Component-Based Networking for Simulations in Medical Education

Sebastian von Mammen*, Timothy Davison*, Hamidreza Baghi* and Christian Jacob*†
*Dept. of Computer Science, †Dept. of Biochemistry and Molecular Biology
University of Calgary
Canada

***Abstract*—For the purpose of medical education, our research team is creating LINDSAY, a 3-dimensional, interactive computer model of male and female anatomy and physiology. As part of the LINDSAY—Virtual Human project, we have developed a component-based computational framework that allows the utilization of various formal representations, computation engines and visualization technologies within a single simulation context. For our agent-based simulations, the graphics, physics and behaviours of our interacting entities are implemented through a set of component engines. We have developed a light-weight client/server component, which spreads its siblings in the system's component hierarchy over a wireless or wired network infrastructure. In this paper we demonstrate how our client/server component paves new ways for organizing, generating, computing and presenting educational contents.**

I. Introduction

As health care systems all over the world struggle to provide affordable and comprehensive care, the need for excellent education and training of medical staff is more important than ever. At the same time, emerging digital technologies render it possible to present complex contents faster and better to large audiences than ever before. In this context, we are developing LINDSAY—Virtual Human, a project at the intersection of Computer Science, Education, and Medicine. LINDSAY provides a collection of computational tools for research and learning in the context of human anatomy and physiology.

As a starting point for this project, we experienced and analyzed lectures in human anatomy given by distinguished instructors. These investigations led to the conclusion that the mere projection of a virtual three-dimensional human body in combination with the ability to easily navigate and label the display could greatly facilitate and improve the learning experience. Consequently, such an application became the first milestone of our LINDSAY project.

To this end, we devised a component-based framework to make our content presentation platform extensible [1], [2]. Hierarchies of components can combine attributes and behaviours on different levels of scale and resolution. They can host information for various visualization techniques (e.g., charts, animations), for interaction interfaces (associated with hardware devices or software user-interfaces) or for simulations computed in real-time and driven by heterogeneous computational engines (physics engines, differential equation solvers, etc.).

In this work, we present the design and implementation of a networking component embedded into our content delivery system. It enables a range of network topologies and configurations that can support novel means of active learning in classrooms supported by wireless devices [3].

The remainder of this paper is structured as follows. Section II introduces the emerging technology of virtual human anatomies in learning environments. It also touches upon agent-based and object-oriented simulation techniques as they are our current focus for the presented networking technology. We also provide references to previous networking solutions for component-based architectures. Section III describes the design and implementation of our client/server networking component and its integration into the larger component framework of our LINDSAY Virtual Human system. Section IV describes examples of distributed computing and visualization in the context of content generation and delivery in an educational environment. The article concludes with a summary and exploration of opportunities for future work in Section V.

II. Related Work

For more than two decades, scientists have been exploring ways to enhance medical research and education by way of computer-based renderings of human anatomy and physiology. The American National Library of Medicine started as early as 1989 with the composition of a comprehensive imagery database of human physiology, also referred to as the Visible Human [4]. Since then, these data sets have been inspiring a large number of virtual anatomy projects for research, patient consultation and education [5].

In the context of education, atlases have been composed that promote the exploration of detailed anatomical terminology in a proper visual context [6]. Some systems have been further extended to incorporate data about actual biomedical processes in the human body, for instance genetic processes [7]. Such systems can be modelled and simulated by means of traditional mathematical methodologies or as large sets of self-organizing bio-agents, or *swarm* systems [8], [9].

Relying on a component-based architecture for a content delivery and generation framework provides the freedom to combine any of these computational modelling and simulation approaches [2]. More specifically, a component can be broadly defined as follows [1]:

> **Component** A component's characteristic properties are that it is a unit of independent deployment; a unit of third-party composition; and it has no persistent state.

The design of component-based software architectures has advantages in regard to various application domains. Frameworks for (human) collaborative work can be implemented by brokering groupware components [10]. The coordinated execution of heterogeneous components works for organizing human collaboration as well as complex code bases that comprise large sets of interoperable, reusable software components. For example, a component-based augmented reality framework could manage components for user interfaces, tracking or object modelling [11]. Computer games, too, face the challenge of integrating vast numbers of software components, whether related to contents, providing networking infrastructure or user interfaces [12]. A tutorial for constructing a component-based framework in the context of Massively Multiplayer Online games is provided in [13]. For practical reasons, multi-facetted game units are defined by aggregating distinct software components rather than by using established methodologies of object-oriented inheritance [14], [15].

