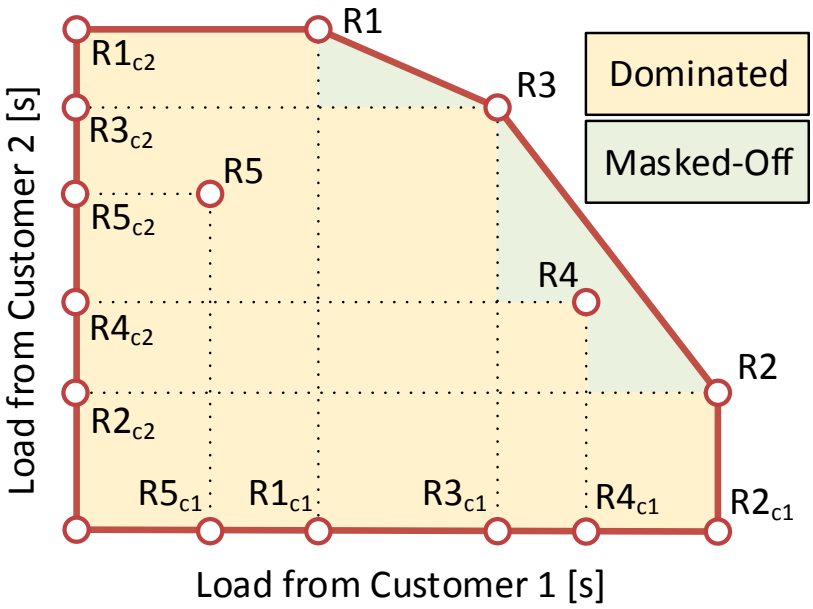


Figure 8.1: Graphical representation of a loading matrix.

types of customers is experienced. On the contrary, Resource $R5$ cannot be the bottleneck because it is dominated by resources $R1$ and $R3$. Finally, $R4$ cannot be the system bottleneck because it is masked-off by the resources $R2$ and $R3$. The figure illustrates that the possible bottlenecks lie on the edge of the convex hull that is defined by the resources.

In order to formally prove that the system bottleneck can be only such a resource that lie on a facet of the convex hull, we will introduce the following terms: The Set of points induced by matrix L is $P(L) = l_i : i \in M$. All possible projection of the vector l_i to the axis representing the customer classes is $\Pi(l_i)$. In general, the points contained in $\Pi(l_i)$ can be also included in $P(L)$, therefore we define $\Pi(L)$ as

$$\Pi(L) = \bigcup_{i=1}^{M} \Pi(l_i) - P(L) \qquad l_i \in P(L) \tag{8.2}$$

Definition 1: Let L be the loading matrix of a multi-class queueing network with M queueing centers and R classes. The characteristic polytope C_L of L is the convex hull of the set $A = P(L) \cup \Pi(L)$.

Theorem 1: The stations mapped to an internal point of C_L cannot be bottlenecks.

Proof: Let l_k be a point inside C_L. Then for any internal point l of $C(L)$ exists an $\epsilon > 0$ that holds that point $v = l_k + \lambda l$ is an internal point of $C(L)$ for all $\lambda \in \mathbb{R} : 0 < \lambda < \epsilon$.

Because v is a point of a convex set, it can be expressed as a convex combination of at most $d + 1$ vertices defining the convex set, where d is the affine dimension of the set (0 for a point, 1 for a line, 2 for a plane, etc.):

$$v = \alpha V_1 + \alpha_2 V_2 + \cdots + \alpha_{d+1} V_{d+1}, \qquad \sum_{i=1}^{d+1} \alpha_i = 1, \qquad \alpha_i \geq 0 \tag{8.3}$$

where V_i are the vertices of the characteristic polytope C_L. Because $C_L = P(L) \cup \Pi(L)$, each V_i belongs either to $P(L)$ or $\Pi(L)$. Based on the fact, that $\Pi(L)$ is projection of $P(L)$, we get for all $V_i \in \Pi(L)$ a vector $L_i \in P(L)$ such that

$$L_i \in P(L) : V_i \leq L_i, \qquad \forall i : 1 \leq i \leq d + 1, V_i \in C(L) \tag{8.4}$$

Based on this, we can express any point of $C(L)$ as a convex combination of vertices from P(L). Thus, we can write

$$l_k < v \leq \alpha L_1 + \alpha_2 L_2 + \cdots + \alpha_{d+1} L_{d+1}, \qquad \sum_{i=1}^{d+1} \alpha_i = 1, \qquad \alpha_i \geq 0 \tag{8.5}$$

Using the relation

$$\frac{L_{ir}}{L_{jr}} = \frac{U_{ir}}{U_{jr}} \tag{8.6}$$

we get

$$U_k < \alpha U_1 + \alpha_2 U_2 + \cdots + \alpha_{d+1} U_{d+1}, \qquad \sum_{i=1}^{d+1} \alpha_i = 1, \qquad \alpha_i \geq 0 \tag{8.7}$$

Because none of the stations represented by vertices $L_i \in P(L)$ is not saturated, it holds $U_i \leq 1$ for all i. Thus we substitute U_i with 1 and finally, we get

$$U_k < \alpha U_1 + \alpha_2 U_2 + \cdots + \alpha_{d+1} U_{d+1} \leq \sum_{i=1}^{d+1} \alpha_i = 1, \qquad \alpha_i \geq 0 \tag{8.8}$$

which proves the theorem.

For further analysis the transformation to the $\lambda-space$, where λ_r denotes the arrival frequency of r-customer class, is needed. $\overline{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ is the arrival frequency vector of all customer classes. This is done only for the bottleneck candidates identified in the previous step.

The transformation is based on the following equation:

$$U = \lambda L \tag{8.9}$$

For each resource r, $r \in P(L)$ in a system with n types of customers has to be guaranteed:

$$\sum_{i=1}^{n} \lambda_i l_{ir} \leq 1 \tag{8.10}$$

8.4 Scheduling Extension of the Directory Facilitator

Attention was paid to the integrability of the designed method with existing systems. The final implementation comprehends extension of two meta-agents – Sniffer and DF (see fig. 8.2). The first one is an extension of the Java Sniffer, which logs the messages and analyzes them in order to derive the loading matrix L. The matrix is then communicated via the regular messaging channel to the DF. This meta-agent utilizes the matrix to detect, whether the series of requests arriving to the system cause saturation of any resource. If the danger of saturation is detected, the particular request is postponed to temporarily decrease the workload.

