Hu, J., et al., Distributed Architecture for Delivery Simulators,
in International Conference on e-Health Networking, Digital Ecosystems and Technologies (EDT 2010),
H. Tan, Editor. 2010, IEEE: Shenzhen. p. 109--112.

Distributed Architecture for Delivery Simulators

Jun Hu, Peter Peters, Frank Delbressine, Loe Feijs
Designed Intelligence Group, Department of Industrial Design
Eindhoven University of Technology, The Netherlands
Email: {j.hu, p.j.f.peters, f.l.m.delbressine, l.m.g.feijs}@tue.nl

***Abstract*—The numbers of high-risk pregnancies and premature births are increasing due to the steadily higher age of pregnancy. Medical simulators are used in training the doctors to deal with emergent perinatal situations. To enhance the training effect, sophisticated simulators are integrated into a realistic training environment that takes into account the medical instruments and team aspects. The training environment becomes increasingly complex and requires a clear structure for different training scenarios and flexible hardware configurations. Distributed multi-agent software architecture with peer to peer communication facilities is developed for this purpose. The architecture is presented in this paper.**

***Keywords*-distributed architecture, medical simulators, medical training, manikin**

I. Introduction

The numbers of high-risk pregnancies and premature births are increasing due to the steadily higher age of pregnancy. As a result, mother and child face increasing risks for miscarriage, premature delivery, birth defects, and health problems later in life. This makes it increasingly important to train doctors and health care personnel to deal with emergency perinatal situations. Medical simulators are used more and more in training, due to the decreasing real-life cases. To enhance the training effect, sophisticated simulators are integrated into a realistic training environment that takes into account the medical instruments and team aspects [1]. The training environment becomes increasingly complex and requires a clear structure for different training scenarios and flexible hardware configurations.

II. The MedSim system

In medical education, how to act in emergency situations is often trained on an individual basis. In practice however , patients are 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 few years, Máxima Medical Center (MMC) in Veldhoven has been providing such multidisciplinary team training using medical simulations. 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 life situation (see Figure 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.

Patient simulators are already commercially available from several suppliers [3], [4]. 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 reality. Especially, most of the commercial products today are designed as a standalone system that does not really take the aspects of the training environment and team training 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. In the training environment, not only the mother and baby manikins should be used, but also the medial monitoring equipment, the space layout and arrangement, lighting and noise conditions should be taken into account. This results in a distributed training environment and the distribution would enhance the training experience and effect [5]. Hence Eindhoven University of Technology (TU/e) is cooperating with MMC, aiming at the next generation simulation based training facilities. The result of this effort is the design and implementation of the MedSim system.

In the design, sensors and actuators are integrated into the mother and baby manikins to simulate the delivery process and to react on the actions taken by the team. For example in the baby manikin, sensors are integrated to detect the position of the baby when the baby is maneuvered inside the birth channel, the pulling and holding force being applied when the baby is pulled through the birth channel using forceps (Figure 2), and positions of the body parts (head, arm and legs). Actuators integrated into the baby manikin simulate important signals that the trainees shall observe, such as the muscle tone, skin

Figure 1. Team training with delivery simulators at MMC

Hu, J., et al., Distributed Architecture for Delivery Simulators,
in International Conference on e-Health Networking, Digital Ecosystems and Technologies (EDT 2010),
H. Tan, Editor. 2010, IEEE: Shenzhen. p. 109--112.

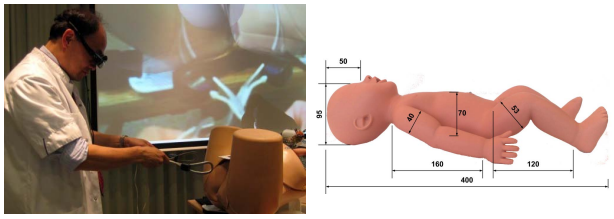

Figure 2. Mother and baby manikins

color, breathing behavior. These sensors and actuators are connected to a microcontroller. The microcontroller has a wireless network component that communicates with other system components, for example, the mother manikin that has sensors and actuators integrated in a similar manner [6].

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 this 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 in other fields at present for several training goals where the real experience is too dangerous or too expensive. 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). In our implementation the augmented reality is used for example to simulate the massive blood flow when things go wrong during the delivery process.

