J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

A Distributed Multi-agent Architecture in Simulation Based Medical Training

Jun Hu and Loe Feijs

Department of Industrial Design
Eindhoven University of Technology
5600MB Eindhoven, The Netherlands
{j.hu, l.m.g.feijs}@tue.nl

Abstract. This paper addresses the issues of distributed interactions in a simulation based medial training environment, where a team of doctors, nurses and assistants are trained for handling difficult delivery situations using simulations. A scripting language is proposed, using a metaphor of play, with which the timing and mapping issues in distributed presentations are covered. A generic architecture for the systems is also presented, which covers the timing and mapping issues of conducting such a script in a medical training environment. The difference between playing a medical training scenario and playing a multimedia entertainment scenario is discussed, based on which the future research and development are proposed.

Keywords: Distribution, multi-agent, delivery simulator, medical training.

1 Introduction

In the United States as many as 98,000 people die each year from medical errors that occur in hospitals, according to a book with the title *To Err is Human*[1]. It was found that in America 75 percent of the failures in rescue were caused by either the diagnosis or the treatment being too late. Many of these deaths could have been avoided by improved the communication and coordination with in the medical teams.

In medical education, how to act in emergency situations is often trained on an individual basis. In practice, patients are however handled by a team from multiple disciplines, hence the training must target on the entire team. A British study shows that regular team training leads to 50 percent less brain damage caused by lack of oxygen during birth [2]. In the last three years, Máxima Medical Center in Veldhoven has been providing such multidisciplinary team training with the help of medical simulation. The team training is given by a gynecologist and an experienced midwife, taking place in a fully equipped delivery room that tries to reproduce the real lift situation (fig. 1). The training uses a patient simulator that is the most advanced up to date. The aim is to increase the skills of a multidisciplinary group of employees in the delivery room and especially to prevent inadequate communication in critical obstetric situations.

Z. Pan et al. (Eds.): Transactions on Edutainment III, LNCS 5940, pp. 105–115, 2009.

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

Fig. 1. Team training with delivery simulators in MMC

Patient simulators are already commercially available from several suppliers (for example the patient simulator "Noelle", a product of Gaumard Miami, Florida. Fig. 2). Although technically advanced, the level of realism is not particularly high. Next to the toy like external appearance, it is also the not really flexible material applied which has the effect that the training experience is still quite remote from the reality. Especially, most of the commercial products today are designed as a stand alone system that does not really take the team training aspects into account. These team training aspects are for example the communication among team members, the position of every member, the monitoring and analysis of the team performance and so forth. Hence a 10 year cooperation plan has been set up between Eindhoven University of Technology and Máxima Medical Center to develop the next generation simulation based training facilities, involving several departments from both institutes as weel as external partners.

For a more realistic experience and an optimal training result it is necessary to involve as many different senses as possible: vision, sound, smell and also importantly a realistic touch experience (moistness, warmth, friction). To realize the technology should allow mixing things that are real and things that appear to be real, and augmented reality seems to be promising. Augmented reality is already applied at present for several training goals where the real experience is

Fig. 2. Noelle from Gaumard

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

too dangerous or too expensive [3]. Examples are training firemen (judging risks for collapsing of a building or the probability of an explosion with a tanker truck overturn), or military personnel (training with realistic impacts of shells).

Next to patient simulators there are also additional possibilities and requirements for visualizing a realistic environment (virtual reality). One can think of adding objects in the background (walls, doors, windows, equipment, but also persons walking by). The advantage of virtual reality is that the very same training room can be used for very different training scenarios with little effort, changing from a delivery room to an emergency room or a mobile situation in an ambulance.

For team performance monitoring and analysis, video based techniques such as 3-D visual signal processing [4] and video content analysis [5] can be applied. D-D Depth map generation techniques create the depth map from a non-calibrated video sequence using the "structure-from-motion" algorithm. This technique facilitates the creation of a 3-D model of a scene from any view angle, which is an essential component in human body and behavior simulation. Human behavior analysis and simulation can be started by the analysis of human motion, since motion reflects the behavior, focusing not only on the global motion estimation, such as human segmentation and tracking, but also on the feature-based motion analysis. Further, human modeling techniques including a 2-D or 3-D human geometry (skeleton) model and a fitting algorithm link the detected motion to the model, generating a reliable human model that tracks the motion with sufficient accuracy. It enables fast semantic analysis of human behaviors. Based on these technologies, combining with sound and facial expression analysis, it is possible to couple emotional state recognition to the imposed conditions of the delivery simulator.