The system architecture is depicted in fig. 8.2. Arrows denoted with α represent regular agent communication, the arrow β stands for logging messages, the

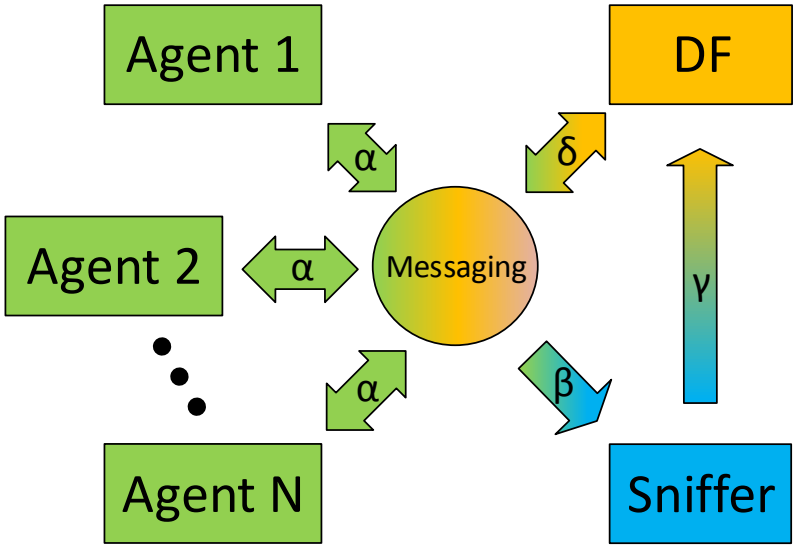


Figure 8.2: Architecture of a MAS with Sniffer and Extended DF.

arrow γ represents communication used by the sniffer to provide data obtained by the workflow analysis to the extended DF, and finally, the arrow δ represents communication between agents and DF. The proposed approach assumes, that the loading matrix remains constant for the whole application run. Thus, the loading matrix is identified during an initialization phase which becomes an integral part of the MAS development.

The DF's part extends the standard Jade's DF in three main parts. The first one is the register implemented as a HashMap, where keys are the request types and the values are timestamps denoting the time of the last occurrence of the request. A request type is a unique combination of an agent and the requested service. For instance, if the agent "A1" requests "cooling" service then the request type is "A1_cooling." The second part is related to the computation of the convex hull. The computation itself is done in Matlab (version R2014a) which provides an implementation of an algorithm for computation of the convex hull in the n-dimensional space[1]:

[1] n denotes the number of the unique request types

$$K = convhulln(L) \tag{8.11}$$

The input L is the loading matrix and the output K is a matrix*[x,y]* - x is number convex hull facets, y is the dimension. In other words, the first row of matrix K contains indices of resources from L, that demarcate the first facet. The bottleneck candidates are such points that appear among the facet members. Set of constraints for these possible bottlenecks is created according to (8.10).

When a new MAS is started and its DF is requested by the first agent to find a provider of a particular service, the DF provides the requested information immediately and temporally (for the time retrieved from the corresponding field of the loading matrix) marks the resources involved in the provision of the service as unavailable (see fig. 8.3). When the DF receives another request, it is checked whether all needed resources are already available. If they are, the requested information is again provided immediately and the temporal unavailability is again denoted. If they are not, the answer to the agent is postponed, until all the resources are available.

The suspensive mechanism is illustrated in fig. 8.4). The arriving requests are served either immediately, or are stored in a buffer of waiting messages. We have proposed a mechanism with separate queues for individual types of requests in order to avoid a situation, when a request waiting for a resource block is blocking another request that has all resources available.

There exist two different approaches on how a DF match-makes agents in a MAS. The DF used in the ACS platform directly forwards the request to registered providers of the particular service (see fig. 8.5 – (a)). Another method is utilized in JADE where the DF returns a list of service providers to the initiator of the communication (see fig. 8.5 – (b)).

The proposed method was tested on JADE. It is important to point out, that JADE provides both synchronous and asynchronous ways of communication with the DF. In this case, it is necessary to use the asynchronous one, otherwise a potential delay created by the DF would block all other actions of the service requester.

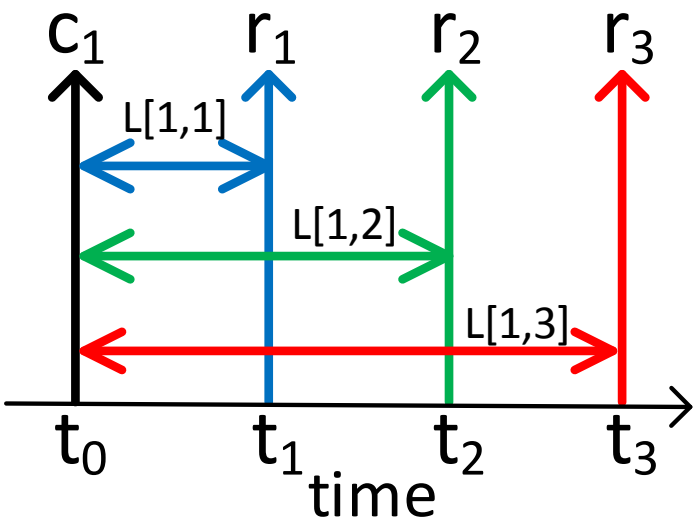


Figure 8.3: Computation of customer's arriving frequencies.

8.5 Operational Regimes

Up to now, the method was fully applicable for an arbitrary CNP-based MAS with no need to change the agents themselves, since the overload protection was implemented within meta-agents (Sniffer and Directory Facilitator). A disadvantage of that approach was a potential accumulation of delayed requests that might exceed an acceptable level. Although such a situation should not occur because the accumulation of delays would indicate a long lasting overload of the system, which should be excluded by the simulation with performance models at design time as described in chapter 7.

However, in some situations it might be effective to utilize the information of the actual delays as an indicator of the current state of the system. Then, the agents might be requested to simplify their actions when a major overload is

