2nd Joint Workshop
on
High-Confidence Medical Devices, Software and Systems
(HCMDSS)
and
Medical Device Plug-and-Play Interoperability
(MD PnP)

Affiliated with the CPS Week

San Francisco
April 16, 2009

Preface

This volume contains technical papers presented at the second Joint Workshop on High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability, held as part of the Cyber-Physical Systems Week on April 16, 2009 in San Francisco, CA.

The workshop series is based on the idea that bringing together the HCMDSS (High Confidence Medical Devices, Software, and Systems) and MD PnP (Medical Device "Plug-and-Play" Interoperability) communities of medical device specialists (researchers, developers, clinicians, regulators, and policy makers) from clinical environments, industry, research laboratories, academia, and government, would accelerate the development of science, technology, and practice to overcome crucial challenges facing the design, manufacture, certification, and use of medical devices. More reliable medical devices and interoperability standards will form the building blocks for patient-centric integrated medical device systems to improve the safety and efficiency of healthcare in diverse clinical settings.

The previous joint workshop was held on June 25-27, 2007, in Cambridge, MA. One-hundred-forty-five attendees from academic and clinical organizations, government and industry, demonstrated their commitment to the vision and goals of the conference over three information-rich days. Prior meetings on HCMDSS and MD PnP were held separately: the first HCMDSS workshop was held in June 2005 in Philadelphia, PA, and the MD PnP Interoperability program has held plenary meetings in May 2004, in November 2004 (at the FDA), and in June 2005. The synergies between the HCMDSS and MD PnP goals led to the joint workshop series to continue the momentum produced by the prior meetings, and to provide a forum to exchange new research and development results by the emerging community of researchers, developers, regulators, users, and manufacturers. (See **Background** section below for further information on both programs.)

Workshop Organizers:

Julian M. Goldman, MD	Massachusetts General Hospital
Insup Lee	University of Pennsylvania
Oleg Sokolsky	University of Pennsylvania
Susan Whitehead	CIMIT

Background

High Confidence Medical Devices, Software & Systems (HCMDSS). The development and production of medical device software and systems is a crucial issue, both for the US economy and for ensuring safe advances in healthcare delivery. As devices become increasingly smaller in physical terms but larger in software terms, the design, testing, and eventual Food and Drug Administration (FDA) device approval is becoming much more expensive for medical device manufacturers in terms of both time and cost. Furthermore, the number of devices that have recently been recalled due to software and hardware problems is increasing at an alarming rate. As medical devices are becoming increasingly networked, ensuring even the same level of health safety is a challenge.

Several federal and regulatory agencies have identified this growing problem and are interested in establishing a research agenda directed at improving the design, certification, and operation of current and future medical device software and systems. The 2005 High-Confidence Medical Device Software and Systems (HCMDSS) workshop was sponsored by various federal agencies, including the FDA, the National Institute of Standards and Technology, the National Security Agency, and the National Science Foundation, along with the National Coordination Office for Networking and Information Technology Research and Development.

The purpose of the first HCMDSS workshop was to provide a working forum for leaders and visionaries from industry, research laboratories, academia, and government concerned with medical devices. More than 90 experts from these sectors attended the workshop. They represented a mix of the relevant stakeholders — including researchers, developers, certifiers, and users — who can help identify emerging systems and assurance needs. The main goal of the workshop was to develop a road map for overcoming crucial issues and challenges facing the design, manufacture, certification, and use of medical device software and systems. An additional goal was to identify and form a sustainable research and development community for the advancement of HCMDSS. Of particular interest was the crystallization of technology needs and promising research directions that could revolutionize the way HCMDSS are designed, produced, and validated in the future but that are beyond the range of today's devices because of time-to-market pressures and short-term R&D practices. More information about the first HCMDSS workshop, including the presentations of the working groups, keynote speakers, and panelists, as well as the submitted position statements of participants in this workshop are available at www.cis.upenn.edu/hcmdss/.

Findings from the HCMDSS workshops provided the basis for the report by the Networking and Information Technology Research and Development (NITRD) Program, titled "High-Confidence Medical Devices: Cyber-Physical Systems for 21st Century Health Care." The NITRD Program is an interagency activity within the U.S. Federal Government that coordinates research and development (R&D) in the areas of advanced networking, computing, and related information technologies. The report, released in February 2009, establishes national priorities in medical device R&D.

MD PnP Program. Medical devices are essential for the practice of modern medicine. However, unlike the inter-connected "plug-and-play" world of modern computers and consumer electronics, most medical devices are designed to operate independently, and do not employ open networking standards for data communication or for device control. The integration of

individual medical devices into patient-centric networked systems can provide real-time comprehensive data for the electronic health record (EHR) and can create integrated clinical environments to support innovation in patient safety, workflow improvements, and reductions in medical errors and healthcare costs throughout the continuum of care: from the home, to pre-hospital transport, and to hospital areas as diverse as the OR, ICU, and general hospital ward.

The MD PnP program was established in 2004 to lead the evaluation and adoption of open standards and technology for medical device interoperability to support clinical innovation. The program is affiliated with Massachusetts General Hospital (MGH), CIMIT (Center for Integration of Medicine and Innovative Technology), and Partners HealthCare Information Systems, with additional support from TATRC (U.S. Army Telemedicine & Advanced Technology Research Center). Having evolved from the OR of the Future program at MGH, the MD PnP program remains clinically grounded. The program has a geographically dispersed, multidisciplinary, multi-institutional team of collaborators representing diverse stakeholder groups (clinicians, biomedical and clinical engineers, healthcare delivery systems, regulatory agencies, medical device vendors, standards development experts). Since the program's inception, more than 700 clinical and engineering experts and representatives of over 85 institutions that share a vision of medical device interoperability have participated in ongoing convening activities, including elicitation of clinical scenarios for improving healthcare through interoperability, implementation of selected clinical scenarios in a laboratory environment, and drafting an initial standard for the integrated clinical environment (ICE) required to support device interoperability (ASTM F-2761, being published in Spring 2009).

To support these goals, the CIMIT MD PnP Lab opened in May 2006 to provide a vendor-neutral "sandbox" to evaluate the ability of candidate interoperability solutions to solve clinical problems, model clinical use cases (in a simulation environment), develop and test related network safety and security systems, and support interoperability and standards conformance testing. In the Lab we are developing demonstrations of interoperability-based patient safety improvements, such as improving the safety and quality of portable x-rays, and patient-controlled analgesia systems that are used for pain management. Our team of collaborators includes participants from: Kaiser Permanente, Johns Hopkins, FDA, Univ. of Penn. Dept. of Computer and Information Science, Dräger Medical Systems, LiveData Inc., DocBox Inc., Moberg Research Inc., Univ. of Illinois at Urbana Champaign, Univ. of Waterloo, NIST, NSF, Geisinger Health System, as well as the Partners HealthCare System community (Massachusetts General Hospital Anesthesia, Biomedical Engineering at MGH and Brigham & Women's Hospital, PHS Information Systems, and PHS Materials Management).

Current activities include:

- Developing clinical scenarios to inform interoperability solutions, and refining methodologies to analyze clinical scenarios to derive engineering requirements
- Developing an ASTM International standard to define the "ecosystem" requirements of a patient-centric "Integrated Clinical Environment" (ICE), which includes system functions that could meet clinical, technical, regulatory, and legal requirements: data logging, data security, decision support, and connectivity to the hospital information system.
- Collaborating with the FDA and others to elaborate a regulatory pathway for patient-centric networked medical devices
- Sharing contract language to support the preferential acquisition of standards-conformant

systems by healthcare organizations (as developed by Kaiser Permanente, Johns Hopkins, and Massachusetts General Hospital / Partners HealthCare)
- Defining an open source platform to support ICE-compliant implementations of interoperability-driven clinical use cases.

Learn more at http://www.mdpnp.org (including a link to streaming video coverage of the June 2007 Joint Workshop on MD PnP and HCMDSS).

Program

8:00 - 8:15am	Welcome
8:15 - 9:15am	**Invited keynote address**
	David Whitlinger President & Board Chair, Continua Health Alliance Director, Healthcare Device Standards, Intel Corp.
9:00 - 10:00am	***Break***
10:00 - 11:30am	**HCMDSS**
10:00 - 10:10am	CPS projects in the HCMDSS domain *Insup Lee*
10:10 - 10:35am	Engineering High Confidence Medical Device Software *A. Ray, R. Jetley, P. Jones*
10:35 - 11:00am	A Tool for Designing High-Confidence Implantable BioSensor Networks for Medical Monitoring *S. Gupta*
11:00 - 11:15am	An Event Driven Framework for Assistive CPS Environments *F. Makedon, Z. Le, H. Huang, E. Becker*
11:15 - 11:30am	Safety Enhancements of Home Lift, Position, & Rehabilitation (HLPR) Chair *J. Zalewski, D. Guo, C. Csavina, J. Sweeney, R. Bostelman, K. Kirsner*
11:30 - 12:30pm	***Lunch***
12:30 - 2:30pm	**MD PnP**
12:30 - 12:40pm	MD PnP update *Julian Goldman*
12:40 - 1:05pm	A Concept for a Medical Device Plug-and-Play Architecture based on Web Services *S. Pöhlsen, S. Schlichting, M. Strähle, F. Franz, C. Werner*
1:05 - 1:30pm	A Publish-Subscribe Architecture and Component-based Programming Model for Medical Device Coordination and Integration *A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, S. Weininger*

1:30 - 1:55pm	A Modular Framework for Clinical Decision Support Systems: Medical Device Plug-and-Play is Critical *M. Williams, F. Wu, P. Kazanzides, K. Brady, J. Fackler*
1:55 - 2:10pm	Standards for Physiological Data Transmission and Archiving for the Support of the Service of Critical Care *J.M. Eklund and C. McGregor*
2:10 - 2:30pm	***Break***
2:30 - 4:00pm	Panel: **R&D Collaboration Opportunities in HCMDSS and MD PnP**
	Confirmed participants: *Helen Gill*, NSF *Paul Jones*, FDA *Lui Sha*, University of Illinois
4:00 - 4:30pm	***Break***
4:30 - 5:15pm	**Medical Networks**
4:30 - 4:45pm	Monitoring and Diagnosis of Networked Medical Hardware and Software for the Integrated Operating Room *S. Bohn, M. Lessnau, O. Burgert*
4:45 - 5:00pm	Wireless Health and the Smart Phone Conundrum *J. Woodbridge, A. Nahapetian, H. Noshadi, M. Sarrafzadeh, W. Kaiser*
5:00 - 5:15pm	Flexible RFID Location System Based on Artificial Neural Networks for Medical Care Facilities *H-J. Wu, Y-H. Chang, I-Ch. Lin, M-Sh. Hwang*
5:15 - 5:25pm	Wrap-up
5:25 - 6:00pm	***Break - Reception***
6:00 - 8:00pm	Clinical Requirements Session

Technical Session I

High-Confidence Medical Devices, Software, and Systems

Session Chair:

Insup Lee

University of Pennsylvania

Engineering High Confidence Medical Device Software

Arnab Ray
Fraunhofer Center for Experimental Software Engineering
ARay@fc-md.umd.edu

Raoul Jetley Paul Jones
US Food and Drug Administration, Center for Devices and Radiological Health
{raoul.jetley, paull.jones}@fda.hhs.gov

Abstract

The increasing complexity of medical device software has created new challenges in ensuring that a medical device operates correctly. This paper discusses how two technologies — model-based development and static analysis — may be used to facilitate the successful engineering of medical software and some possible regulatory side benefits.

Keywords model-based development, formal verification, static analysis, instrumentation based verification

1. Introduction

The amount of software present in medical devices has dramatically increased over the last decade. Many infusion pumps today contain tens of thousands of lines of code. This number can run into the millions for proton beam therapy devices. Software is considered by many to be easier to configure, change and re-use than hardware. It is a technology that enables robust device designs. The need for high-integrity software in the health-care industry has become more important than ever as remote surgery, intelligent operating rooms, autonomous assisted living environments, and bio-feedback based prosthetics become the norm in the not-so-distant future.

The increasing complexity of device software presents considerable engineering challenges. In 1998, close to 8% of device failures could be traced to software errors [5]. Currently, the number of device recalls due to software problems is believed by some to be about 18%. It is likely that device failures and subsequent recalls will continue to increase until software is better engineered.

Figure 1 depicts a generalized software development workflow process typically followed by device manufacturers. The quality of the code in this workflow process is generally ensured through verification activities such as manual inspections, code walkthroughs and testing. Integrated system testing typically takes place at the end of the development lifecycle. Such verification activities, in the context of a quality system, have historically been considered sufficient for developing quality software. However, history has shown that common practices within this workflow process are insufficient for developing highly dependable software [10]. The reason for this is that these largely human resource intensive activities simply cannot fathom, unaided, the interdependencies of complex requirements and code.

Some of the limitations associated with traditional software development techniques can be summarized as follows:

- No formal, mathematics-based verifiable relationship is established between the design and the code
- Without a formal relationship established, it is difficult to demonstrate that the design and the code conform to each other structurally as well as behaviorally
- Without a formal methods foundation, rigorous verification and validation results are difficult to demonstrate throughout the life-cycle process
- Without the use of statistically based testing methods code coverage is difficult to characterize objectively
- No formal relationship is established between system property requirements (e.g., safety, security, privacy, etc), code, and the test suite used to verify the software. As a result, there is no reliable way to ensure that the software addresses these property specific requirements.

The risks associated with current software development practices will likely increase as medical device cyber-physical systems[1] such as non-homogeneous interoperable medical devices begin to enter the health care system. Configurations of such devices will be highly variable and reconfigurable in order to provide support in operating rooms, in hospital rooms, and in home care environments. For example, during a surgical operation, a number of "off-the-shelf" medical devices may be networked together to monitor and safely react to a patient's changing physiology.

[1] The term cyber-physical systems refers to the tight conjoining of and coordination between computational and physical resources

HCMDSS 2009 April 16 2009, San Francisco.

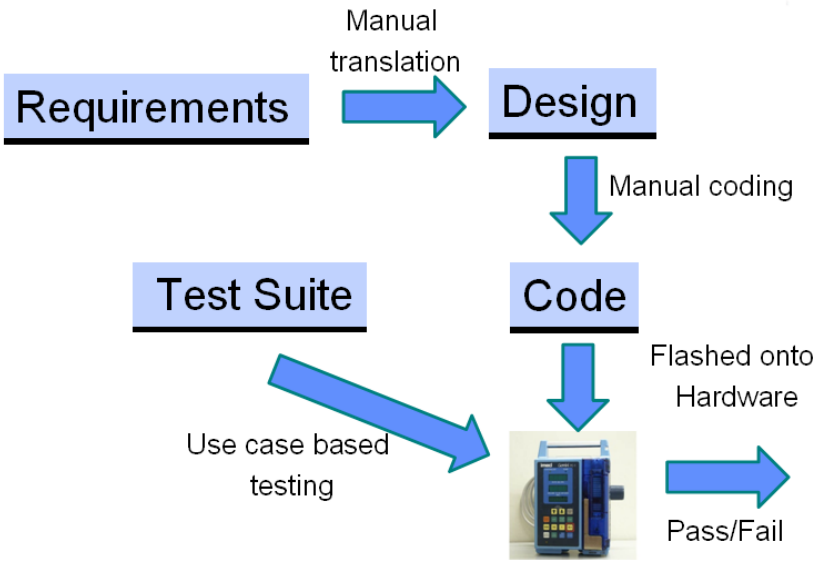

Figure 1. Traditional Software Development Workflow

Emerging medical device cyber-physical systems, such as interoperable medical device systems and prosthetics, bring new engineering challenges to the medical device development space in terms of scale, security, privacy, timing, human factors, composition, sensing, coordination, control, and certifiable evidence based verification and validation. Ultimately, research and development is needed to establish cyber-physical device composition and integration technologies and certifiable tool chains that address issues of logical and physical interoperability [8].

The remainder of this paper discusses some mathematically well-founded technologies for design, development and verification of medical device software. These techniques have been used with considerable success in other safety-critical industries such as aerospace and automotive engineering. In particular, we discuss model-based development and static analysis, and discuss how these technologies might be leveraged in a regulatory environment.

2. Model-based Development

A model can be thought of as a formal representation of a design specification. A model can also be used to capture the essential structure and behavior of a component or system. Of particular interest to developing high-confidence medical device software is the notion of ”executable modeling notations.” Executable modeling notations can be distinguished from conventional design notations by the fact that they are based upon a mathematically precise notion of what it means for a model to perform a behavioral action. This means that designs rendered in an executable modeling notation can be simulated and debugged just like normal code that has been written in a traditional programming language like C or C++.

The principal advantage of using executable models, hereafter referred to simply as models, over conventional programming languages is that they free the developer from implementation details like pointer management and memory allocation. This is analogous to the way programming languages abstract away low-level details of processor instruction sets, facilitating the separation of software and hardware concerns. Modeling makes it easier for the developer to focus resources on various aspects and properties of a particular design.

To verify models on a particular hardware platform they must be converted to code through a process called “automatic code generation.” Modeling tools are used to convert modeling notations into code. Compilers then transform this code into machine language that can then be executed on the hardware.

This kind of software development, where the model serves as the primary artifact, is often referred to as model-based development. Over the years, model-based development techniques have become standard practice in the production of high-integrity embedded software in the aerospace and automotive industries.

A major advantage of a model-based development workflow is that it facilitates catching and correcting errors early in the development lifecycle. Since models can be constructed much faster than code, designers can rapidly create prototypes of their system and study various design alternatives before committing to a final implementation. Also, owing to the executable nature and formal semantics of these models, various analytical verification and validation (V&V) methods like model-checking [4] or instrumentation-based verification (IBV) [1] can be used to formally prove

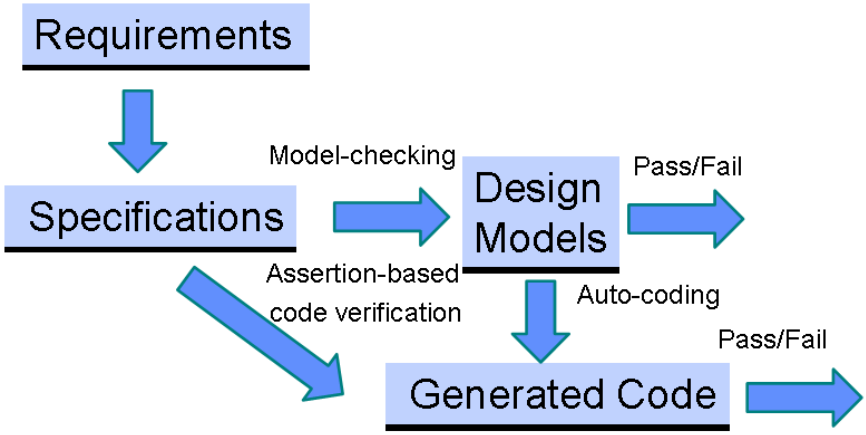

Figure 2. Model-checking based Verification Workflow

that the software design satisfies functional and specific property requirements (e.g. safety, security, etc).

When using model-based V&V techniques, natural language requirements are first converted to formal specifications, which may be expressed as either temporal logical formulae [2] or monitor models (*monitors* for short) [1]. Temporal logic formulae are typically employed for model-checking. Monitors may be thought of as encodings of idealized system behavior that are executed concurrently with the models to guarantee consistent results during IBV.

If model checking is used (as shown in Figure 2), then the logical specifications (or temporal logic formulae) are checked against finite-state representations of the design model using either sophisticated graph traversal or equation solving techniques. A model is said to be verified against a set of specifications, if for all possible model executions, it is not possible for any of the specifications to be violated. On the other hand, if a specification is violated by an execution trace, the model is deemed to be erroneous. (This execution trace is often generated by the model-checker as proof of the specification violation.)

If IBV is used as the V&V method of choice, as shown in Figure 3, the design model is first instrumented with monitor models. A test-generation engine is then used to check the composite of the design model and the monitor against a series of automatically generated tests. The aim here is to determine whether the actual behavior of the design model and the idealized behavior of the monitor instrumentation diverge from each other. In other words, the test-generation engine takes the role of a pessimistic observer and generates tests so as to "break" the design. If it is successful in observing a divergence between the design model and the monitor, it outputs the relevant test case as the rationale for why the specification is not satisfied.

The metric that specifies how extensively the model's behavior is covered by the tests is known as a coverage criterion. Various coverage criteria can be used to verify the model based on how rigorous the test cases need to be. For example, line coverage stipulates that each model element needs to be executed at least once for the test suite to be complete. Decision coverage, on the other hand, enforces that boolean expressions tested in control structures (such as the if-statement and while-statement) must evaluate to both true and false. The coverage criterion typically used by IBV is known as MC/DC (modified condition decision coverage). MC/DC stipulates that tests should be generated until each boolean sub-expression in a conditional expression has been shown to independently affect the outcome of the expression. MC/DC is considered by the Federal Aviation Agency (FAA) to be the most exhaustive coverage criterion and is used for testing the most critical type of aerospace code.

Once the model has been verified, using either model checking or IBV, automatic code generation is used to derive the core source code for the device. The generated code typically needs to be instrumented by hand in the same way outputs of compilers need to be optimized for certain applications[2].

There are two principal ways to perform code verification in the model-based development process. The first is applied using model checking techniques. In this method, the logical specifications are first converted to assertions, the generated code is instrumented with these assertions, and code verification tools [6] run on the modified code. In contrast to this rather direct method of re-verifying the requirements on the code, one may adopt an alternative strategy, where the code and the design are shown to be behaviorally equivalent to each other [12]. Since the design has already been verified, we may conclude that the code also satisfies the requirements. This alternative strategy makes use of IBV work, wherein the test suite generated as part of the model

[2] With advances in code-generators we expect to see production-level highly-optimized code being produced directly from models in the future.

