Title
Service-based Systems in Clinical Environments

Author
Stantchev, Vladimir

Publication Date
2008-04-01

Service-based Systems in Clinical Environments

By
Vladimir Stantchev (vstantch@icsi.Berkeley.edu)
International Computer Science Institute
University of California, Berkeley

UCB iSchool Report 2008-023
April 2008

Abstract:
In this report we present an architectural approach to add quality-of-service (QoS) assurance and location awareness to service-based systems within existing clinical infrastructures. To address typical design requirements of such systems (e.g., cooperating services, performance and availability) the work proposes a service-oriented architecture (SOA) as architectural concept and architectural translucency to provide stable QoS. We evaluate position sensing systems, QoS assurance approaches and present design principles for service-based health care applications. Furthermore, we present a clinical application scenario and an architectural approach to integrate existing infrastructure into a human centric assistance system.

Keywords:
public services, quality of service, health care, location awareness

Contents

Chapter 1

Introduction

Services in clinical environments typically take place under high workloads and have to handle many different problems in a very short time. Digital assistance through software systems and electronic devices is able to reduce administrative workload and free physician and nurses for their core competence, taking care of patients. Examples for such technologies are hospital information systems (HIS) and electronic patient records (EPR) or electronic health records (EHR) [7] which simplify the access to patient data and medical information. Furthermore, there are devices that can provide information about the location of patients, staff and medical devices in the hospital. Such localization can be done via different technologies, for example Ultra Wide Band (UWB), Bluetooth (BT) or Wireless LAN (WLAN) location applications. UWB has the advantage that it works independently of other systems, allows very precise location and sends with a very low signal strength. WLAN localization on the other hand can be done in existing WLAN networks and allows the localization of computers, such as handhelds and laptops. Hybrid approaches that use two or more such technologies typically provide higher precision and more robust position sensing [10]. Handhelds and laptops are widely used in hospital environments. They enable mobile access to hospital information systems and patient records. We can improve the access to patient data by combining the localization with the data from the HIS and EPR. A doctor can use a laptop while doing the ward round to display only information that is relevant to the patients of the room he is currently in. This would prevent the physician from searching for the patient and therefore save time.

A service-oriented architecture (SOA) is an emerging paradigm to offer functionality in such distributed environments [30].

A key requirement when using SOA in such scenarios is the assurance of acceptable service levels of Quality of Service (QoS) parameters. QoS parameters are part of the nonfunctional properties (NFPs) of a service, typically specified in service level agreements (SLAs). We distinguish between run-time related and design-time related NFPs. Run-time related NFPs are performance oriented. Examples are response time, throughput, availability. Design-time

related NFPs such as language of service and compliance are typically set during design time and do not change during runtime. Run-time related NFPs can change during runtime when service usage patterns differ (times of extensive usage by many users are followed by times of rare usage), or when failures occur. Such failures can occur within the service, as well as in the network components that lie between user and service.

Formalization and specification of NFPs and their SLAs is currently a very active research field (see Section 3). The enforcement of these levels for runtime-related NFPs cannot be done automatically *a priori*, due to the changes in service usage and network availability. An approach to dynamically adapt service performance to these changes can ensure continuous meeting of service levels. This approach should employ service reconfiguration at runtime, as changes in source code of a service are not a feasible option. One approach to identify possible reconfigurations in a SOA and evaluate their implication is called architectural translucency [38].

This report evaluates position sensing techniques, how we can use them in healthcare scenarios, and how we can employ architectural translucency to ensure QoS for such scenarios. Furthermore, we present an application that demonstrates the concept.

The rest of this report is structured as follows: in Chapter 2 we describe related work for similar application scenarios. In Chapter 3 we present technologies for position sensing and our architectural approach to assure QoS in such environments. In Chapter 4 we present a case study clinical application. Useful insights and our future activities are outlined in Chapter 5.

Chapter 2

Related Work

One application that combines location-aware access with HIS is described in [33]. It is implemented using agent-oriented middleware (SALSA) [44]. As the implementation is PDA-based (device-based position calculation), neural networks were used to implement position sensing. A key shortcoming is the insufficient precision for delivering relevant patient information.