The aforementioned concepts brings more software and hardware devices and components into the training room than a single patient simulator. We also aim at an open system architecture that is flexible and extensible enough for the industry to introduce further development and future technologies into simulation based team training. This paper presents the concepts of such an architecture. First the notion of "Play" is introduced as a metaphor for scripting the training scenarios, then the main architectural constructs needed for bringing a "Play" to live are described, followed by a discussion of the difference between playing a medical training scenario and playing a multimedia entertainment scenario, and its implication for future research and development.

2 Scenario Based Scripting: Play Metaphor

In the delivery room, a team of experts must work together, reacting to each other on expected and unexpected situations at right and sometimes critical moments. For a successful training session with certain focused training purposes, the communication, the situations, the reactions and the timing must be planned in advance for a optimal training result. At the moment this is often done by manually preparing a training scenario and carrying out under the supervision

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

of a leading gynecologist. Observing this team training practice, we find the Play metaphor proposed by Hu (2006) to be applicable, although it is originally designed for distributed interactive multimedia [6].

A *play* is a common literary form, referring both to the written works of dramatists and to the complete theatrical *performance* of such. Plays are generally performed in a *theater* by *actors*. To better communicate a unified interpretation of the text in question, productions are usually overseen by a *director*, who often puts his or her own unique interpretation on the production by providing the actors and other stage people with a *script*. A script is a written set of directions that tell each actor what to say (*lines*) or do (*actions*) and when to say or do it (*timing*). If a play is to be performed by the actors without a director and a script from the director, the results are unpredictable, if not chaotic. Here we use the word "play" for both its written form of a script, and the stage performance form of this script.

There are obvious similarities between a theater play and a training session, and the metaphor can be directly used by taking a training room as a theater. The trainees, the simulators and other hardware and software devices are actors, and the training room is their performing theater. The script describes the training scenario, where actions must be taken with right timing. One might also argue why a "Play" metaphor is needed and why the training scenarios can not be directly used. There are two reasons for this. First, we are not only aiming at the delivery simulations, but also other medical training simulations such as newborn babies, a more generic set of terms are necessary to keep the system flexible for extensions. Second, we need a striping language that imposes timing and communication among the trainees, the simulators and other system components. A stricter definition is necessary when the machines are included in interaction loops.

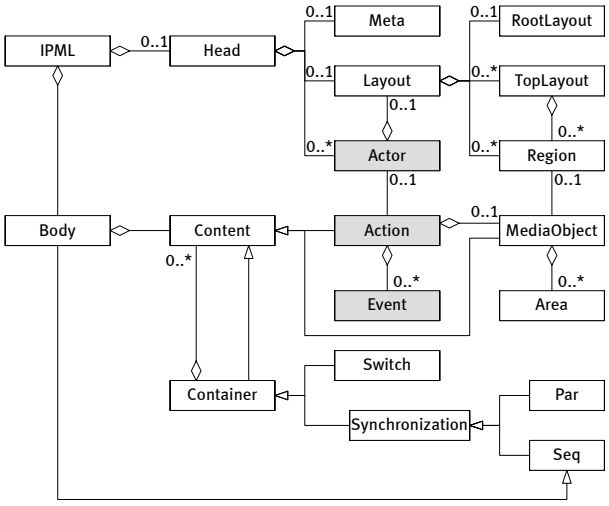

Fig. 3. Scripting language extends SMIL

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

We propose an extension to SMIL (Synchronized Multimedia Integration Language) [7] for scripting the scenarios, by introducing the metaphor of play, adding event-based linking elements and keeping timing and synchronization elements [6]. Fig. 3 shows an object-oriented view on the structure of the language IPML.

3 Actors: Distributed PAC Agents

Many interactive architectures have been developed along the lines of the object-oriented and the event driven paradigms. Model-View-Controller (MVC) [8] and Presentation-Abstraction-Control (PAC) [9] are the most popular and often used ones [10].