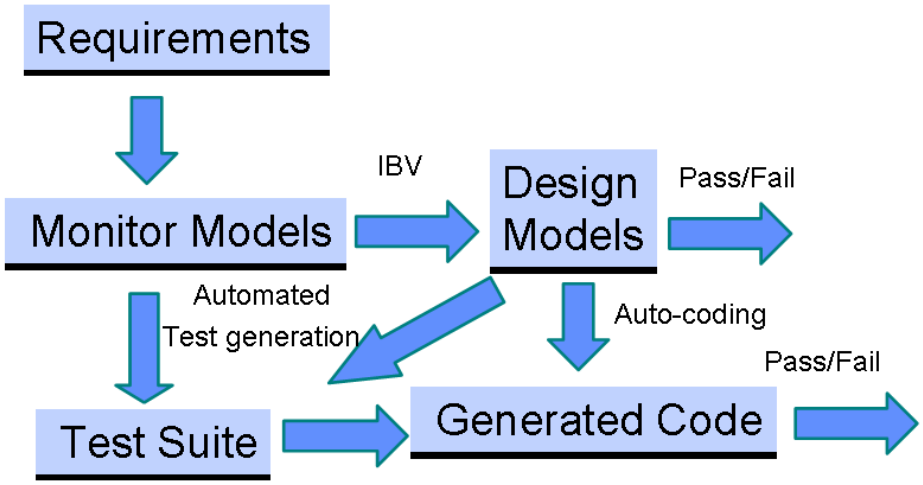

Figure 3. Instrumentation-Based Verification Workflow

verification is re-used for code. The code is verified to be correct if outputs of the model and the code are equivalent. If they are not, one may suspect that behavior has been introduced in the code that may lead to the violation of a requirement.

It should be noted that both these techniques for code verification are driven by requirements. In the case of assertion-based verification, the code is checked against assertions that are derived directly from the requirements. In the testing equivalence method, the test set that is used to prove behavior conformance between design and code is generated primarily by referencing the requirements. In both approaches, a direct traceable connection between the requirements and code verification activities is established. This traceability, base in mathematics, can help establish a convincing argument that the software has been checked with respect to its requirements at each stage of the development life-cycle.

The use of model-based V&V techniques reduces the dependence on testing as the principal means for verification, while at the same time providing a means for detecting design errors early in the development life-cycle. Clearly, the earlier errors are detected and corrected, the greater are the benefits in terms of time and cost; a fact expressed succinctly by the great architect Frank Lloyd Wright — "You can use an eraser on the drafting table or a sledge hammer on the construction site".

The nature of these design formalisms is such that they could be used in a regulatory context to challenge manufactured products for specific properties, such as safety, security, etc., acting as pseudo reference standards. In the FDA/CDRH/OSEL[3] software laboratory we were able to establish an infusion pump safety model using these methods. From this model, we were able to establish a set of alarm safety assertions and insert them in code from a real infusion pump implementation. The Verisoft[4] tool was used to perform systematic state space exploration of the code and check if any of these assertions were ever violated without triggering the appropriate alarm. Several alarms were not triggered that should have been [9]. An advantage of using Verisoft was that the assertions could be checked on all possible paths of the program and not just a specific execution path, as with runtime checking.

Clearly, it is impractical for regulators to develop such reference models for all medical devices. However, it is eminently practical for device manufacturers to carry out their own property-specific verification activities, and get "regulatory credit" for the work. One way of presenting this work is in the form of an assurance (or dependability) case [10]. For example, the *claim* might be that the device is safe. The *evidence* might be a test result report showing that all safety properties are met. And, the *argument* might be a safety model and an explanation of how it relates to the test results.

3. Static Analysis

Static analysis can be defined as an analysis of software that is performed without executing code, i.e., by analyzing some static artifact like source code or object files. Using static analysis facilitates detecting errors while the code is under development, thus reducing development and maintenance costs and the risk of expensive device recalls. In the context of high-confidence medical software, static analysis may be carried out for two principal purposes: a) checking the

[3] Food and Drug Administration/Center for Devices and Radiological Health/Office of Science and Engineering Laboratories

[4] Verisoft is a freely available state-space exploration tool for C programs. The use of Verisoft for the research study does not imply FDA endorsement of the tool.

source code to ensure that architectural constraints are not violated, and b) discovering errors in the source code.

3.1 Checking Architectural Constraints through Static Analysis

In the previous section, we described how assertion-based code verification and testing equivalence aims to establish the identical behavior of design models and code with respect to satisfying the requirements. However, these techniques do not check whether structural constraints defined in the design architecture are actually implemented in code. Static analysis can be used to make such checks.

The structural constraints that designers impose on code stem from considerations of extensibility and maintenance. For example, in a layered protocol, a layer is only allowed to use functions provided by its immediate subordinate so that a layer implementation may be replaced easily with another. However, such constraints formulated at the design phase are often not followed in the implementation. This often leads to spaghetti (highly-coupled) code that while perhaps still functionally correct, is extremely difficult to maintain and modify. In order to prevent this architectural degeneration, the code needs to be checked for off-specification dependencies. This can be done by using static analysis techniques to extract the implemented architecture from code [11]. The extracted architecture can then be compared to the required architectural specifications. This comparison can help identify dependencies that are present in the code but should not be and dependencies that should be present in the code but are not.

As an example, consider the architecture diagrams shown in Figure 4. Figure 4(a) shows a design architecture where component A is expected to communicate with B and B with C. However, after performing static analysis, we find that even though a dependency exists between A and B as planned, the expected dependency between B and C is missing and an extra dependency between A and C, that was not supposed to exist, is now present.

3.2 Detecting Runtime Errors using Static Analysis

While dependency analysis on extracted architectures may help guard against design errors, it does not afford any kind of protection against low-level coding errors. Coding errors usually manifest themselves as run-time bugs, such as null pointer dereferences, buffer overruns, arithmetic errors and memory leaks. Until recently, the only way to detect these errors was by means of rigorous code reviews and dynamic testing. However, with advances in lightweight formal methods techniques, a number of these defects can now be detected using static analysis.

While dependency analysis on extracted architectures may help guard against design errors, it does not afford any kind of protection against low-level coding errors. Coding errors usually manifest themselves as run-time bugs, such as null pointer dereferences, buffer overruns, arithmetic errors and memory leaks. Until recently, the only way to detect these errors was by means of rigorous code reviews and dynamic testing. However, with advances in lightweight formal methods techniques, a number of these defects can now be detected using static analysis.

There are many different types of static analysis techniques for detecting run-time bugs, such as symbolic execution [7] and abstract interpretation [3]. These techniques focus on assessing run-time bugs by evaluating intricate interactions within the software. For example, values of variables as they are manipulated down a path through the code, or the relationship between how parameters of functions are treated and the corresponding return values. To analyze code with this level of sophistication, all possible paths in the software are exhaustively analyzed to check for potential software anomalies.

By searching exhaustively through all paths in the program, these static analysis techniques can uncover bugs that may not be caught by testing alone. Since each test case follows only a specific path in the program, a finite number of tests can only check a limited set of possible execution paths. Usually these paths cover only a small fraction of the total possible paths in the software. Static analysis, on the other hand can evaluate all possible execution paths through the program; subject to the constraints of the tool employed.

Despite providing greater code coverage than testing, static analysis does have its limitations. Since the analysis is performed at compile-time, it is impossible to ascertain the actual values of input parameters and program variables used during execution. Static analysis tools therefore have to assume all possible values for these variables. This makes the analysis computationally intensive and causes high false positive[5] rates. Alternatively, the analysis tools may use heuristics to improve performance, yielding false negatives[6] as a result. In the ideal case, static analysis tools should have no false positives, no false negatives, and run in approximately the same amount of time as is required for compilation. However, this is not possible given the current state of technology. Therefore, most effective static analysis tools instead try to find the elusive sweet spot between false positives, false negatives, and performance to make results useful for every day software development.

It must be noted that static analysis is most effective when used in combination with traditional V&V techniques. It must be viewed as a complement to, rather than a replacement for, conventional V&V methodologies. Ideally, of course, static analysis should be integrated with a manufacturers' software development life-cycle process. Using it as code, is developed helps developers identify and repair defects prior to adding the code to a code baseline. Similarly,

[5] A false positive is any result that a static analysis tool reports that is not actually a defect in the source code.

[6] A false negative is any defect in the code that a static analysis tool does not report.

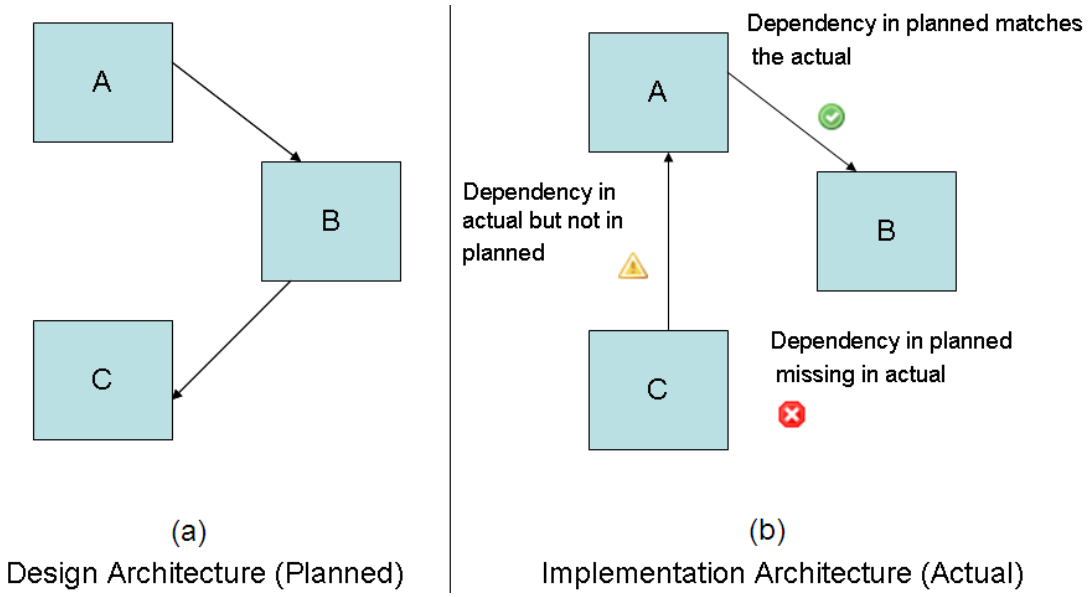

Figure 4. Architecture based comparison between design and implementation

using static analysis during code integration can provide an integrated analysis of the entire software system at a holistic level.

Static analysis technology can play a role in a regulatory context as well. In this context regulators can obtain device code and apply this technology to expose errors, without knowing much about the design or code. And, like the modeling technology discussed earlier, manufacturers could get "regulatory credit" for using this technology when presented in an assurance case format. One could further imagine that a verification claim would be strengthened by arguing that both model-based development and static analysis techniques were used in the verification process.

4. Conclusion

In this paper we have presented two complementary software development technologies that can be used to help develop high integrity medical device software: model-based development, which allows the developer to check that the design and implementation adhere to the system (software) requirements and static analysis that helps ensure that the implementation itself is free of errors.

Though these technologies have been used with great success in the aerospace and automotive industries, it should be remembered that the medical device environment has its own idiosyncrasies to consider. This environment is based on the practice of medicine (a rather inexact science) on patients with widely varying physiological conditions and with devices that rely on the notion of "competent human intervention" as a primary means for risk control. In this environment, the consequence of a device malfunction may be death or serious injury.

The technologies discussed provide a glimpse of how the development of high-confidence medical device cyber-physical systems might begin to be realized. An open-systems based research environment seems warranted to facilitate broad involvement in addressing issues underlying the composition and integration of cyber-physical medical device and infrastructure technologies through certifiably dependable tool chains that can represent and resolve cyber-physical properties. At the same time, these tool chains need to explicitly support implementation assurance claims. A broad national research agenda is warranted that brings academics, manufacturers, and regulators together to refine existing technologies; and through innovation, develop new technologies such that future implementations can be established as certifiably dependable.

References

[1] C. Ackermann, A. Ray, R. Cleaveland, J. Heit, C. Shelton, C. Martin. Model-Based Design Verification. A Monitor Based Approach. Society of Automotive Engineers World Congress 2008

[2] M. Ben-Ari, A. Pnueli and Z. Manna. The temporal logic of branching time. Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), 1981

[3] P. Cousot and R. Cousot. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. POPL 1977

[4] E. Clarke, O.Grumberg, D.Pereld. Model Checking. MIT Press. 2000

[5] General Principles of Software Validation; Final Guidance for Industry and FDA Staff. January 11, 2002

[6] P. Godefroid. Model checking for programming languages using Verisoft. Proceedings of the 24th ACM SIGPLANSIGACT symposium on Principles of programming languages, ACM Press, 1997

[7] H. Hampapuram, Y. Yang, and M. Das. Symbolic path simulation in path-sensitive dataflow analysis. In SIGSOFT Software Engineering Notes, Jan 2006

[8] High-Confidence Medical Devices: Cyber-Physical Systems for 21st Century Health Care A Research and Development Needs Report, Prepared by the High Confidence Software and Systems Coordinating Group of the Networking and Information Technology Research and Development Program, February 2009

[9] R. Jetley and P. L. Jones. Safety Requirements based Analysis of Infusion Pump Software, Proceedings of the IEEE Real Time Systems Symposium, Tuscon, December 2007

[10] D. Jackson, M. Thomas, and L. I. Millet editors. Software for Dependable Systems: Sufficient Evidence? Committee on Certifiably Dependable Software Systems, National Research Council, National Academies Press, 2007

[11] J. Knodel, D. Muthig, M. Naab, M. Lindvall. Static Evaluation of Software Architectures. 10th European Conference on Software Maintenance and Reengineering 2006

[12] A. Ray, R. Cleaveland, S. Jiang, T. Fuhrman. Model-Based Verification and Validation of Distributed Controller Architectures. Society of Automotive Engineers Convergence 2006

A Tool for Designing High-Confidence Implantable BioSensor Networks for Medical Monitoring

Sandeep K. S. Gupta
IMPACT Lab (http://impact.asu.edu)
School of Computing and Informatics
Arizona State University
Sandeep.gupta@asu.edu

Abstract - In this work we describe a software tool for designing implantable *biosensor network* (BSN) applications. BSNs are next generation medical monitoring systems, which provide continuous monitoring and actuation capabilities to medical personnel. They usually form a wireless network on a subject's body and can be controlled remotely. Before deploying any mission critical systems, it is important to be able to evaluate their performnce in the appropriate settings, and fine tune the design choices made. This is especially important for BSNs which are *cyber-physical* in nature – they interact and influence their environment of deployment.

Toward this goal we present the development of a software tool which can be used by developers and medical personnel to emulate an actual deployment of biosensor applications and evaluate its performance in different scenarios. We use Architecture Analysis and Design Language (AADL) in order to implement our tool as it provides an easy to use interface for specifying complex systems, and their environments. In this paper we discuss various aspects of developing such a tool including prinicipal characteristics of BSNs that need to be considered by it along with its functional architecture. We also provide an example scenario of how the tool can be used to evlauate a specific biosensor application.

1. Introduction

Recent technological advances in the fields of MEMS, integrated circuits, and low power design have lead to the development of implantable network of health monitoring sensors and devices. The RAND Corp. report on future technologies [RAND] predicts that the first applications resulting from the synergistic efforts of various disciplines will be out for public use by the year 2015. These **Biosensor Networks** (BSNs) are *cyber-physical systems* which have the potential to save lives by continuously monitoring the human body and taking corrective actions by triggering a response in case of medical anomalies. Biosensors communicate using the wireless medium with one another and with the external world. Medical personnel can use the Internet to remotely monitor and control implanted sensors, which not only provides them with valuable diagnostic feedback but also actuation capabilities. Figure 1 illustrates an example BSN embedded inside the human body.

Given the cyber-physical nature of BSNs – their close coupling with the human body – care has to be taken to understand the effects of their operation on their environment (body). As it may not be feasible to test BSNs in

a real-life setting (through actual deployment), it is important to develop tools which can 'emulate' such deployments and allow designers of BSNs to be able to evaluate the consequences of their design choices and thereby improve the performance of the BSNs.

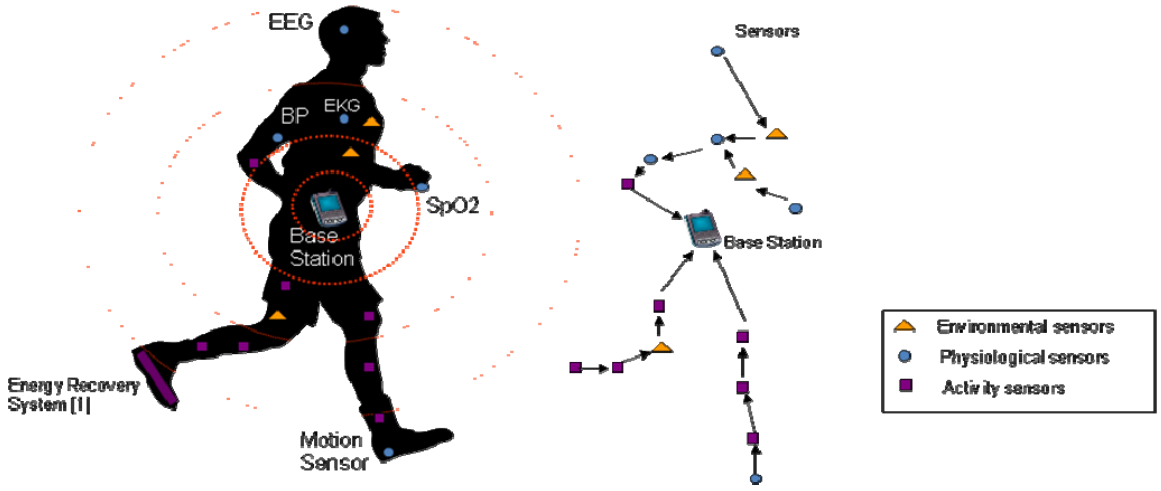

Figure 1: Example Biosensor Network

Typically, developing any application including those based on biosensors begins with an idea and conceptualization. This is then followed by the preliminary design. The design will then have to be analyzed against models of the target environment. Design and analysis are iterative steps and are repeated until the design team is confident that they have taken care of all issues. This refined design is then used to build a prototype that will be tested in the target environment (tissue medium). Here, the design and analysis phases are especially important as they are used to identify potential problems and address them at a very early stage. The goal of this paper is to present an overview of our tool and some of the issues involved in developing it. We use the Architecture Analysis and Design Language (AADL) in order to implement the tool. Some of the applications where such a tool could be useful include: analyzing the effects of signal propagation through the human body [GLP+03]; and studying energy-efficient coding and modulation techniques for biosensor networks [PG03] [69], techniques for minimizing heat dissipation in biosensor networks [TSG] [TTG], energy-efficient wireless communication protocols [SNG+01] [SGA+02], and cyber-physical security solutions of BSNs [VBG08].

2. Preliminaries

Biosensor applications can be of many types. Table 1 shows some of the important applications of BSNs. Even though the individual application requirements vary, all BSN applications have some properties in common. Each of these characteristics has to be considered carefully within our tool. In this section we summarize some of the prominent characteristics of BSNs.

Table 1: Types of monitoring needed for different applications of biosensors [SGW+01].

Type of sensors	**Continuous/Discrete monitoring**
Organ monitoring (Heart, Liver, Kidney)	Continuous
Cancerous Cell monitoring	Continuous
Glucose monitoring	Discrete
General Health monitoring	Discrete

Network Topologies: Unlike individual medical devices, BSNs have a group of devices (sensors) working in tandem performing patient monitoring and actuation. To be energy efficient the sensors typically organize themselves into different topologies. However, this organization of the sensors into different network topologies directly affects the deployment environment. For example, if sensors are located too close to each other, the cumulative heat generated between the sensors during their operation may be difficult to drain away and may result in unsafe temperature rise. But if the sensors are locate too far from each other the longer distance may need higher power wireless communication between sensors, which means higher RF power consumption and higher radiation into the surrounding tissue. The tool should be able to specify and handle a variety of sensors with wide ranging capabilities.

Sensor Hardware: The type of sensor used in building the network is of importance. Smaller sensors can only be equipped with low capacity battery and limited computational capability. Since such sensors cannot cover a

large area, they may require a higher density of distribution. Then more scalable and complicated network algorithms have to be designed to support more powerful and efficient data exchange for the large number of sensors. The tool should be able to specify and handle a variety of sensors with wide ranging capabilities.

Bio-safety Considerations: Bio-safety is a critical issue that should be considered at every step of biosensor design and implementation. Strict regulation of bio-safety may require smaller antenna and lower radiation. Also, sensors may not be allowed to recharge continuously in order to avoid sustained heating of sensors and the surrounding tissue medium. The ability to consider these requirements into the analysis of the BSN design, in an automated manner within the tool is extremely important for achieving a practical design.

It should be noted that looking at each of these requirements in isolation is not sufficient. Every aspect of the biosensor application influences the characteristics and performance of other parts of the system. Trade-offs between all the factors and requirements must be thoroughly considered and measured. Coordination and integration among different parts are essential to achieving a successful design.