A concept of context-aware computing in health care is presented in [6]. It includes a scenario with scenes such as *Entering an Active Zone*, the *Context-aware Hospital Bed* and the *Context-aware Hospital EPR*. Key lessons learned are:

- Context-awareness can improve targeted data access to EPRs by clinicians.
- Access to physical objects (e.g., container, x-ray image, wheelchair, bed) reveals activity. Based on information about this access we can present proper data to support this activity.
- We can use context-awareness to suggest courses of action, not to automatically react to context changes.

There are several design principles for runtime infrastructures presented in [6]. These include:

- Distributed and Cooperating Services (e.g., a SOA),
- Security and Privacy,
- Lookup and Discovery (e.g., provided by foundation services in SOA), as well as basic design principles such as
- Performance and Availability.

The application we present applies these design principles by employing a SOA.

Chapter 3

Technology and Architecture

In the first section of this chapter we describe technologies for position sensing and which of them we are using in our application. In the following sections we define architecture as applied to service-based software systems and present our approach for QoS assurance.

3.1 Position Sensing Technologies

Several research projects and commercial products are offering positioning using WLAN or other wireless technologies. Depending on the system and used technology different levels of precision can be reached. The Horus system developed at the University of Maryland [46] and the RADAR system by Microsoft Research [5] were one of the first viable efforts to provide WLAN-based position sensing. They are using a WLAN signal strength map for positioning and reach precision of less than three meters. A downside is that they are requiring huge initial effort to set up the signal strength map. A similar system is being developed and offered by Ekahau [16]. These systems work only in areas where signals from enough WLAN access points are received (at least three) and need to be recalibrated in case the infrastructure is changed. The Place Lab system by Intel Research [24] is pursuing an opposite approach minimizing the needed initial effort. The system provides WLAN positioning in a whole city. The needed data is gathered by "war driving" (driving around in a car and collect WLAN access point signals). The achieved precision is much less but the system is more prone to changing infrastructure.

Systems based on RFID technologies or BT for positioning are offering slightly better precision [28, 19, 4] but cannot be built on existing infrastructure. Additional RFID tags and readers, or BT infrastructure have to be installed.

Hybrid positioning systems use two or more wireless communication technologies for position sensing. One example is MagicMap [10] where WLAN, RFID and ZigBee [3] are used. MagicMap delivers 33% better precision when at least two technologies are available.

3.2 Software System Architecture

The term *architecture* is intuitively associated with what humans see and experience. From such perspective people understand architecture as physical structures and their construction to form a coherent whole. Beside this external view there is the view of the building architect. This perspective allows the architect to provide the required properties in his building – shelter, light, heat, safety, accessibility, etc. [2].

A general definition of *software architecture* is as a structure composed of components, and rules for the interaction of these components [25]. Other definitions include the view of software architecture as components, connections, constraints and rationale [8], as elements, form and rationale [31], and as components, connectors and configurations [22]. All these definitions represent more or less the external view of a software system. In order to account for the inner view of the architect a *software systems architecture* should include also a collection of stakeholder-need statements and a rationale that demonstrates that the external view (components, connections and constrains) can satisfy these statements [21].

The usage of managed environments to facilitate the development of distributed applications leads to transparency of component location – the developer can write a procedure call once and the environment takes care to find the needed component (local or remote) and forward the call to it. Such transparency makes the task of writing distributed applications easier, but rarely accounts for typical NFP constrains related with remote calls, e.g., the need to limit the number of such calls or to process them asynchronously.

3.3 Service-based Environments

Web services are emerging as a dominating technology for providing and combining functionality in distributed systems. As shown in Figure 3.1 service-based environments are not something fundamentally new, but are rather an evolutionary step. Thus, the transparency of location and the NFP-related problems that we can observe in other distributed environments are also inherent to SOA. Furthermore, the SOAP-based interchange paradigm adds a huge performance overhead. Not only is the message size considerably larger, but every web service request using complex data types involves a serialization of object status by the requesting service before the request is submitted and its deserialization by the service that then processes this request.