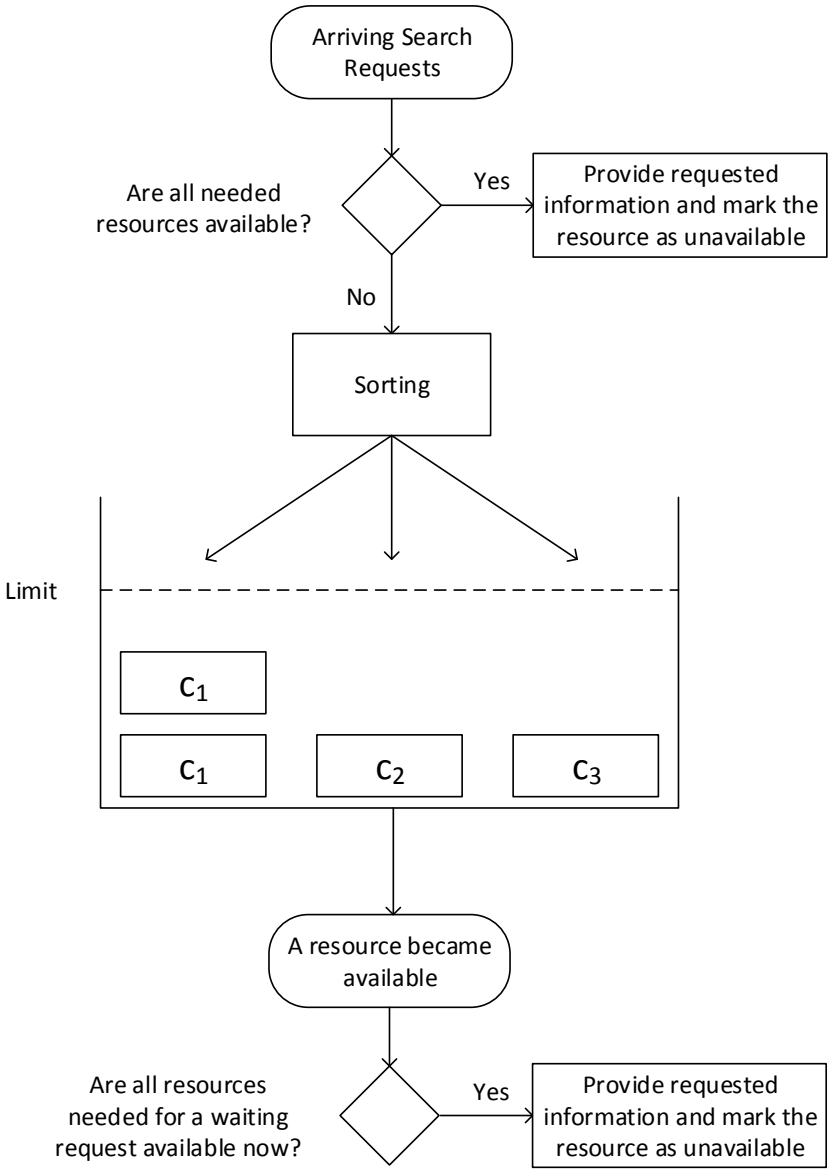


Figure 8.4: Suspensive mechanism of DF scheduling.

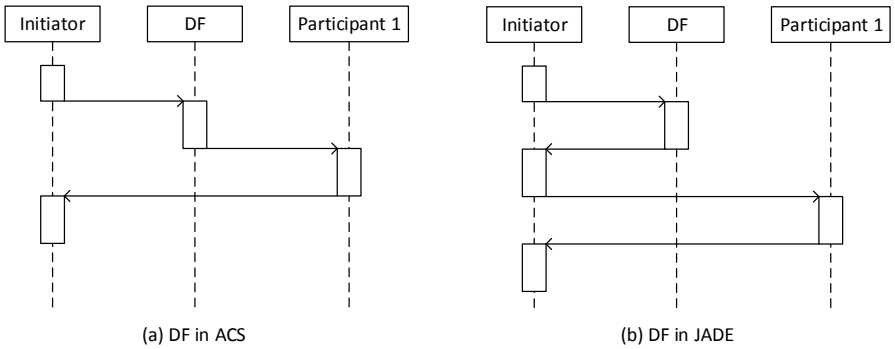


Figure 8.5: Different match-making mechanisms.

detected.

The response time can be expressed as a function of time:

$$R(t) = (\lambda L_n)t - t = ((\lambda L_n) - 1)t \tag{8.12}$$

If $(\lambda L_n) > 1$ the function $R(t)$ grows above any limit and the number of requests delayed in the DF also constantly grows. If we cannot reduce λ we have to decrease L_n to L_s for which holds $(\lambda L_s) < 1$ in order to decrease the response time. The system eliminates the accumulated delay ($R(t) = 0$) in time t_2:

$$t_2 = \frac{\lambda L_n - 1}{\lambda L_s - 1} t_1 \tag{8.13}$$

The proposed approach has to be supported also on the level of agents. The ADE can contribute to this approach on the level of agent templates which might contain a skeleton of code where the developer implements the desired agent behavior.

8.6 Communication Timeouts

As we have already mentioned above, the correct setting of communication timeouts is a challenging and time-consuming process and we will discuss it in detail in this section.

Optimally, a timeout is equal to the corresponding response time of the system. As the utilization of the system resources grows, it becomes more likely that a new request will have to wait until requests that arrived before will be finished. The relation between the utilization of a resource and the response times is derived as follows:

The response time is the service time of the request itself + service times of all the requests ahead, i.e.,

$$R = S(1 + N) \tag{8.14}$$

applying Littles Law

$$N = XR \tag{8.15}$$

we get

$$R = S(1 + XR) = S + SXR = \frac{S}{1 - XS} \tag{8.16}$$

Application of Utilization Law brings the final formula:

$$R = \frac{S}{1 - U} \tag{8.17}$$

where R is response time, S is service time and U is current utilization. The characteristic of this relation is depicted in Fig. 8.6. This shows, how the sensitivity of the response time grows with the utilization and why the correct setup of the timeouts is so difficult.

As the proposed method prevents the system from triggering new tasks unless there are available computational resources to serve them, the jobs do not queue in service centers and the response times equals to service times. Thus the timeouts can be adjusted according to the response times observed during the initialization phase, when all expected types of jobs were triggered one by one to hinder their mutual influence. A response time of an agent that involves into the cooperation other agents is composed of several summands (see Fig. 8.7).

In the ideal case, the response time of a job fulfillment is constant and in fact represents the optimal timeout. However, the system behavior is influenced by other aspects (e.g. load of the communication network) that are not covered by this modeling approach, the precision of the method is limited (see (**?**) where the problem of performance model accuracy has been already discussed). This has to

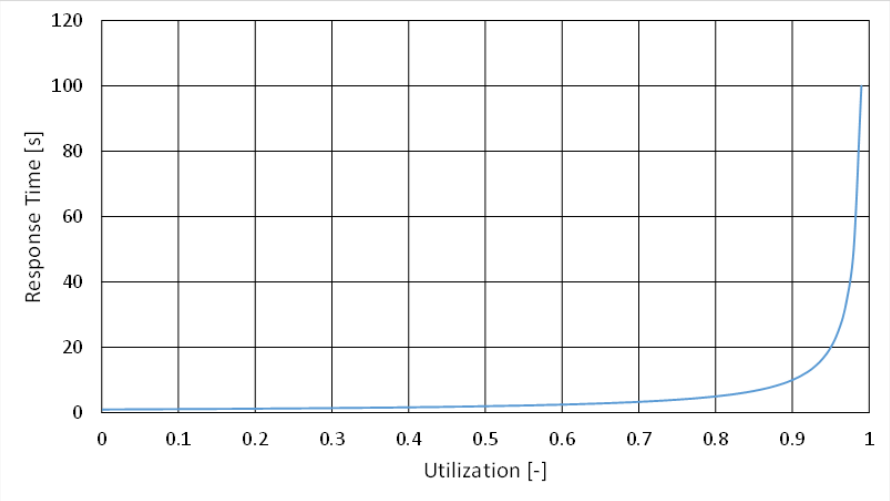


Figure 8.6: Relation between response time and utilization.