3. BSN Design Tool Requirements

It is desirable for the tool to be applicable to a wide variety of biosensor applications and hence its analysis capabilities should be a common denominator of the various possible analysis methods. At the heart of such a tool will be a generic workflow control mechanism that is customized by specifying the application-specific plug-in modules and a user-specified array of third party tools. A modular design with well-defined interfaces will allow different researchers to work on different problem domains and implement their work as modules that can be plugged into the tool. As more knowledge becomes available to the community through ongoing research, the tool can be refined by swapping specific modules with newer and better ones. Thus the tool will be flexible. The tool will also be extensible in the sense that new functionality can be added later. This entails an open architecture design from the very beginning. We are using the *Architecture Analysis & Description Language* (AADL) from the Software Engineering Institute (SEI) for implementing the tool. The AADL provides an easy to use language with various constructs allowing system architecture model specifications.

3.1 Functional Requirements

In this section we present some of the principal functional requirements of the BSN design tool. We divide the requirements into two parts –operation specification and usability.

3.1.1 Component Specification

These describe the ability of the tools to specify the various components of the BSN application, how they function and how the results from the operation of the application are analyzed. Some of the requirements in this category are:

Workflow Specification: This will allow BSN developers to describe their application to the tool without having to modify the tool itself. Workflow is specified using AADL, which will be extended as and when required using new constructs which will be incorporated as an annex to the language.

Unified Bio-heat, Communication and Energy Consumption Analysis: BSNs work in a difficult environment. The wireless channel in the human body is prone to high path loss factors due to high water content. Further due to organ, bone and blood vessel boundaries there will be severe multipath fading. Specific propagation models have to be developed for the biosensors. Based on the model used, the range of a transmission can be estimated. Further, if there are multiple transmissions, propagation models that can be used to estimate the level of interference and hence the bit error rate that can be expected in the transmitted data. While analyzing heat, we have to account for multiple sources and sinks. We also have to take note of the fact that heat and communication affect each other. Communication produces heat which in turn affects the communication channel.

Sensor Deployment Specification: This will allow the developers to take sensor properties, placement and energy supply requirements into consideration. As these parameters can significantly affect bio-heat, communication and energy consumption, it is important for them to be explicitly specified by the biosensor application developer. Further, these parameters are hardware implementation dependant and vary from manufacturer to manufacturer.

Regulatory Requirements Management: The tool should have a controller that will maintain data consistency and enforce government mandated rules and regulations. The controller should be flexible enough to allow changes since different countries or states may have varying and conflicting guidelines

3.1.2. Usability

These requirements describe the tool in terms of its utility to the application developers in developing and analyzing the BSN application. Some of the requirements in this category are:

Graphical User Interface: The user interface provided by the tool should hide the complexity of models from users not well versed in the intricacies of specific models. At the same time, the interface should also provide enough flexibility for a researcher to change modules and models easily. New modules should be addable in a plug-n-play fashion and users should be able to control simulation parameters without having to write scripts.

Interfacing the Third Party Analysis Tools: In order for a tool to be useful, its data should be available for further analysis. Instead of developing new data analysis and simulation tools specifically for this application, we should take advantage of the numerous tools that are already available to the research community. To avoid being tied to any particular third party tool, we have to have some data exchange interface in which data from our tool is output in a standard format. We can then write tool specific drivers that will convert the standard data format to a format amenable to the tool. As and when researchers want to add new tools to the simulator environment, drivers can be written specifically for those tools without affecting anything else.

Customizability and Extensibility: Users should be able to use this tool with as many biosensor applications as possible. Since the nature of individual applications cannot be known beforehand, users should be able to easily customize the tool to their specific applications. It should have pluggable and swappable modules for models of body (2-D or 3-D shape, size, density, resolution), radio propagation, heat absorption, energy consumption, sensor properties (obtainable from manufacturer) etc.

4. Tool Architectural Description

Our vision of the tool architecture is shown in Fig. 3. It consists of 5 logical blocks as described below.

Design and Evaluation Tool: At the heart of the *Design and Evaluation Tool (DET)* will be the modules for bioheat, communication and energy analysis. Parameters and instructions to these modules can be fed through the Graphical User Interface (GUI). These modules will perform the required analyses using inputs from the other four modules. A workflow execution engine will perform the biosensor application operations as specified in the application workflow that is fed to the tool via the GUI. The GUI will allow the user to specify application parameters such as sensor type and location, energy constraints etc. It will provide an interactive visual interface to help place sensors in the body and also provide a visualization of the human body model that will be constructed with information obtained from third party databases. Only the important functionality will be executed in the DET. Functionality that are prone to change with the biosensor application and supporting functionality such as checking for regulations compliance are implemented in the other modules. In order to provide flexibility and extensibility, the Tool will have standard interfaces for each of the four surrounding modules as shown in Figure 3.

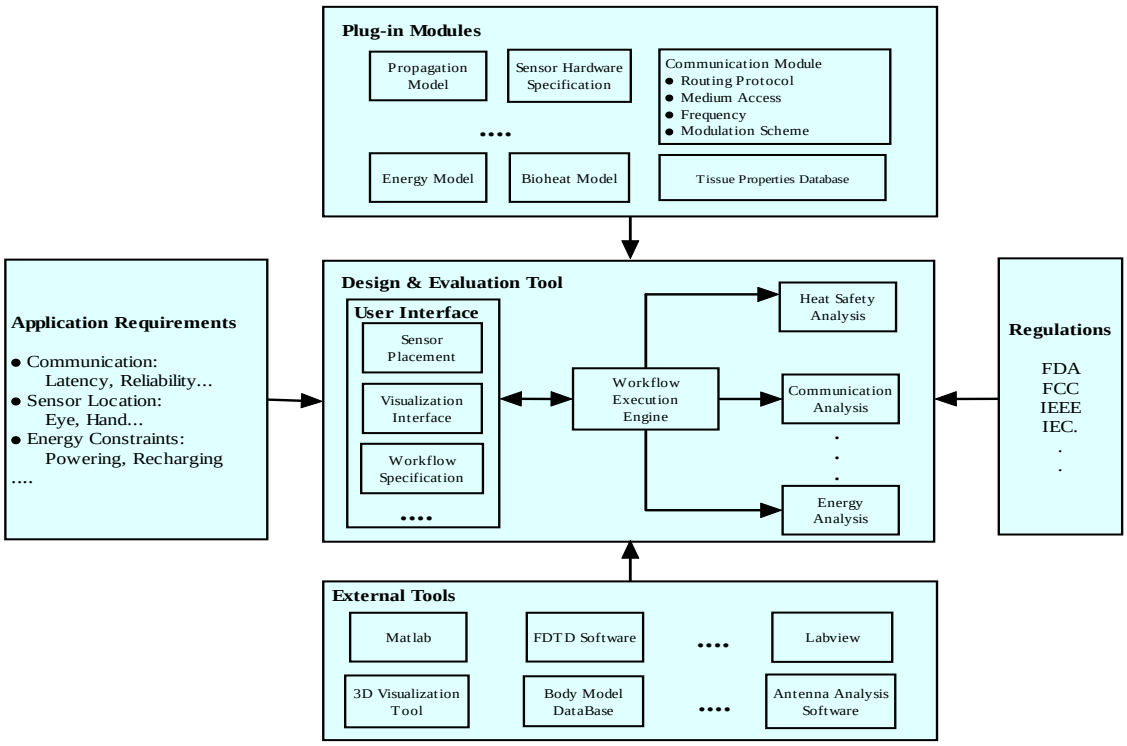

Figure 3: System Architecture.

Plug-in-Modules: Biosensors applications are very complicated may involve many technnologies. We can think of the biosensor system as composed of several subsystems including those for wireless communication, energy supply, and tissues. Each of these subsystems are research problems in their own right and have a continually evolving body of knowledge associated with them. Researchers have developed theoretical and empirical models to mathematically formulate these research problems. Therefore each subsystem in the of the biosensor application will be a model or mathematical approximation of the real world. As research in these areas continue, better and more accurate models will be developed. By implementing these models as plug-in models, we can easily replace old models with new ones. The design and analysis tool should also be flexible enough to allow each subsystem to be implemented in different ways. For example, power supply to sensors can be through RF induction, supersonic powering or an embedded battery. We should be able to change the way power model for an application by simply plugging in the appropriate module. The actual model used will affect the outcome of the analysis but will not affect the architecture of the tool. Some of the plug-in modules that may be implemented are: 1) *Propagation model*: The medium in which EM waves are transmitted can be homogeneous, heterogeneous, or layered. Its impact on attenuation and phase shift of EM waves would be various. We will provide some widely used propagation models but users can customize by adding their own propagation models; 2) *Energy model*: Provides some regular and basic options for various power supply methods such as RF induction, supersonic, and B-field. Parameters such as the capacity of battery and performance of transductor can be adjusted to meet different requirements. Users' own customized power supply models are acceptable as well; and 3) *Tissue properties database*: Has information on tissue properties that can be obtained from publicly avalaible sources [EMF].

External Tools: Commercial and open-source software are available to perform several useful tasks that may be performed in the design and analysis of biosensor applications. To avoid reinventing the wheel and save development time, it may be useful to exploit the capabilities offered by these third party software tools rather than rewrite them. However, these tools may be developed by different parties and may be incompatible. To simplify interaction, it will be necessary to develop a data exchange standard. For any new software tool that

has to be supported, a tool driver can be written that will convert between the standard format and the format accepted by the new tool. Useful tools include but are not limited to: 1) Mathematical computing and signal processing tools such as MatLAB and LabVIEW; Electromagnetic simulation tools such as FDTD or FED analysis software; 2) Antenna analysis and propagation simulation tools; 3) Human body modeling and simulations tools. We are considering some publicly available human model, such as Visual Man Project [NLM] or other public tissue model, such as NIH model organization [NIH]. These models will be used by our tool in the form of an input data file or database; and 4) Visualization and graph-plotting software that will help researchers better understand analysis results.

Application Requirement Specifications: This module consists of those parts of the application specifications that change often during the iterative process of design and analysis. It has data structures to store parameters that can be used to tweak an application. The actual parameter definitions and values have to be provided by the application developer and are specific to that application. Some of common application requirements specifications include: 1) Communication*:* Frequency of operation, data latency limits, data loss tolerance; 2) *Sensor Location*: Placement of individual sensors and base station. Depending on the human body model used, location may be specified in terms such as right eye, left elbow, heart or may be specified Cartesian co-ordinates; and 3) *Energy Information*: Power supply (embedded battery, RF inductance), energy consumed by individual sensors for different operations.

Regulation Compliance: This module is used to define the International or governmental regulations that should be followed for implantable biosensor applications. Regulations cover issues such as the permitted operation frequency for implanted medical devices, the Specific Absorption Rate limit and maximum temperature rise allowed for bio-safety. This functionality has been put into a separate module because regulations change with time and different applications may use a different set of regulations. Also, different countries and regions may have different regulations. In our implementation, we plan to support most widely used regulations of biomedical or electrical engineering. These include: 1) *Food and Drug Administration (FDA)* regulations on medical devices; 2) *Federal Communications Commission (FCC)* regulations on using

the ISM (Industrial, Scientific and Medical) band and other requirements on frequency and bandwidth management. FCC also has the Medical Implant Communications Service (MICS) standard for communication between medical implants; 3) IEEE defines many standards on issues such as the measurement of SAR, IEEE C95.1 RF human exposure standard, and Standard for Medical Device Communications (IEEE 1073); and 4) *IEC* defines the safety limit of exposure to RF radiation. IP68 (IEC 601.2.2) and IP20 have stringent requirements for medical devices. Other regulating organizations will also be considered including EU and ISO. It is important to take these regulations into account while developing biosensor applications.

5. Example of an Analysis Model: Bioheat Problem

Operation of implanted devices inside human body will cause tissue heating. Heating is caused by both the sensor circuitry as well as absorption of radiation by tissue. Specific Absorption Rate (SAR) is a measure of the rate of radiation energy that is absorbed by dielectric materials, such as biological tissues. Normally it is expressed in watts per kilogram (W/kg) or milliwatts per kilogram (mW/kg). These limits, which are based on the current International Electrotechnical Commission (IEC) 60601-2-33 standard, are 8 W/kg in any gram of tissue in the head or torso for 15 minutes, or 12 W/kg in any gram of tissue in the extremities for 15 minutes [FDA99]. Also ANSI/IEEE C95.1-1992 has a limit on partial body exposure, to 8 or 1.6 W/kg (controlled or uncontrolled exposure) averaged over any gram of the exposed tissue.

Different parts of body have various sensitivities to temperature rise. For example, the eye is expected to be more sensitive to heating because of a lack of blood supply to cool down its temperature once increased. And exposure to RF fields results in increased retinal temperatures, which can lead to eye dryness and ocular discomfort. Some research results show that a long-term exposure to RF could also lead to cataract. Understandably, bio-safety is an essential issue and should be considered when implementing implanted biosensor. Strict calculation and prediction should be done to estimate the SAR (Specific Absorption Rate) and temperature rise inside body tissue.

5.1. Heating Factors

In our previous work, we have studied the temperature rise inside body tissue due to an implanted biosensor [TTG05]. In that case, the internal sensors are powered by RF inductive power. Sensors and an external base station exchange data wirelessly using the 2.4 GHz ISM frequency band. The heat factors we considered were:

Heating caused by RF inductive powering: If the implanted devices are powered by RF inductive power supply, the frequency of RF power supply is normally operated in 2 MHz to 20 MHz range [He88] [MS01].

Radiation from Implanted device communication: Implanted devices need to exchange data between other implanted devices or an external device using wireless communication. The wireless signal also has radiation effect on tissues surrounding the implanted sensors.

Power dissipation by implanted node circuitry: When a sensor node processes the data, there will be power consumed by its circuitry. The sensor circuitry may also need to perform data aggregation and various other functions which consume power. This power is transformed into heat which may add to the already heated tissue. The power consumed by the sensor circuitry depends on its implementation technology and architecture.

Effect of Metallic Implants: Several research findings show that metallic implants may couple with the RF used in magnetic resonance imaging (MRI) and may lead to a heating hazard [YSA02] [Ho]. The presence of the metallic implant results a local amplification of the SAR, and this effect is not seen with the external transmitter alone.

5.2. Calculating Temperature Rising

The above mentioned sources for heating the tissue can cause a rise of temperature inside the control volume. The rate of rise in temperature is calculated by using the Pennes bioheat equation as follows.

$$\rho C_p \frac{dT}{dt} = K\nabla^2 T + \rho SAR - b(T - T_b) + P_{circuitry} + Q_m$$

The left hand term measures the rise in temperature in the control volume, the terms on the right side respectively indicate the heat transfer rate by conduction, heat transfer due to radiation, heat

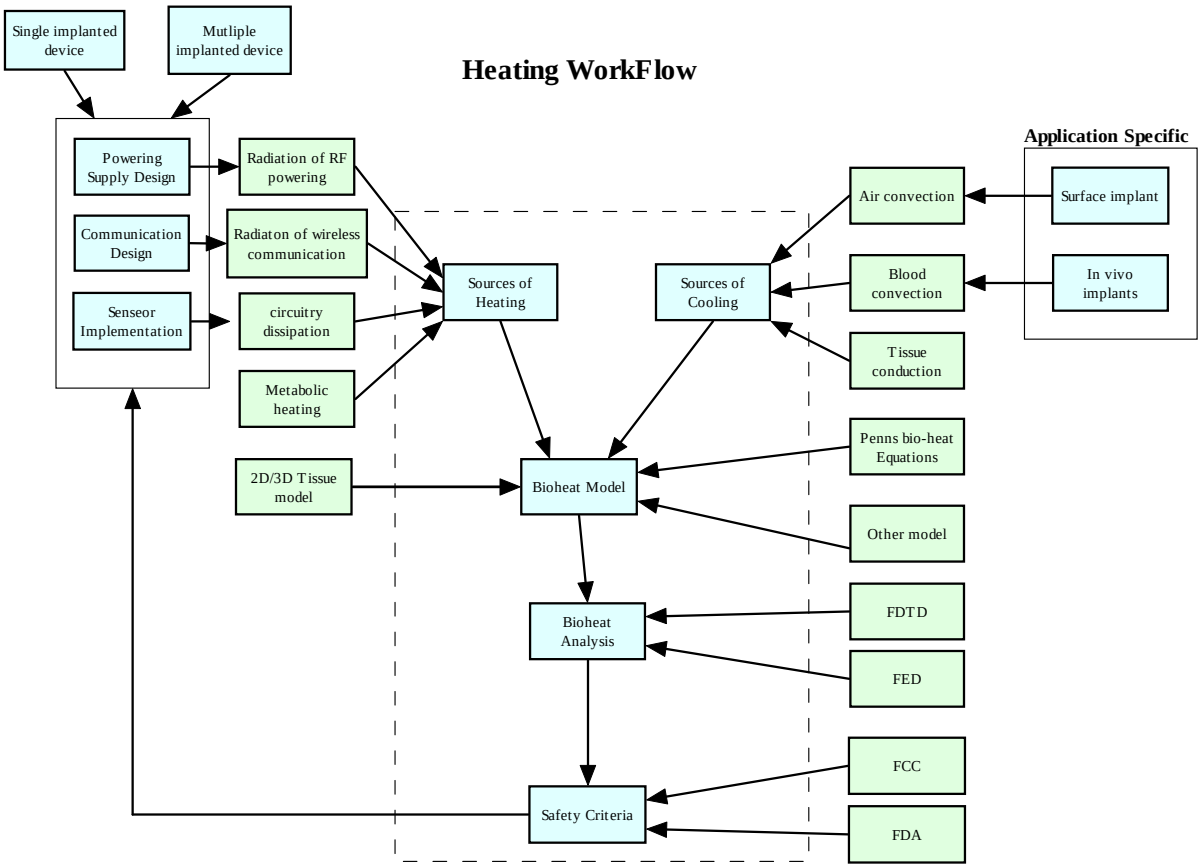

Figure4: Heating Workflow.

transfer due to blood perfusion, power consumed by circuitry and heat generated by metabolic heating .Where is the rate of rise in temperature in the control volume, ρ is the mass density, C_p is the specific heat of the tissue, K is the thermal conductivity of the tissue, b is the blood perfusion constant which indicates how fast the heat can be taken away by blood flow inside the tissue, T_b is the temperature of the blood. Once we know the properties of mediums and blood flow, and the power or heat absorbed by the tissue, we can calculate the temperature change rate within a period of time by $\frac{dT}{dt}$. With this equation we can predict the SAR inside tissue and the resulting temperature rise.

5.3. Workflow of Heating Problem

Actual calculation of the temperature rise is not be straightforward as many factors are dependent on the design and implementation of other parts of the biosensor system as shown in Figure 4. Factors that affect heating include power supply design, communication design and sensor implementation. Researchers may work with Matlab, LabView or other wireless simulation software, propagation software to design those parts. If part of the communication design is changed (ex. different encoding scheme or radio frequency), then its impact on heating would be changed too. Sources of cooling have a similar problem and depend on where to implant the sensors, the properties of tissues etc. Further, researchers may use different models, or use different software tools (Matlab, VC++) and algorithms (FDTD, FEM) to evaluate the SAR and temperature rise. The final result would be compared with different regulations according to the specific application. Final results may lead to redesign of other parts of the system.

All the blocks outside the dash box are factors that depend on other application requirements and system implementations. Any change to any of them would lead to a change in the heating effect. Researchers have to work with several different software packages and repeat the whole heating work flow several times.

With proper interface drivers and software platform, the heating analysis inside the dash box can be automated. Researchers only need to interact with the integrated platform to use the different software. The data generated from different sources will be managed and aggregated together to realize an automatic heating estimation workflow. If any part of the system has changed to a different scheme, researcher only needs to re-execute the automated workflow again without having to deal with individual software packages.

5.4. Communication System Workflow

A researcher may use various software tools for antenna simulation and signal modulation simulation. With different antenna schemes and transmission medium, RF signals have different attenuation and phase shift. Signal strength also depends on the location and distance between sensors or base-station. This interdependence of various components and design decisions are shown in Figure 5. If we consider propagation model, modulation scheme and environment interference together, link budget analysis can be performed. The output can be used in designing communication hardware for the sensor.

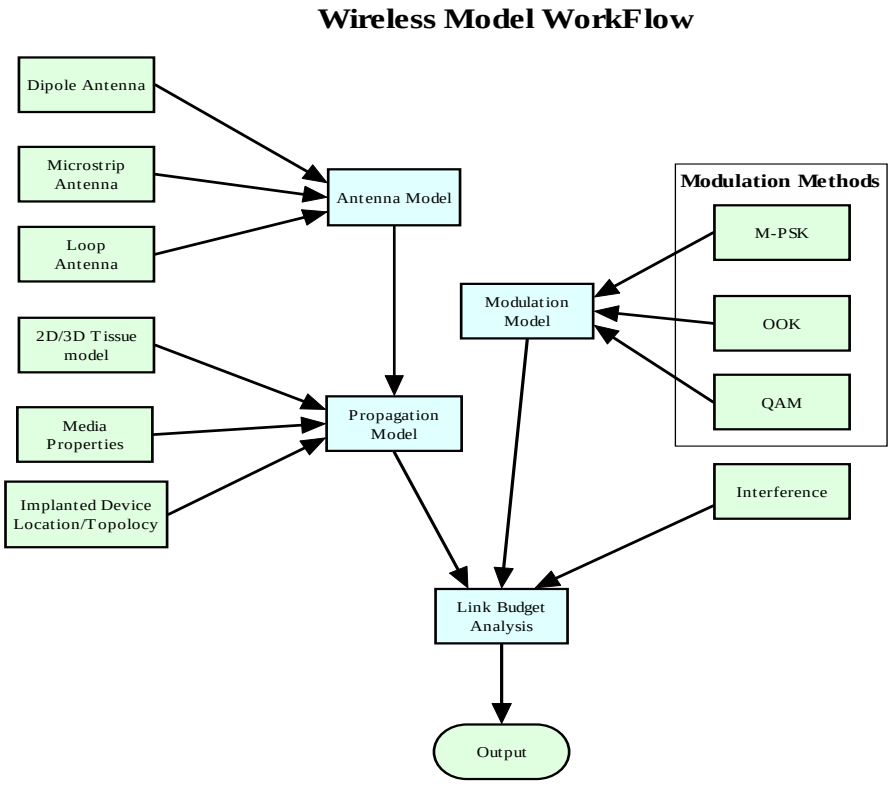

Figure 5: Communication Workflow.