A SOA is defined by its components (services), connections and constrains (what other service can consume them and how). Hereby, it offers native capabilities, such as publication, discovery, selection and binding. An extended view of SOA (see Figure 3.2) [29] also allows to specify different roles involved in the provision of services. Since services are basic building blocks for the creation of new applications, the area of composite services is introduced on top of native capabilities. It governs the way applications are developed from basic services.

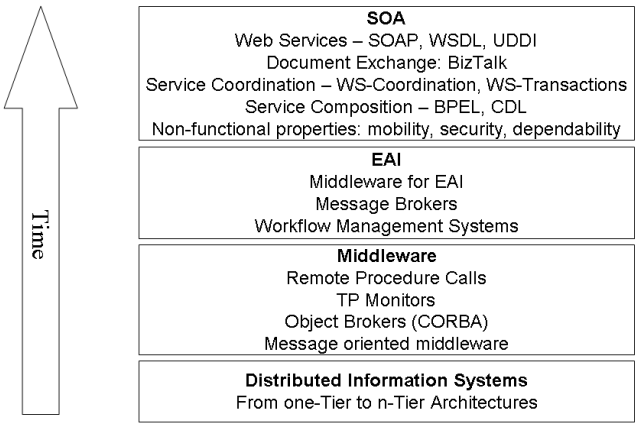

Figure 3.1: Development of Distributed Computing Architectures

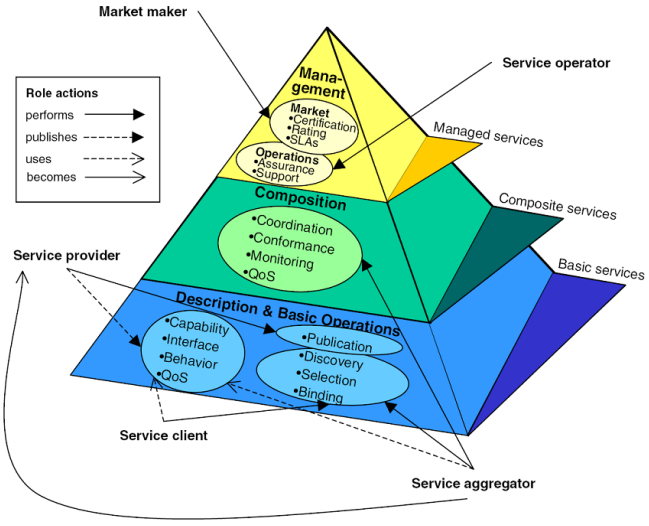

Figure 3.2: Extended Service Oriented Architecture

There are two basic aspects of a successful service offering: to provide the needed functionality and to provide the needed Quality of Service (QoS). QoS parameters are part of NFPs of a service, typically specified in service level agreements (SLAs). We distinguish between runtime-related and design-time-related NFPs. Runtime-related NFPs are performance oriented. Examples are response time, throughput, availability. Design-time-related NFPs such as language of service and compliance are typically set during design time and do not change during runtime. Runtime-related NFPs can change during runtime when service usage patterns differ (times of extensive usage by many users are followed by times of rare usage), or when failures occur. Such failures can occur within the service, as well as in the network components that lie between user and service. NFPs and QoS are regarded (together with semantics) as topics that encompass all three levels of services within an SOA (basic services, composite services, managed services) [30].

3.4 Quality of Service

Much work has been done in the area of QoS-aware web service discovery [26], QoS-aware platforms and middleware [45], and context-aware services [43].

Extensive research concerning assurance of NFPs exists in the field of CORBA (Common Object Request Broker Architecture), particularly in the areas of real-time support [32], replication as approach for dependability [17] as well as adaptivity and reflection [14]. There are various efforts to address NFPs in distributed computing environments that look at different system levels. Under the term performability, originally introduced in 1997 [12], there are works in the area of resource control [37] and addressing fault-tolerance at system level and application level [23].

Furthermore, there are architectural approaches that aim to improve NFPs in one location, e.g., reliability at the OS level [42, 41], scalability by clustering of web servers [20] or email servers [35], as well as introducing software RAID approaches [9].