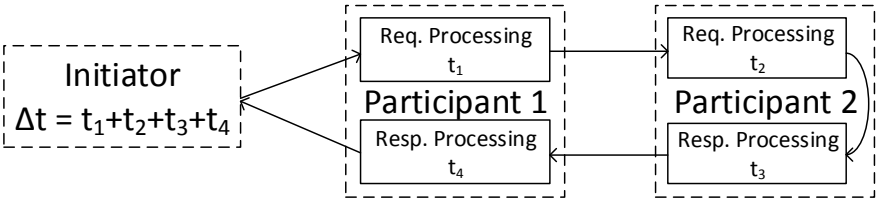


Figure 8.7: Response time.

be reflected in the final timeout setup. Therefore the final formula for the timeout setup is as follows:

$$T_{max} = RC \tag{8.18}$$

where T_{max} is the recommended timeout, R is the observed response time during the test run during initialization and C is a correction coefficient that increases the response time. The setup of the coefficient has to be done with respect to the specific application and it is not possible to provide a universal method for its computation. Nevertheless, there are several hints that should be followed.

- The parameter has to be always greater than 1 as the system will never react faster in run time than during the initialization.
- The higher the error of service time estimation is the bigger the parameter C has to be.
- The developer of the MAS has also to take into account the trade off between potential impact of too short or too long timeouts, respectively.

8.7 Experimental Evaluation

Using the testbed already mentioned in the section 7.5.2, we set up a system with 6 Jade agents respecting the layout of the queueing network experiment (see Fig. 7.13). The plot (see fig. 8.8) depicts the relation between frequency of request entering the system (Input Frequency) and the frequency of finishing the jobs – system throughput. The measured characteristic confirms the existence of a system throughput maximum, i.e., the frequency at which the system returns the optimum throughput.

The positive impact of using the extended version of the DF is clearly shown in fig. 8.9. The points marked by crosses represent measurements done with the regular JADE system and it demonstrates that after crossing a certain frequency the amount of jobs passed by a certain deadline steeply decreases. On the other

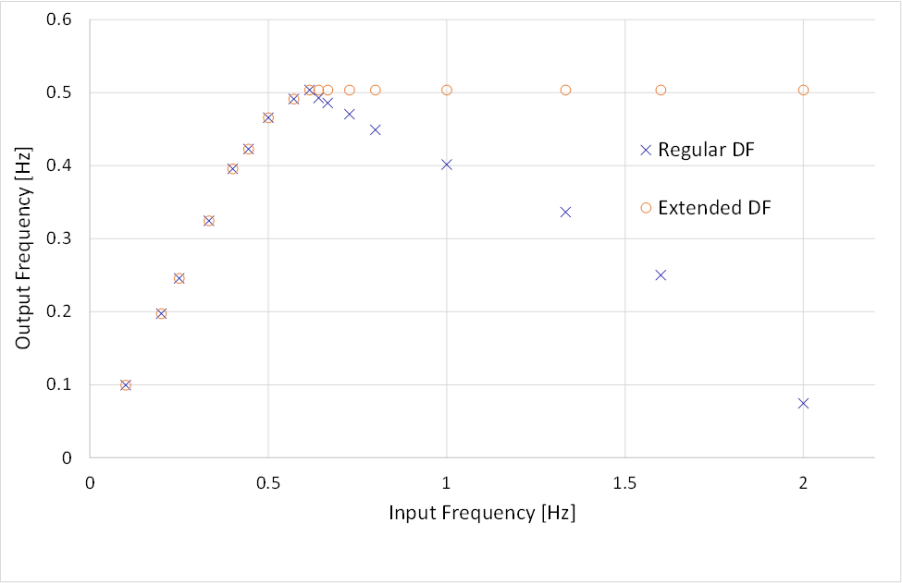


Figure 8.8: System Throughput

hand, the modified version of the DF regulates the frequency of the input jobs, which ensures the preservation of the passing rate on the level of 100 %.

The further experiments were focused on systems with more customer classes. We present the contribution of the proposed approach on a 2-customer system, i.e., on a system, where two types of customer requests occur. It was created by extending the previous 1-customer experiment by a new type of request, but still using the same testbed. The measurements go along with the previous results regarding the performance degradation, but on top of that also demonstrate the strong influence across input frequency of individual customer classes (see Fig. 8.10 and Fig. 8.11).

8.8 Conclusion

This chapter described how bursts of events can endanger the ability of a MAS to converge to a solution. The reasons of the steep increase of response times with growth of utilization were also explained. Consequently, a congestion management

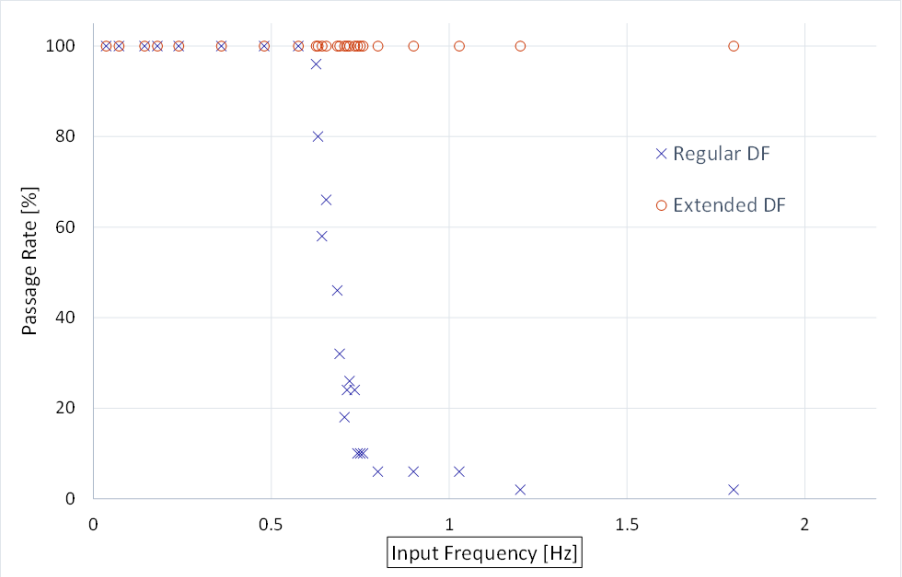


Figure 8.9: Passage Rate.