Models inside the dash box will be integrated as an automatic process in our platform. Once a user changes the requirements and implementation antenna model, modulation model, network routing model, the workflow of communication system will automatically run and the result will be input to other related workflow and will trigger those workflow to process again.

5.5 Power Workflow

Power system design is influenced by the application requirements, the implementation of sensors and design of the communication system. The work flow that specifies the inter-relationship among the different requirements and design decisions are shown in Figure 6. The total power consumption is composed of power consumed by sensor circuitry, communication system and the base station. Users may select different implementation options of power system which have different impact on heating the surrounding tissue. At the

same time, regulations or application may have some strict requirements on lifetime of battery, size of power supply etc. This workflow can be automated by our proposed platform.

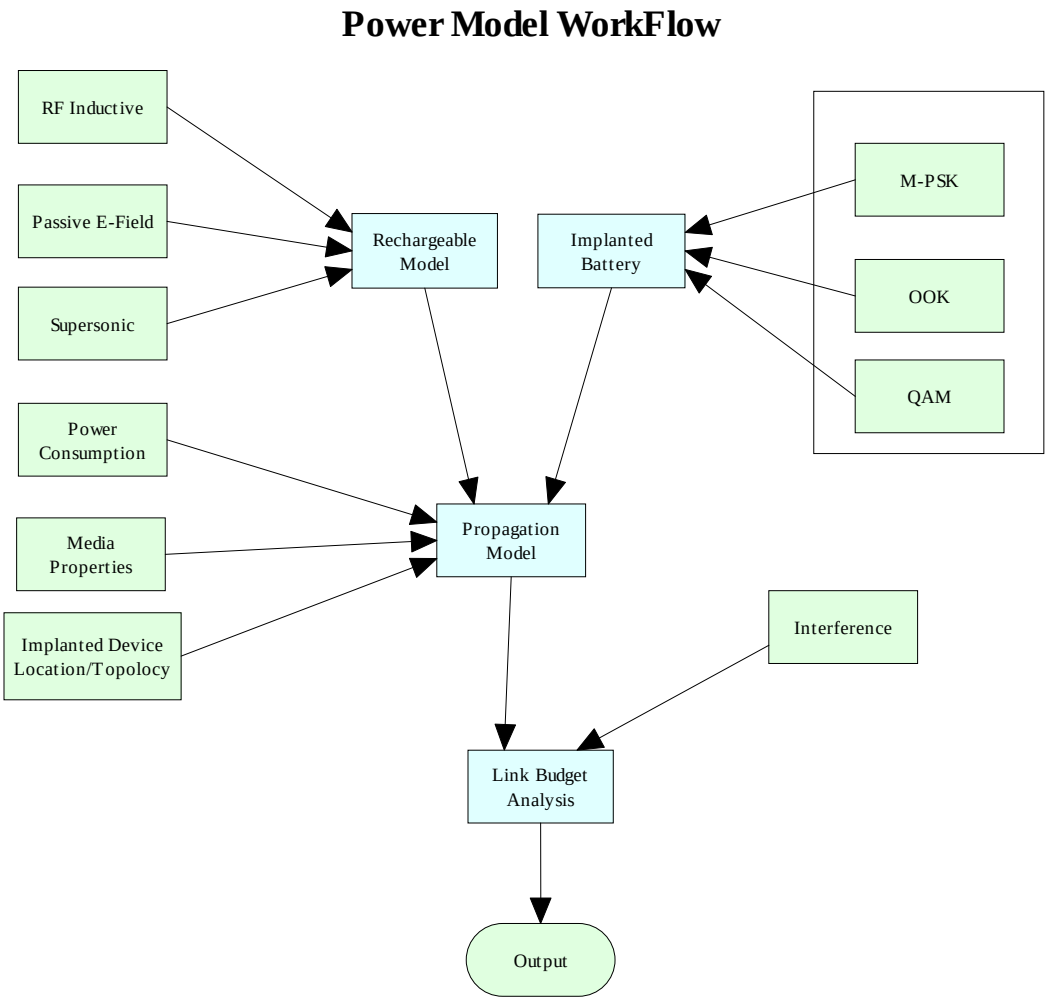

Figure 6: Power Workflow.

5.6. Interaction between different workflow

The relationship between the different models is shown in Figure 7. The whole design process will work on an integrated and automated mode. All the subsystems rely on the detailed application specifications and regulations. These subsystems work together to decide whether reasonable power consumption criteria, stable communication, and bio-heat safety criteria can be met. The change of input of one workflow would automatically generate new results for this workflow and then trigger the re-computation of related workflows.

Finally, users can expect to get simulation and analysis results with a few mouse clicks and the output will show if the result is in accordance with specifications and regulations.

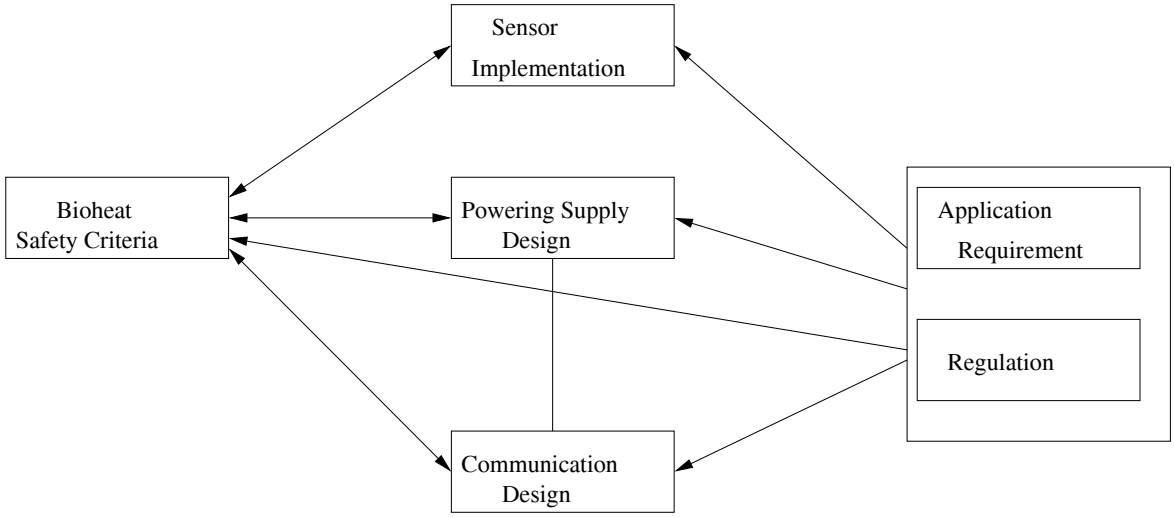

Figure 7: Module Interaction.

6. Development with AADL

A model description language is required to develop the biosensor network application analysis tool. In this regard we use AADL [FGH] which has the following properties:

- High level architectural model of the tool can be specified in AADL using various constructs that it provides.
- The functionality of the AADL model can be extended with the help of appropriate *annex* to the language
- Analysis of the entire system can be performed in AADL by designing appropriate plug-ins.

  Given the system architecture of the tool in Figure 3 we endeavor to develop an AADL specification. Figure 8 [CPS] shows the AADL specification for the biosensor network that is designed for analyzing the heat safety of the system.
- To specify the system we first need models of the physical objects and devices, that comprise the system. These are required to be specified in AADL specific constructs. As shown in Figure 8 we specify the model of the human tissue in the physical component named *Tissue* where we can incorporate its thermal characteristics as a set of attributes. The sensors are modeled as the *Node* device wherein we need to specify the computing states of the sensors and the associated energy dissipation.

- In order to incorporate the effect of energy dissipation of the sensors nodes on the tissue we have to specify the bioheat model using AADL language constructs. Specification of models of different physical phenomenon in AADL involve the development of appropriate annexes.

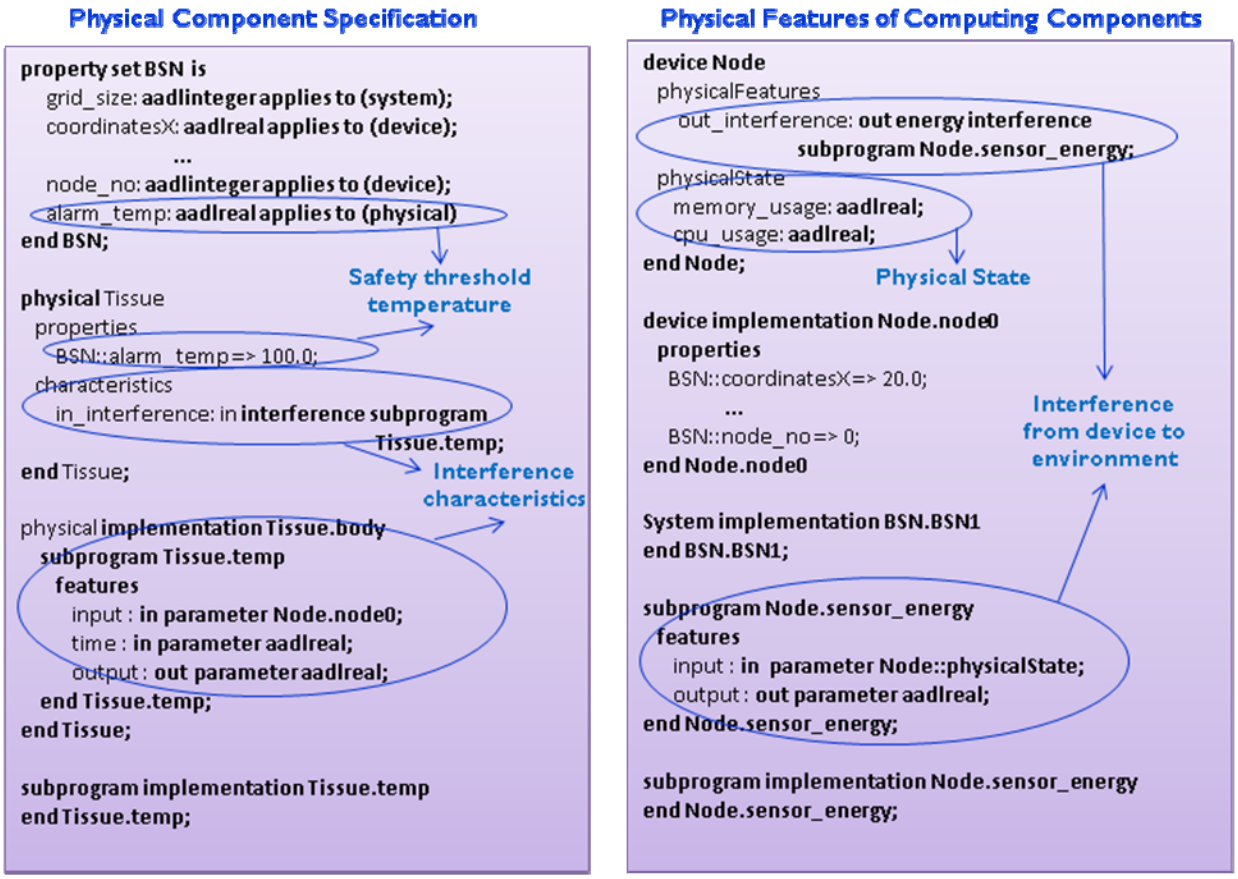

Figure 8: AADL specification for Heat Safety analysis

We specify the bioheat model as a subprogram *Tissue.temp* in the physical component specification of the tool (as shown in Figure 8). In this subprogram we can specify the input variables and the output variables of the bioheat model and also specify a mapping between them.

- The application requirements and regulations of the tool need to be specified in AADL by properly setting values of attributes of different componenets. An application requirement in our tool is to provide a system alarm whenever the temperature of the tissue crosses a threshold. This requirement is specified in the AADL model as an attribute to the *Tissue* component.
- The tool in order to analyze the thermal behavior of the BSN will require inputs from external softwares. The

bioheat model would require several parameter values that are not possible to compute in AADL given the present status of its infrastructure. In our tool we need to parse the output provided by Matlab and convert it to AADL recognizable format and provide as input to the bioheat subprogram. Thus ouputs from different software modules need to be properly parsed and provided to the analysis infrastructure of AADL.

- An important aspect of this tool is the analysis of the interaction of the BSN with the physical environment (Tissue). The interaction is very complex and current AADL infrastructure does not provide a method to specify and analyze this cyber-physical property of the BSN. We are currently working towards the development of an annex for AADL that supports the evaluation of the cyber-physical interaction of a system. In the case of the tool for BSN we have incorporated the physical interaction of the BSN with the body tissue by implementing two subprograms in each of the componenets *Tissue* and *Node.* The subprogram *Node.sensor_energy* specifies the amount of energy that is being transferred as heat to the sensor's physical environment (body tissue). *Tissue.temp* subprogram is then utilized to calculate the temperature rise in the tissue due to the energy dissipation in the sensor.
- An intuitive GUI is essential for the tool in order to provide an easy interface to the user. The GUI that currently exists in the AADL framework does not enable the user to fully utilize all the functionalities of AADL. One of our goals in this development of the tool is to develop a GUI through which the user can provide complete information about the system such as placement location of sensors, tisue heating parameters, work flow for analysis of different policies.

7. Conclusions

In this paper we have discussed some important issues in developing a software tool for designing and analyzing BSN applications. In this regard we presented a design of a software tool which can be used by developers and medical personnel to emulate an actual deployment of biosensor applications and evaluate its performance in different scenarios. We use Architecture Analysis and Design Language (AADL) in order to implement our tool as it provides an easy to use interface for specifying complex systems, and their environments. Further, we discussed various aspects of developing such a tool including the principal

characteristics of BSNs that need to be considered by it along with the tool's functional architecture.

Acknowledgements

The authors would like to thank Ayan Banerjee, Krishna Venkatasubramanian, Tridib Mukherjee, and Qinghui Tang for their technical contributions. This work is supported in part from a grant from NSF #0831544.

References

[RAND] P.S. Anton, R. Silberglit and J. Schneider,"*The global technology revolution: Bio/Nano/Materials trends and their synergies with information technology by 2015"*, *http://www.rand.org/publications/MR/MR1307/*

[SGW+01] L. Schwiebert, S. K. S. Gupta, J. Weinmann et al., Research Challenges in Wireless Networks of Biomedical Sensors, In *The Seventh Annual International Conference on Mobile Computing and Networking*, pp 151-165, Rome Italy, July 2001.

[PG03] Y. Prakash, S.K.S Gupta, Energy Efficient Source Coding and Modulation for Wireless Applications, IEEE Wireless Communications and Networking Conference, 2003. WCNC 2003. Volume: 1, 16-20 March 2003, Page(s): 212 -217.

[SNG+01] V. Shankar, A. Natarajan, S.K.S. Gupta, L. Schwiebert. Energy-efficient Protocols for Wireless Communication in Biosensor Networks, IEEE Personal, Indoor and Mobile Radio Communications Conference, San Diego, 2001.

[GLP+03] S.K.S. Gupta, S. Lalwani, Y. Prakash, E. Elsharawy, L. Schwiebert. Towards a Propagation model for Wireless Biomedical Applications, *IEEE International Conference on Communications, Alaska, May 2003.*

[SGA+02] L. Schwiebert, S.K.S. Gupta, P.S.G. Auner, G. Abrams, R. Lezzi, P. McAllister, "A Biomedical Smart Sensor for Visually Impaired", IEEE Sensors 2002, Orlando, FL, June 11-14.

[FDA98] U.S. Department of Health and Human Services, Food and Drug Administration, Center for Devices and Radiological Health. Guidance for the Submission of Premarket Notifications for Magnetic Resonance Diagnostic Devices, November 14, 1998.

[He88] W.J Heetderks, "Powering of Millimeter and Submillimeter-Sized Neural Prosthetic Implants", *IEEE Transactions on Biomedical Engineering, Vol. 35, No. 5, May 1988.*

[MS01] W. Mokwa, U. Schnakenberg, *Micro Transponder Systems for Medical Applications, IEEE Transaction on Instrumentation and Measurements 30 (6) (2001).*

[YSA02] C.J. Yeung, R.C. Susil, E. Atalar, "RF heating due to conductive wires during MRI depends on the phase distribution of the transmit field", Magnetic Resonance in Medicine 48:1096-1098, 2002.

[Ho] Henry S. Ho, Safety of metallic implants in magnetic resonance imaging. Journal of Magnetic Resonance Imaging, Volume 14, Issue 4 , Pages 472 – 477.

[TGS05] Q. Tang, S. K. S. Gupta, and L. Schwiebert. BER performance analysis of an on-off keying based minimum energy coding for energy constrained wireless sensor application. November 2005. In Proceedings of the IEEE International Conference on Communications.

[TTG05] Q. Tang, N. Tummala, S. K. S. Gupta, and L. Schwiebert. Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue. IEEE Transactions on Biomedical Engineering, 52(7):1285–1294, July 2005.

[VBG08] Krishna K. Venkatasubramanian, Ayan Banerjee, Sandeep K. S. Gupta, Plethysmogram-based Secure Inter-Sensor Communication in Body Area Networks In Proc of IEEE Military Communications Conference, (MILCOM'08), San Diego, CA, November 2008

[CPS] Cyber-Physical Systems and AADL – Sandeep Gupta AADL User's Day Agenda Feb 2, 2009 Architecture Analysis & Design Language (AADL) University of Southern California, Los Angeles, CA http://csse.usc.edu/csse/event/2009/AADL/pages/program1.html

[FGH] Peter H. Feiler, David P. Gluch and John J. Hudak, The Architectural Analysis & Design language (AADL): An introduction, In Performance-Critical Systems, Technical Note.

AN EVENT DRIVEN FRAMEWORK FOR ASSISTIVE CPS ENVIRONMENTS

Fillia Makedon, Zhengyi Le, Heng Huang, Eric Becker
{makedon, zyle, heng, becker}@uta.edu
Computer Science and Engineering Department
University of Texas at Arlington, USA

Abstract

Assistive Cyberphysical Systems *(ACPS) are pervasive and ubiquitous systems connecting the cyber with the physical, with the aim to assist a human's daily activities both at home and at work. We propose an* ***event driven*** *framework with event identification mechanisms that drive actuators, transform a substrate and alter human behavior in a feedback loop process in ACPS. This framework is a dynamic, context aware, adaptive, self-repairing and high-confidence system that couples computational power with physical substrate (testbed) control and command; it recognizes events, human needs from lifestyle, environmental and longitudinal health data.*

1. INTRODUCTION

Assistive Cyber-Physical Systems or ACPS are systems that collect data and provide assistance to humans interacting with the physical and digital environment around them. In this special CPS category, the human is dependent on the input and output of instruments and on the environment's sensors embedded in @home and @work spaces. ACPS is a complex, dynamic and pervasive CPS that is human-centered and responsive to human needs.

In this paper, we describe an "***event identification mechanism"*** for ACPS that enables seemingly meaningless human activity and interaction data to gain meaning with regard to human behavior. **Event identification (EI),** is a 2-phase process of first assimilating continuous and discrete types of data or streams collected through various types of sensors over time and then identifying "events" of interest through the use of various mining and feature extraction tools applied to multi-channel data in a non-invasive and privacy-preserving approach. Through the EI mechanism, the framework can summarize and filter non-interoperable information over time and space in order to reach higher levels of awareness and intelligence that responds to patterns and enhances human capabilities in significant ways.

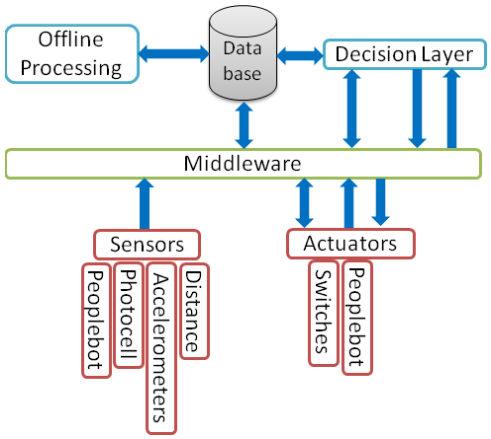

Figure 1 . ACPS Framework for Event Detection

ACPS desired properties include (1) a self-correcting and adjusting feedback loop that refines the identification of an event by adjusting sensors and data collected; (2) flexible and easy to use human-centered and customizable functions; (3) an expandable architecture to accommodate new types of instruments and devices; (4) responsiveness to emergency and non-emergency situations; (5) supportive prediction and decision making by identified events; (6) an EI mechanism that adapts to different situations, the types of data that need to be collected, new technologies or software; and used to discover behavioral and environmental "markers" of social and community importance that can trigger alerts or warnings or address social and community needs, ranging from better learning or training environments, to improving human performance in stressed situations, such as recognizing depression, pain, lack of understanding, or even recklessness in the

face of risk or danger; (7) innovation and design of new instruments, and testing new ways to use them; (8) powerful analysis engines to recognize low-level and higher level events, and support meta-analysis impacting numerous CPS applications, from healthcare, to manufacturing, house/car design, training, school scheduling [2,5-10]. Figure 1 shows the ACPS framework for event detection.