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 3D visual signal processing and video content analysis are applied. 3D 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 3D model of a scene from any view angle. Human behavior analysis and simulation are started by the analysis of human motion, since motion reflects the behavior. Further, human modeling techniques including a 2D or 3D human geometry (skeleton) model and a fitting algorithm link the detected motion to the model, generating a reliable model that tracks the motion with sufficient accuracy. It enables fast semantic analysis of human behaviors. These technologies, combined with sound and facial expression analysis, make it possible to couple emotional state recognition to the imposed conditions of the delivery simulator.

The aforementioned concepts bring more software and hardware devices and components into the training room than a single patient simulator. We 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. In the design of the delivery simulation system, distributed sensors and actuators are integrated into the mother manikin, the baby manikin, as well as the environment for the purpose of medical team training. We introduce the software architecture that supports open and flexible integration, in which XML (Extensible Markup Language) based messaging mechanisms and P2P network technology are used.

III. Software Architecture

The MedSim system uses a script driven, agent based architecture [3], [4], [7], [8], similar to the structure proposed in [9]. During a training session many distributed components are employed, including the mother and baby manikins, tracking cameras, augmenting projections, medical monitoring equipments, etc. Every physical component has integrated sensors and actuators to detect the actions of the team and the status of delivery process, and to react upon these actions and status according to prescripted scenarios and fetal-maternal models. Each physical component has a software counterpart and is modeled as a software agent. Distributed agents communicate over a messaging bus using a dedicated messaging protocol. During a training session, the director agent interprets the training scenario script and coordinates the actions and reactions of the distributed agents accordingly, as shown in Figure 3. XML-based scripting enables the distribution of the events and messages, and the coordination of the actions taken by the agents. It does not rely on a particular system implementation and it is easily extensible [7], [10].

A. Physical components

Each physical component has it own internal structure that implements its functionalities, as well as a communication service to connect to its software agent. This is usually done through its embedded software. The baby manikin is a good example.

The software in the baby manikin will have to take care of reading the available sensors, driving the actuators present, doing conversions possibly according to a physiological model, take care of timing relations and communication to and from the internal communication service which in turn connects to the software agent (Figure 4, [11]). Although the actual implementation of the prototype software is still simple compared to what it will be in a full fledged manikin, the same architecture will be used for future prototypes.

1) Sensors and Actuators: All connected sensors are read sequentially, their data values are stored and the mean value of the last 10 measurements is calculated and used as data to send in the sensor message. The calculation creates a low pass filter that reduces the amount of noise that might be present on the signal but it also prevents quick

Hu, J., et al., Distributed Architecture for Delivery Simulators,
in International Conference on e-Health Networking, Digital Ecosystems and Technologies (EDT 2010),
H. Tan, Editor. 2010, IEEE: Shenzhen. p. 109--112.

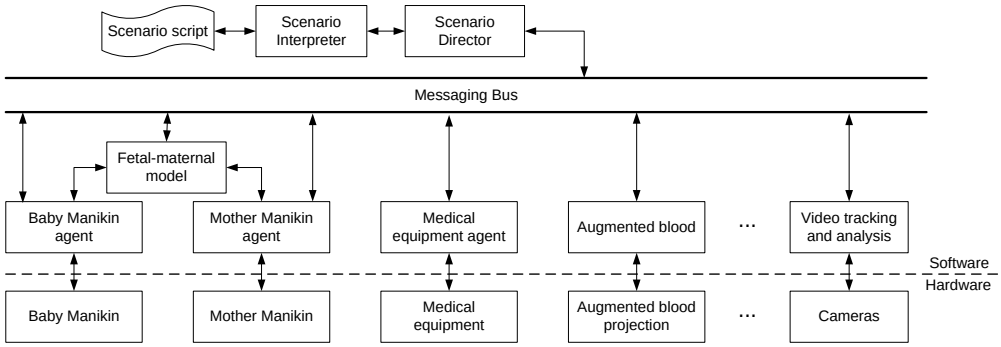

Figure 3. MedSim system overview

changes to be detected and introduces delays. Actuators are driven as soon as the type of the incoming message indicates it is an actuator message. The message will also contain an indication of which actuator to drive and a value that determines what to do with the actuator.