An overview of component-based client/server frameworks is provided in [16]. Sets of component-based distributed embedded systems facilitate coordinated interactions [17]. Redeployment of components across a network infrastructure results in improvements regarding service availability [18]. Alternatively, mobile devices can exchange software components in peer-to-peer networks in order to address user requests [19]. The exchange of software components in heterogenous hardware infrastructures might require adaptation of the control over the respective components or of their data. In [20], such an adaptation strategy is presented for transferring components among devices of varying degrees of computational power, e.g., from a desktop/server to a set of mobile devices. In particular, adaptation is realized by specific drivers that serve as middleware to translate the broadcast software components.

III. Component-based Simulation & Networking

In our component-based simulation framework, the LINDSAY Composer, a simulation is represented as a hierarchy of components. Figure 1 shows a prototypical component hierarchy of a simulation. Darker shaded boxes represent nested components at a lower level of the system hierarchy. Components may be registered with and interface with various computational engines. As an example, Figure 2 illustrates how components on a higher hierarchical level, such as the Blood Cell, can aggregate several subcomponents that are registered with their respective component engines. Here, the Blood Cell component has children registered with Graphics and Physics component engines.

In particular, a component engine accesses a component's data, possibly relying on the presence of various sibling components. State updates of the components drive the computational processes. As an example, a Physics component engine updates the Transform component (position and orientation) of the Blood Cell depicted in Figures 1 to 3 .

Although the aggregation of different components for individual objects results in a flat functional hierarchy [14], [15], a tree hierarchy facilitates object management in large simulation environments. Parent-child relationships, as emphasized in Figure 1, are primarily semantically motivated. They can, however, also play a role in the context of certain modelling representations such as Membrane Computing, where biological systems are described based on inner-compartmental particle interactions and inter-compartmental particle transfers [21]. In such a case, the component hierarchy in combination with the means of object introspection can provide the data structure required for a component engine that works across hierarchies.

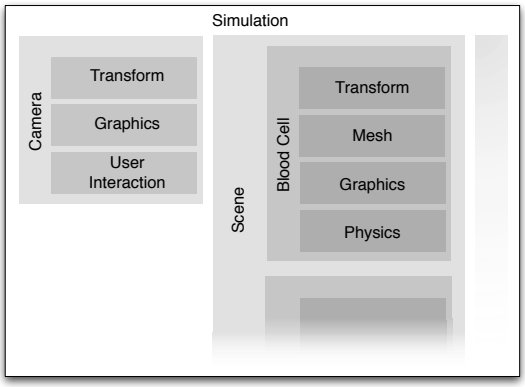

Fig. 1. Components at different hierarchical levels are defined by the aggregation of their child components. Usually only the leaves of the tree are registered with respective component engines.

A. Client/Server Components

We have embedded our Server and Client components into the simulation's component hierarchy. Thereby, we determine which parts of the simulation should be broadcast over the network, or where the received components should be embedded in the recipient's simulation context. Figures 2 and 3 show a Server and Client component within the hierarchies of two prototypic simulation contexts—on a server and a client system, respectively. In accordance with the shaded hierarchy schema introduced in Figure 1, the Server component in Figure 2 is parented by a Scene component and is a sibling of a Blood Vessel and a Blood Cell.

On the implementation side, our Client/Server component architecture works as follows. The Server tries to connect with a given Bonjour service name. Once a connection between the Server and the Client component has been established, the Server component first traverses and then broadcasts its sibling components. Next, changes to the properties and hierarchical organization of these siblings are aggregated over each frame of the simulation, and are then packaged and distributed across the network link. On the other side of this link, the Client component first creates, then maintains and updates a mirrored

set of these components as it receives updates from the server. These updates also consider structural changes to the hierarchy, including creating, destroying, and rearranging the mirrored components. In addition to the selection of subtrees, a Server component can apply predicates to filter its siblings and their children which results in an arbitrary selection of transmitted and updated components. As an example, one might only be interested in sending component types that encapsulate spatial and visual information but do not process the simulation. Another predicate might filter out a set of components based upon their spatial position within a simulation.

The Client component, on the other hand, skips over the recursive integration of those received components that would require a component engine that is not provided by the client-side system. In the example shown in Figure 3, a Client component receives and maintains the Blood Vessel and the Blood Cell components sent by the Server depicted in Figure 2. As the Server component has filtered out all Physics components, the client system only processes the visualization of the transferred components, relying on the Graphics component engine and the Transform and Mesh component types.