Failures at the network level lead to network partitions. There is currently no convincing way to mathematically model network partitions [48]. Furthermore, it is NP-hard to derive a partition model from link and node failure models [34]. A convincing approach to incorporate network failures in availability metrics that define $Avail_{client} = Avail_{network} \times Avail_{service}$ is presented in [48]. A novel approach to further improve availability in such settings by better assignment of object replicas to nodes is presented in [47].

3.5 Architectural Translucency

Functional composition and orchestration, as well as formalization and specification of NFPs and their SLAs are currently very active research fields. The enforcement of these levels for runtime-related NFPs cannot be done automati-

cally *a priori*, due to the changes in service usage and network availability. An approach to dynamically adapt service performance to these changes can ensure continuous meeting of service levels. Providing such dynamically reconfigurable runtime architectures is regarded as one of the main research challenges in the area of service foundations [30]. Such approach should employ service reconfiguration at runtime, as changes in source code of a service are not a feasible option.

The approach can be based on experimental computer science [15, 18]. Experimental computer science uses experiments to enhance theory in two general ways – to confirm or refute theory predictions and to find new phenomena [1]. Thereby an experimental approach needs three key building blocks – a *hypothesis* to be tested, an *apparatus* to be measured, and systematic *analysis of the data* to see whether it supports the hypothesis [15].

One approach to dynamically adapt service performance to changes in load and usage is called architectural translucency [38]. It can ensure continuous meeting of service levels by employing service reconfiguration at runtime, as changes in source code of a service are not a feasible option.

Chapter 4

Application in Clinical Environments

Our application scenario focuses on the surgical sector – one of the largest cost factors in health care and at the same time a place, where high creation of value takes place.

For the effective utilization of the surgical sector pre- and postoperative processes are crucial. The surgery (equipments and specialists) is a fixed (and expensive) resource. So pre- and postoperative processes need to be aligned and provide for an optimized utilization of this resource.

4.1 Perioperative and Postoperative Processes

The perioperative processes start with a notification from an operating room nurse or an anesthesia nurse, that the next patient should be transported to the operating room. Then the patient is moved by a transport service or a nurse from the ward to the operating room area. In the main registration area the patient is transferred from the ward bed to an operating room table. Afterward the patient is taken to the induction area, where the patient is anesthetized. Then the patient is moved to the operating room, where the preparation for the operation starts, for example operation specific bedding, sterile coverage etc. The surgery starts with the cut and finishes with the suture. After the surgery the patient is transported to the post anesthesia recovery unit, where he is moved again to the ward bed and recovers from anesthesia. After the recovery the patient is transported back to the ward.

There are many visions how to redesign and reorganize perioperative patient flow and work processes for maximum operating room productivity, which also bring changes in operating room architecture. Figure 4.1 [36] shows a ground plan and flow diagram of patient movement through the operating room of the future. Patients are brought from the main registration area (1 and arrow) to the induction area (2). Perioperative preparation and induction of anesthesia

occur in the induction area (2), concomitantly with instrument setup taking place in the operating room (3). The sequence is timed so that anesthetized patients are transferred to the operating room (2, arrow, and 3) for surgery as instrument setup is completed. At the conclusion of surgery, patients emerge from anesthesia in the operating room and are promptly transferred to the early recovery area (3, arrow, and 4), or emergence occurs in the early recovery area. After approximately 15 min of recovery, patients are transferred to the postanesthesia care unit (4 and arrow) by the perioperative nurse. The work space provides access to the hospital information system. It is used by surgeons between cases for dictation, order writing, and teleconsultation with patients' families and by the anesthesia team during surgery for perioperative planning for subsequent cases.

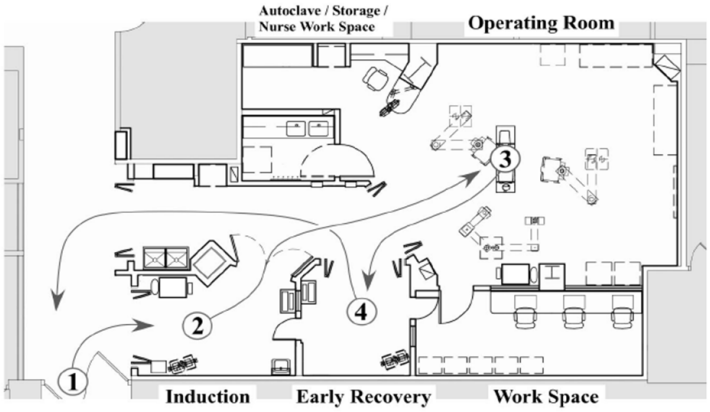

Figure 4.1: Example for Optimized Process Flow in Surgery.