2) Timing: The most influential factor in all timing of the software will be the actual speed that is needed for reading the sensors and driving the actuators. The microcontroller platform used in the implementation allows for reading sensors (and driving actuators) with a repetition rate of 20 mS, which is more than enough for most of the sensors and actuators. For signals that would require the fastest reaction time of the microcontroller, for example the muscle reflex signals, a sensor with local processing capability, a sensor with direct coupling to the related actuator, or a more powerful microcontroller is suggested.

3) Communication: The baby manikin communicates through the communication service. This layer of service allows the separation of the hardware component from its software agent. In an ideal implementation, the software agent can be integrated into the hardware component as part of the embedded software. Separation can be necessary especially during the prototyping phase, or when the physiological model applied is too complicated to to be implemented in the embedded software.

B. Messaging protocol

In the system the agents operate separately in a decentralized manner. Communication among the agents is designed not to be managed through a central server, but to be carried out by following an XML based protocol that is independent of network protocols, hardware and software platforms and implementation languages. The aims are that not only in a running system can a component join or leave the system at any moment, but also such a component can be developed and provided separately by a third party as long as it follows the protocol.

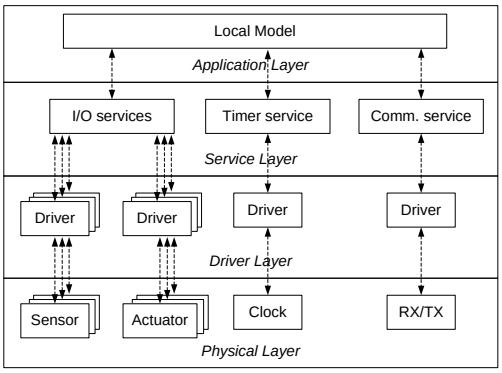

Figure 4. Inside the baby manikin

XML based messages are easily extensible, easy to read, process and exchange. There are a variety of XML protocols, including XML-PRC, SOAP, WDDX, XMI, Jabber, ebXML, WSDL, WIDL, SCL - just to name a few of them. The Messaging protocol for the MedSim system went through several design iterations and the final version (0.1.3) is based on SOAP (Simple Object Access Protocol) [12]. Although SOAP was originally designed as a basic messaging framework upon which web services can be built, it was later extended for general purposes in exchanging structured information in a decentralized, distributed environment in a way that is independent of any particular programming model and other implementation specific semantics, which satisfies the requirements for the communication between the MedSim agents. Simply saying, the MedSim messaging protocol is an application of SOAP. As an application, implementation of such a protocol can easily make use of a rich set of existing SOAP-based communication libraries for many different hardware and software platforms.

A MedSim message has a header part and a body part. The header part defines the related information such as version control, transportation source and destination, identifications for communication sessions, and quality of service requirements. In the body part concrete message entries are defined, each with a list of parameters. A simple example of such a message is shown in Figure 5.

C. Messaging bus

The communication channels in the system architecture are implemented as a function of the “Messaging Bus”, with which the agents can send messages to or receive messages from the other agents. The structure of the messaging bus is shown in Figure 6. Although the messaging bus is programmed in Java in our reference implementation, the components in this structure are specified as

Hu, J., et al., Distributed Architecture for Delivery Simulators,
in International Conference on e-Health Networking, Digital Ecosystems and Technologies (EDT 2010),
H. Tan, Editor. 2010, IEEE: Shenzhen. p. 109--112.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
xmlns:medsim="http://www.deliverysimulator.id.tue.nl/medsim/">
   <SOAP-ENV:Header>
      <medsim:Sender>Baby</medsim:Sender>
      <medsim:Receiver>Mother</medsim:Receiver>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <medsim:HeartBeat>
         <rate>100</rate>
         <isNormal>true</isNormal>
      </medsim:HeartBeat>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5. A MedSim message

abstract interfaces such that it can be implemented by a third party using different tools on different platforms.

