Simulation and Implementation of Logistics Systems based on Agent Technology

Alexander Pokahr, Lars Braubach, Jan Sudeikat
Wolfgang Renz, Winfried Lamersdorf

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
{pokahr | braubach | lamersdorf}@informatik.uni-hamburg.de

Multimedia Systems Laboratory,
Department of Information and Electrical Engineering
Hamburg University of Applied Sciences,
{wr | sudeikat}@informatik.haw-hamburg.de

Abstract
The logistics domain offers challenging problems, which are often characterized by specific properties that render them hard to solve. The development of IT-systems for the logistics domain has to adequately address these characteristics in order to provide acceptable solutions. One key problem of traditional software development approaches is that mainstream software engineering paradigms such as object orientation do not offer sufficiently rich abstractions for complex logistics problems. In order to address this drawback in this paper an agent-based perspective for logistics problems is advocated. On the one hand it is demonstrated what new concepts agent and multi-agent systems provide and how these concepts can contribute to the description of logistics problems and on the other hand a new development approach for multi-agent systems is presented. This approach is specifically designed for domains in which a simulation of the application scenario is beneficial in beforehand of the application implementation. It allows a seamless transition between simulation and operation models of a multi-agent system. This means that an agent-based simulation model of the application domain can be analyzed, optimised, and tested in a first stage of the development. Thereafter, it can be directly used as starting point for the multi-agent implementation and does not require the business logic code to be changed. The approach is tool supported by the Jadex agent framework and its usefulness will be further explained in the context of example applications from the health care and transportation logistics domains.

Introduction

According to Davidson and Kowalczyk (1997, p.1): “Logistics is the process of managing the flow and storage of materials and information across the entire organization with the aim to provide the best customer service in the shortest time at the lowest cost.” This definition highlights that logistics has to provide solutions for resource planning and transport in the broadest sense. Examples of logistic problems include fleet management, order management, route planning, scheduling and cargo management. Most of these problems have in common that they are very difficult to solve (e.g. NP-hard) and therefore no fast algorithms exist, which can deliver optimal solutions in complex real-world situations. In the following some of the most important characteristics will be discussed in more detail (cf. Davidson and Kowalczyk 1997, Perugini et al. 2003):

High complexity: Logistic problems often consist of numerous components, which exhibit complex behavior and are interconnected in various ways. Solving logistic problems requires understanding these dynamics and providing means for managing the complexity.

Large decision space: Typically, the solution strategies for solving logistics problems can have a multitude of options at their disposal and many different decision variables need to be taken into account. Furthermore, these options are often difficult to evaluate and prioritize.

Utilization of real-time data: Today’s logistic departments are confronted with a fast changing world, in which many unanticipated situations can arise. In order to stay competitive data has to be collected and processed in realtime.

Uncertainty: It is an inherent property of many business environments that only partial or incomplete knowledge is available and decisions have to be made on basis of these imperfect knowledge. In addition, unexpected events might occur (e.g., emergencies, machine breakdowns) that could have severe influence on ongoing activities and have to be handled.

Numerous decision makers: Logic processes often involve multiple decision makers, who are involved in processes with different responsibilities. In this respect it can be e.g. distinguished between the different departments a decision maker is responsible for, e.g. marketing, managerial or operational.

Highly constrained: There are a lot of constraints that need to be fulfilled in order to plan and carry out logistic activities. These constraints e.g. include physical constraints such as available storage and machine capacities as well as business objectives such as production efficiency or customer satisfaction.

Distributed domains: Typically, logistics needs to solve problems that involve complex settings consisting of physically dispersed entities and/or data. Furthermore, involved actors often have individual objectives such as keeping their rest times, which have to be coordinated with business objectives such as achieving on time delivery of goods.

Even though logistic settings might not expose all the aforementioned properties at once, logistics solutions and software have to embrace the existing characteristics and handle them in an intelligent way. Considering the difficulty of logistics problems the proposed software solutions should also fulfill some general requirements. Some key factors are explained in the following:

Understandability: Despite the complexity of the logistics problems the provided software should try to mask this complexity as far as possible and provide not overly complex usage interfaces. Furthermore, it is often beneficial that a software system makes transparent what it does and allows users to understand how the applied solution strategy works. If decision support systems are considered this may lead to an increased acceptance of the software (Graudina and Grundspenkis 2005, Himoff et al. 2005).

Seamless software / operator interaction: In many logistics scenarios manual operators work hand in hand with software tools supporting them. As software cannot always be aware of all currently relevant knowledge and additionally the operator might have long experience with certain tasks, the software should in those scenarios play the role of a subordinated assistant. This means that the software should make autonomous decisions only if explicitly authorized by the operator. Otherwise the software should make recommendations leaving the final decisions about its execution by the human operator (Dorer and Calisti 2005).

Robust system behavior: Logistics software systems should exhibit robust system behavior also in unanticipated situations especially due to the great amount of uncertainty in the domain. Concretely the software should be able to cope with unexpected situations and produce acceptable results also in those situations.

Existing logistics software systems try to address some of these issues but many existing logistics problems are far from optimally solved (Davidson and Kowalczyk 1997). Besides the inherent difficulties of the logistics problems one key problem in the development of logistics software is that mainstream software paradigms do not offer suitable concepts for capturing the complex entities of many logistics scenarios. E.g. object oriented concepts are suitable for passive business objects but fail to capture the characteristics of autonomous actors.