The MVC model divides an interactive agent into three components: model, view and controller, which respectively denotes processing, output and input. The model component encapsulates core data and functionality. View components display information to the user. A View obtains the data from the model. There can be multiple views, each of which has an associated controller component. Controllers receive input, usually as events that encode hardware signals from input devices.

Coutaz (1987) proposed a structure called Presentation-Abstraction-Control [9], which maps roughly to the notions of View-Controller pair, Model, and the Mediator pattern [11]. It is referenced and organized in a pattern form by Buschmann et al. [10]: the PAC pattern "defines a structure for interactive software systems in the form of a hierarchy of cooperating agents. Every agent is responsible for a specific aspect of the application's functionality and consists of three components: presentation, abstraction, and control. This subdivision separates the human-computer interaction of the agent from its functional core and its communication with other agents."

In the design of the simulation based training system, PAC is selected as the overall system architecture, and the actors are implemented as PAC agents that are managed by the scheduling agents in a PAC hierarchy, connected with the channels, and performing the actions.

This structure separates the user interface from the application logic and data with both top-down and bottom-up approaches (fig. 4). The entire system is regarded as a *top-level* agent and it is first decomposed into three components: an *Abstraction* component that defines the system function core and maintains the system data repository, a *Presentation* component that presents the system level interface to the user and accepts the user input, and in between, a *Control* component that mediates the abstract component and the presentation component. All the communications among them have to be done through the control components.

At the *bottom-level* of a PAC architecture are the smallest self-contained units which the user can interact with and perform operations on. Such a unit maintains its local states with its own *Abstraction* component, and presents its own state and certain aspects of the system state with a *Presentation* component. The communication between the presentation and the abstraction components is again through a dedicated *Control* component.

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

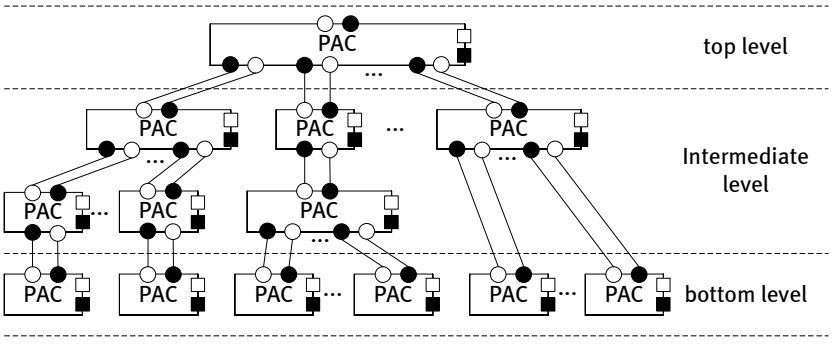

Fig. 4. Distributed PAC in a hierarchy

Between the top-level and bottom level agents are *intermediate-level* agents. These agents combine or coordinate lower level agents, for example, arranging them into a certain layout, or synchronizing their presentations if they are about the same data. The intermediate-level may also have its interface *Presentation* to allow the user to operate the combination and coordination, and have an *Abstraction* component to maintain the state of these operations. Again, with the same structure, there is a control component in between to mediate the presentation and the abstraction.

The entire system is then built up as a PAC hierarchy: the higher-level agents coordinate the lower level agents through their *Control* components; the lower level agents provide input and get the state information and data from the higher level agents again through the *Control* components. This approach is believed more suitable for distributed applications and has better stability than MVC, and it has been used in many distributed systems and applications, such as CSCW [12], distributed real-time systems [13] , web-based applications [14,15], mobile robotics [16,17], distributed co-design platforms [18] and wireless services [19].

To a large degree the PAC agents are self-contained. The user interface component (Presentation), the processing logic (Abstraction) and the component for communication and mediation (Control) are tightly coupled together, acting as one. Separations of these components are possible, but these distributed components would then be regarded as PAC agents completed with minimum implementation of the missing parts. Thus the distribution boundaries remain only among the PAC agents instead of composing components.