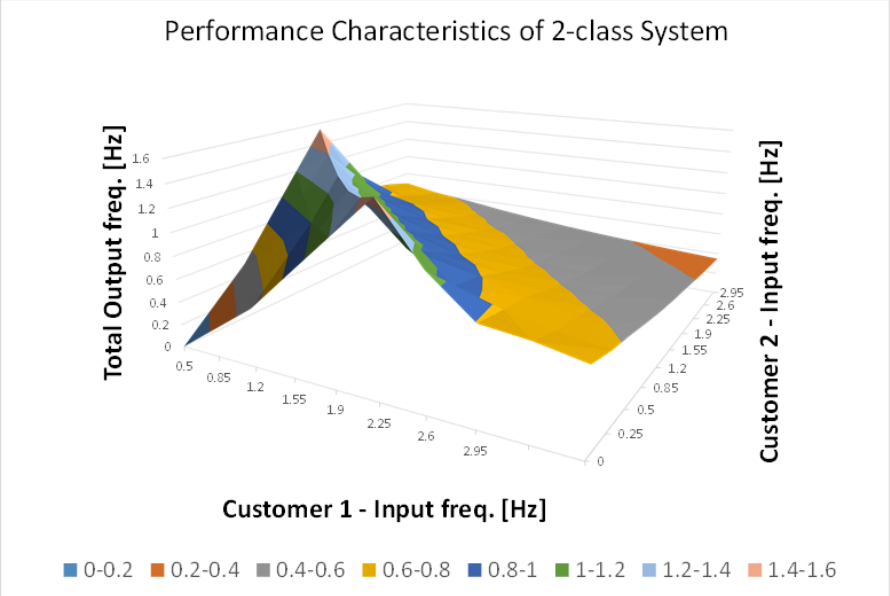


Figure 8.10: Original System.

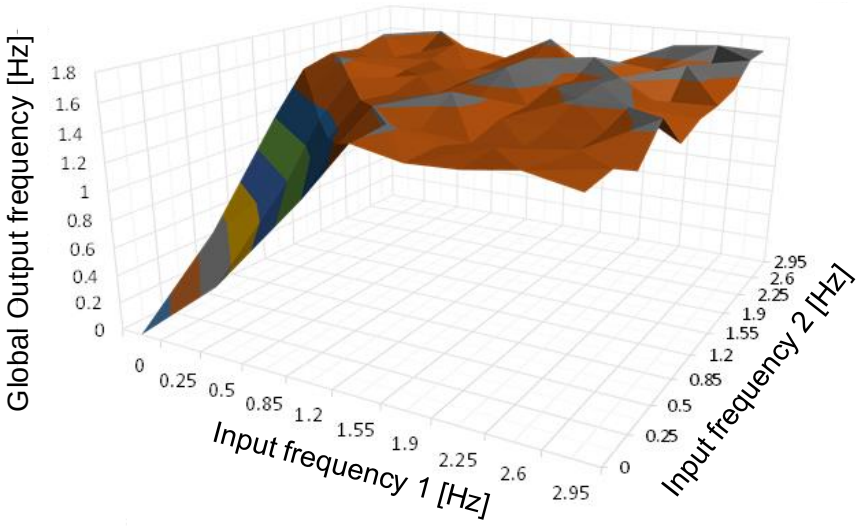


Figure 8.11: System with Extended DF

control was introduced. It is based on an efficient algorithm that identifies from all resources the potential bottlenecks whose utilization is controlled at runtime. Furthermore, this chapter provides guidelines on how to set up communication timeouts in MASs. Results of experiments conducted on a test bed (6 minicomputers Raspberry Pii) illustrate the contribution of the proposed method.

Chapter 9

Conclusion

This thesis provided an insight into development of industrial MASs. It started with an introduction of the industrial application areas for the agent technology, which demonstrated that the agents penetrate the industrial domain on multiple levels. Not only agents control flow of material in complex networks, but also they are utilized to plan and schedule manufacturing operations and control assembly in flexible manufacturing systems.

The advent of the agent-based solutions is followed by the development of supportive tools. Their survey was presented in chapter 3. Although a lot has been done in this field, a need for a comprehensive methodology, which would provide guidelines to developers of industrial MASs, is apparent.

The existence of this methodological gap is reflected by introduction of the ADE. This tool guides a developer of a new industrial MAS through the entire development process. This tool interconnects the high-level object oriented approaches used for development of agents with low-level control designed according to the standard IEC 61131, which is used for implementation of the local low-level control with real-time requirements.

The remainder of the thesis solves performance aspects of agent-based approaches, since the ability to estimate and guarantee certain performance is essential for the wide acceptance of MASs by industrial enterprises. The first performance-related method eases the performance and capacity planning of a

new MAS at design time. It automatically transforms communication logs into performance models, which provides insight into the inner states of the system resources. Moreover, the method enables to derive the maximal throughput of the system.

The second performance-related method is designed to increase the resistance of MASs against temporal bursts of arriving requests. The ability of MASs to handle bursts of events is tightly coupled with correct setting of communication timeouts that have to respect the corresponding response times. We have shown, how the response times depend on the actual load of the system resources and we have proposed a congestion management method that continuously observes the cooperation among agents and protects the system from potential overload by postponing of new requests.

9.1 Fulfillment of the Thesis Goals

- A set of methods has been developed and integrated into a tool named Agent Development Environment. The tool guides a programer of a new MAS through the whole development process and contributes to its fast and error free design.

- A method that enables an automated compilation of performance models based on Queueing Network notation has been developed. The obtained model provides insight into the behavior of individual components under various loads, which speeds up and refines the development of a new MAS. The utilization of the method has been demonstrated on a distributed hardware platform.

- An extension of the existing management of the social knowledge within a community of agents has been proposed in order to control the load of individual computational resources. It has been shown that only some resources can be system bottlenecks and thus only load of those has to be monitored and eventually controlled. The monitor and control functions have been integrated into meta-agents Sniffer and Directory Facilitator. This makes this

approach directly applicable with an arbitrary FIPA-compliant multi-agent system.

9.2 Contribution of the Thesis

This thesis contributes to the development of industrial MASs on multiple levels.

1. The thesis proposes Agent Development Environment that supports developers of MASs. Particularly, it provides methods that enable the automated integration of high-level object oriented programing (e.g. Java, C++) used for implementation of agents and the low-level programing of Programmable Logic Controllers (e.g. IEC 61131, IEC 61499) designed for real-time control on the shop-floor. ADE is a team project. The author of this thesis designed and developed mechanisms for synchronized instantiation of High-Level Control parts (agents) and generation of the Low-Level Control Parts (e.g. concepts of inheritance and macro-instructions).

2. The thesis introduces a method for automatic compilation of performance models from logs of messages. This enables to speed up the the process of finding the optimal distribution of the agents across the computational resources. The corresponding part of the work has been completely elaborated by the author of this thesis.

3. The thesis proposes a new role of a Directory Facilitator. Originally, a Directory Facilitator is a meta-agent managing the social-knowledge within an MAS. We propose an extension that builds upon the central position of the DF agent in the negotiation process. The extended DF keeps a track of all the requests that entered the system and derives the current load of the system. The thesis proposes a new approach to dealing with setting of communication timeouts. The approach is based on the congestion management that enables to face to uncontrolled growth of response times that would be caused by highly utilized resources. First, the forthcoming saturation of a computational resource is detected in advance. Second, the new arriving requests are postponed in order not to exceed the edge of saturation.