This paper will contribute to the improvement of software development for logistics problems at two different layers. In the next section, it will be argued at a generic level and by using illustrative examples that the multi-agent paradigm can provide many conceptual abstractions, which fit very well to the already introduced logistics domain characteristics. Additionally, it will be shown that the multi-agent properties support achieving the logistics software requirements more easily.

In section three, a new development approach for logistics applications will be presented. The approach is suitable for scenarios in which a simulation of the scenario in beneficial in beforehand to the actual system implementation. The key idea here is to provide a seamless transition between scenario simulation and implementation in the sense that main software components need to be developed only once.

The application of the proposed development approach will be exemplified with the help of different logistics problems. For these problems it is shown how agent-based solutions can be derived and how the approach facilitates their development. Section four concludes the paper with a summary and an outlook to prospective future work.

How logistics can benefit from agent-based solutions

In this section it will be discussed which properties make agent-based approaches attractive for handling typical logistic problems. In general it can be stated that multi-agent systems provide natural key metaphors which facilitate a high-level and understandable description of the problem domain and the aspired solution. In the following the agent-based characteristics will be discussed on two different levels. First the properties of individual agents and then of multi-agent systems will be presented. For further illustration purposes, at the end of this section some real-world applications using multi-agent technology will be sketched.

Agent characteristics

The agent characteristics discussed in the following rely on an agent definition of Wooldridge called “strong notion of agency” (Wooldridge 2002). In general an agent is seen as a situated entity, which interacts with its environment through sensors and actuators (Luck et al. 2005).

Autonomy: Autonomy describes the property of an agent to act on his own. On a conceptual level this means that an agent has control not only over its state but also over its actions, i.e. it can decide on the basis of its own perception of the world what to do next.

This autonomy reflects the local decision power of the numerous decision makers within typical logistic settings. In a software system these different responsibilities can adequately be expressed using the agent metaphor. Each decision maker can be represented by an agent, which has the purpose to act on behalf of its principal. Despite the possibility of autonomous action the degree of autonomy is controllable and should be adapted to the specifics of the concrete application domain and the responsibilities of the agent in the system.

Reactivity: An agent should be capable of performing fast reactions to changes that might occur. Typically, an agent perceives events from its environment and should be enabled to react to them is a timely fashion. This possibly means that the agent has to shift its attention from the ongoing activity it performs towards the newly occurred event and if it needs immediate processing the corresponding activity should be executed with priority. In order to allow fast reactions an agent architecture has to deal with those issues and provide appropriate means for specifying reactive behavior.

Regarding the logistics domain reactivity is extremely important for coping with uncertainties. One important aspect of these uncertainties are unexpected occurrences such as breakdown of machines or delays in delivery of goods, which need to be considered by the logistics system as soon as possible. If the environment is monitored and occurrences are propagated to agents with reactive capabilities a timely handling can be enforced.

Proactivity: A proactive agent has goals that it tries to achieve. Hence, the behavior of a proactive agent is driven by internal motivations and steered not only by reactions to environmental percepts. This allows a proactive agent to act also strategically and plan its actions in a long-term manner. In order to combine such proactive behavior with reactive capabilities mentioned beforehand, in the field of multi-agent systems so called hybrid agent architectures such as PRS (procedural reasoning system) (Rao and Georgeff 1995) have been developed. These kinds of architectures ensure that reactive and proactive influences are balanced within the agent and especially that proactive behavior does not prevent fast reactions.

Concerning the logistics domain proactivity allows to specify the individual objectives of the different participating entities. This means that e.g. in a transportation scenario the vehicles as well as the hubs could be represented as agents, which are seeking to fulfill their aims. In this respect, one important vehicle objective could be to perform transportations with high utilization.

Social abilities: An agent is equipped with communication mechanisms allowing it to asynchronously send/receive messages to/from other agents. Agent communication is typically speech-act based (Searle 1969), i.e. agents do not only transmit a message content but also their intention towards this content. These intentions are described with performative verbs such as “request” for asking another agent to perform an action or “inform” for sending knowledge to another agent. In order to build up more complex communication forms than request-reply, interaction protocols have been devised, which specify the allowed message sequences in beforehand. Interaction protocols have been developed and standardized by FIPA[1] for e.g. English and Dutch auctions and contract-net.

The social abilities combined with the decision freedom of agents allow them to communicate with others whenever they see need for it. In a transport setting, truck agents could e.g. proactively communicate to other trucks nearby that the used highway is jammed. This new knowledge gives the other truck agents the chance to replan their current route and possibly avoid the jam. Further advantages of coordination through communication will be discussed with regard to multi-agent characteristics, later in this section.

Mentalistic notions: Mentalistic notions are descriptions of human mental attitudes such as *beliefs* or *goals*. These notions have been used for explaining human behavior in the context of folk psychology (Christensen and Turner 1993). Follow-

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

ing the ideas of the *intentional stance* (Dennett 1971) using mentalistic notions facilitates the understanding of complex artifacts by ascribing mental attitudes to them, i.e. a truck drives down Church Street because it has the aim to get to the main station, which is located in that direction. The mentalistic framework allows for taking up an abstract point of view and helps distinguishing the underlying motivations from its concrete actions. Typically these motivations are described using top-level goals, which are further refined into a hierarchy of plans and subgoals.