2. EVENT DRIVEN ACPS

An "Event" in ACPS is any extraordinary occurrence or observation involving the subjects, objects, and environmental status in the entire environment. We focus on identifying three types of events [3,4]: (a) prevention of accidents such as falls or other injuries; (b) abnormal behavior or activity involving either the human or the system, (e.g., malfunctioning of instruments, physical or digital intrusions, aberration from normal human activity (e.g. missed medication or meals) in order to better guide the human or signal the need for the system and/or its components to self-correct or ask for human intervention; (c) social/psychological need detected (e.g., depression, pain, loneliness).

The event identification component in the ACPS framework gets data fed from multimodal sensors and devices (e.g. Micaz/Cricket/Mote Invent sensors, VICON cameras, audio sensor, ECG sensor). In ACPS, event identification and fusion exist in both low level (e.g. identify abnormal activities from image/video) and high level (e.g. identify events from correlated events and data from different sensors/devices). All events in ACPS can be organized as a hierarchical tree: the top-level events are derived from the low-level events.

3. FEEDBACK LOOP

This ACPS includes a self-correcting and adjusting feedback loop that refines the identification of an event by adjusting the way the sensors and the environment collect data. For example, it may add more sensors, improve the window of time over which data are collected, reduce noise by some other mechanism, improve the algorithm applied on the data, etc. Figure 2 shows the feedback loop of the ACPS framework for event detection. (a) The static sensors provide raw sensory data, to be processed by the behavior recognition components and create events. (b) The events are fed into the *Event Processing/Identification Module*. (c) Unusual behaviors are identified and sent to the human operator (who closes the loop) and/or to the actuator controller who may trigger a certain set of actions. (d) The actuator controller activates the robot or moving cameras to focus their sensors on the target. (e) Better sensory input is acquired and sent to the behavior recognition module. (f) Some live video is transmitted to the human operator, who closes the loop.

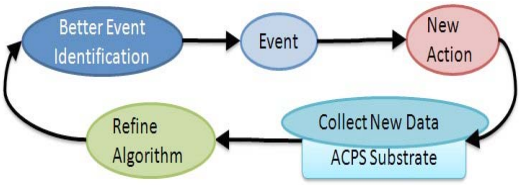

Figure 2. Event Refinement Feedback Loop

4. PRIVACY AND SECURITY ISSUES

The ACPS system produces, by its nature, valuable behavioral and other human-centered data that relate to sensitive health records of a person. It can also connect to biomedical data such as brain scans, history of a condition, physical characteristics and even genetic characteristics. It is very important, therefore, that in order to produce robust and feasible ACPS systems for e-health and pervasive environments of the future, that we also address important issues of privacy and security. In particular, we ask the question, at which points of the event identification process is there risk for privacy violations or security lapses? Or, how does each stage of the EI process rank in terms of data sensitivity? In this section we provide an analysis of the different types of security issues that can arise. In a series of PETRA'08 and 09, papers [1, 11-13], we introduce a security framework for ACPS. We divide the key privacy and security issues in two

different types: Low level security that applies to raw data or streams and high level security that applies to high level events that have more semantic meaning regarding behavior. For low-level security, we consider *Data Integrity* (nobody tampered with the data and no parts are missing), *Confidentiality* (data is encrypted and only the proper receiver can decrypt it), and *Availability* (continues and robust service). For high-level security, we consider sensitive events that are defined by the user or the data owner.

4.1 Low Level Security on Raw Data Sets/Streams

Data Integrity: When data are generated in devices and ready to be sent to a receiver, such as a base station, a router or another device which will further process the data, are supposed to guarantee they can tell that (1) it receives the entire package of the data instead of a portion of it, and that (2) the data is from that device as it claimed and not someone else [2,3]. Usually it could be provided by generating the hash value or fingerprint of the entire message or data sequence, and then signing the hash value with the private key of the device if public key operations are affordable on that device, or using a keyed hash function to generate the fingerprint if public key operations are not affordable and the two parties has established shared symmetric key. The receiver will use the public key of the sender or the symmetric key they shared priori to verify the package it received. The finger print makes sure that all the original information was included and the keyed hash or the digital signature makes sure that the data is generated by the device which holds the same key or corresponding public key.

Data Confidentially: When data is transmitted among devices, secure channels must be established for the communication to protect the data against eavesdroppers. So besides the data origination authentication and integration checking, the data must be encrypted during transmission. For those devices that can afford public key operations, they will use their public/private key pairs to establish session keys to encrypt the data exchanged. For those which cannot afford public key operation, either a key escrow or an initial key distribution phase will be introduced to set up the symmetric key sharing among the devices in a group-wise or pair-wise fashion.

Data privacy could be viewed as "selective confidentially" granted by data owner. We could let users/patient to configure the access of their data and this is not enough since privacy information may leak in an unintentional way. For example, the traffic data in a health center may already be de-indentified. However, if an attacker is equipped with an appropriate tracking algorithm and also obtains other information about a person, such as the type of room (e.g., bathroom) he is visiting at a certain time, he will be able to link a daily habit with the data set and associate this with the actual corresponding person. So we have to sanitize the collected data before we actually publish it, even if it is aimed at general-purpose research. We will need to annonymize such a dataset against popular data mining methods. At the same time, we must also keep certain dataset properties associated with the dataset in order for it to be valuable for further analysis. For example, we may wish to use the dataset to optimize the layout of the physical apartment space of a person and the associated sensors for data collection in order to schedule better services [13].

Data Availability: The availability is the basis of the confidentiality and integrity, and requires the data or service to be continuously available. The required security must provide intrusion detection and fault-tolerant-and-recover mechanisms, and work against attacks like Denial-of-Service (DoS) attacks. The resiliency is more critical in assistive applications since any network misconnection or dysfunction of the medical devices may endanger human lives.

4.2 High Level Security on Sensitive Events

After raw data is processed and transformed into events, a higher level of security should

be provided to protect sensitive events. The mechanism we propose is to allow patients to authorize the access of his/her events and to enforce the system to check the compliance when someone requests them. We propose to define *event access* using widely accepted Semantic Web standards such as OWL, RDF, and XML.

```
<owl:Class rdf:type="#event">
<owl:DataTypeProperty rdf:ID="timestamp"/>
      <rdfs:domain rdf:resource="#time"/>
       <rdfs:range rdf:resource="&xsd;time"/>
<owl:DataTypeProperty rdf:ID="location"/>
      ...
</owl>
<event id="8739173917">
<timestamp> 2009.2.18.12:50pm </timestamp>
      <location> x: xxxx; y: yyyy </location>
      <type> bathroom visit </type>
      <physical quantities type="hit">
      <dooropen newton> 5 </dooropen newton>
      <velocity> 4 m/h </velocity>
      <duration> 10 m </duration>
      </physical quantities>
</event>
<Rules>
      <Rule type="home">
      <condition type="event">
      <before> B ← all other events detected
      within 1 hour </before>
      <present> P ←  Newton > 10 v velocity >
      10 m/h v duration > 30m </present>
      </condition>
      <action> report injury ← P v (B =
      abnormal) </action>
      </Rule>
</Rules>
<Access>
<action> check role of requester R ←
Requester </action>
      <Access type="bathroom">
      <condition duration="9a-5p">
      <allowed> ALL(TRUE) </allowed>
      </condition>
      <condition duration="6p-12a&12a-8a">
      <allowed> Doctor(R) and Nurse(R)
      </allowed>
      </condition>
      <condition assistance="robot">
      <allowed> on call ← injury </allowed>
      <denied> video ← false </denied>
      </condition>
</Access>
```

Figure 3. Sensitive Event

Figure 3 gives an example of the access description of a *bathroom-visit event*. This event is characterized by 3 or more parameters, such as, the type of force used to open the bathroom door, the walking speed to the bathroom door, and the duration of the bathroom visit. The *event access* description is to be executed when such an event has been requested. In this example, only doctors and nurses can see the event records during the night and an assistant robot is allowed to be on call in case there is an accident associated with this event.

5. EVENT VISUALIZATION (ZSCOPE)

As part of our framework, there is a front-end information visualization interface (ZSCOPE) that summarizes a history of key current and past events taking place. We will use the Heracleia Apartment (HA) of the Human Centered Computing Laboratory for the study of device training in Figure 4. ZSCOPE will be used to help a human operator or user locate active devices, their function and priority and provide ways to interact with them. Ongoing empirical studies collect human activity data from HA in order to: (1) to understand needs for human assistance and how the devices and instruments could be involved; and (2) to track the visualization and training needs of human operators.

Figure 4 The Heracleia Apartment

ZSCOPE is also used to illustrate the interactive nature of the EI mechanism and the generated events regarding patient behavior, or worker performance. The events will tell whether the patient follows the medical instructions and how well such a worker performs his tasks. Mistakes need to be corrected. Training regarding new rehabilitation or new devices that were added, will predict risk or neglect.

6. CONCLUSIONS

This paper proposed an event identification (EI) framework with a multi-level data to knowledge approach on multimodal human activity data in complex physical pervasive environments, converting low level data semantics to high level, with privacy and security built-in. In addition, events are

visualized in this framework through an information visualization interface, ZSCOPE.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation under award number CT-ISG 0716261. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Zhengyi Le, Matt Bishop, and Fillia Makedon, Strong Mobile Device Protection from Loss and Capture Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[2] Alan Bowling, Zhengyi Le, and Fillia Makedon, SAL: A simulation and analysis tool for assistive living environments Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[3] Kyungseo Park, Eric Becker, Jyothi K. Vinjumur, Zhengyi Le, and Fillia Makedon, Human Behavioral Detection and Data Cleaning in Assisted Living Environment using Wireless Sensor Networks Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[4] Eric Becker, Zhengyi Le, Kyungseo Park, Yong Lin, and Fillia Makedon, Event-based Experiments in an Assistive Environment using Wireless Sensor Networks and Voice Recognition Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[5] Eric Becker, Gutemberg Guerra-Filho, and Fillia Makedon, Automatic Sensor Placement in a 3D Volume Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[6] Eric Becker, Vangelis Metsis, Roman Arora, Jyothi Vinjumur, Yurong Xu, Fillia Makedon, SmartDrawer: RFID-Based Smart Medicine Drawer for Assistive Environments Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[7] Yong Lin, Eric Becker, Kyungseo Park, Zhengyi Le, Fillia Makedon, Decision Making in Assistive Environments using Multimodal Observations Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

[8] Roman Arora, Vangelis Metsis, Rong Zhang and Fillia Makedon, Providing QoS in Ontology Centered Context Aware Pervasive Systems Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'09), Corfu, Greece, June 9-13, 2009.

2008

[9] Eric Becker, Yurong Xu, Steven Ledford, Fillia Makedon, A wireless sensor network architecture and its application in an assistive environment Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Athens, Greece, July 16-18, 2008.

[10] Eric Becker, Yurong Xu, Heng Huang, Fillia Makedon, Requirements for implementation of localization into real-world assistive environments Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Athens, Greece, July 16-18, 2008.

[11] Vangelis Metsis, Zhengyi Le, Yu Lei, and Fillia Makedon, Towards An Evaluation Framework for Assistive Environments Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Athens, Greece, July 16-18, 2008.

[12] Zhengyi Le, Yi Ouyang, Yurong Xu, and Fillia Makedon, Mobile Device Protection against Loss and Capture Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Athens, Greece, July 16-18, 2008.

[13] Yi Ouyang, Yurong Xu, Zhengyi Le, Guanling Chen, and Fillia Makedon, Providing Location Privacy in Assisted Living Environments Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Athens, Greece, July 16-18, 2008.

Safety Enhancements of Home Lift, Position, & Rehabilitation (HLPR) Chair

Janusz Zalewski[1)] **Cristy Csavina**[2)] **Roger Bostelman**[3)]
Dahai Guo[1)] **Jim Sweeney**[2)] **Kenneth Kirsner**[4)]

[1)] Dept. of Computer Science
[2)] Dept. of Bioengineering
[4)] School of Nursing
Florida Gulf Coast University
Ft. Myers, FL 33965-6565

[3)] Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

1. Introduction

Recent studies show [1] that "In 2000, people aged 65 and older made up 12.3 percent of the U.S. population, while by 2030, they will constitute 19.2 percent, after which growth is projected to level off so that this cohort represents 20.0 percent of the population in 2050. The most rapid growth will occur within a subgroup of this cohort—the so-called 'oldest old,' or people over the age of 80. Today this group makes up 3.2 percent of the U.S. population, while by 2030 that number will increase to 5.0 percent, and by 2050, to 7.2 percent". This group is subject to both physical and cognitive impairments. Such situation will have a profound impact on maintaining the elderly independent from caregivers.

As stated in [2], mobility is fundamental to health and social integration of human beings, and therefore is viewed as being essential to the outcome of the rehabilitation process of wheelchair dependent persons. It is estimated that some 2.5 million people in Europe and 1.25 million in the US depend upon a wheelchair for their mobility. Equally important as wheelchairs are the lift devices. As far as assistive technology for the mobility impaired including the wheelchairs, lift aids and other devices, is well established, the patient typically requires assistance to use the device. With more and more elderly, there is a need for improving these devices to make them more intelligent to ensure them independent assistance. The need for patient lift devices will also increase as generations get older.

In response to these needs, the National Institute of Science and Technology (NIST), Intelligent Systems Division, began the Healthcare Mobility Project to address this healthcare issue of patient lift and mobility, and began developing the Home Lift, Position, & Rehabilitation (HLPR) chair to investigate these specific areas of mobility, lift and rehabilitation [3]. The prototype of the HLPR chair, shown in Figure 1, is based on a manual, off-the-shelf forklift. The forklift includes a U-frame base with casters and a vertical frame. The patient seat mechanism is a double, nested and inverted L-shape where the outer L is a seat base frame that provides a lift and rotation point for the inner L seat frame. The outer L is bolted to

the lift device while the inner L rotates with respect to the seat base frame at the end of the L. Drive and steering motors, batteries and control electronics along with their aluminum support frame provide counterweight for the patient to rotate beyond the wheelbase. When not rotated, the center of gravity remains near the middle of the HLPR Chair. When rotated to π rad (180 deg.) with a 136 kg (300 Lb) patient on board, the center of gravity remains within the wheelbase for safe seat access.

Fig. 1. HLPR Chair [3].

The HLPR chair exists as a prototype and is being used for research purposes at the Florida Gulf Coast University (FGCU). A significant number of unresolved issues are being researched. The objective of this particular project is to investigate the computer and software safety issues in the design, implementation and use of the HLPR chair.

2. Specific Goals of the Project

One particular problem is that currently there are no standards, or even adequate research, to guide developers and manufacturers regarding intelligent rehabilitation chairs and forklift technologies that use advanced sensors, computers and actuation systems. There is a strong sense that before

intelligent chairs are commercialized and sold to the general public, a research based target safety practice should be in place. Our study is meant to address this gap.

The issue of primary importance is the patient safety. The chair is designed toward enabling safe mobility to all users, including all sorts of impairments that may include blindness, paralysis, Alzheimer's, obesity, etc. Because the HLPR Chair's ultimate purpose is to help disabled persons, it is important to remember the human impact during all of these projects. For safety purposes, humans should be assumed unreliable and unpredictable, thus safety studies must expect the extraordinary in regards to the functions to which the user will subjugate the chair.

As an example, in current design, when the patient is using joysticks to control the move and turn the chair, steering wheel design allows stopping the chair at just beyond 180 degrees for safety of the steering system. Steering is reverse Ackerman controlled as joystick left rotates the drive wheel counterclockwise and joystick right rotates the drive wheel clockwise [4]. The steering rotation amount can be limited by the amount of drive speed so as not to roll the frame during excessive speed with large steering rotation. This raises several important questions, however, regarding the implementation of safety functions in software. Some of the most critical issues include: chair stability (is there sufficient safety margin for abrupt rotations and tilt adjustment?), chair mobility (is the chair traceable, so it could be remotely controlled in case of a hazard?), software itself (is there enough protection embedded in software to keep the patient safe in case of equipment failures?). Some of these issues are discussed in the next section.

3. Selected Issues Related to HLPR Chair Safety

3.1 Tilt Awareness and Stability

Some safety standards for the wheelchairs and forklifts already exist [5], and are implemented in current design. However, incorporating simple safety measures, such as electronic level detection and sensors to determine the chair lift's current height into the HLPR software, not allowing the user to take the chair past a predetermined angle, are necessary but not sufficient for full patient protection. This project is taking it a step further.

Under existing collaboration with NIST, testing for the HLPR chair's stability in load carrying situations has been conducted [6]. The research was aimed at looking for discrete angle of tip in the most and least stable configuration (Fig. 2). Several factors were included in the analysis, such as: load/lift height, load orientation, HLPR orientation on platform.

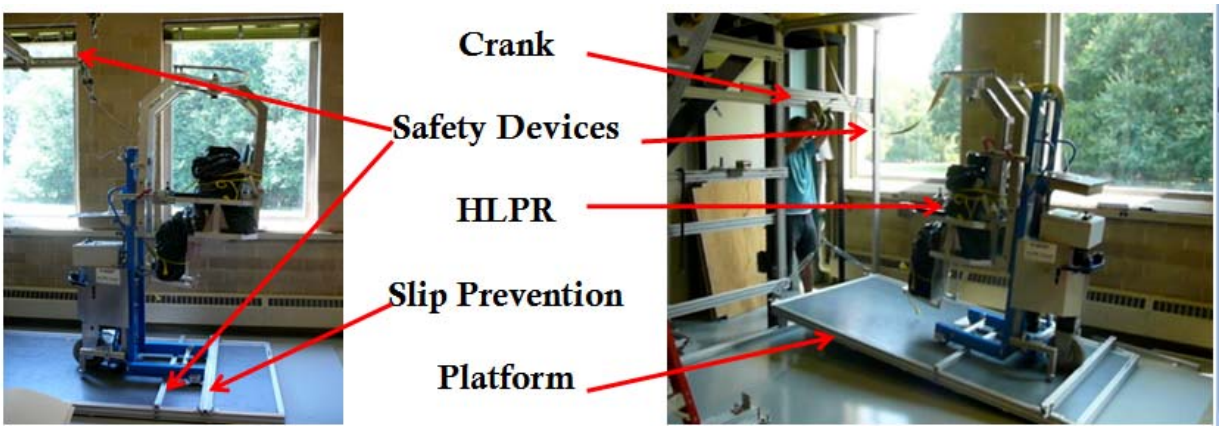

Fig. 2. Illustration of Basic Chair Safety Issues [6].

Several areas for stability testing were identified, including those listed below, and will be addressed in this research by developing respective stability algorithms and implementing them in software: forward and rearward dynamic stability on ramp, lateral dynamic stability on ramp, lateral dynamic stability while turning in circles, lateral dynamic stability while turning suddenly, dynamic stability while traversing a step

3.2 Autonomous Motion

By incorporating a new controller feature of near real-time validation and execution, the HLPR chair could be made autonomous. In particular, previous studies have determined that the incorporation of RFID tags on the HLPR chair along with RFID readers in a building, would allow tracking of any HLPR chair and its user within a designated area. This implementation could also serve as a safety measure for prohibiting entrance to certain areas and automatically unlocking certain doors. However, full safety analysis regarding potential hazards and failure modes requires more significant attention, with specific requirements coming from the nursing objectives and is the subject of this study.

At this point, we developed a preliminary RFID tracking system that allows: (1) collecting the RFID tag data with search abilities, and (2) making the data available via the Internet (Fig. 3). The software allows a remote access to a server and pulling from it logs of what tags have passed through and when [7]. It has a user-friendly interface and socket-protocol accessibility, and will be used as a basis for the HLPR safety analysis with RFID.

Fig. 3. RFID Device Employed in the Project (left) and the User Interface (right).

3.3 Software Safety

Software safety analysis is typically done by identifying potential hazards that may be caused by software failures. Analyzing software architecture is very helpful, in this respect, because it identifies the major components that may be potential sources of such hazards. The architecture of current HLPR chair controller is based on the RCS concepts outlined in [8] and illustrated in Figure 4. It does not, however, include any safety features.

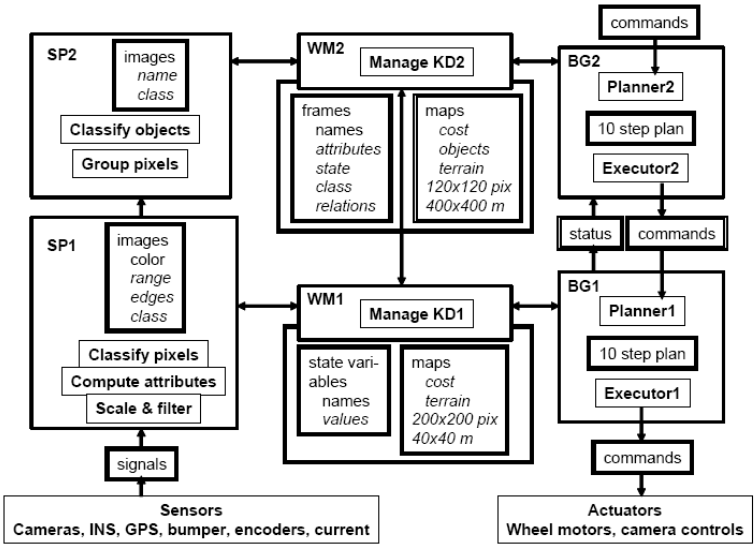

Fig. 4. RCS Architecture Used in HLPR Chair Controller Design [8].