During the project we created three reference implementations in Java, firstly "local" then "unicasting" and "multicasting". The local messaging bus was created firstly for testing purposes, which only channels the communication among agents on the same local machine. The unicasting and multicasting messaging bus was implemented using JXTA protocols [13]. The JXTA protocols are a set of six protocols that have been specifically designed for ad hoc, pervasive, and multi-hop peer-to-peer (P2P) network computing. Using the JXTA protocols, peers can cooperate to form self-organized and self-configured peer groups independently of their positions in the network, and without the need of a centralized management infrastructure. The multicasting messaging bus propagates the message to all the peers connected to the same bus. Although multicasting may cause higher demand on the network traffic, for a small amount of peers, it is convenient. If the system scales up, the multicasting bus can be easily switched to a different type, namely, a unicasting bus, by requesting a different bus from the *MessagingBusFactory*.

IV. Conclusion

The software architecture presented in this paper is implemented in the MedSim system, which is now used at the MedSim Center in Eindhoven for medical training.

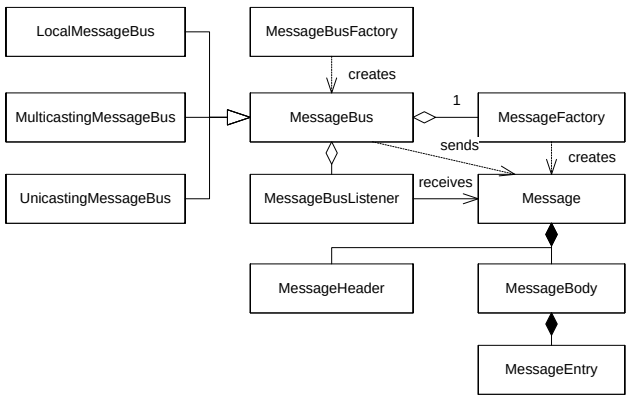

Figure 6. Messaging bus

The XML based messaging mechanisms and the P2P network technology support the integration of distributed sensors and actuators into the training environment, which has been proven in practice to be solid and flexible. TU/e researchers from different departments and medical staff from MMC are working together towards the future perinatal solutions.

References

[1] W. Chen, J. Hu, S. Bouwstra, S. B. Oetomo, and L. Feijs, "Sensor integration for perinatology research," *International Journal of Sensor Networks*, 2010, to appear.

[2] M. James, "Does training in obstetric emergencies improve neonatal outcome?" *British journal of Obstetrics and Gynaecology*, vol. 113, no. 8, 2006.

[3] J. Hu and L. Feijs, "A distributed multi-agent architecture in simulation based medical training," *Transactions on Edutainment*, vol. III, LNCS 5940, pp. 105–115, 2009.

[4] ——, "A distributed multi-agent architecture in simulation based medical training," in *Learning by Playing. Game-based Education System Design and Development*, ser. Lecture Notes in Computer Science, M. Chang, R. Kuo, Kinshuk, G.-D. Chen, and M. Hirose, Eds. Banff, Canada: Springer Berlin / Heidelberg, 2009, vol. 5670/2009, p. 49.

[5] J. Hu, M. Janse, and H.-j. Kong, "User experience evaluation of a distributed interactive movie," in *HCI International 2005*, Las Vegas, 2005.

[6] P. Peters, L. Feijs, and G. Oei, "Plug and play architectures for rapid development of medical simulation manikins," in *WMSCI 2008 - The 12th World Multi-Conference on Systemics, Cybernetics and Informatics*, vol. 2, Orlando, Florida, USA, 2008, pp. 214–219.

[7] J. Hu and L. Feijs, "Ipml: Extending smil for distributed multimedia presentations," in *Interactive Technologies and Sociotechnical Systems*, ser. Lecture Notes in Computer Science. Xi'an, China: Springer, 2006, vol. 4270/2006, pp. 60–70.

[8] ——, "An agent-based architecture for distributed interfaces and timed media in a storytelling application," in *The 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, Melbourne, Australia, 2003, pp. 1012–1013.

[9] J. Hu, "Design of a distributed architecture for enriching media experience in home theaters," PhD Thesis, Department of Industrial Design, Eindhoven University of Technology, 2006.

[10] ——, "Storyml: Enabling distributed interfaces for interactive media," in *The Twelfth International World Wide Web Conference*, vol. CDROM, p135, Budapest, Hungary, 2003.

[11] P. Peters, "Design of a medical simulator hard- and software architecture," 2010, to be published.

[12] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon, "Soap version 1.2 part 1: Messaging framework. available from http://www.w3.org/tr/soap12-part1/," 2003.

[13] L. Gong, "Jxta: A network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95, 2001.