A well-known mentalistic framework is the philosophical BDI-model, which originally aims to explain human behavior with beliefs, desires and intentions (Bratman 1987). On basis of this model, Rao and Georgeff (1995) have proposed the PRS architecture, which refines the BDI ideas in a software technical sense. Using the PRS architecture an agent is defined using beliefs for representing its individual world view, desires for stating its current motivations and intentions for expressing the courses of actions it already has committed itself to.

In the context of logistic scenarios mentalistic agent descriptions can help managing the complexity of behavior descriptions. As an example one can consider the scenario in which one top-level goal of a truck agent is to bring a packet to the main station. Depending on the delivery context different routes may be applicable, but this does not to be considered on the highest abstraction level. Instead, lower level plans can handle the route planning according to the delivery context and e.g. prefer freeways if cost effectiveness is important or also consider routes liable to charges for time-critical deliveries.

Multi-agent Characteristics

Agent-based approaches to distributed systems development exploit the agent design metaphor and conceive applications as sets of interacting agents that are integrated in a common environment. In these *multi-agent systems* (MAS), the application functionality results from individual agent coaction and interaction. It has been argued that this development viewpoint extends established modeling practices and leads to software designs that model today's application domains in more expressive abstractions (Jennings, 2001). Particularly, the development of distributed software systems benefits from abstractions that model system components as autonomous actors, because it often reflects existing structures. Understanding software systems as sets of autonomous actors poses novel challenges on software development processes, but also provides a common toolset to coordinate software elements and exploit synergies between otherwise statically connected system elements. In the following, the main characteristics of multi-agent systems will be explained and their usefulness for logistic applications will be sketched.

Decentralized organization: MAS are inherently distributed software systems, enabling agents to transparently communicate regardless of their location. This transparency makes MAS subject to inherently decentral organizations, where the

physical location is abstracted and systems operate in a decentralized network of distributed application components. The inherently decentralized nature of MAS-based applications is a major building block for the MAS characteristics discussed below and contributes to fault tolerance and scalability, since local failures only have minor effects on the software application itself and new components/agents can be connected/removed at run-time. This decentralized infrastructure is particularly attractive for open environments where agents and hosts enter and leave the system at run-time. Concerning Logistics applications, this feature, e.g, facilitates the addition and removal of automatic guided vehicles or manufacturing machines (cf. section 2.3).

Environment abstraction: While the physical environment is transparently hidden from agent developers, agents themselves are expected to inhabit an application dependent environment. The MAS environment can either be implicitly perceivable (only message passing agents) or explicit represented (situated MAS). The environment provides a first class abstraction to interact with MAS external components and software frameworks are available that support modeling environment properties and agent interactions (Viroli et al., 2007). In case of situated MAS, the agents can interact indirectly by concurrently modifying their shared environment. These indirect interactions are particularly useful for exploiting self-organizing phenomena (Serugendo et al. 2006), i.e. to achieve large-scale coordination solely by local interactions.

Since logistics is often intrinsically related to the spatial movement of vehicles, it is particularly attractive for developers to represent the system context explicitly. Developers can choose from established environment abstractions (Gouaich and Michel, 2005) that are supported by software frameworks and allows to represent the dynamics within the application context. E.g. concerning logistics the availability of routes will be influenced by external factors like traffic jams. These application internal events are to be represented in environment models allowing the agent population to perceive and adjust.

Self-organizing Behavior: The inherent support for decentralized agent organizations and explicit environment models facilitates the utilization of self-organizing phenomena, as known from biological, physical and social systems (Sudeikat and Renz, 2008b). In these systems, agent societies coordinate themselves implicitly by local interactions that are ignorant of the system wide implications. By providing agents with sets of behaviors and means to select among these, based on local perceptions, the adaptivity of system wide properties can be enforced. A prominent example is the foraging behavior of ant colonies. Ant populations manage to form shortest paths between their nest and sources of resources by exploiting stigmergy (Brueckner and Czap, 2006). Ants that are traveling from a resource to the nest modify their environment by releasing chemical substances (so-called pheromones) that attract the attention of other ants. These follow the scent of evaporating pheromones and are recruited to exploit resources repeatedly. Since pheromones diffuse

and evaporate, the trail is enforced the most, which allows the quickest passages of individuals.

Self-organized adaptation is in principle not related to spatial environments but can be applied to task allocations and role adopting behaviors as well. The utilization of decentralized coordination mechanisms as means to purposefully engineering self-organizing dynamics is an active topic of research (Sudeikat and Renz, 2008a, b, c). Particularly, for logistic settings it is interesting to allocate resources and tasks in adaptive ways. E.g. transportation routes can be subject to adaptation as to react on vehicles unavailability's (e.g. repairs) and availability of new transporters to address high workloads as well as to allocate trucks to specific routes. In manufacturing line control, working examples are available that show the benefits of the self-organizing adaptation of the routes of items in production lines (cf. section 2.3).

Coordination mechanisms: A key concern in multi-agent systems is the coordination of agent behavior, i.e. managing the dependencies between distributed activities. A cornucopia of coordination techniques and strategies are available, each of which represents a well-understood pattern of local activities and interaction activities that allows steering the behavior of individual agents in a desired way to achieve overall design objectives. Coordination mechanisms can broadly be categorized into *cooperative* and *competitive* approaches.