Third, if the cumulative delay of the arriving request reaches a certain level, the agents are informed to switch from "normal" to a "stressed" mode, in which the agents simplify their actions (for instance, instead of seeking for the optimal provider of a service the last one used is contacted directly). The corresponding part of the work has been completely elaborated by the author of this thesis.

9.3 Future Work

The proposed methods have a great potential for further development as they are not limited only to the domain of MASs. For instance, the continuously growing field of service oriented architectures is another promising application area. The first step in this direction has been already done by the introduction of a sniffer for JBoss ESB (Vrba et al., 2014). Further, the proposed principles are also applicable for event-driven control systems (e.g. IEC 61499) which have to be designed with respect to the performance limits of the computational resources. Otherwise, a part of the system might get saturated and cause a failure. We have experienced this type of problems when working on system connecting IEC 61499 control system with an HMI ScadaBR. Under certain circumstances, the control system sent updates to the HMI too frequently which resulted in a failure and consequently in a loss of data.

Bibliography

Andreev, S., Rzevski, G., Shviekin, P., Skobelev, P. and Yankov, I. (2009), A multi-agent scheduler for rent-a-car companies, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer, pp. 305–314.

Baskett, F., Chandy, K. M., Muntz, R. R. and Palacios, F. G. (1975), 'Open, closed, and mixed networks of queues with different classes of customers', *Journal of the ACM (JACM)* **22**(2), 248–260.

Bause, F. (1993), Queueing petri nets-a formalism for the combined qualitative and quantitative analysis of systems, *in* 'Proceedings of the 5th International Workshop on Petri Nets and Performance Models', IEEE, pp. 14–23.

Bellifemine, F. L., Caire, G. and Greenwood, D. (2007), *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*, John Wiley & Sons. ISBN 0470057475.

Bertoli, M., Casale, G. and Serazzi, G. (2009), 'JMT: performance engineering tools for system modeling', *ACM SIGMETRICS Performance Evaluation Review* **36**(4), 10–15.

Bitting, E., Carter, J. and Ghorbani, A. A. (2003), Multiagent system development kits: An evaluation, *in* 'Proceedings of the 1st Annual Conference on Communication Networks and Services Research', pp. 15–16.

Black, G. and Vyatkin, V. (2007), On practical implementation of holonic control principles in baggage handling systems using iec 61499, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer, pp. 314–325.

Bratman, M. E. (1999), *Intention, Plans, and Practical Reason*, Cambridge University Press.

Brennan, R. W., Vrba, P., Tichý, P., Zoitl, A., Sünder, C., Strasser, T. and Mařík, V. (2008), 'Developments in dynamic and intelligent reconfiguration of industrial automation', *Computers in Industry* **59**(6), 533–547.

Buschmann, F., Henney, K. and Schimdt, D. (2007), *Pattern-oriented Software Architecture: On Patterns and Pattern Language*, Vol. 5, John Wiley & Sons.

Bussmann, S., Jennings, N. and Wooldridge, M. (2004), *Multiagent systems for manufacturing control: a design methodology*, Springer.

Bussmann, S. and Schild, K. (2000), Self-organizing manufacturing control: An industrial application of agent technology, *in* 'Fourth International Conference on MultiAgent Systems', IEEE, pp. 87–94.

Camarinha-Matos, L. M. (2002), Multi-agent systems in virtual enterprises, *in* 'Proceedings of international conference on AI, simulation and planning in high autonomy systems', pp. 27–36.

Casale, G. and Serazzi, G. (2004), Bottlenecks identification in multiclass queueing networks using convex polytopes, *in* 'Proceedings of the IEEE Computer Society's 12th Annual International Symposium onModeling, Analysis, and Simulation of Computer and Telecommunications Systems', IEEE, pp. 223–230.

Christensen, J. H. (1994), 'Holonic manufacturing systems: initial architecture and standards directions', *Proceedings of the 1st Euro Workshop on Holonic Manufacturing Systems* .

Ciocchetta, F., Duguid, A., Gilmore, S., Guerriero, M. and Hillston, J. (2009), The bio-pepa tool suite, *in* 'Proceedings of the 6th international conference on the quantitative evaluation of systems.', pp. 309–310.

Commission, E. (2003), Futman – the future of manufacturing in europe 2015-2020 – the challenge for sustainability, Technical report, European Commission.

Commission, E. (2007), Manufuture – strategic research agenda, assuring the future of manufacturing in europe, Technical report, European Commission.

Cruz, R. (1991), 'A calculus for network delay. i. network elements in isolation', *IEEE Transactions on Information Theory* **37**(1), 114–131.

Denning, P. J. and Buzen, J. P. (1978), 'The operational analysis of queueing network models', *ACM Computing Surveys (CSUR)* **10**(3), 225–261.

Di Caro, G. and Dorigo, M. (1998), 'Antnet: Distributed stigmergetic control for communications networks', *Journal of Artificial Intelligence Research* **9**, 317–365.

Di Marco, V. C. A. and Inverardi, P. (2011), *Model-based software performance analysis*, Springer.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische mathematik* **1**(1), 269–271.

European SmartGrids technology platform: vision and strategy for Europe's electricity networks of the future (2006).
URL: *http://cordis.europa.eu*

Evertsz, R., Fletcher, M., Jones, R., Jarvis, J., Brusey, J. and Dance, S. (2004), Implementing industrial multi-agent systems using jack™, *in* 'Proceedings of the 1st International Workshop on Programming Multi-Agent Systems', Vol. 3067, Springer, p. 18.

FIPA (2002), 'Fipa acl message structure specification'.
URL: *http://www.fipa.org/specs/fipa00061/*

Hall, K. H., Staron, R. J. and Vrba, P. (2005), Experience with holonic and agent-based control systems and their adoption by industry, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer.

Hallenborg, K. and Demazeau, Y. (2006), Dynamical control in large-scale material handling systems through agent technology, *in* 'IEEE/WIC/ACM International Conference on Intelligent Agent Technology', pp. 637–645.

Harrison, P. and Strulo, B. (1995), Stochastic process algebra for discrete event simulation, *in* 'Quantitative Methods in Parallel Systems', Springer, pp. 18–37.

Hart, P. E., Nilsson, N. J. and Raphael, B. (1968), 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Transactions on Systems Science and Cybernetics* **4**(2), 100–107.

Hillston, J. (2005), Process algebras for quantitative analysis, *in* 'Proceedings of 20th Annual IEEE Symposium on Logic in Computer Science', pp. 239–248.

Ivaschenko, A., Khamits, I., Skobelev, P. and Sychova, M. (2011), Multi-agent system for scheduling of flight program, cargo flow and resources of international space station, *in* V. Mařík, P. Vrba and P. Leitão, eds, 'Holonic and Multi-Agent Systems for Manufacturing', Vol. 6867 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 165–174.
URL: *http://dx.doi.org/10.1007/978-3-642-23181-0*$_1$6

Jackson, J. R. (1963), 'Jobshop-like queueing systems', *Management science* **10**(1), 131–142.