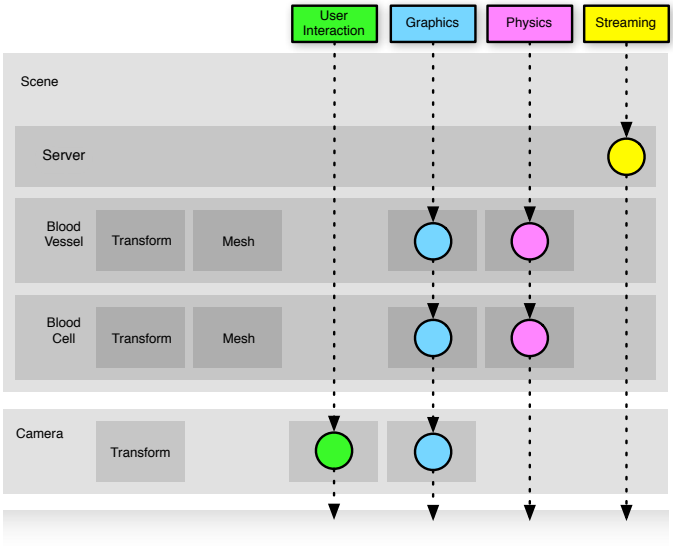

Fig. 2. A Server component is embedded within the scene of the simulation. It transmits and updates its siblings, Blood Vessel and Blood Cell, across the network. Grey boxes represent the component hierarchies. Circles denote registered components. Component engines (coloured boxes) consider the registered components in the order indicated by the dashed arrows. Parts of the diagram were adapted from [15].

B. Technical Aspects

The implemented Client and Server components rely on the Apple Bonjour protocol [22]. A service name is the only information necessary to establish a handshake and a connection link. Mac OS X Snow Leopard on Mac Pro desktop machines with 2.26GHz eight-core Intel processors computed and streamed the physics simulations. Multiple iPhone 3G phones (16GB and 32GB) and iPod touches (3rd generation, 32 GB) were used as wireless devices. In our experiments,

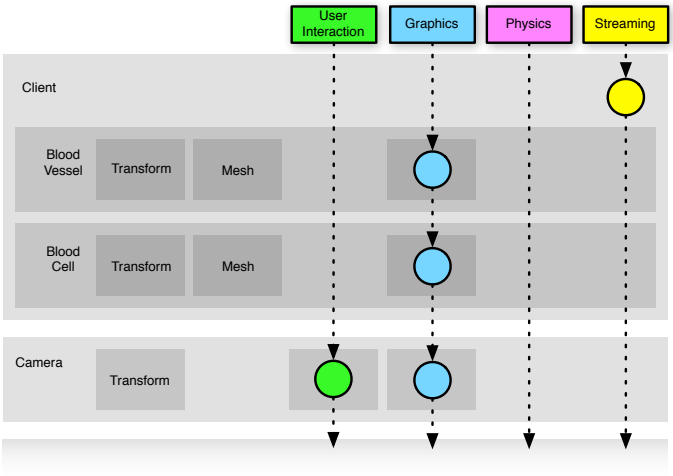

Fig. 3. A Client component integrates the components it receives as children. In the given case, the Blood Vessel and Blood Cell are stripped off their Physics component as it was filtered out by the streaming server in Fig. 2.

we measured about 800KB/s data throughput, rendering the presented configurations more feasible to be deployed on the IEEE wireless standards 802.1g and n, as opposed to b. The Server component's filtering method is implemented based on the NSPredicate class of the Apple Foundation framework [23]. Changes to components are observed via the NSKeyValueObserving protocol.

IV. Example Scenario

The introduced Client and Server components allow for a broad variety of network configurations. In this section, we present an example scenario for the classroom that utilizes these components in different ways. Server and Client components can exist on any device in a heterogeneous network. Multiple Client and Server components may even exist within the same component hierarchy on a single device. This raises several interesting possibilities. For example, one can have a centralized simulation running on a single powerful machine, with less powerful devices connecting to this device to only visualize (not compute) smaller portions of the simulation. Distributed simulation is also possible, with multiple computers computing different pieces of the simulation. In another example multiple handheld devices are connected to a simulation that is run on a powerful shared device, where each handheld controls different portions of that simulation.