Based on this observation, each component is formally described by Hu (2006) [6], modeling the communication among PAC agents with push style channels from the Channel pattern. In fig. 4, a "●" indicates a data supplier component, a "○" indicates a data consumer component and a connecting line in between indicates the channel. The symbol "■" indicates that the attaching component has a function of physically presenting data to the user, and the symbol "□" indicates the function of capturing input from the user interface or the environment.

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

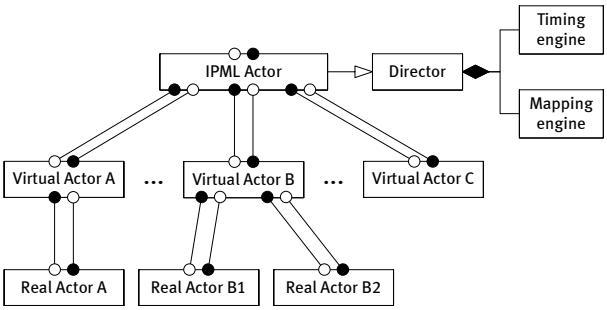

Fig. 5. IPML system: an IPML actor

An actor is then basically a PAC agent. It reacts on the user input events and scheduling commands, and takes corresponding actions. The final system is simply an IPML actor, or in other words, an *Actor* implementation that is capable of presenting the IPML scripts. The IPML actor implements the role of a *Director*, which has a mapping engine [20], creates, manages and connects the virtual actors, and has a timing engine which schedules the timed actions for the delegating virtual actors (fig. 5). Depending on the physical configuration of the "theater" – the presenting environment, the mapping engine may also connect appropriate "real actors" to virtual actors, where the virtual actors keep the role of software drivers for the "real actors". The mapping engine may make use of distributed lookup and registration services such as UPNP [21] and JINI [22] to locate and maintain a list of "real actors", but this architecture leaves these possibilities open to the implementation of the mapping engine. The timing engine takes the timing relations defined using the SMIL timing module, translates them into an internal representation (based on Peri Nets [23,24] and OCPN [25], see also a detailed description in [6]), and schedules the actions for the actors, no matter whether the actors are distributed over the network.

4 Discussions

When comparing the requirements to playing a medical training scenario and the requirements to playing a multimedia entertainment scenarios, the following differences and points of attention appear:

A. In the medical training scenario the need for explicit control of timing aspects is even more important than in entertainment applications.
B. In the medical training scenario the need for team training is very essential. Whereas many entertainment applications nowadays under development are still geared towards and individual experience, team training is the most important issue in present day medical training.
C. In the medical training scenario there will be variations among the capabilities of the various manikins and instrumentations of the training rooms.

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

At present most training rooms are pioneering the role of team training anyhow and the scenarios are developed locally and tuned to the available manikins and instruments. But as to the training centers and the manikin industry manufactures, there will be more and more different versions of the manikins with formalized capability lists. Then it must be possible that the IPML plays execute in different training rooms without elaborate manual adaptation or rewriting. Automated mapping software as a part of the software architecture can run the same scenario's by adapting them to the manikins at hand.

D. In the medical training scenario there is a larger need for evaluating the performance of the users. It is not enough to optimize their experience according to certain dramatic, physical and emotional goals. Each session will be followed by a debriefing session in which the actions performed, their timing, their precise locations, the forces used, are essential. When the session is part of a qualification or examination this becomes even more obvious.

These points are easily illustrated by examples from obstetrics, notably breech deliveries, which is precisely the field in which our group cooperates with Prof G. Oei MD and others of the MMC (Máxima Medisch Centrum), Veldhoven, The Netherlands together with the Máxima Academie, the training center.

A. the timing of the various phases of the delivery and the various degrees of oxygen shortage associated will determine whether or not the baby will suffer brain damage. Particularly when the umbilical cord is blocked, every second counts.

B. we just give a very simple example of a team aspect. From the instructions for Lovset's maneuver we quote "Grasp his thighs and pelvis with both hands (if he is slippery use a gauze swab or small towel), your thumbs along his sacrum, your forefingers on his symphysis, and your remaining fingers round his thighs" [26]. Clearly the doctor's assistent has to anticipate and it is essential that the doctor communicates his strategy. Only then can the assistant make sure the gauze swab or the small towel is at hand precisely when the doctor needs it.