In cooperative approaches, agents work together in a benevolent way in order to achieve one or more shared goals. Therefore, the required tasks and activities are allocated to individual agents in a way to increase the efficiency and effectiveness of the overall system and to dynamically reallocate tasks in the presence of unexpected events (e.g. traffic jam) or partial system failures (downtime of a machine).

In competitive approaches each agent has individual goals that might be in conflict with goals of other agents. Usually, market-based mechanisms such as auctions or negotiations are used for coordinating agents in competitive settings. These approaches also allow to represent conflicting goals inherent in the problem domain. E.g. in transportation logistics one usually wants to maximize utilization of trucks (i.e. avoid tours of only partially loaded trucks), but also wants to minimize packet delivery time. By representing individual resources (e.g. trucks and packets) as agents that negotiate with each other, appropriate trade-offs between conflicting goals can be established using suitable coordination strategies that move solutions in the direction of a global optimum.

Organizational structures: The multi-agent system metaphor also naturally provides an organizational perspective. This means that organizational and social concepts can be exploited for modeling software systems in analogy to human organizations. In this respect three different dimensions can be distinguished (Hübner et al. 2002). The *structural dimension* relates to the setup of an organization and provides concepts for a meaningful (often hierarchical) decomposition into smaller units and for the description of their relationships. Typical notions within this area

are groups, roles and positions as proposed within the AGR (agent-group-role) model of Ferber and Gutknecht (1998). The *behavioral dimension* deals with the problem, how different agents can work together in a coordinated way to achieve an overall objective. In this dimension typically the teamwork of agents is considered and aspects such as team formation, operation and termination play an important role. One well-known approach here is the joint intention framework of Cohen and Levesque (1990), which introduces mentalistic concepts such as joint persistent goals on the group layer and ensures that the coordination between agents working on a shared persistent goal is automatically performed. Finally, the *deontic dimension* is concerned with normative aspects of agent communities. The key idea is that social norms and obligations can be established in a multi-agent system for monitoring and enforcing benevolent behaviour of the inhabiting agents. The observation and enforcement is typically performed by an electronic institution, which also provides the area of validity for the norms and obligations respectively.

In logistic scenarios organizational ideas can e.g. be used for naturally mapping real-word settings. In military transport logistics the existing hierarchical troop structure consisting of groups, subgroups and individual vehicles can be directly used in the software design. Also, in manufacturing logistics different production cells and their contained machines can be modeled as groups and agents. This allows viewing the design at different levels and different aspects can be emphasized if the top-level or lower-level layers are under consideration.

Agent-based Logistics Applications

The agent development paradigm plays out its strengths to handle turbulent environments, where the activities of individual software components are subject to failures. We conclude this section by exemplifying successful applications of agents in logistics applications to clarify the benefits and challenges of this modeling and development approach. Here, we outline a selection of agent-based designs that are related to commercial applications. First, we outline two examples that support the scheduling decisions by domain experts, and then we denote two approaches that directly control logistic processes. These systems exemplify the usage of agent and MAS characteristics to model logistic applications, where different environment models, coordination mechanisms and organizational models are applied to enable MAS adaptivity.

Regarding scheduling applications one major challenge is that they must timely respond to unforeseen events that enforce derivations from previously adopted schedules. In dynamic transportation environments these events may comprise traffic jams, transporter breakdowns, or accidents. Due to these external turbulences, logistics companies need to respond quickly and wisely. These turbulences and the growing complexity of large scale transportation networks challenge conventional transportation optimization approaches (Dorer and Calisti, 2005).

The commercial *MAGENTA* agent platform and its *I-Scheduler*[2] application have been applied for the management of tanker fleets (Himoff, Skobelev and Wooldridge, 2005). This application particularly addresses the oil transportation market, where the frequent and unexpected fluctuations of transportation costs need to be considered. Agent instances, which are equipped with their individual constraints (parameterized by domain experts), inhabit a so-called *Virtual Marketplace* and negotiate their individual transportation routes. An external influence, e.g. the availability of new shipping tasks, triggers the creation of MAS internal events. Agents respond to these events by negotiating alternative schedules, where agents with free transport capacities propose their availability and agents are free to swap cargo assignments when necessary. This allows for an effective search of the space of possible schedules, where agents ensure that their transportation constraints are met.

This radical approach, where every transport is modeled by an individual agent is opposed by the approach that has been adopted by Whitestein Technologies[3], where the commercial Living Systems® Adaptive Transportation Networks (LS/ATN) platform guides the dispatchment of cargo transporters (Dorer and Calisti, 2005). This platform particularly addresses large scale transportation networks that challenge traditional planning techniques. Therefore, the transportation environment is separated into so-called *dispatching regions*. Each region is handled by dedicated agents that manage the locally available trucks. Arriving orders are tentatively allocated and locally optimized. When pickup or delivery locations involve different regions, the region's representatives are informed and may handle the order themselves. The system distinguishes between parameters that have to be met and constraints that may potentially be violated, within tolerance ranges.