To address the software safety issues we employ the concept of safety shell (Fig. 5), developed in collaboration with NASA [9], which relies on an architectural concept similar to that of RCS architecture [10] and fits well into the RCS scheme, enabling design of control systems [11]. Its essential element is the implementation of a "Test First" design element to prevent dangerous situations from occurring. In case of the HLPR chair, this design element is initiated with every change in input from the user and encoder. It is meant to catch any hazardous situation at its beginning; by "testing first" the processor will either validate or invalidate the current motion and/or the desired motion.

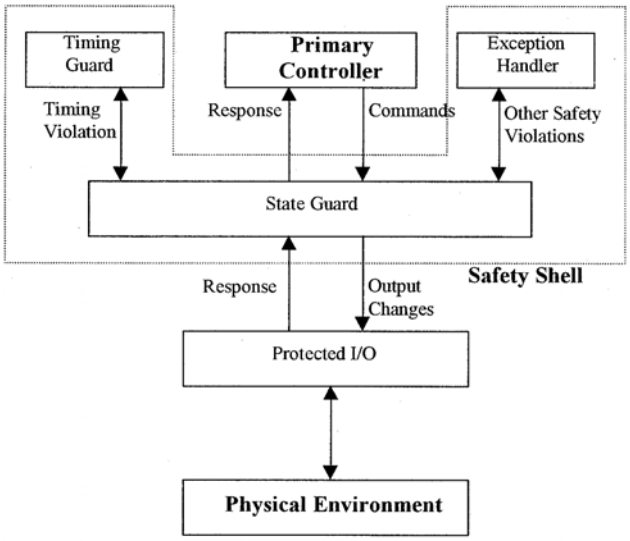

Fig. 5. Safety Shell Architecture [9].

The safety shell for the HLPR chair is being implemented in the MOAST/UsarSims environment [12], which allows for virtual simulation, thereby simplifying safety testing for mapping and planning operations. Software safety requirements have been developed for this project, including requirements for external interfaces, input requirements, output requirements, processing requirements, and performance requirements, for all system modes and user classes. A sample of input requirements is given below:

- Software shall accept input from the designated sensors/user input device.
- Software shall validate or invalidate an input according to the environment.
- Software shall accept changing inputs from sensors/user input device.
- Software in the failsafe mode shall be able to override sensor/user input devices.

Conclusion

The HLPR chair was designed at NIST to be a revolutionary patient lift and mobility system for wheelchair dependents, the elderly, stroke patients, and others requiring personal mobility and lift access.

The system shows promise for moving these groups of patients into the work force and removing the burden placed on the healthcare industry. It has been prototyped to show the basic concept of such a patient lift and mobility system. However, the complete development of the HLPR chair from its current state all the way to its realization as an assistive technology in the hospital setting requires a significant additional work in several areas, including computer and software safety.

The current proposal addresses the safety issues in a comprehensive, interdisciplinary way, and aims at developing a safety model and its verification, to enable subsequent steps towards chair certification by the U.S. Food and Drug Administration (FDA).

The multidisciplinary approach is intended to cover all aspects of the complex problem of ensuring chair safety, both at the product level (to reconcile discrete safety assurance algorithms and continuous algorithms to ensure stability) and at the process level (to address domain requirements originating from computing, bioengineering and nursing).

References

[1] Pollack M., Intelligent Technology for an Aging Population, *AI Magazine*, pp. 9-24, Summer 2005

[2] van der Woude L.H.V. et al. (eds.), *Biomedical Aspects of Manual Wheelchair Propulsion: The State of the Art II*, IOS Press, 1999

[3] Bostelman R., J. Albus, HLPR Chair – A Service Robot for the Healthcare Industry, *Proc. 3rd Int'l Workshop on Advances in Service Robotics*, Vienna, Austria, July 7, 2006

[4] Bostelman R., J. Albus, Sensor Experiments to Facilitate Robot Use in Assistive Environments, *Proc. PETRA2008, 1st ACM Int'l Conference on Pervasive Technologies Related to Assistive Environments*, Athens, Greece, July 16-18, 2008

[5] ANSI/ITSDF Std B56.5-2005, *Safety Standard for Guided Industrial Vehicles and Automated Functions of Manned Industrial Vehicles*, 2004

[6] Johnson J. *Development of Static and Dynamic Stability Test Standards for A Load Carrying Device*, Project Report, Florida Gulf Coast University, Ft. Myers, FL, 2008

[7] Gallegos J., *Applied RFID Technology*, Project Report, Florida Gulf Coast University, Ft. Myers, FL, 2008

[8] Albus J.S., Barbera, A.J., RCS: A Cognitive Architecture for Intelligent Multi-agent Systems, *Annual Reviews in Control*, Vol. 29, No. 1, pp. 87-99, 2005

[9] van Katwijk J., H. Toetenel, A.K. Sahraoui, E. Anderson, J. Zalewski, Specification and Verification of a Safety Shell with Statecharts and Extended Timed Graphs. *Proc. SAFECOMP 2000, Int'l Symp. on Computer Safety, Reliability and Security*, Springer-Verlag, 2000, pp. 37-52

[10] Zalewski J., Real-Time Software Architectures and Design Patterns: Fundamental Concepts and Their Consequences, *Annual Reviews in Control*, Vol. 25, No. 1, pp. 133-146, July 2001

[11] Sanz R., J. Zalewski, Pattern-Based Control Systems Engineering, *IEEE Control Systems*, Vol. 23, No. 3, pp. 43-60, July 2003

[12] Scrapper C., S. Balakirsky, E. Messina, MOAST and USARSim – A Combined Framework for the Development and Testing of Autonomous Systems, *Proc. SPIE Defense and Security Symposium*, Orlando, FL, April 17-21, 2006

Technical Session II

Medical Device Plug-and-Play Interoperability

Session Chair:

Julian Goldman
Massachusetts General Hospital

A Concept for a Medical Device Plug-and-Play Architecture based on Web Services

Stephan Pöhlsen*, Stefan Schlichting†, Markus Strähle‡, Frank Franz†, and Christian Werner*

*Institute of Telematics – University of Lübeck
Ratzeburger Allee 160 – D-23538 Lübeck, Germany
Email: {poehlsen|werner}@itm.uni-luebeck.de

†Research Unit – Drägerwerk AG & Co. KGaA
Moislinger Allee 53–55 – D-23542 Lübeck, Germany
Email: {stefan.schlichting|frank.franz}@draeger.com

‡Dräger Medical AG & Co. KG
Moislinger Allee 53–55 – D-23542 Lübeck, Germany
Email: markus.straehle@draeger.com

Abstract

Medical device interoperability is still an issue. Standards exist only for specific areas like HL7 and DICOM, or have not been widely adopted like ISO/IEEE 11073 except for the domain information model at the semantic level. An approach that covers interoperability below the semantics is proposed. It is based on Web services which are widely accepted outside the medical device application domain. In particular the architecture is build on the upcoming Device Profile for Web Services (DPWS). It is a collection of existing Web services specifications for service discovery, interface description, event notification, and security. It is designed for resource-constrained devices and thus seems to be suitable as a basis for medical device plug-and-play.

1. Introduction

Interoperability is defined as “ability of two or more systems or components to exchange information and to use the information that has been exchanged” [1]. According to Lesh et al. [2] the interoperability continuum distends from the least complex endpoint of physical interoperability to the most complex of data interoperability. To achieve data interoperability not only the capability to exchange information without an error has to be given, but also correct interpretation of the information to use it in an algorithm. Benefits of medical device interoperability range from less development time for data-driven clinical decision support algorithms or medical device safety interlocks to improved patient safety. Many initiatives like the MD PnP Interoperability program [3] or IHE PCD [4] also address this problem.

The ISO/IEEE 11073 is a mature standard addressing the interoperability issue. Especially the provided domain information model is used frequently and enables medical devices to exchange data on the semantic level. The lower layers are complex and do not support current technologies like Ethernet or TCP/IP.

MediCAN is a solution proposed by McKneely et al. [5] and is described as a vendor-independent network architecture for interfacing medical devices based on proprietary protocols that are built on top of UDP/TCP as well as CAN-Bus. There is no direct communication between the devices and the access to the closed network of medical devices is controlled by a proxy server.

Another approach establishes interoperability between components of imaging systems built from OEM-components using special purpose CANopen device profiles. There is also a CANopen based application profile addressing acute care systems [6].

1.1. Service-oriented Architecture

The idea of service-oriented architectures (SOA) is coming from the business environment. The operation and maintenance of computer systems there is complex and thus very costly. Each time business processes change, new computer systems have to be integrated in the existing infrastructure. Often, these new computer systems have new interfaces and an integration with old systems is problematic.

A SOA solves this problem by introducing a new standardized interface technology for all systems, called “service”. All services are now communicating with their service interface in a common messaging

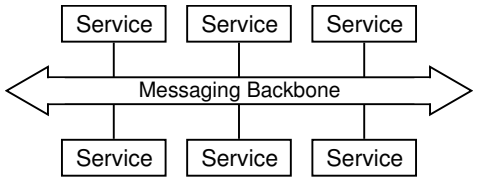

Figure 1. In a SOA all services communicate with each other over a common messaging backbone. The realization of this abstract backbone depends on the concrete SOA implementation. It can be a network or a central communication server for example.

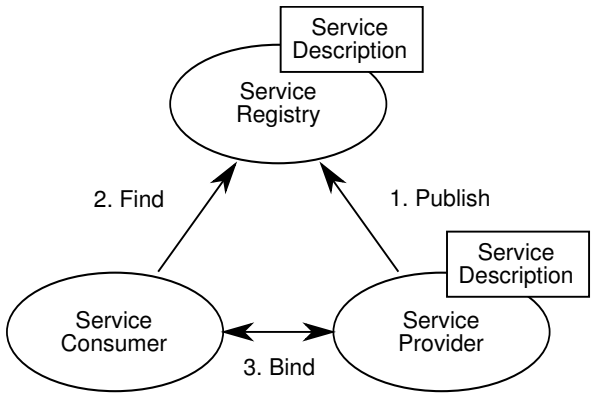

Figure 2. The three primary roles in a service-oriented architecture with the interaction among each other

backbone according to Figure 1.

A SOA does not dictate a specific technology, it is just a concept how to build such an architecture. Figure 2 depicts that concept with the three primary roles: service provider, service registry, and service consumer. A service provider publishes its service description to a service registry. Then, a service consumer is able to find a corresponding service in the registry. Afterwards, the service consumer binds to the service provider to use that service.

Web services are a realization of a SOA with Internet technologies [7]. The benefit of Web services in contrast to other SOA solutions is that it uses well-known technologies and a vendor lock-in can be avoided.

The fundamental and widely accepted standards on which Web services are build are *SOAP* [8] and the *Web Services Description Language (WSDL)* [9]. WSDL covers the service description in the role model, while SOAP is an XML-based messaging transfer format. Together WSDL and SOAP cover the roles of the service provider and the service requester.

The registry role is not covered by these two standards, but by the *Universal Description Discovery &*

Integration (UDDI) [10] that specifies an interface for such a registry. However, the standard is not so widely accepted as WSDL and SOAP. This is owed to the fact that the main idea behind UDDI to build one global service registry failed.

1.2. First Architecture Concepts

The architecture developed in the FUSION project, funded by the German Federal Ministry of Education and Research, is built upon Web services to realize a SOA. Two approaches have been explored: a complete centralized and a more decentralized approach. The latter to overcome the problem of a single point of failure and due to performance issues.

The centralized approach utilizes an enterprise service bus (ESB) to provide a common data-centric middleware structure for reliable transport and traceability of synchronous and asynchronous messages [11]. The distributed approach does not depend on a central structure, but more on standards from the Web service domain. Lookups of services are handled by a UDDI server. Web services of a provider are invoked directly by a consumer without sending the request/response through the ESB. Furthermore, events are transported using the WS-Eventing specification: either point-to-point or via an event broker. In most cases, point-to-point event notification will be used, if the event is safety critical and could not be send via a centralized event broker. This approach yields more flexibility, less configuration effort, and – by far the most important – communication between devices will not break down, even if one of the basic service components crashes. With regard to semantic interoperability, both approaches employ an XML vocabulary based on the ISO/IEEE 11073 domain information model for hemodynamic and respiratory data as well as HL7 for ADT data.

The problem with both solutions – even if reduced for the latter one – is the usage of centralized structures and hence the introduction of a single point of failure as well as scalability issues. Furthermore some effort has to be expended for configuration of the medical devices with the result that no plug-and-play feeling comes up.

For that reason, an architecture based upon current Web service standards is presented that is locally de-centralized and only needs central components if communication over subnet boundaries is needed. This concept is described in the following section.

2. Proposed Architecture

The architecture proposal for medical device connectivity leverages the upcoming *Devices Profile for Web Services (DPWS)* standard [12] that defines services for discovery, interface description, messaging, event propagation as well as secure information transmission. It is out of scope of the proposed architecture to define the semantic side of interoperability. This can be handled using data models from mature standards like HL7 or ISO/IEEE 11073.

DPWS – becoming an OASIS standard in June 2009 – is a minimal set of Web services specifications for resource-constrained devices. It includes discovery and description of Web services as well as the possibility for event propagation. The origins of DPWS are in the consumer electronics domain where it is used in modern network printers or image scanners to allow plug-and-play. To sum up, DPWS achieves the same ease of use for consumer electronics for Ethernet as USB does for serial connected devices. It is pushed forward by Microsoft for future printer integration and consequently Microsoft Windows Vista, Windows Server 2008, and Windows Embedded CE 6.0 R2 already have a native DPWS stack (called WSDAPI) on-board.

The services of the proposed architecture are discussed in detail below.

2.1. Dynamic discovery

Web Services Dynamic Discovery (WS-Discovery) [13] is a service localization protocol and will be standardized in the context of the DPWS standardization process. By default it operates in an ad-hoc mode without any configuration. Therefor it uses a predefined multicast group to reach all services within the same sub-network. Multicast is a transmission mode where the underlying network distributes the data to all subscribed nodes. The discovery functionality is similar to the probably known discovery procedure in SoHo (Small Office Home Office) firewall appliances or in media center systems that distribute music and video in home networks. In the managed mode the discovery process uses a centralized component called discovery proxy that caches all information. With this proxy the discovery process scales to a larger number of endpoints since the usage of multicast is reduced to a minimum in contrast to the ad-hoc mode.

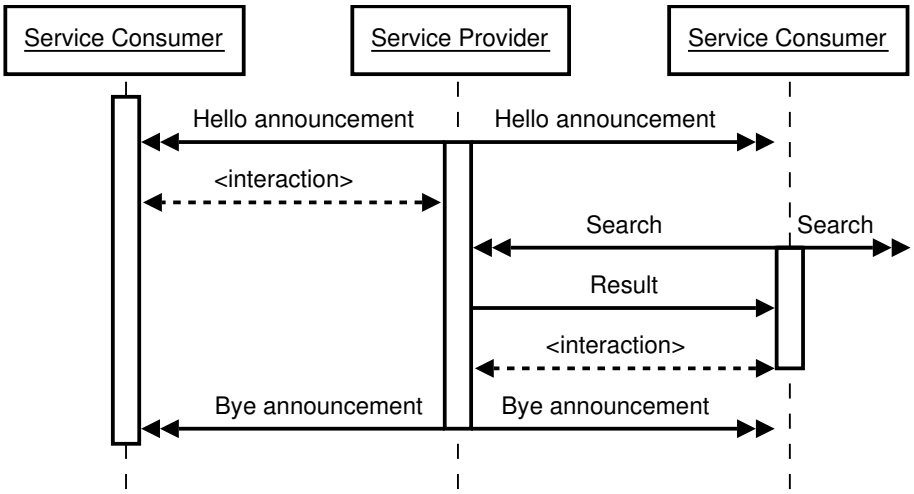

Figure 3. WS-Discovery sequence diagram for two discovery procedures: The service consumer on the right searches for services, while the consumer on the left waits for announcements.

The decentralized nature of the ad-hoc mode is applicable to the application domain of medical devices, since it avoids a single point of failure. It is important to have a robust discovery system in case of network failures outside the currently interacting devices. Medical devices in an operation room for example should function well in case of network failures outside their room. This scenario forbids a sole centralized registry for discovery.

In WS-Discovery service providers announce themselves when they join the network. It avoids the need for polling in service consumers for the same service periodically. With this feature it is possible to automatically update lists of available services or directly respond in case a new service appears in the network.

In Figure 3 two discovery procedures are shown. First, the service consumer on the left side is already running while the service provider starts up. The service consumer receives the hello announcement of that service provider and thus is able to directly start the interaction without the need for periodically polling for new services. In the second discovery sequence with the service consumer on the right, the service provider started before the consumer. Thus, the consumer missed the hello announcement and must search for that service.

WS-Discovery is designed for use within a single subnet. The multicast announcements and search requests do not scale very well, because all messages have to be sent to all WS-Discovery nodes in the network. Also the specification restricts the usage to a single subnet. Thus, this specification seems to be

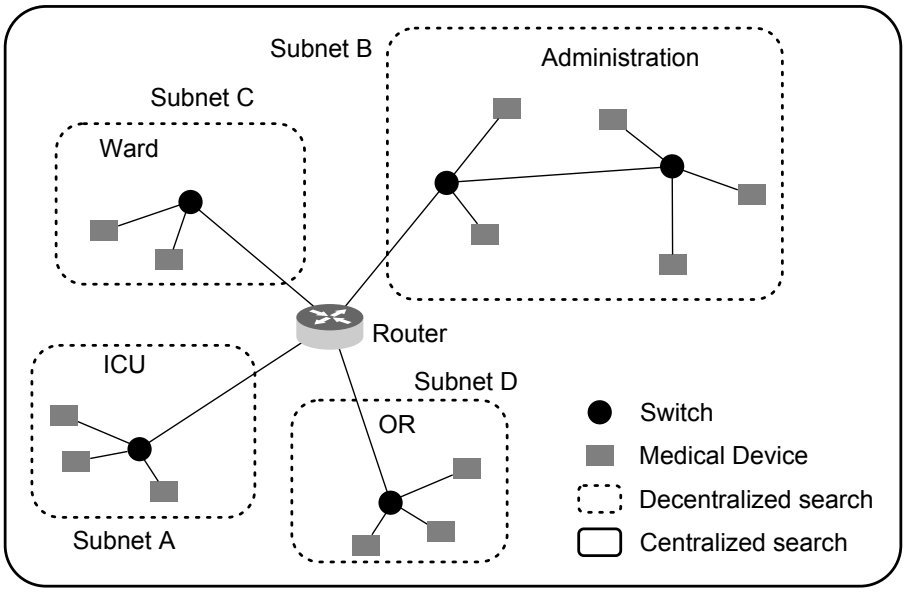

Figure 4. Two different discovery layers: Within the same subnet services can be found decentralized, while in the whole network a centralized scheme is necessary to go across subnet boundaries.

impractical for use in enterprise or hospital networks. In such large networks it is much more efficient to have a centralized component for discovering services.

On the one hand the reliability of the decentralized multicast discovery is required on the other hand an enterprise scale discovery is required. An approach to solve this conflict is to use both variants in combination with each other. For the managed mode WS-Discovery specifies a so-called discovery proxy. Originally it is used to cache all available information from the local service providers and response to search requests from service consumers within the same subnet. Instead of using this discovery proxy to reduce multicast traffic in subnets it can be used as a central component to search for services hospital wide – outside the current subnet. A benefit of using a WS-Discovery proxy server instead of a UDDI server is that the discovery process is more consistent for the architecture.

The proposed feasible 2-layer discovery architecture [14] is shown in Figure 4. The WS-Discovery compliant decentralized multicast discovery within a subnet is as robust as possible against network failures. It is sufficient, that a network connectivity between service provider and consumer exist. Practically this is no restriction, since the two services cannot communicate further. To discover services in the whole network it is necessary to query a centralized discovery proxy.

Different approaches are possible to reach the centralized discovery proxy. In [14] the DHCP protocol

is used to transfer the IP address of the discovery proxy server to the service consumers. DHCP offers the possibility to add vendor specific data on the server side. Then DHCP clients are able to request this data. Another approach [15] is to use the domain name system. Here, the clients query the name servers for the IP address of the discovery proxy. The DNS based solution has the benefit of being easier to implement platform independent and has to be configured only in one name server instead of all subnet limited DHCP servers.

2.2. Event-driven Architecture Concepts

Web services started with typical request-response scenarios, where a client requests a service from the Web service provider and gets a response. A typical example for this are use cases from travel agencies where a flight and a hotel must be booked for a customer. Events are disregarded in this scenario.

In the medical application domain a lot of communication is event driven. For example, alarms were sent and real-time data is transmitted. That means that publish-subscribe has to be supported in addition to the typical request-response pattern for device control.

In IP-based networks two different solutions for publish-subscribe exists in general. The first one uses a dedicated point-to-point connection for each subscriber; while the second solution utilizes the functionality from the underlying network using UDP multicast.