A different setting where position sensing is used to better plan and control resources during the perioperative and postoperative processes is described in [39, 40].

While there the focus is on process optimization and technology selection process for WLAN-based positioning systems, here, our focus lies on the provision of data in such environment. Our objective is to use the existing infrastructure (WLAN, RFID, HIS) to allow for clinicians to access EPRs from the HIS in a context-aware way.

4.2 Service-oriented Technology Environment and Architectural Approach

Our WLAN positioning system is Ekahau [16], our HIS is FD Klinika [13], we are using Tablet PCs as mobile devices for the clinicians. Our architectural approach (see Figure 4.2 is to use an SOA with wrappers that provide web service interfaces to the enterprise service bus (ESB) [11]. The AT engine is responsible for service QoS assurance by monitoring and management. When it notices that for example a service is experiencing higher loads, it dynamically reconfigures the replication settings of the service to further provide the expected QoS. Integration of other systems (e.g., enterprise resource planning, external partner applications, health insurance systems). Representation and further information processing are depicted in the upper part of the figure. During the first stage (depicted green) we plan to provide portal-based access to EPRs that are extracted from the HIS and visualized on the Tablet PC depending on the current location of the Tablet PC, particularly other WLAN-enabled objects surrounding it (e.g., patient tags). During the second stage (depicted red) we plan to introduce more complex planning and evaluation functions. These will be realized by composite services.

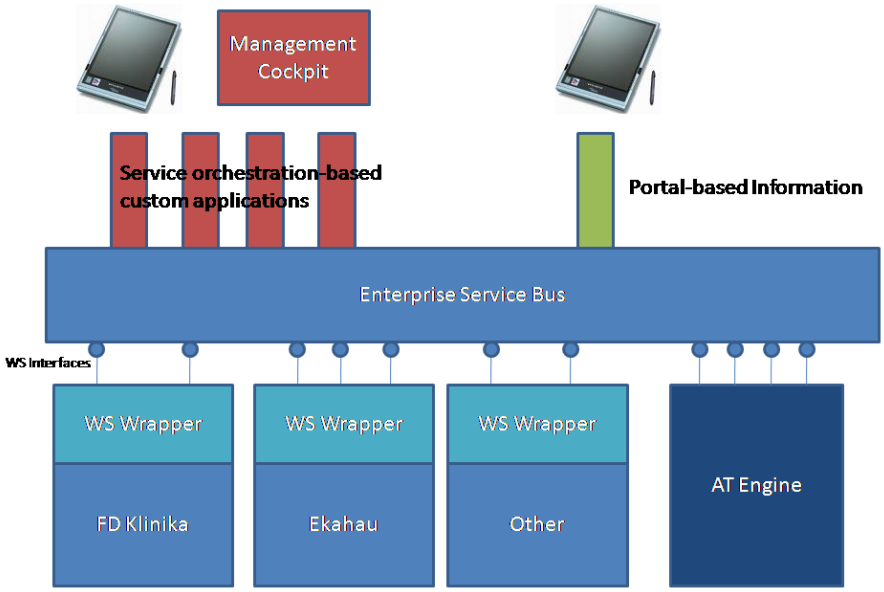

Figure 4.2: Systems, Technologies and Architecture for Service Delivery in Clinical Environments.

Chapter 5

Conclusion

In this report we present an architectural approach to enhance to an existing clinical infrastructure with QoS assurance and location awareness. To address typical design requirements of such systems (e.g., cooperating services, performance and availability) we propose SOA as architectural concept and architectural translucency to provide stable QoS. Location awareness is provided by an *off-the-shelf* WLAN-based positioning system. Open issues that we are currently addressing include the standardization of healthcare-related services within an ESB and service priorities (e.g., wireless transmitted alarms regarding vital data) within a limited resource environment.