Besides these systems that support human decision makers, agents have also been applied to directly manage transports. E.g. *DaimlerChrysler* (Bussmann and Schild, 2000) addressed the control of manufacturing processes. Typical manufacturing lines connect machines in series and perform sequences of operations on individual items, finally leading to the addressed product. These lines suffer from their inherent inflexibility. When machines fail, time consuming reconfigurations are required to enable the required sequence of operations. Bussmann and Schild (2000) have proposed to equip machines with *shifting tables* that enable items to travel freely between machines. Production items, the tables and the machines are represented by agents and items negotiate the sequence of machine interactions, e.g. to bypass failing or fully loaded machines.

Weyns et al. (2005) examined the decentralized control of automated logistics services for warehouses and manufacturing. Automatic guided vehicles are utilized to transport loads within specified environments. These vehicles are typically con-

[2] http://www.magenta-technology.com/en/solutionsandservices/smartresource/

[3] http://www.whitestein.com/autonomic-business-solutions/logistics-supply-chain-management

trolled by a centralized server that ensures timely responses to transportation requests, collision avoidance and the absence of deadlocks (i.e. the vehicles block each others ways). In order to meet the demand for scalability and the adaptive scheduling of transportations, vehicles have been enabled to coordinate themselves via a virtual environment (Weyns et al., 2005), therefore allowing to add and remove vehicles at any time.

The presented agent-oriented designs partition the application domain in autonomous actors (cf. section 2.1) and place them in an environment and organizational context (cf. section 2.2). Each active participant (vehicle, storage facility, etc.) is equipped with its own constraints that need to be met. Therefore, typically centralized planning problems are transferred into a set of decentralized agent interactions that allow for concurrent, distributed processing. Explicitly decentralized models of the application domain are particularly beneficial for large scale logistics problems as they facilitate scalability and increases robustness by the absence of single point-of-failures. Moreover, events can be handled locally by limiting the propagation of changes as for every event only small groups of individual agents need to (re-)coordinate their activities.

An integrated development approach combining simulation and operation

The most critical property of logistics systems is the quality of the proposed solutions, commonly measured as cost or cost reduction. Due to the complex nature of most logistics scenarios (i.e. regarding the number of involved entities and interdependencies), they are in general not open to purely analytical approaches. Therefore, simulation plays an essential role in the area of logistics applications. On the one hand, computational models are a helpful tool for analysts to gain a thorough understanding of the problem domain and to identify areas for improvement of structures and/or processes. On the other hand, simulation is important during the development of IT systems that aim at supporting or automating activities in the logistics planning process. Upfront simulation experiments of the system under development provide numerous benefits:

– The employed coordination algorithms can be tested and validated against artificial as well as real data sets.
– Using real data sets, the coordination algorithms can be benchmarked against the current status-quo strategy.
– In simulation experiments, alternative algorithms can be investigated and compared and parameters can be fine-tuned.

In the following, a short overview of some available logistics simulation tools will be presented. Furthermore, it will be shown, how simulation tools in general currently fit into the process of developing software application for the logistics area. Some limitations of the current situation regarding the integration of simulation and application development will be highlighted. Finally, a new approach will be pre-

sented that aims to overcome these limitations by providing a unified simulation and application execution environment.

Simulation and Software Development

For analysis tasks, many specialized simulation tools are available that can be used, e.g., for process optimization, what-if analysis or demand estimation. For these purposes, a wide variety of domain specific (logistics) simulation tools exists, such as SimFlex, eM-Plant, or Citilabs Cube. Besides these, also generic simulation environments like SeSAm, RePast or AnyLogic allow to build proprietary simulation models for concrete, domain specific analysis purposes.[4]

Many of these tools expose a high level of maturity and have proven their usefulness for logistics applications. Nevertheless, neither domain specific simulation tools nor generic simulation environments are meant to be used for the deployment of productive software systems. They can be used for analysis purposes, only. In figure 1 this situation is depicted for the case of using a generic agent based simulation environment.. Separate tools and platforms are used for developing and analyzing simulation models and for building and deploying the final application (left: *generic simulation environment*, right: *generic execution platform*).

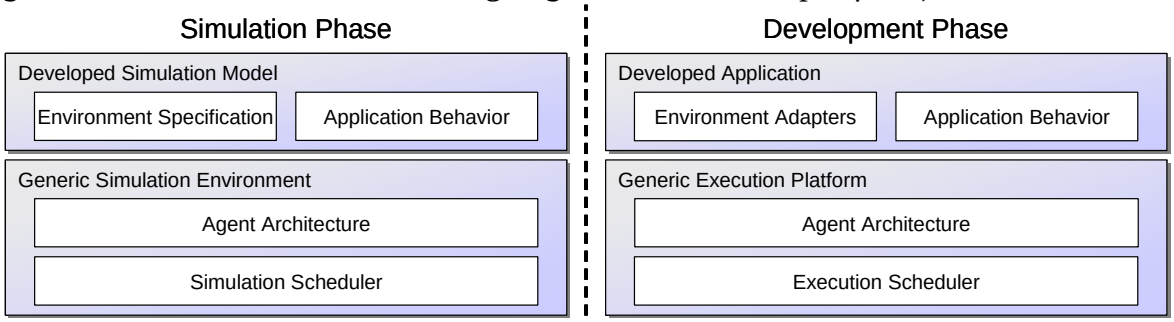

Figure 1: Separate simulation and development environments

The simulation environment is usually based on a *simulation scheduler*, which controls simulation runs based on event-driven or time-stepped simulation algorithms (Page and Kreutzer 2005). E.g. in event-driven simulation, relevant occurrences of the simulated world are kept in an event-list and are executed in order and skipping times where no events happen. This enables simulation tools to calculate days or weeks of simulated time using only minutes our hours of real time. On top of the scheduler, agent-based simulation environments provide a specialized *agent architecture* that usually closely corresponds to the simulation algorithm (e.g. in time-stepped simulation, agents may have tasks that are executed in each time step). The developer uses this specialized agent architecture to develop a simulation model, which is usually composed of the *application behavior* as well as a simulation