Jennings, N. R., Sycara, K. and Wooldridge, M. (1998), 'A roadmap of agent research and development', *Autonomous agents and multi-agent systems* **1**(1), 7–38.

Kadera, P. and Tichy, P. (2009), Plan, commit, execute protocol in multi-agent systems, *in* 'Proceedings of 4th International Conference on Industrial Application of Holonic and Multi-Agent Systems', pp. 155–164. PT: S; CT: 4th International Conference on Industrial Application of Holonic and Multi-Agent Systems; CY: AUG 31-SEP 02, 2009; CL: Linz, AUSTRIA; UT: WOS:000270319700015.

Kartson, D., Balbo, G., Donatelli, S., Franceschinis, G. and Conte, G. (1994), *Modelling with generalized stochastic Petri nets*, John Wiley & Sons, Inc.

Kumar, V. and Cole, E. (2005), An ant colony optimization model for wireless ad-hoc network autoconfiguration, *in* 'IEEE International Conference on Systems, Man and Cybernetics', Vol. 1, IEEE, pp. 103–108.

Lastra, J. L. M., Torres, E. L. and Colombo, A. W. (2005), A 3d visualization and simulation framework for intelligent physical agents, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer, pp. 23–38.

Lee, E. A. and Seshia, S. A. (2011), *Introduction to Embedded Systems.*

Leitão, P. and Rodrigues, N. (2012), Modelling and validating the multi-agent system behaviour for a washing machine production line, *in* 'Proceedings of IEEE International Symposium on Industrial Electronics', pp. 1203–1208.

Li, M. and Georganas, N. D. (1990), 'Coloured generalized stochastic petri nets for integrated systems protocol performance modelling', *Computer Communications* **13**(7), 414–424.

Manikonda, V., Satapathy, G. and Levy, R. (2004), 'Cybele: An agent infrastructure for modelling, simulation, and decision support', *AgentLink News* (15), 25–26.

Mařík, V., Vrba, P., Hall, K. H. and Maturana, F. P. (2005), Rockwell automation agents for manufacturing, *in* 'Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems', ACM, pp. 107–113.

Maturana, F. P., Staron, R., Hall, K., Tichý, P., Šlechta, P. and Mařík, V. (2005), An intelligent agent validation architecture for distributed manufacturing organizations, *in* 'Emerging Solutions for Future Manufacturing Systems', Springer, pp. 81–90.

Maturana, F. P., Tichý, P., Staron, R. J., Discenzo, F. M., Hall, K. et al. (2003), A highly distributed intelligent multi-agent architecture for industrial automation, *in* 'Multi-Agent Systems and Applications III', Springer, pp. 522–532.

Mohd, A., Ortjohann, E., Schmelter, A., Hamsic, N. and Morton, D. (2008), Challenges in integrating distributed energy storage systems into future smart grid, *in* 'Proceedings of IEEE International Symposium on Industrial Electronics', IEEE, pp. 1627–1632.

Mönch, L., Stehli, M. and Zimmermann, J. (2003), Fabmas: An agent-based system for production control of semiconductor manufacturing processes, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer, pp. 258–267.

Odell, J. J., Parunak, H. V. D. and Bauer, B. (2001), Representing agent interaction protocols in uml, *in* 'Agent-oriented software engineering', Springer, pp. 121–140.

Pearl, J. (1984), *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-05594-5.

Pěchouček, M. and Mařík, V. (2008), 'Industrial deployment of multi-agent technologies: review and selected case studies', *Autonomous Agents and Multi-Agent Systems* **17**(3), 397–431.

Pěchouček, M., Rehák, M., Charvát, P., Vlček, T. and Kolář, M. (2007), 'Agent-based approach to mass-oriented production planning: Case study', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **37**(3), 386–395.

Richter, K., Jersak, M. and Ernst, R. (2003), 'A formal approach to mpsoc performance verification', *Computer* **36**(4), 60–67. ID: 1. ISBN 0018-9162.

Rzevski, G., Skobelev, P. and Andreev, V. (2007), Magentatoolkit: A set of multi-agent tools for developing adaptive real-time applications, *in* 'Holonic and Multi-Agent Systems for Manufacturing', Springer, pp. 303–313.

Sallez, Y., Berger, T. and Trentesaux, D. (2009), 'A stigmergic approach for dynamic routing of active products in fms', *Computers in Industry* **60**(3), 204–216.

Schliecker, S., Rox, J., Negrean, M., Richter, K., Jersak, M. and Ernst, R. (2009), 'System level performance analysis for real-time automotive multicore and network architectures', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **28**(7), 979–992. ID: 1. ISBN 0278-0070.

Smith, R. G. (1980), 'The contract net protocol: High-level communication and control in a distributed problem solver', *IEEE Transactions on Computers* **C-29**(12), 1104–1113. ID: 1. ISBN 0018-9340.

Strasser, T., Andrén, F., Merdan, M. and Prostejovsky, A. (2013), Review of trends and challenges in smart grids: An automation point of view, *in* 'Industrial Applications of Holonic and Multi-Agent Systems', Springer, pp. 1–12.

Sycara, K., Pannu, A., Williamson, M., Zeng, D. and Decker, K. (1996), 'Distributed intelligent agents', *IEEE Intelligent Systems* **11**(6), 36–46.

Sycara, K., Widoff, S., Klusch, M. and Lu, J. (2002), 'Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace', *Autonomous agents and multi-agent systems* **5**(2), 173–203.

Systems, R. (2011), 'http://www. agentbuilder. com'.
URL: *http://www.agentbuilder.com*

Thiele, L., Chakraborty, S. and Naedele, M. (2000), Real-time calculus for scheduling hard real-time systems, *in* 'Proceedings of the IEEE International Symposium on Circuits and Systems', Vol. 4, pp. 101–104 vol.4. ID: 1.

Tichý, P., Šlechta, P., Staron, R. J., Maturana, F. P. and Hall, K. H. (2006), 'Multiagent technology for fault tolerance and flexible control', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **36**(5), 700–704.

Tichý, P., P. and Staron, R. J. (2010), Multi-agent technology for fault tolerant and flexible control, *in* 'Innovations in Multi-Agent Systems and Applications-1', Springer, pp. 223–246.

Tichý, P., Šlechta, P., Maturana, F. and Balasubramanian, S. (2002), Industrial mas for planning and control, *in* 'Multi-agent systems and applications II', Springer, pp. 280–295.

Tichý, P. (2003), Social knowledge in multi-agent systems, PhD thesis, Czech Technical University in Prague.

Valckenaers, P., Kollingbaum, M. and Van Brussel, H. (2004), 'Multi-agent coordination and control using stigmergy', *Computers in Industry* **53**(1), 75–96.