For the first solution with n point-to-point connections for n subscribers different Web services specifications exist that were pushed by different companies: WS-Eventing (Microsoft) [16], WS-Notification (IBM) [17], and WS-Events (HP) [18]. In March 2006 they released a joint white paper to harmonize the existing specifications [19]. For event propagation it results in a new specification called WS-EventNotification that builds on WS-Eventing and has to be specified in the future. It seems to be best practices to use WS-Eventing. It will also be referenced in the DPWS standard. In WS-Eventing the event source can delegate the management of subscriptions to and distribution of events to another Web service. This is practical for scenarios where the event source cannot or should not handle the list with all subscriptions.

The second solution is based on the multicast functionality from the underlying network. In this case, messages are sent via UDP multicast, which results in only one message transmission from the publisher to the multicast address. The distribution to the recipients is managed by the network routers. Multicast is

designed from the group up for such distribution scenarios that is why it scales better than point-to-point transmission. The required specification to employ multicast messaging in Web services is SOAP-over-UDP [20] which is also included in the OASIS standardization process of DPWS.

2.3. Security

For sensitive data and control commands information security is important. In the context of medical device connectivity data integrity, confidentiality, availability, and non-repudiation is interesting for risk management.

Data integrity refers to the validity of data. The integrity can be harmed by accident or maliciously. Confidentiality means that the data is only accessible to those that are authorized to have access. Availability stands for a system that has all information available that are needed to serve its purpose. In the considered context of medical device interoperability, it is important to have a correctly functioning communication channel. Non-repudiation is required to ensure that a party in a dispute cannot repudiate a message that it sent out. This might be interesting for control commands.

Information security can be achieved on transport layer or at message level. For transport layer security a secure channel is established for end-to-end communication. This secure channel provides data integrity and confidentiality as long as both endpoints authorize each other at the beginning. HTTPS is an example of transport layer security that uses the TLS (Transport Layer Security) protocol which is the successor of SSL (Secure Socket Layer) [21].

For message level security each message is secured individually instead of sending unsecured messages through a secure channel. The benefit of message level security is that the message can be passed through insecure nodes. Furthermore, non-repudiation is achieved on the message level utilizing digital signatures. In this case the signed messages must be logged for later proof of message transmission However, the message level security is more complex in contrast to the transport layer security.

For transport as well as message security a public key infrastructure (PKI) is a basic requirement for medical device authentication. A centralized approach with user name and password is inapplicably. It contradicts to the plug-and-play nature, introduces a single point of failure, and nevertheless requires a certificate to authenticate the server to the clients.

Availability for Web services highly depends on the underlying network. Ethernet is a best effort

network that supports prioritization. Anyhow, a network node with unsocial behavior might disturb other data transmission and important data packets might get dropped in overloaded switches. For hard real-time constraints proprietary Ethernet modifications exists. These are not compatible with Web services. A tradeoff between normal Ethernet and proprietary solutions can be a fair network switch [22] that equally shares the available data rate between sending nodes. Thus the effect of unsocial nodes can be reduced and a minimal data rate can be guaranteed.

3. First Feasibility Evaluations

First feasibility evaluations have been made with demonstrator programs on standard computer hardware. For WS-Eventing a few performance measurements were made to gain experience for multiple point-to-point transmissions. They were done with the DPWS toolkits WS4D-gSOAP and WS4D-JavaME [23]. The results are summarized in Table 1.

Table 1. Exemplary performance results for event distribution according to WS-Eventing for different toolkits

Event source	**Listener**	**Total delay**
WS4D-gSOAP	1 × WS4D-gSOAP	3.0 ms
WS4D-gSOAP	2 × WS4D-gSOAP	5.7 ms
WS4D-gSOAP	1 × WS4D-JavaME	3.8 ms

The performance measurements showed for a WS4D-gSOAP device, that for one WS4D-gSOAP listener it takes 3.0 ms to transmit the event and receive an acknowledgement. For two listeners the event is delivered after 5.7 ms. This time does not only depend on the devices implementation it also depends on the listeners. For one Java listener implemented with the WS4D-JavaME toolkit it took 3.8 ms instead of the 3.0 ms with the WS4D-gSOAP listener.

Table 2. Serialization overhead in WS4D-gSOAP for normal messages and messages with a digital signature for different payloads on PC hardware

Payload	**Normal messages**	**Signature w/o certificate**	**Signature with certificate**
1 Integer	1.3 ms	6 ms	8 ms
50 Integer	2.4 ms	12 ms	14 ms

For measurement of the overhead especially for message level security the WS4D-gSOAP toolkit was used. It is an add-on for the gSOAP toolkit [24] that has support for digital signatures. Table 2 summarizes first results for the serialization time in different scenarios. From the second column it can be concluded that the overhead for the SOAP envelope and the DPWS compliant SOAP header takes 1.3 ms. The serialization of an additional integer takes merely 22 μs. In the third column the whole body of a SOAP message is signed according to WS-Security with a 1024 bit RSA key and the SHA1 hash algorithm. The certificate for the RSA key is not included because it is already known to recipient in this scenario. An additional integer in a signed SOAP message takes 122 μs which is 5 to 6 times the overhead according to the unsigned message. This overhead solely results from the SHA1 hashing in the signature procedure. From the last column it can be concluded that an embedded certificate leads to an overhead of 2 ms for serialization and hashing.

Van Engelen et al. [25] did further performance measurements concerning security overhead. They compared HTTPS transmissions with symmetric and asymmetric message level security.

4. Results

A concept for a plug-and-play architecture was introduced. It is build on current Web services standards and covers the technical layers of interoperability. It is an infrastructure architecture and thus semantics is out of scope and existing standards should be used. To discover other devices an enhanced version of WS-Discovery is used to offer the possibility to reach services outside the current sub-network. For event propagation two different solutions WS-Eventing and SOAP-over-UDP multicast coexist with different advantages. Information security is covered at the transport layer as well as at the message level. First performance tests showed a timing overhead that results from the use of Web services. If security is needed due to safety reasons, this adds of course an additional overhead that cannot be avoided.

5. Discussion

In the medical application domain the proposed DPWS-based architecture seems to provide a basis for device connectivity. It is designed to achieve full plug-and-play capabilities with devices from different vendors. It runs on resource-constrained devices as well as on high-end computer systems. Web services

in general are one of the widest accepted building blocks for a SOA that probably has the most tool support. The reasons for this are supposably its platform and programming language independency in combination with the use of well-known Internet technologies such as HTTP.

Even, if there some reservations concerning the performance of Web services for medical device connectivity. The performed measurements show that they could be used for communication even if low transmission latency is needed.

For event and real-time data propagation SOAP-over-UDP multicast should scale better since multicast was designed for such purposes in contrast to WS-Eventing that tried to add the required functionality on top of existing Web services standards. The exemplary measurements for WS-Eventing support that assumption. The delay depends on the listener implementation and the scheduling strategy of the sender. From a devices perspective the listener implementation can not be influenced. The sender is able to inform the listener in sequential order or concurrently. When transmitting the messages in sequential order the last subscribed listener will be informed with a large delay. In contrast to that a programmatically concurrent transmission is more resource expensive, since multiple threads are required.

Also the UDP multicast has a drawback. It does not support a reliable transport. It is assumed that all transmitted data is received by the listener. Thus, the decision between WS-Eventing and SOAP-over-UDP multicast is not a Web service specific decision.

Anyhow, the major problem in both cases is: What to do with failed transmissions? Queue them and try it x minutes later or just fire and forget an event? All these different aspects lead to the conclusion that it depends on the specific event (i.e., alarm or real-time data) which solution might be more practical.

In case of information security this decision is even more complex. Van Engelen et al. [25] propose to use HTTPS whenever it is possible since it performs much better then message level security. It is only possible to use HTTPS in conjunction with WS-Eventing, because of the nature of multicast transmissions that require message level security.

A de-centralized plug-and-play architecture like the proposed one might have the best chances to smoothly migrate into the existing infrastructure since it does not require complex components – two modern devices are sufficient.

References

[1] *IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries*. New York, NY, USA: IEEE Computer Society Press, Jan 1991.

[2] K. Lesh, S. Weininger, J. M. Goldman, B. Wilson, and G. Himes, "Medical Device Interoperability-Assessing the Environment," in *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2007, pp. 3–12.

[3] "MD PnP Program." [Online]. Available: http://mdpnp.org/MD_PnP_Program.html

[4] "IHE Patient Care Device Domain." [Online]. Available: http://www.ihe.net/pcd/

[5] P. K. McKneely, F. Chapman, and D. Gurkan, "Plug-and-Play and Network-Capable Medical Instrumentation and Database with a Complete Healthcare Technology Suite: MediCAN," in *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2007, pp. 122–130.

[6] R. Zitzmann and T. Schumann, "Interoperable Medical Devices due to Standardized CANopen Interfaces," in *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2007, pp. 97–103.

[7] M. P. Papazoglou, *Web Services: Principles and Technology*. Pearson Education, 2008.

[8] W3C, "SOAP Version 1.2," Apr 2007. [Online]. Available: http://www.w3.org/TR/soap12/

[9] W3C, "Web Services Description Language (WSDL) 1.1," Mar 2001. [Online]. Available: http://www.w3.org/TR/wsdl

[10] OASIS Open, "UDDI version 3.0.2," 2004. [Online]. Available: http://uddi.org/pubs/uddi_v3.htm

[11] M. Strähle, M. Ehlbeck, V. Prapavat, K. Kück, F. Franz, and J.-U. Meyer, "Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications," in *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2007, pp. 153–155.

[12] OASIS, "Devices Profile for Web Services Version 1.1 – Public Review Draft 01," Jan 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/wsdd-dpws-1.1-spec.html

[13] OASIS, "Web Services Dynamic Discovery (WS-Discovery) Version 1.1 – Public Review Draft 01," Jan 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html

[14] S. Pöhlsen and C. Werner, "Robust Web Service Discovery in Large Networks," in *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC '08)*, vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, Jul 2008, pp. 521–524.

[15] S. Pöhlsen, C. Buschmann, and C. Werner, "Integrating a Decentralized Web Service Discovery System into the Internet Infrastructure," in *Proceedings of the 6th IEEE European Conference on Web Services (ECOWS '08)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov 2008, pp. 13–20.

[16] D. Box et al., "Web Services Eventing (WS-Eventing)," Mar 2006. [Online]. Available: http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/

[17] S. Graham et al., "Web Services Notification," Mar 2004.

[18] N. Catania et al., "Web Services Events (WS-Events) Version 2.0," Jul 2003. [Online]. Available: http://xml.coverpages.org/WS-Events20030721.pdf

[19] K. Clineand, J. Cohen, D. Davis, D. F. Ferguson, H. Kreger, R. McCollum, B. Murray, I. Robinson, J. Schlimmer, J. Shewchuk, V. Tewari, and W. Vambenepe, "Toward Converging Web Service Standards for Resources, Events, and Management," Mar 2006. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa480724.aspx

[20] OASIS, "SOAP-over-UDP Version 1.1 – Public Review Draft 01," Jan 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/wsdd-soapoverudp-1.1-spec.html

[21] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Aug 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[22] S. Pöhlsen, F. Franz, K. Kück, J.-U. Meyer, and C. Werner, "Fair-Queued Ethernet for Medical Applications," in *Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications (NCA '08)*. Los Alamitos, CA, USA: IEEE Computer Society, Jul 2008, pp. 152–159.

[23] E. Zeeb, A. Bobek, H. Bohn, S. Prüter, A. Pohl, H. Krumm, I. Lück, F. Golatowski, and D. Timmermann, "WS4D: SOA-Toolkits making embedded systems ready for Web Services," in *Proceedings on the Second International Workshop on Open Source Software and Product Lines 2007 (OSSPL07)*, Limerick, Ireland, Jun 2007.

[24] R. A. van Engelen and K. A. Gallivan, "The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks," in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '02)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2002, pp. 128–135.

[25] R. A. van Engelen and W. Zhang, "An Overview and Evaluation of Web Services Security Performance Optimizations," in *Proceedings of the 2008 IEEE International Conference on Web Services (ICWS '08)*. Los Alamitos, CA, USA: IEEE Computer Society, Sep 2008, pp. 137–144.

A Publish-Subscribe Architecture and Component-based Programming Model for Medical Device Coordination and Integration

Andrew King, Sam Procter*
Dan Andresen, John Hatcliff†, Steve Warren
Kansas State University
{aking,samuel3,dan,hatcliff,swarren}@ksu.edu

William Spees, Raoul Jetley
Paul Jones, Sandy Weininger
US Food & Drug Administration
{William.Spees,Raoul.Jetley,PaulL.Jones,
Sandy.Weininger}@fda.hhs.gov

March 7, 2009

Abstract

Medical devices historically have been monolithic units – developed, validated, and approved by regulatory authorities as stand-alone entities. Modern medical devices increasingly incorporate connectivity mechanisms that offer the potential to stream device data into electronic health records, integrate information from multiple devices into single customizable displays, and coordinate the actions of groups of cooperating devices to realize "closed loop" scenarios and automate clinical workflows.

In this paper, we describe a publish-subscribe architecture for medical device integration based on the Java Messaging Service. We provide a overview of a model-based development environment that we have built for rapidly programming device coordination scenarios. We assess the extent to which this framework is capable of supporting and complementing the Integrated Clinical Environment that has been proposed by the Medical Device Plug and Play Interoperability Project The implementation of this framework is free available and open source. One of the primary goals of the framework is to provide researchers in acadaemia, industry, and government with an open test bed for exploring development, quality assurance, and regulatory issues related to medical device coordination.

1 Introduction

Historically, medical devices have been developed as monolithic stand-alone units. Current Verification and Validation (V&V) techniques used in industry primarily target single monolithic systems. Moreover, the US Food and Drug Administration's (FDA) regulatory regimes are designed to approve single stand-alone devices.

This state of affairs stands in direct contrast to the pervasive integration and cooperation among computing devices in our world today, and it is quite clear that numerous clinical motivations exist to deploy systems of integrated and cooperating medical devices. It is anticipated that medical systems will undergo a paradigm shift to provide functionality such as device data streaming directly into patient electronic health records (EHRs), integration of information from multiple devices in a

*Author's current affiliation: University of Nebraska, Lincoln
†Corresponding Author.

clinical context (e.g. hospital room) into a single tailorable composite display, automation of clinical workflows via computer systems that control networks of devices as they perform cooperative tasks, remote-controlled/robotic surgery, and even automatic construction and execution of patient treatments. Indeed, companies including Cerner, CapsuleTech, Philips/Emergin, Sensitron, and iSirona are bringing to market infrastructure that facilitates streaming of device data into medical records. In addition, large-scale research projects such as the Medical Device "Plug and Play" Interoperability Program [9] (MDPnP), funded by the U.S. Army's Telemedicine and Advanced Technology Research Center (TATRC), are developing standards and prototypes for systems of cooperating devices.

Numerous challenges presently exist that are preventing this vision of deeply integrated and highly beneficial medical systems from being realized. These include: (a) lack of domain knowledge and infrastructure on the part of academic researchers as they seek to develop appropriate V&V technologies, safety-critical system components, and programming models, (b) lack of awareness in industry of formal modeling and verification technologies that could tackle the problems of compositional construction of highly interactive safety-critical systems, and (c) lack of realistic applications of cutting edge V&V and programming technologies in the device integration space that might provide science-based inputs for guiding the formation of new regulatory policy. We believe that only a broad-based community effort of academics, industry, and regulatory officials can solve these interrelated challenges.

Progress must be made on a number of fronts to address the challenges described above.

- Which middleware and integration architectures are candidates to support device integration across multiple interaction scenarios?
- Which programming models are suitable for rapid development, validation, and certification of systems of interacting medical devices?
- What V & V techniques are appropriate for compositional verification of envisioned medical systems, and how can the effectiveness of the techniques be demonstrated so as to encourage adoption among commercial vendors?
- Can existing regulatory guidelines and device approval processes that target single devices be (a) extended to accommodate component-wise approval of integrated systems and (b) established in a manner that encourages innovation and rapid transition of new technologies into the market while upholding a mandate of approving safe and effective technologies?
- What interoperability and security standards are necessary to encourage development of commodity markets for devices, displays, EHR databases, and infrastructure that can support low cost deployment of integrated systems and enable flexible technology refresh?

To facilitate industry, academic, and government exploration of these issues, we are developing an open Medical Device Coordination Framework (MDCF) for designing, implementing, verifying, and certifying systems of integrated medical

devices.

Below we list the design goals for our MDCF.

1. Provide distributed networking middleware infrastructure that enables devices/displays/databases from different vendors to be integrated with minimal effort.
2. Provide payload capabilities that support common data formats used in the medical device and medical informatics domains.
3. Provide an architecture that enables tailoring, integrating, and transforming device information streams.
4. Support the requirements of realistic device integration contexts.
5. Develop an architecture whose performance and application-level programmability scales gracefully as the number of integrated devices and computational resources (e.g. server machines) increases.
6. Provide basic functionality needed for reliability including options for guaranteed message delivery, logging/auditing, and persistent storage of messages.
7. Support a programming model that makes it easy to assemble new functionality from building blocks.
8. Use infrastructure that is freely available and open source (to enable academic research).
9. Use standards-based frameworks that are supported by enterprise-level implementations that can provide suitable performance in a realistic enterprise setting.
10. Because it will be difficult for academics to obtain real devices, support both real and simulated devices.
11. The architecture should enable health care providers to mix and match components from different vendors best suited to meet patient needs, without undue concern for the safety of the resulting system. This should be done in a way that assures a level playing field such that vendors compete on the basis of features and performance.
12. Understand the limitations and safety implications of the architecture to establish risk boundaries.
13. Ultimately, we aim to support the capabilities similar to those called for in the MD PnP project by providing an implementation of a notion of Integrated Clinical Context or similiar capabilities but realized in a component-based integration environment supported by model-driven development [14].

In previous work [7], we overviewed the MDCF, discussed multiple clinical device integration scenarios that it aims to support, and reported on experimental studies that described the performance of the MDCF under computational loads that are similar to what might be encountered in realistic deployments. This paper describes in greater detail the architecture of the MDCF and a MDCF-specific environment for component-based model-driven development of device coordination/integration scenarios. The specific contributions of this paper are as follows:

- Describe a device coordination framework built on top of an open standards based middleware (the Java Messaging System).
- Describe an architectural layer built on top of JMS that provides extended functionality for robustly managing the interaction of medical devices, and which supports a model based programming model.
- Explain a model-based development process for the development of device coordination scenarios and components. We present an Eclipse-plugin based on our Cadena tool [2] which directly supports and aides the development of medical device coordination scenarios.
- Summarize experiments which indicate that the architecture offers satisfactory performance for most integration contexts.
- Describe the open-source infrastructure that is now available for use building with and ontop of the MDCF (Including JMS providers, mock devices, example device drivers and example scenario components.)

We are submitting this paper to HDMCSS in order to directly engage medical professionals and systems builders focused on integrating medical devices. We hope that the MDCF can be used to rapidly prototype integration systems geared towards a high level of reliability and patient safety. These prototypes in turn would provide both system designers and medical professionals with insight concerning potential issues systems of medical devices are likely to face. In addition, the MDCF architecture in combination with its development environment serves as an example on how rigorous development practices could be automatically enforced. The contents of this paper should not be interpreted as an endorsement by the FDA of any particular technology, software infrastructure, or direction for regulatory policy. However, we expect experience with frameworks like the one presented here to provide science-based input to ongoing regulatory policy and standards development efforts. We encourage additional experience building efforts with this infrastructure by others in the software engineering and medical device communities to help shape the vision and realization of systems that we believe will be central to future health care enterprises.

The MDCF infrastructure is available for public download at [8].

2 JMS Overview

2.1 MOM Foundation

The design of our core architecture is driven by practical realities of the clinical device integration, such as (a) flexible, dynamic information flow (frequently needing privacy), (b) heterogeneous systems, mechanisms, and needs, (c) many listeners, and many sources, and (d) time-critical, scalable performance. A message-oriented, publish-subscribe architecture with decentralized hubs, dynamic queuing, reliable message passing, and enterprise-grade deployment fits these criteria nicely. We have found it convenient to consider message-oriented-middleware (MOM) based on the Java Message Service (JMS)

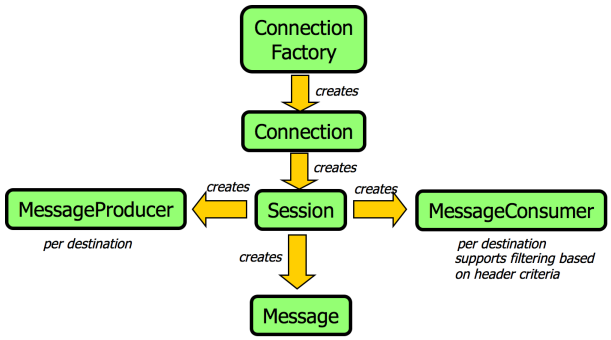

Figure 1: JMS primary objects

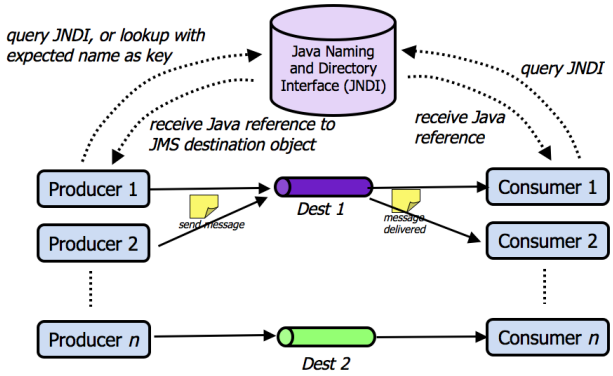

Figure 2: JMS destinations

standard. JMS satisfies the criteria (a-d) above, while providing low-cost, open-source implementations for low barriers to entry and easy integration into research environments. In addition, there are multiple commercial enterprise-quality JMS implementations such as those found in IBM's WebSphere and Oracle's AQ products JMS provides point-to-point or publish/subscribe topologies reliable or unreliable message delivery and high performance It enables distributed communication which is "loosely coupled, reliable, and asynchronous." In our application environment, its ability to pass simple data types as well as complex objects enables a clean integration with structured text standards such as HL7, as well as complex objects for seamless framework control .