[4] See http://flextronics.com/en/SimFlex/, http://www.emplant.com/, http://www.citilabs.com/, http://www.simsesam.de/, http://repast.sourceforge.net/, http://www.xjtek.com/

model of the environment (*environment specification*) providing the external events, to which the application should react.

The basis of execution platforms is an *execution scheduler* that is responsible for executing agents concurrently on the available system infrastructure like e.g. Java EE application servers, which provide efficient mechanisms such as thread-pooling or load-distribution. *Agent architectures* on these platforms range from simple task-based agents to deliberative agents with advanced reasoning capabilities. To deploy an application on such an execution infrastructure, the developer has to provide the *application behavior* and additionally *environment adapters*, which provide the interfaces to external system components and the outside world.

The separation of simulation and execution environments leads to a number of consequences when software systems should be based on coordination strategies, which are analyzed and designed in upfront simulation experiments:

– The already simulated application behavior has to be reimplemented in the desired agent platform leading to a doubled development effort.
– The reimplementation needs to be validated again to check if it correctly resembles the simulated coordination behavior.
– The concepts available in the simulation and runtime environment might differ (e.g. using different agent architectures), so no one-to-one reimplementation might be possible, thereby requiring a completely new agent design.

In the remainder of this section, a new approach is presented that unifies simulation with application concepts and allows for an easy transition from a MAS simulation model to a real-world MAS application. This approach completely removes the aforementioned issues, because no reimplementation of the simulated strategy is necessary.

Unified Approach

The big picture of the unified approach is shown in figure 2. The *generic simulation / execution platform* provides a single *agent architecture*, which allows agents to be executed by a *simulation scheduler* as well as a real-time *execution scheduler*. This ensures that all developed *application behavior* can be used in the simulation as well as development phase. During design and implementation of the application behavior, the developer therefore does not have to consider simulation or deployment issues, as these are abstracted away by the execution environment.

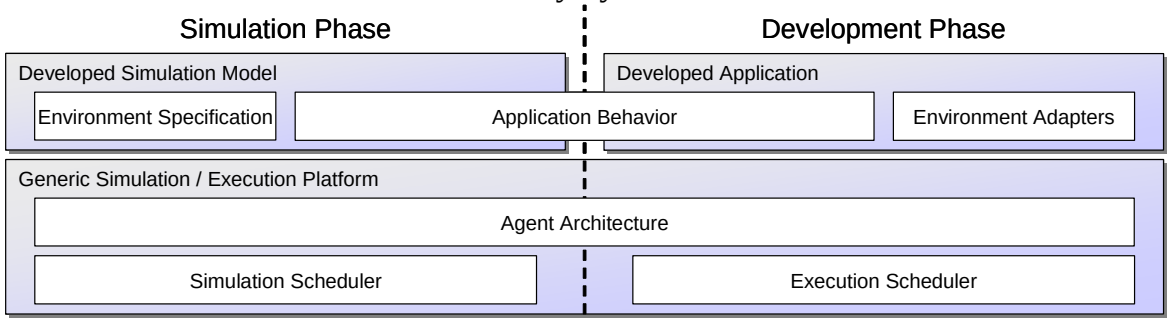

Figure 2: Unified simulation and execution approach

Therefore, the approach permits the complete reuse of the agent business logic and thereby largely reduces the application development effort, as only simulation specific components such as the artificial environment (*environment specification*) have to be replaced against their real-world counterparts (*environment adapters*).

Besides resolving the issues mentioned in the last section (necessity of reimplementation and revalidation, potential differences in concepts), the approach exhibits a number of additional advantages. These advantages are mainly due to the fact that all developed application components can be used in simulation settings as well as the productive execution.

– The approach facilitates an incremental software development process. The development can start from abstract specifications that are only useful for preliminary simulations and can be iteratively refined to more and more concrete application behavior until the application is finally ready for deployment.
– Simulation can be used as a testing tool. During the iterative application refinement, simulation runs can be performed for validating the newly created application components. Also initial acceptance tests can be performed on application prototypes that still run in simulation mode.
– Application components can be incrementally deployed. For developing the required adapters for interfacing with the real environment, developers can perform a one-by-one replacement of the virtual environment with real components. This allows testing each adapter in isolation before fully deploying the complete application.
– Simulation can be used for training and education. Once the application is ready to deploy, the prospective users have to be trained on how to operate the system. As the complete system (including GUI components) can still be run on the simulation platform, simulation scenarios can be devised in order to teach the users on how to react in different situations.

The approach requires the availability of a generic simulation and execution platform. Therefore, the different simulation and real-time execution modi have been implemented in the Jadex agent platform. Jadex is an open source agent framework[5] that allows building belief-desire-intention (BDI) agents using established technologies such as XML and Java. Agents implemented in Jadex can be deployed on a variety of execution environments, including standalone Java applications and the FIPA-compliant JADE platform.