Bibliography

[1] Experimental computer science. *Commun. ACM*, 50(11):24–26, 2007.

[2] C. Alexander. *Notes on the Synthesis of Form*. Harvard University Press, 1970.

[3] ZigBee Alliance. ZigBee Specification. *hup://www.zigbee.org*.

[4] G. Anastasi, R. Bandelloni, M. Conti, F. Delmastro, E. Gregori, and G. Mainetto. Experimenting an indoor bluetooth-based positioning service. *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 480–483, 19-22 May 2003.

[5] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM*, pages 775–784, 2000.

[6] Jakob E. Bardram. Applications of context-aware computing in hospital work: examples and design principles. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1574–1579, New York, NY, USA, 2004. ACM.

[7] Richard J. Baron, Elizabeth L. Fabens, Melissa Schiffman, and Erica Wolf. Electronic Health Records: Just around the Corner? Or over the Cliff? *Ann Intern Med*, 143(3):222–226, 2005.

[8] B. Boehm. *Megaprogramming*. University Video Communications Stanford, CA, 1994.

[9] A. Brown and D.A. Patterson. Towards Availability Benchmarks: A Case Study of Software RAID Systems. *Proceedings of the 2000 USENIX Annual Technical Conference*, 2000.

[10] Stefan Bruning, Johannes Zapotoczky, Peter Ibach, and Vladimir Stantchev. Cooperative positioning with magicmap. *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, pages 17–22, March 2007.

[11] D. Chappel. *Enterprise Service Bus*. O'Reilly, 2004.

[12] C.M.Krishna and Kang.G.Shin. *Real-Time Systems*. The McGraw-Hill Companies, Inc, New York, 1997.

[13] Fliegel Data. FD Klinika. *http://www.fliegel-data.de/fd-online/*, 2007.

[14] Pierre-Charles David and Thomas Ledoux. An infrastructure for adaptable middleware. In Meersman and Tari [27], pages 773–790.

[15] Peter J. Denning. Acm president's letter: What is experimental computer science? *Commun. ACM*, 23(10):543–544, 1980.

[16] Inc. Ekahau. *Ekahau Positioning Engine 3.1.* http://www.ekahau.com/file.php?id=129, 2005.

[17] Pascal Felber and Priya Narasimhan. Reconciling replication and transactions for the end-to-end reliability of corba applications. In Meersman and Tari [27], pages 737–754.

[18] Jerome A. Feldman and William R. Sutherland. Rejuvenating experimental computer science: a report to the national science foundation and others. *Commun. ACM*, 22(9):497–502, 1979.

[19] S. Feldmann, K. Kyamakya, A. Zapater, and Z. Lue. An indoor Bluetooth-based positioning system: concept, implementation and experimental evaluation. *International Conference on Wireless Networks*, pages 109–113, 2003.

[20] Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer, and Paul Gauthier. Cluster-based scalable network services. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 78–91, New York, NY, USA, 1997. ACM.

[21] C. Gacek, A. Abd-Allah, B. Clark, and B. Boehm. On the Definition of Software System Architecture. *Proc. of ICSE 17 Software Architecture Workshop*, 1995.

[22] D. GARLAN and M. SHAW. AN INTRODUCTION TO SOFTWARE ARCHITECTURE. *Advances in Software Engineering and Knowledge Engineering*, 1993.

[23] Joshua Haines, Vijay Lakamraju, Israel Koren, and C. Mani Krishna. Application-level fault tolerance as a complement to system-level fault tolerance. *The Journal of Supercomputing*, 16(1-2):53–68, 2000.

[24] J. Hightower, A. LaMarca, and I.E. Smith. Practical lessons from place lab. *Pervasive Computing, IEEE*, 5(3):32–39, July-Sept. 2006.

[25] AK Jones. The Maturing of Software Architecture. *Software Engineering Symposium, Software Engineering Institute, Pittsburgh, PA*, 1993.

[26] Y. Makripoulias, C. Makris, Y. Panagis, E. Sakkopoulos, P. Adamopoulou, M. Pontikaki, and A. Tsakalidis. Towards Ubiquitous Computing with Quality of Web Service Support. *Upgrade, The European Journal for the Informatics Professional*, VI(5):29–34, 2005.