We developed an example scenario around a physics-based blood clotting simulation. Figure 4 shows a screenshot of the simulation run by the LINDSAY Composer on a desktop computer. One can see a clot forming over the breach in the vessel wall.

A. Distributed Computation

In order to increase the scale of the simulation, the introduced network components can be used to establish a distributed computing network. Figure 5 presents a network configuration in which different parts of a simulation are

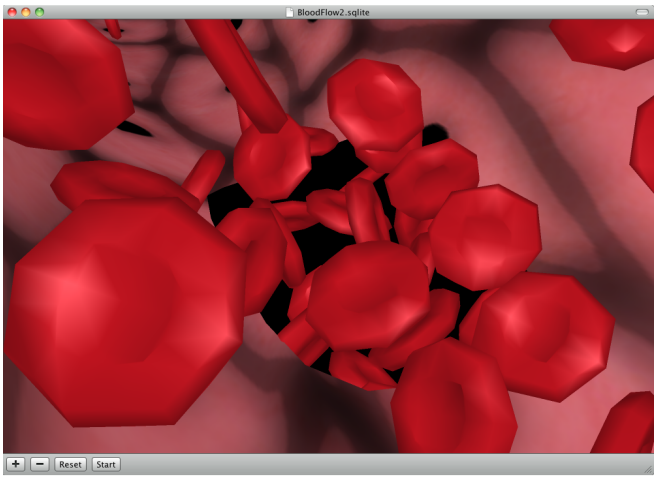

Fig. 4. Screenshot of a blood clotting simulation run with the LINDSAY Composer. Red blood cells are streaming through a blood vessel.

computed on different machines. In addition, one machine is dedicated to retrieve and aggregate the information of the separately computed parts, which are then visualized in a single simulation context. Figure 5(a) shows a conceptual diagram of the network configuration: Three Client components load their data into the same simulation context for visualization. The screenshots in Figures 5(b-d) illustrate those three independent simulations running on separate machines. Figure 5(e) shows the resulting combined visualization in one simulation context.

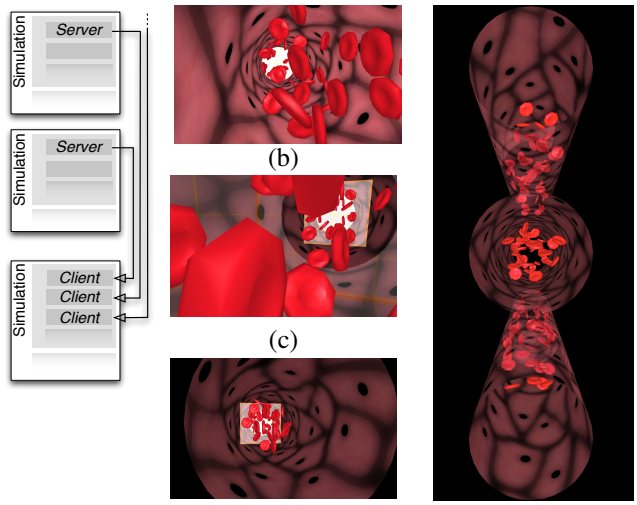

Fig. 5. (a) One computer receives data sent from three other machines and integrates it into one simulation context. (b-d) Snapshots of the simulation processes on the server machines. (e) The visualization of the merged simulation data on the client machine.

B. Distributed Learning

Currently, a room for demonstration and development purposes is setup for the LINDSAY project that is similar in size to a medium-sized classroom (ca. 20 students). Simulations of human anatomy are projected onto a large backlit projection screen. This setup is designed so that one lecturer can teach the students while guiding them through various simulation contents. The Server/Client component tandem can, however, add a new dimension of student involvement to the classroom experience. In particular, students can connect with their wireless handhelds, cell phones, laptops or tablets and explore the shared simulation space by themselves. This means that each student could take a different perspective of following the shared simulation. For our example scenario, students could observe the blood clotting from inside or outside the blood vessel, hop onto a blood cell and follow the action, while the rest of the class simply join the instructor's perspective. Alternatively, wireless devices can be used as remote controls for steering the actual scene presented on the screen—whether an inquiring student or a guiding teacher controls the simulation remotely would depend on the educational context and the stage of the class.