Figure 3 shows the simulation control panel, which is part of the Jadex runtime tool suite. The *clock settings* panel (top left) allows observing and controlling execution settings. The platform supports event-driven, time-stepped and continuous time execution, where continuous execution is based on the system clock but allows specifying an offset and a dilation factor for accelerating or slowing down the

systems execution speed. It is also possible to change these settings while the system is running (e.g. changing the dilation factor or even switching from event-driven simulation to real-time execution). The *execution control* (bottom left) allows to pause and single-step the execution, which is especially useful for debugging either simulation models or the application implementation. Also for debugging purposes, the current list of *active timers* (i.e. to be performed time events, right) can be inspected.

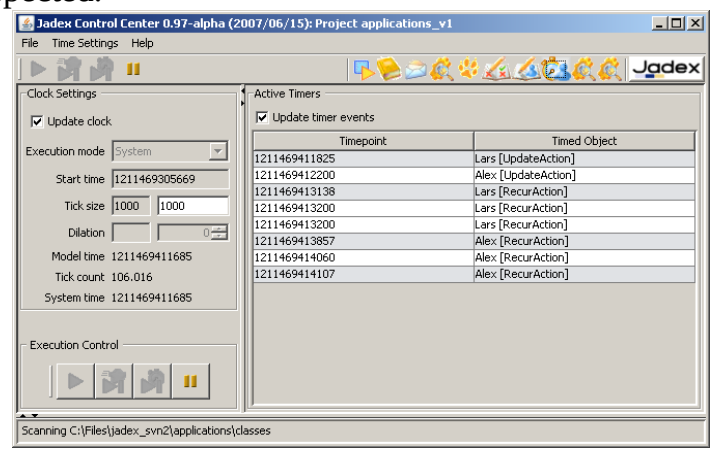

Figure 3: Screenshot of the Jadex simulation control panel

Application Scenarios

The combined simulation and operation facilities of the presented approach will be illustrated in the following with two example applications. The first one is a transportation logistics scenario, which makes use of simulation techniques. The second example deals with appointment scheduling in hospitals and exploits the combined simulation and operations.

Packet Delivery Scenario

Ruwinski und Timotin (2007) examined the applicability of the Jadex platform to simulate a logistics transportation scenario. For this purpose, a simplified application setting has been adopted where parcels are to be transported by heavy goods vehicles (HGV) between redistribution centers. The utilization of a general purpose agent platform facilitated the utilization of third party software packages, i.e. database integrations, and facilitated software engineering practices. A dedicated simulation setting has been conceived that enabled the parameterization, execution and analysis of simulation runs. This support allows domain experts to parameterize the simulation setting and agent population members and to observe (measure) system properties, e.g. package throughput at local and global scales.

[5] http://jadex.sourceforge.net

In the examined setting, a market-based coordination strategy has been applied, i.e. parcels were equipped with certain amounts of a virtual currency to bid for transportation by an HGV. HGVs travel between distribution centers and try to optimize their profit by serving different routes and negotiating transport cost with the individual packages. A round-based negotiation protocol has been revised and is concurrent execution is coordinated by the individual distribution centers. Simulation users can adjust the negotiation strategies of the parcels and HGVs. The simulations history is saved for later examination and is visualized via a dedicated graphical user interface.

Appointment Scheduling Scenario

The objective of the DFG-funded MedPAge (Medical Path Agents) project, conducted in cooperation by the Universities of Mannheim and Hamburg, was to realize cross-functional patient scheduling in hospitals (Paulussen et al. 2006). Patient scheduling is concerned with the optimal assignment of the scarce hospital resource to the patients, whereby patients want to minimize their stay time and resources intend to minimize their idle time. Patient scheduling is a complex task because the hospital environment is characterized by a high degree of uncertainty so that emergencies and complications are likely to occur.

In order to respect the distributed hospital setting and the local responsibilities of the different wards, patient scheduling is approached via decentralized agent coordination. Patient and resource agents use protocol-based negotiation strategies for determining the next appointments. In the first project phase promising negotiation strategies have been conceived and subsequently been implemented within a simulation model. This model was used to benchmark the different approaches against each other and especially with respect to the existing mechanism currently applied in hospitals. In the second phase a field study within the hospital was conducted. For this purpose a user interface was developed, which offers different views for patient admittance, wards and resources. The interface replaces the inputs that have been generated automatically from the simulation environment in simulation mode. Due to the unified simulation and execution approach, the agent behavior needed not to be changed when switching from simulation to real-time operation.

Conclusion

In this paper we argued for the applicability of agent technology to simulate and control logistic applications. The characteristics of agents and agent-based software systems have been outlined and related to the properties of logistics applications. Particularly the notions of *agents* as autonomous, pro-active actors as well as the agent *environment* provide suitable abstractions for logistics software systems. However, simulation systems and logistics control applications typically rely on

different kinds of agent execution platforms. This gap enforces manual effort to transfer once simulated system behavior into operational applications. Therefore, it has been discussed how to bridge this gap and a development approach has been proposed that allows the seamless transition between simulation and operation of a target system. The approach is tool supported by the general purpose Jadex agent development platform, which integrates a simulation infrastructure with an agent execution kernel.