Vrba, P., Kadera, P., Myslik, M. and Klima, M. (2014), Jboss esb sniffer, *in* 'Proceedings of the 23rd IEEE International Symposium on Industrial Electronics', IEEE, pp. 1724–1729.

Wagner, I. A., Lindenbaum, M. and Bruckstein, A. M. (1999), 'Distributed covering by ant-robots using evaporating traces', *IEEE Transactions on Robotics and Automation* **15**(5), 918–933.

Weiss, G. (2013), *Multiagent Systems*, MIT Press.

Wong, H. C. and Sycara, K. (1999), Adding security and trust to multi-agent systems, *in* 'In Proceedings of Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies', pp. 149–161.

Wooldridge, M., Fisher, M., Huget, M.-P. and Parsons, S. (2002), Model checking multi-agent systems with mable, *in* 'Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2', ACM, pp. 952–959.

Wooldridge, M. and Jennings, N. (1999), 'Software engineering with agents: pitfalls and pratfalls', *IEEE Journal Internet Computing* **3**(3), 20–27.

World Energy Outlook 2013 (n.d.).
URL: *http://www.worldenergyoutlook.org/*

Zoitl, A., Strasser, T. and Ebenhofer, G. (2013), Developing modular reusable IEC 61499 control applications with 4DIAC, *in* 'Industrial Informatics (INDIN), 2013 11th IEEE International Conference on', pp. 358–363.

List of Author's Publications

Publications in Journals with Impact Factor

Vrba, P. (16%), Tichý, P. (14%), Mařík, V. (14%), Hall, K. H. (14%), Staron, R. J. (14%), Maturana, F. P. (14%) and Kadera, P. (14%) (2011), 'Rockwell automation's holonic and multiagent control systems compendium', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* **41**(1), pp. 14–30.

Tichý, P. (20%), Kadera, P. (20%), Staron, R. J. (20%), Vrba, P. (20%)and Mařík, V. (20%) (2012), 'Multi-agent system design and integration via agent development environment', *Engineering Applications of Artificial Intelligence* **25**(4), pp. 846–852.

Publications in Reviewed Journals

Obitko, M. (20%), Vrba, P. (20%), Mařík, V. (20%), Radakovic, M. (20%) and Kadera, P. (20%) (2010), 'Applications of semantics in agent-based manufacturing system.', *Informatica* **34**(3), pp. 315–330.

Kadera, P. (70%), Novák, P. (20%), Jirkovskỳ, V. (5%) and Vrba, P. (5%) (2014), 'Performance models preventing multi-agent systems from overloading computational resources', *Automation, Control and Intelligent Systems* **6**, pp. 96–102.

Publications excerpted in ISI

Kadera, P. (90%) and Tichý, P. (10%) (2009), Plan, commit, execute protocol in multi-agent systems, *in* proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems.

Obitko, M. (25%), Vrba, P. (25%), Kadera, P. (25%) and Jirkovskỳ, V. (25%) (2011), Visualization of ontologies in multi-agent industrial systems, *in* 'proceedings of the 16th IEEE Conference on Emerging Technologies and Factory Automation'.

Jirkovskỳ, V. (34%), Kadera, P. (33%) and Vrba, P. (33%) (2012), Semantics for self-configurable distributed diagnostics, *in* 'proceedings of the 17th IEEE Conference on Emerging Technologies and Factory Automation'.

Kadera, P. (34%), Vrba, P. (33%) and Jirkovskỳ, V. (33%) (2012), Deviation detection in distributed control systems by means of statistical methods, *in* 'proceedings of the 38th Annual Conference of IEEE Industrial Electronics Society'.

Vrba, P. (25%), Kadera, P. (25%), Myslík, M. (25%) and Klíma, M. (25%) (2014), JBoss ESB Sniffer Message Flow Visualization for Enterprise Service Bus, *in* 'proceedings of the IEEE International Symposium on Industrial Electronics'.

Other Publications

Kadera, P. (70%) and Tichý, P. (30%) (2009), Chilled water system control, simulation, and visualization using java multi-agent system, *in* 'proceedings of the international conference on Information Control Problems in Manufacturing'.

Tichý, P. (35%), Kadera, P. (20%), Staron, R. J. (20%), Vrba, P. (15%) and Mařík, V. (15%) (2010), Agent development environment for multi-agent system design and integration, *in* 'proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems'.

Vrba, P. (20%), Kadera, P. (20%), Jirkovskỳ, V. (20%), Obitko, M. (20%) and Mařík, V. (20%) (2011), New trends of visualization in smart production con-

trol systems, *in* 'proceedings of the 5th International Conference on Industrial Applications of Holonic and Multi-Agent Systems'.

Vrba, P. (34%), Mařík, V. (33%) and Kadera, P. (33%) (2012), Mast: From a toy to real-life manufacturing control, *in* 'proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing'.

Lepuschitz, W. (25%), Jirkovskỳ, V. (25%), Kadera, P. (25%) and Vrba, P. (25%) (2012), A multi-layer approach for failure detection in a manufacturing system based on automation agents, *in* 'proceedings of the 9th International Conference on Information Technology: New Generations'.

Jirkovskỳ, V. (25%), Kadera, P. (25%), Obitko, M. (25%) and Vrba, P. (25%) (2012), Diagnostics of distributed intelligent control systems: Reasoning using ontologies and hidden markov models, *in* 'proceedings of 14th IFAC Symposium on Information Control Problems in Manufacturing'.

Novák, P. (70%), Kadera, P. (15%), Vrba, P. (10%) and Šindelář, R. (5%) (2013), Architecture of a multi-agent system for scada level in smart distributed environments, *in* 'proceedings of the 18th IEEE Conference on Emerging Technologies and Factory Automation.

Jirkovskỳ, V. (25%), Obitko, M. (25%), Novák, P. (25%) and Kadera, P. (25%) (2014), Big data analysis for sensor time-series in automation, *in* 'proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation'.

Novák, P. (20%), Kadera, P. (20%), Vrba, P. (20%) and Biffl, S. (20%) (2014), Engineering of Coupled Simulation Models for Mechatronic Systems, *in* 'proceedings of the International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing.

Patents

Vrba, P. (17%), Hall, K.H. (17%), Kadera, P. (17%), Mařík, V. (17%), Obitko, M. (17%), Radakovič, M. (17%) (2012), 'Ontology-based system and method for industrial control', U.S. Patent No. 8,145,333. Washington, DC: U.S. Patent and Trademark Office.

Obitko, M. (25%), Vrba, P. (25%), Kadera, P. (25%), Jirkovsky, V. (25%) (2013), 'System and method for implementing a user interface to a multi-agent distributed control system'U.S. Patent No. 20,130,055,115. Washington, DC: U.S. Patent and Trademark Office.