Figure 6(a) illustrates the network configuration for such a classroom setup. The projected simulation (top box) streams simulation data to a number of wireless clients. Among them is an iPod that is used as a remote control (middle box) whose directions are fed back into the simulation. As complex physics simulations are typically not suited for handheld devices, we only send the visualization data of the simulation to the iPhone/iPod clients (Figure 6(b)). In Figure 6(c), an on-screen joystick is drawn on top of the visualization of the simulation. It can be used to control the camera on the iPhone. In particular, the virtual joystick can be used to move the camera forward, backward, left and right. Additionally, the camera can be rotated by touching on the screen and dragging the fingers in the desired direction—up/down pitches the camera, whereas left/right rotates the camera around its y-axis.

Since the camera and its transformation is transferred to the workstation, any change to the camera on the iPhone will change the camera on the workstation. Hence, the camera of the main simulation running on the workstation can be controlled using an iPhone, which would typically be used by the course instructor.

V. Summary & Future Work

We introduced the LINDSAY Composer as a generic, component-based simulation platform. Its primary purpose is the simulation and visualization of complex biomedical systems for teaching in medical education. We present Client and Server components that can be embedded in the hierarchical data structure of simulation within the LINDSAY Composer. These networking components can be easily used to send and receive parts of a simulation over a network. Since multiple Server and Client components can be integrated into one system, and since they are able to handle arbitrary component data, complex networking scenarios are possible.

With several examples, we have demonstrated the flexibility and simplicity of using Server/Client components to implement interesting networking scenarios. These scenarios are designed for, but not limited to a classroom context. In

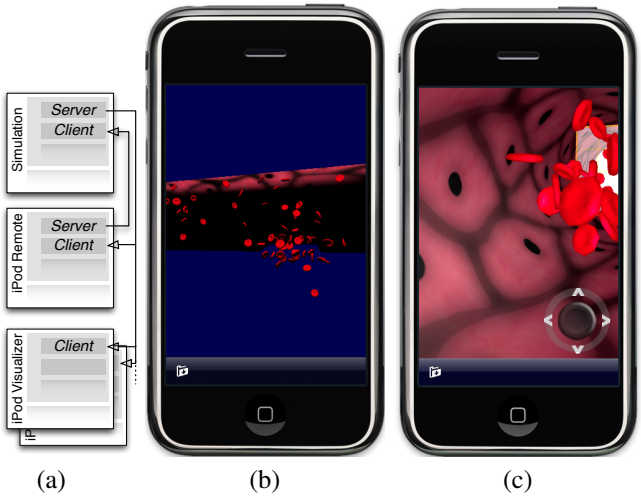

Fig. 6. (a) A network configuration for an interactive classroom: A shared simulation is presented on individual wireless devices. Information from one of these devices is fed back into the simulation. (b) Visualization of the simulation on an iPhone—as described in Section III-A, Physics components are not transferred. (c) On a second iPhone, a virtual joystick (in the bottom right corner) is used to guide the camera.

particular, we showed how a complex physics-based simulation can be computed on several machines and how the results can be visualized by one client. Filtering out Physics components from the transmitted simulation data renders it possible to have complex simulations visualized on wireless devices with limited computational power. In particular, we stream visualization data from a blood clotting simulation to a set of iPhones/iPods that are used by students to individually explore a shared simulation. By adding a Server component to a wireless device, it can be used as a remote control and feed data back into another system. In an example, we used an iPhone to direct the camera of a remote simulation.

The combination of a component-based simulation framework and easy-to-use networking components allows for a vast number of possible networking scenarios. In addition to the presented examples, students could use networked devices to simultaneously manipulate the data of a shared simulation. In this context, the idea of turning iPhones into augmented reality devices might prove invaluable: depending on the environment captured by an iPhone's camera, additional information displayed on the handheld devices could overlay the projected simulation. Mixed-reality elements could be introduced, for instance, using the overlay technique to display simulation data in the context of a human body.

In addition to exploring new technological prototypes for interactive classroom settings, an easy-to-use graphical user interface will have to be provided so that lecturers can quickly design and setup distributed simulation scenarios. The Client/Server components should be extended by an authentication mechanism. This could help to maintain confidentiality about certain course material, which is required, for instance, in the context of human corpses. Authentication would also empower the instructor to individualize the learning experience of the students, e.g., by assigning different means to manipulate simulation data, or by giving access to different supplementary data sets. Associating specialized configurations with handheld devices could also support group work.

Acknowledgment

This research was supported by the Office of Undergraduate Medicine of the University of Calgary, Alberta, Canada.