C. Different manufacturers of manikins have their own product lines. Important examples are Gaumard, Laerdal, Limbs and Things and Meti. These companies each have their own strengths and weaknesses. For example Meti excels in physio-pharmacological models, Laerdal excels in matters of circulation and respiration, and so on. Moreover they release new versions every few years. In view of the costs, a training center cannot have them all.

D. A very clear example of the need to record and check quantitative performance data is the problem of shoulder dystocia. When the fetus is in breech position, the legs will be delivered first, instead of the head. Then it can happen that the shoulders get stuck in which case the doctor has to perform very specific maneuvers such as the Van Deventer maneuver or the Lovset maneuver. The latter maneuver involves rotating the baby in a certain manners of 180 degreesȮnly after that it is allowed to pull. Measuring the leftward

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

> and rightward rotations in degrees and the subsequent force in Newton is essential because in real life these parameters will determine the fait and the health of the baby. If the manoeuvres have not been done correctly, the higher force needed will have caused damages such as partial paralysis.

IPML satisfies the needs of A perfectly and it is well-prepared for B in the sense that the play for the actors is already organized as a team effort, not as a single actor. IPML is also perfectly prepared to cope with the demands of C because of the mapping structure and mechanisms [6, Chapter 9]. For D, however, it is necessary to extend IPML with Logging and Verification constructs. We propose an extended language version IPML/V, with V for verification. Because the language is XML based this presents no difficulties at the syntactic level. Moreover, the original play is still written in IPML. But during execution it is transformed into an IPML/V file, essentially a structured logging file. Semantically this is what happens: whenever a play is executed, the execution engine keeps track of which alternative actions have been chosen and at which time the events have actually happened. The result is not just a sequential log, but a log in which the events and the time stamps appear as annotations in their original structured setting. Checking any logical condition φ written in a logical formalism, against a given IPML/V model M can be done either in run-time or by postprocessing. Logically it amounts to checking the truth or falsehood of a formula in a model, usually written as $M \vDash \varphi$.

A further development in the description of the training scenarios is providing a graphical WYSIWYG authoring and analyzing tool on top of the IPML and IPML/V. IPML is a scripting language that is based on XML, which is designed to satisfy the needs of a strict definition of the timing and mapping strategies and relations in a scenario. The primary purpose of XML or any XML based language is to facilitate the sharing of structured data across different information systems and to encode documents and serialize data. It can be read and written using a plain text editor however the process can be cumbersome and trivial. Visualizing the interactive components and defining the timing and mapping relations among them graphically would help the doctors and the trainers to concentrate on the training scenarios instead of a scripting hassle.

5 Conclusions

The concepts of the Play metaphor, the IMPL scripting language and the Distributed PAC based multi-agent architecture are found to be applicable in simulation based medical training, because of the similar requirements on timing and mapping in applications of both distributed multimedia entertainment and simulation based medial training. However there are also clear differences between these two application areas. Further research and development need to be done to deal with the issues such as multiple participants in team training and the verification of the actual performance of a training session.