[27] Robert Meersman and Zahir Tari, editors. *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 Irvine, California, USA, October 30 - November 1, 2002, Proceedings*, volume 2519 of *Lecture Notes in Computer Science*. Springer, 2002.

[28] M. Menz. RFID-basierte Positionsbestimmung. *Informatik, Humboldt Universität*, 2005.

[29] Michael P. Papazoglou. Service-oriented computing: concepts, characteristics and directions. *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 3–12, 2003.

[30] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, Nov. 2007.

[31] D.E. Perry and A.L. Wolf. Foundations for the Study of Software Architecture. *ACM SIGSOFT SOFTWARE ENGINEERING NOTES*, 17(4):40, 1992.

[32] A. Polze and M. Malek. Responsive computing with corba. *isorc*, 00:73, 1998.

[33] M.D. Rodriguez, J. Favela, E.A. Martinez, and M.A. Munoz. Location-aware access to hospital information and services. *Information Technology in Biomedicine, IEEE Transactions on*, 8(4):448–455, Dec. 2004.

[34] A. Rosenthal. Computing the Reliability of Complex Networks. *SIAM Journal on Applied Mathematics*, 32(2):384–393, 1977.

[35] Yasushi Saito, Brian N. Bershad, and Henry M. Levy. Manageability, availability, and performance in porcupine: a highly scalable, cluster-based mail service. *ACM Trans. Comput. Syst.*, 18(3):298, 2000.

[36] W.S. Sandberg, B. Daily, M. Egan, J.E. Stahl, J.M. Goldman, R.A. Wiklund, and D. Rattner. Deliberate perioperative systems design improves operating room throughput. *Anesthesiology*, 103(2):406–18, 2005.

[37] Kang G. Shin, C. M. Krishna, and Yann-Hang Lee. Optimal dynamic control of resources in a distributed system. *IEEE Trans. Softw. Eng.*, 15(10):1188–1198, 1989.

[38] V. Stantchev and M. Malek. Architectural Translucency in Service-oriented Architectures. *IEE Proceedings - Software*, 153(1):31–37, February 2006.

[39] Vladimir Stantchev, Tino Schulz, and Trung-Dang Hoang. Ortungstechnologien im op-bereich. In *Mobiles Computing in der Medizin (MoCoMed) 2007: Proceedings of the 7th Workshop on*, pages 20–33, Aachen, Germany, 2007. Shaker.

[40] Vladimir Stantchev, Tino Schulz, Trung-Dang Hoang, and Ilja Ratchinski. Optimizing clinical processes with position sensing. *IT Professional (to appear)*, Mar/Apr 2008.

[41] Michael M. Swift, Brian N. Bershad, and Henry M. Levy. Improving the reliability of commodity operating systems. *ACM Trans. Comput. Syst.*, 23(1):77–110, 2005.

[42] M.M. Swift, M. Annamalai, B.N. Bershad, and H.M. Levy. Recovering device drivers. *Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation*, 2004.

[43] Y. Tokairin, K. Yamanaka, H. Takahashi, T. Suganuma, and N. Shiratori. An effective qos control scheme for ubiquitous services based on context information management. *cec-eee*, 00:619–625, 2007.

[44] Carlos Varela and Gul Agha. Programming dynamically reconfigurable open systems with salsa. *SIGPLAN Not.*, 36(12):20–34, 2001.

[45] S.S. Yau, Yu Wang, Dazhi Huang, and H.P. In. Situation-aware contract specification language for middleware for ubiquitous computing. *Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of*, pages 93–99, 28-30 May 2003.

[46] Moustafa Youssef and Ashok K. Agrawala. Handling samples correlation in the horus system. In *INFOCOM*, 2004.

[47] Haifeng Yu and Phillip B. Gibbons. Optimal inter-object correlation when replicating for availability. In *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 254–263, New York, NY, USA, 2007. ACM.

[48] Haifeng Yu and Amin Vahdat. The costs and limits of availability for replicated services. *ACM Trans. Comput. Syst.*, 24(1):70–113, 2006.