References

[1] G. T. Leavens and M. Sitaraman, Eds., *Foundations of component-based systems*. New York, NY, USA: Cambridge University Press, 2000.
[2] G. T. Heineman and W. T. Councill, *Component-Based Software Engineering*. ACM Press, 2001.
[3] D. Ebert-May, C. Brewer, and S. Allred, "Innovation in large lectures: Teaching for active learning," *BioScience*, vol. 47, no. 9, pp. 601–607, 1997.
[4] M. J. Ackerman, T. Yoo, and D. Jenkins, "From data to knowledge - the visible human project®continues," *MEDINFO 2001*, pp. 887–890, 2001. [Online]. Available: http://iospress.metapress.com/content/9008x58feh87ft0d
[5] M. F. Seifert, "Visible human projects special issue," *Clinical Anatomy*, vol. 19, no. 3, pp. 191–289, 2010.
[6] N. Mitsuhashi, K. Fujieda, T. Tamura, S. Kawamoto, T. Takagi, and K. Okubo, "BodyParts3D: 3D structure database for anatomical concepts," *Nucl. Acids Res.*, p. gkn613, 2008. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/gkn613v1
[7] C. W. Sensen and J. Soh, *Advanced Imaging in Biology and Medicine*. Berlin, Heidelberg: Springer, 2009, vol. II, ch. CAVEman, An Object-Oriented Model of the Human Body, pp. 289–300.
[8] I. Burleigh, G. Suen, and C. Jacob, "Dna in action! a 3d swarm-based model of a gene regulatory system," in *ACAL 2003, First Australian Conference on Artificial Life*, Canberra, Australia, 2003.
[9] C. Jacob, A. Barbasiewicz, and G. Tsui, "Swarms and genes: Exploring λ-switch gene regulation through swarm intelligence," in *CEC 2006, IEEE Congress on Evolutionary Computation*, 2006.
[10] D. A. Tietze, "A framework for developing component-based cooperative applications," Ph.D. dissertation, Technische Universität Darmstadt, Darmstadt, Germany, 2000.
[11] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riß, C. Sandor, and M. Wagner, "Design of a component-based augmented reality framework," in *International Symposium on Augmented Reality*, Los Alamitos, CA, USA, 2001, p. 45.
[12] C. Stoy, *Game Programming Gems 6*. Charles River Media, 2006, ch. Game Object Component System, pp. 393–403.
[13] (2007, September) Entity systems are the future of mmog development. [Online]. Available: http://t-machine.org/index.php/2007/09/03/entity-systems-are-the-future-of-mmog-development-part-1/
[14] (2002, July) Game object structure: Inheritance vs. aggregation. [Online]. Available: http://gamearchitect.net/Articles/GameObjects1.html
[15] (2007, January) Evolve your hierarchy. [Online]. Available: http://cowboyprogramming.com/2007/01/05/evolve-your-heirachy/
[16] S. M. Lewandowski, "Frameworks for component-based client/server computing," *ACM Comput. Surv.*, vol. 30, no. 1, pp. 3–27, 1998.
[17] K. Årzén, A. Bicchi, G. Dini, S. Hailes, K. Johanesson, J. Lygeros, and A. Tzes, "A component-based approach to the design of networked control systems," *European Journal of Control*, vol. 13, no. 2, pp. 261–279, 2007.
[18] M. Mikic-Rakic, S. Malek, and N. Medvidovic, "Improving availability in large, distributed component-based systems via redeployment," *Component Deployment*, pp. 83–98, 2005. [Online]. Available: http://dx.doi.org/10.1007/11590712_7
[19] H. Roussain and F. Guidec, "Cooperative component-based software deployment in wireless ad hoc networks," *Component Deployment*, pp. 1–16, 2005. [Online]. Available: http://dx.doi.org/10.1007/11590712_1
[20] E. de Lara, D. S. Wallach, and W. Zwaenepoel, "Puppeteer: Component-based adaptation for mobile computing," in *USENIX Symposium on Internet Technologies and Systems*, 2000.
[21] G. Paun and G. Rozenberg, "A guide to membrane computing," *Theoretical Computer Science*, vol. 287, no. 1, pp. 73–100, 9 2002/9/25.
[22] (2010, March) Apple bonjour website. [Online]. Available: http://www.apple.com/support/bonjour/
[23] *Foundation Framework Reference*, Apple Inc., Cupertino, CA, 2009.