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

References

1. Kohn, L.T., Corrigan, J., Donaldson, M.S.: To Err Is Human: Building a Safer Health System. National Academies Press (2000)
2. Draycott, T., Sibanda, T., Owen, L., Akande, V., Winter, C., Reading, S., Whitelaw, A.: Does training in obstetric emergencies improve neonatal outcome? BJOG: An International Journal of Obstetrics and Gynaecology 113, 177–182 (2006)
3. Azuma, R.T.: A survey of augmented reality. Presence: Teleoperators and Virtual Environments 6(4), 355–385 (1997)
4. Lao, W., Han, J., de With, P.H.N.: 3d modeling for capturing human motion from monocular video. In: Proc. Int. Symp. On Information Theory in the Benelux (WIC), Noordwijkerhout (NL), pp. 299–306 (2006)
5. Han, J., Farin, D., de With, P.H.N., Lao, W.: Real-time video content analysis tool for consumer media storage system. IEEE Transactions on Consumer Electronics 52(3), 870–878 (2006)
6. Hu, J.: Design of a Distributed Architecture for Enriching Media Experience in Home Theaters. Technische Universiteit Eindhoven (2006)
7. Ayars, J., Bulterman, D., Cohen, A., Day, K., Hodge, E., Hoschka, P., Hyche, E., Jourdan, M., Kim, M., Kubota, K., Lanphier, R., Layaïda, N., Michel, T., Newman, D., van Ossenbruggen, J., Rutledge, L., Saccocio, B., Schmitz, P., ten Kate, W., Michel, T.: Synchronized multimedia integration language (SMIL 2.0), 2nd edn. W3C recommendation (2005)
8. Krasner, G.E., Pope, S.T.: A cookbook for using the model-view controller user interface paradigm in smalltalk-80. Journal of Object Oriented Program 1(3), 26–49 (1988)
9. Coutaz, J.: PAC, an implemention model for dialog design. In: Interact 1987, Stuttgart, pp. 431–436 (1987)
10. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture, A System of Patterns, vol. 1. John Wiley & Sons, Inc., Chichester (1996)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns – Elements of Reusable Object-oriented Software. Addison-Wesley, Reading (1995)
12. Calvary, G., Coutaz, J., Nigay, L.: From single-user architectural design to PAC*: a generic software architecture model for cscw. In: CHI 1997 Conference, pp. 242–249. Addison-Wesley, Reading (1997)
13. Niemelä, E., Marjeta, J.: Dynamic configuration of distributed software components. In: ECOOP 1998: Workshop ion on Object-Oriented Technology, London, UK, pp. 149–150. Springer, Heidelberg (1998)
14. Illmann, T., Weber, M., Martens, A., Seitz, A.: A pattern-oriented design of a web-based and case oriented multimedia training system in medicine. In: The 4th World Conference on Integrated Design and Process Technology, Dallas, US (2000)
15. Zhao, W., Kearney, D.: Deriving architectures of web-based applications. In: Zhou, X., Zhang, Y., Orlowska, M.E. (eds.) APWeb 2003. LNCS, vol. 2642, pp. 301–312. Springer, Heidelberg (2003)
16. Khamis, A., Rivero, D.M., Rodriguez, F., Salichs, M.: Pattern-based architecture for building mobile robotics remote laboratories. In: IEEE International Conference on Robotics and Automation (ICRA 2003), Taiwan, vol. 3, pp. 3284–3289 (2003)
17. Khamis, A.M., Rodriguez, F.J., Salichs, M.A.: Remote interaction with mobile robots. Autonomous Robots 15(3) (2003)

J. Hu and L. M. G. Feijs, "A Distributed Multi-agent Architecture in Simulation Based Medical Training," Transactions on Edutainment , vol. III, LNCS 5940, pp. 105-115, 2009.

18. Fougeres, A.-J.: Agents to cooperate in distributed design. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2629–2634 (2004)
19. Niemela, E., Kalaoja, J., Lago, P.: Toward an architectural knowledge base for wireless service engineering. IEEE Transaction on Software Engineering 31(5), 361–379 (2005)
20. Feijs, L., Hu, J.: Component-wise mapping of media-needs to a distributed presentation environment. In: The 28th Annual International Computer Software and Applications Conference (COMPSAC 2004), Hong Kong, China, pp. 250–257. IEEE Computer Society Press, Los Alamitos (2004)
21. Michael Jeronimo, J.W.: UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play. Intel Press (2003)
22. Edwards, W.K.: Core JINI, 2nd edn. Prentice Hall PTR, Englewood Cliffs (2000)
23. Petri, C.A.: Kommunikation mit Automaten. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2 (1962)
24. Petri, C.A.: Kommunikation mit automaten. New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377 1 (1966); 1–Suppl. 1 English translation
25. Little, T.D.C., Ghafoor, A.: Synchronization and storage models for multimedia objects. IEEE Journal on Selected Areas in Communications 8(3), 413–427 (1990)
26. Awori, N., Bayley, A., Beasley, A., Boland, J., Crawford, M., Driessen, F., Foster, A., Graham, W., Hancock, B., Hancock, B., Hankins, G., Harrison, N., Kennedy, I., Kyambi, J., Nundy, S., Sheperd, J., Stewart, J., Warren, G., Wood, M.: Primary Surgery: Non-trauma, vol. 1. Oxford Medical Publication (1990)