Future work will address the systematic usage of the approach by conceiving a methodology for its usage. It remains to be examined how simulation and execution environments can be transferred from each other, e.g. to support iterative development in a systematic, methodological way. In addition, the utilization of self-organizing processes as MAS design elements and implementation components has been proposed (cf. section 2.2). Providing these mechanisms in the Jadex agent platform (as approached by Sudeikat and Renz, 2008c) promises a novel toolset to steer the adaptively of logistic applications.

References

Bratman, M. (1987). Intention, Plans, and Practical Reason, Harvard University Press.

Braubach, L./Pokahr, A. /Lamersdorf, W. (2006) Tools and Standards. Multiagent Engineering - Theory and Applications in Enterprises, Springer Series: International Handbooks on Information Systems. Springer-Verlag.

Brueckner, S./Czap, H. (2006) Organization, Self-Organization, Autonomy and Emergence: Status and Challenges International Transactions on Systems Science and Applications, 2, 1-9

Bussmann S./Schild K. (2000) Self-Organizing Manufacturing Control: An Industrial Application of Agent Technology. In Proc. of the 4th Int. Conf. on Multi-agent Systems (ICMAS'2000), Boston, MA, USA, 87-94

Cohen, P. R./Levesque, H. J. (1990). Intention is Choice with Commitment. In: Artificial Intelligence 42, S. 213–261

Christensen, S. M./Turner, D. R. (1993). Folk Psychology and the Philosophy of Mind. Lawrence Erlbaum Associates.

Davidson, I./Kowalczyk, R. (1997). Towards Better Approaches To Decision Support in Logistics Problems, Industrial Logistics, Feb 1997.

Dennett, D. (1971). Intentional Systems. In: Journal of Philosophy (1971), Nr. 68, p. 87–106.

Dorer, K./Calisti, M. (2005) An adaptive solution to dynamic transport optimization AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM, 45-51

Ferber, J./Gutknecht, O. (1998) Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems, Third International Conference on Multi-Agent Systems, Paris, IEEE.

Gouaich, A./Michel, F. (2005) Towards a Unified View of the Environment(s) within Multi-Agent Systems Informatica (Slovenia), 29, 423-432

Graudina, V./Grundspenkis, J. (2005). Technologies and multi-agent system architectures for transportation and logistics support: An overview. In: International Conference on Computer Systems and Technologies - CompSysTech, Varna, Bulgaria

Himoff, J. (2005). Magenta Logistics i-Scheduler. In Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems (The Netherlands, July 25 - 29, 2005). AAMAS '05. ACM, New York, NY, 159-160.

Himoff, J./Skobelev, P./Wooldridge, M. (2005) MAGENTA technology: multi-agent systems for industrial logistics AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM, 60-66

Hübner, J./Sichman, J./Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), Springer.

Jennings, N. R. (2001) Building complex, distributed systems: the case for an agent-based approach Comms. of the ACM, 44 (4), 35-41

Luck, M./ McBurney, P./Shehory, O./Willmott, S. (2005). Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing), AgentLink

Page, B./Kreutzer, W. (2005). The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java, Shaker Verlag

Paulussen, T.O./Zöller, A./Heinzl, A./Braubach, L./Pokahr, A./Lamersdorf, W. (2006). Agent-Based Patient Scheduling in Hospitals.Multiagent Engineering, Springer, Berlin, S. 255-275.

Perugini, D./Wark, S./Zschorn, A./Lambert, D./Sterling, L./Pearce, A. (2003). Agents in logistics planning - experiences with the coalition agents experimental project. Agents at Work, AAMAS Workshop, 2003

Rao, A./Georgeff, M. (1995). BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multiagent Systems, The MIT Press, 312-319.

Ruwinski, W./ Timotin, D. (2007) Entwicklung eins Multiagentsystem-basierten Frameworks zur Simulation logistischer Prozesse, Hamburg University of Applied Sciences

Searle, R. (1969). Speech Acts: an essay in the philosophy of language. Cambridge University Press

Serugendo, G. D. M./Gleizes, M. P./Karageorgos, A. (2006) Self-Organisation and Emergence in MAS: An Overview Informatica, 30, 45-54

Sudeikat, J./ Renz, W. (2008a) Toward Systemic MAS Development: Enforcing Decentralized Self-Organization by Composition and Refinement of Archetype Dynamics Proceedings of: Engineering Environment-Mediated Multiagent Systems, LNCS, Springer – to appear

Sudeikat J./Renz, W. (2008b) Building Complex Adaptive Systems: On Engineering Self-Organizing Multi-Agent Systems, Applications of Complex Adaptive Systems, IGI Global, 229-256 R. (1969).

Sudeikat, J./Renz, W. (2008c) On the Encapsulation and Reuse of Decentralized Coordination Mechanisms: A Layered Architecture and Design Implications
Communications of SIWN, ISSN 1757-4439, to appear

Viroli, M./Holvoet, T./Ricci, A./Schelfthout, K./Zambonelli, F. (2007) Infrastructures for the environment of multiagent systems, Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, 14, 49-60

Weyns, D./Schelfthout, K./Holvoet, T./Lefever, T. (2005). Decentralized control of E'GV transportation systems. AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM, 67-74.

Wooldridge, M. (2002). An Introduction to MultiAgent Systems, John Wiley and Sons