Figure 1 presents the primary objects involved in JMS publish/subscribe communication. When a client wishes to originate a connection with a JMS provider, it uses the Java Naming and Directory Interface (JNDI) to locate a *Connection Factory* that encapsulates a set of connection-configuration parameters for the provider. The client then uses the Connection Factory to create an active *Connection* to the provider (typically represented as an open TCP/IP socket between the client and the provider's service daemon). In our architecture, clients will do all of their messaging with a single Connection. A Connection supports an *Exception Listener* that will be called when an connection fails (which we will use to handle situations in which a device unexpectedly disconnects in the middle of an activity). Once a connection is established, a client uses the connection to create a JMS *Session*.

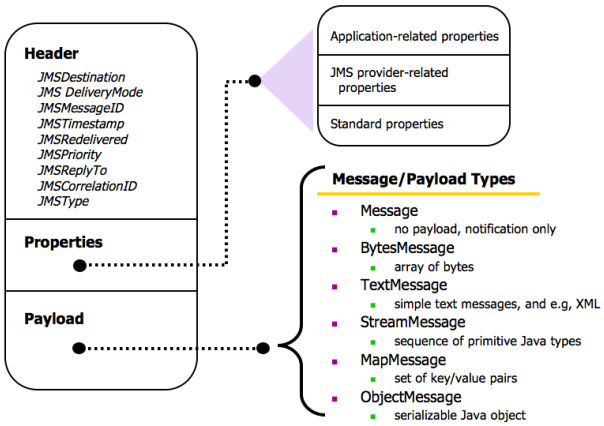

Figure 3: JMS message format

Figure 2 illustrates that a JMS *destination* is an abstract entity to/from which a client publishes or receives a message. Destinations are located/retrieved via JNDI calls. A session serves as a factory for creating *MessageProducers* or *MessageConsumers* for a particular destination. To send a message, a client requests a session to create an empty message (of a particular type supported by JMS), the message contents are filled in, and a MessageProducer is called to send the message. To receive messages asynchronously (which is the method we will use in our framework), the client creates an object (a handler) that implements the MessageListener interface and sets that object as the listener for a particular MessageConsumer.

A session is a single-threaded context designed for serial use by one thread at a time. It conceptually provides a thread for sending and delivering messages for all message producers/consumers created from it, and it serializes delivery of all messages to all of its consumers.

Figure 3 illustrates that the abstract structure of a JMS message is divided into three parts: a header containing values used by both clients and providers to identify and route messages, a properties section containing application-defined or JMS-provider-defined key-value pairs that provide additional metadata about the message, and the payload of the message. A number of these fields such as Destination, DeliveryMode, MessageID, Timestamp, and Redelivered are not set by the client but by the infrastructure layer as a message is transmitted. We use the Timestamp field to gather performance information reported on in Section 5. Other fields such as CorrelationID and ReplyTo are set by the client to guide responses to messages. We use CorrelationID to support the situation where we have multiple integration scenarios running on the same server. There are a few base administrative destinations (communication channels) that are shared among all running scenarios; each scenario sets a unique correlationID and watches for responses from the scenario administrator using the same ID.

Property values are set by the client prior to sending a message. When constructing a message consumer, a client can specify a filter expression that references fields in message headers and properties; only messages that pass the filter are delivered to clients. Thus, the primary purpose of message properties is to expose attributes for filtering. We currently use filtering only on header fields, but the property mechanism provides significant flexibility for enhanced functionality moving forward.

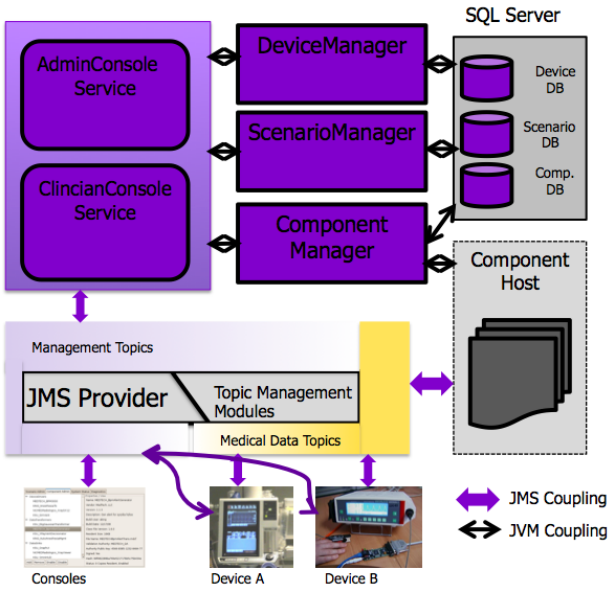

Figure 4: High Level MDCF architectural diagram

JMS provides a number of different formats for message payloads. We primarily use text messages (e.g. HL7 and most other data) and object messages (e.g. for DICOM images) (see Section 5).

3 Architecture

3.1 MDCF Components

Figure 4 contains a high level overview of the modules that compose the MDCF. The figure denotes that two different types of coupling are in place between the various modules. Two modules with 'JVM Coupling' simply means that those modules communicate with standard Java method calls, and that they must live in the same JVM. Two modules with 'JMS Coupling' communicate with each other via the JMS message bus. Two modules with 'JMS Coupling' need not exist in the same JVM.

3.2 Message Bus Modules

The modules in Figure 4 are grouped according to their general functionality. The JMS message bus (*JMS Provider*) and device relavent extensions (*Topic Management Modules*) provide an abstraction of the JMS topic management interface and abstract access to the JNDI. As mentioned in Section 2 JMS provides a publish subscribe framework where JMS clients either publish to or subscribe to 'global topics.' The *Topic Management Modules* hide the global nature of JMS publish / subscribe and instead expose the notion of virtual inter-component channels (see Figure 5). A MDCF scenario component then only communicates to other scenario components via these virtual channels. There are 2 main benefits to this approach; 1) Human and automated reasoning about information flows at the scenario level are greatly simplified and 2) the MDCF can take advantage of the performance features present in the underlying JMS provider (e.g. If many different clinician terminals are running a scenario that renders data from the same device then the MDCF is smart enough to tap those terminals into the same

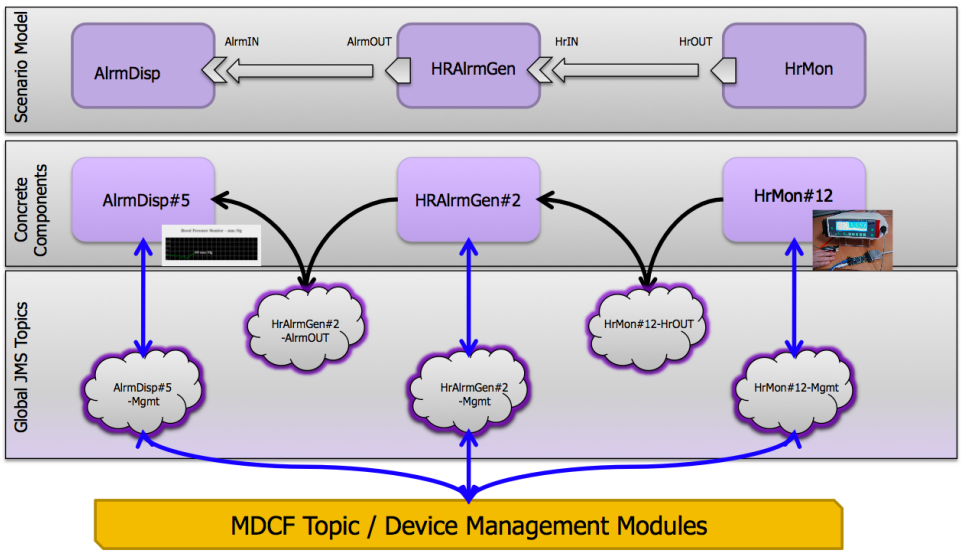

Figure 5: The MDCF hides 'global' topics from the programming model. JMS topics are instantiated for each component output port. The MDCF automatically subscribes a component's input ports in order to instantiate a given scenario. Management data flows are rendered in blue and medical data flows are black.

underlying global topic for information from that device instead of generating a message for each terminal.) In addition, topic subscription management can be abstracted away from the 'business logic' of the MDCF component, allowing the developer to only concern him or herself with the actual data being published or received from a virtual channel.

The *Topic Management Modules* manage two classes of JMS topics; *management topics* and *medical data topics*. The management topics are used by the various MDCF modules to communicate with devices and operator consoles. Management topics are never used to transport medical data. Medical data topics take on the opposite role; they are exclusively used to communicate medical data between devices. This partitioning is in place to support the MDCF programming model (where the scenario developer should not be concerned with low level connection management and component lifecycle) and possible future efforts towards automatic verification of integration scenarios (since information pathways are explicitly partitioned into these two categories, analysis of the functional correctness of scenarios would need not be concerned with management data).

3.3 Operator Services Modules

The *AdminConsoleService* and *ClinicianConsoleService* provide the actual business logic for the various remote operator consoles. Each service manages the authentication of operators at remote consoles and the interactions of those operators with the other modules of the MDCF.

The various consoles at the bottom of Figure 4 could represent either *Clinician* or *Adminstrative* consoles. These consoles are simply a remote graphical frontend to the logic provided by the console services. A *Clinician Console* permits a medical practitioner to request that a given integration scenario be instantiated with an operator specfied set of devices. If specified by the scenario, the console may display data output by that scenario.

The *Adminstrative Console* is somewhat more complicated. This console allows IT staff to configure, install and maintain different aspects of the MDCF. The MDCF requires that devices are registered (in the *DeviceDB*) before they can connect. Likewise, scenarios and software components must be installed into the MDCF prior to use. The *Adminstrative Console* acts as a graphical frontend to the *AdminstrativeConsoleService.* These two modules act in concert to provide sanity checks on the various tasks an administrator may perform. (Verifying version compatibility between the components and scenarios, as well as validating digital signatures.) The *Adminstrative Console* also provides some monitoring facility which allows adminstrative staff to observe the health and activity of the running coordination scenarios.

3.4 Device Integration Management Modules

The *DeviceManager* manages the lifecyle of connected and connecting devices. The *DeviceManager* uses the *management topics* to communicate with devices that are connected or connecting. During this communication the *DeviceManager* will query the remote device for accounting information (i.e. what type of device?) and periodically '*ping*' the remote device to determine the health of the device's connection. The *DeviceManager* also provides information w.r.t the state of connected devices to the rest of the MDCF (e.g. Is device x connected? or Is the device y responding to pings?) The *DeviceManager* uses information in the *DeviceDB* to determine if a given device is allowed to connect to the MDCF and what sort of security level the device is.

The *ScenarioManager* is responsible for instantiating and tracking integration scenarios. The *ScenarioManager* uses scenario specifications stored in the *ScenarioDB* to determine what type of devices and components are required for a given scenario and then communicating the necessary information via the *management topics* to the requisite devices.

Scenarios can include purely software components in their specification (such as an alarm generator). Software components are instantiated per-scenario (each scenario gets its own copy of a component). If the *ScenarioManager* determines that a software component is required in a given scenario, then the *ComponentManager* will retrieve the component bytecode from the *ComponentDB*, instantiate it, connect it to the JMS, and then place the new component in the *ComponentHost*. When a scenario is finished, the *ComponentManager* is responsible for disposing of the component and tearing down any connections to the JMS that component had.

4 Programming Model

We anticipate that device integration scenarios will be implemented either by developers at a company that supplies an integration framework (who would find it advantageous to build up a collection of reuseable components or product lines to serve multiple customers) or by on-site clinical engineers (who may not be familiar with underlying middleware and network concepts). Thus, we have developed a component-based programming model that abstracts away the details of the lower-level infrastructure and facilitates rapid assembly of integration scenarios from reusable components (Goal 7).

The component model supports typed input/output event (asynchronous) ports with multiple categories of components, including data producers such as devices, data transformers that filter, coalesce or transform data streams, and data consumers

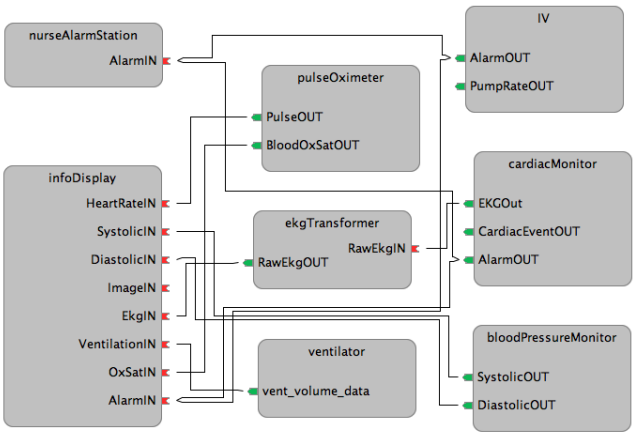

Figure 6: ICU scenario components

that represent displays or data repositories. Some components may be both data producers and consumers, such as devices that may be controlled by others or health information databases.

The MD PnP Integrated Clinical Environment (ICE) and ICE Manager (ICEMan) standards provide a foundational architecture describing the safe interaction of dynamically-assembled components (in keeping with the plug-and-play motif), clearly delineating the various roles within the system [14]. Each device provides a device description to the ICE-defined architecture, detailing the type and frequency of the data/services being provided, and QoS desired. The MDCF complements the ICE standard in several respects - providing a standards-based middleware to support the ICE, proposing a component model for programming device coordination behaviors, and development of a model-based programming environment for rapid assembly of device coordination scripts - while providing a less-developed device model and no support for dynamically adding devices (component relationships are statically analyzed).

While providing a more abstract/general component model than the ICE requires, our component model also provides a natural interaction with systems conforming to the ICE standard. For example, elements from a broader MDCF environment can map easily onto the MEDICAL DEVICE, ICE SUPERVISOR, and ICE NETWORK CONTROLLER components while providing a more detailed view of the medical device ecosystem. Furthermore, MDCF can reduce significantly the overhead of producing compatible, correct systems through extensive code generation capabilities.

We have built an integration scenario development environment in our Cadena framework [2]. Cadena provides component-based meta-modeling that enables us to define a domain-specific language of components for building device integration scenarios. Given a meta-model of the component language, Cadena generates a *component interface* editor that allows one to define component types and a *system scenario* editor that allows one to allocate and connect component instances to form an executable system. Cadena's rich type system allows one to define different type languages for component ports that capture specific properties of data communicated between components. Cadena provides a notion of "active typing" that continuously checks for type correctnesss as a system scenario is constructed in the graphical scenario editor.

Figure 6 shows a device integration scenario built in Cadena's scenario editor. Components corresponding to medical devices such as blood pressure and cardiac monitors appear on the right of the figure. Connections between components

represent publish/subscribe relationships.

We have built a Cadena plugin that provides facilities akin to a very light-weight version of the CORBA Component Model (CCM). Given a Cadena type signature for an MDCF component, autocoding facilities generate a Java skeleton/container for the component. The skeleton contains all logic required by the framework to enable the component implementation to connect to the framework as a framework component (this includes automatically generating the logic for subscription assignment and publishing logic). The component developer then only needs to implement the "business logic" – the code that processes medical information (such as a data transformer or rendering routine) or device access logic (interaction with actual device sensor hardware).

Similar in spirit to CCM's deployment and configuration infrastructure, the plugin can also analyze a Cadena coordination scenario model and generate a MDCF specification file. The MDCF specification file consists of XML that describes the named component graph. The logical name of each component instance and the type of the component is present, as well as what inter-component connections exist. This information is used by the MDCF to locate the appropriate MDCF component class files and instantiate the coordination scenario.

We believe that the use of sophisticated architectural types and component encapsulation can help in constructing assurance cases for integration scenarios. Use of component technology helps prevent unanticipated interference between components by insuring that components only interact through explicitly declared ports. The strong typing in the Cadena modeling environment reduces the possibility of programming errors.

4.1 MDCF Meta-Language

As mentioned in section 3.2 and section 4 the MDCF extends JMS with the notion of abstract inter-scenario component channels. The MDCF meta-language encapsulates the features of this abstraction in a way that allows scenario developers to design both coordination components and scenarios composed of those components within the Cadena MDCF programming environment. The meta-language defines the programming model of the MDCF. What follows is an informal description of the MDCF meta-language.

- *JMSMessage* - An 'abstract' message type that can be transmitted over JMS. (e.g. this is an 'umbrella' type for the TextMessages, ObjectMessages, ByteMessages, etc. described in Section 2)
- *JMSChannel* - An interface type. A message transport between exactely two end points: a message publisher and a message consumer. The *JMSChannel* exclusively transports *JMSMessage*s.
- *JMSPublishPort* - Describes a 'publication port' which can be associated with MDCF components. Data can only leave a component via a *JMSPublishPort* and never enter the component.
- *JMSSubscribePort* - Describes a 'subscription port' which can be associated with MDCF components. Data can enter a component via a subscription port, but will never leave a component via one.

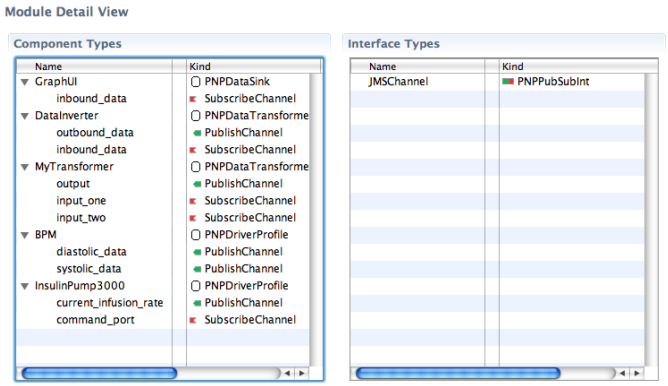

Figure 7: The Cadena MDCF module view

- *DriverProfile* - Components of this kind represent medical devices. *DriverProfiles* can have any number of subscription and publication ports. In the future we anticipate placing a restriction on the types of messages a *DriverProfile* component may subscribe to (e.g. device commands.)

- *DataTransformer* - Components of this kind represent software components that could be used in a coordination scenario. Components of this kind also allow any number of input and output ports.

- *DataSink* - A *DataSink* component only permits subscription ports. Typically components of this kind would be heads up displays or health informatics systems. Restricting this kind to only allow subscription ports permits lightweight analysis of scenario descriptions to determine what class of regulatory oversight a given scenario may fall under.

4.2 Cadena MDCF Module Editor

The Cadena MDCF meta-language defines the kinds (type families) of scenario components that the scenaro developer is permitted to build. The Cadena MDCF uses the meta-language to generate a MDCF specific module editor. MDCF component developers use the module editor to define the type-signature for a MDCF module. (Figure 7 is a screen shot of the module editor with several MDCF component signatures open.)

Component developers refine the component kinds from the meta-language by naming a component signature, explicity specifying what ports that component signature will have, the names of those ports, and the types of interface those ports will use. Constraints on the number and types of ports present in the meta-language are actively enforced by the module editor (i.e. a component type based off of the *DataSink* kind can not have any ports where data is published.)

4.3 Cadena MDCF Scenario Editor

The Scenario Editor allows developers to combine modules defined in the module editor into cohesive coordination scenarios by connecting ports on module instances via channel instances. The plugin actively type checks scenarios as they are being constructed in the scenario editor. For example, developers will not be able to connect two publication ports together or two

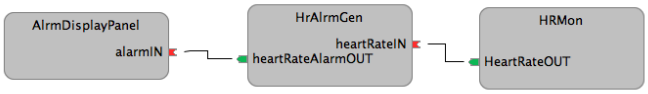

Figure 8: Simple Alarm propagation scenario in the Cadena scenario editor

subscription ports together. Constraints defined in the meta-language and module editor are actively enforced by the scenario editor.

4.4 Building a coordination scenario - start to finish

In order to make the development workflow more concrete, we describe the development of a simple coordination scenario, from the definition of the requisite modules to the assembly of the scenario from those modules. In this example we imagine a simple coordination scenario where patient heart rate information is analyzed, if the heart rate drops below a certain threshold then an alarm is generated and forwarded to a display at a nurses' station. We also assume that modules for both the heart rate monitor (*HRMon*) and alarm display (*AlrmDisplayPanel*) have been implemented and exist as component type signatures in the development environment. In order to realize the described scenario the developer must define a component type for the alarm generator (*HrAlrmGen*), implement its logic, and then combine all three components into a useable scenario specification.

The type signature for *HrAlrmGen* is straightforward. *HrAlrmGen* will be a software component that subscribes to one data stream (heart rate information) and publishes one data stream (alarm events.) The most appropriate kind from the meta-language to refine for this component type is *DataTransformer*, which will permit this signature to have both publish and subscribe ports. For clarity, we name the subscribe port *heartRateIN* and set its type to *JMSChannel*. Likewise, a publish port will be defined called *heartRateAlarmOUT* which also has the type *JMSChannel*. This completes the type signature for *HrAlrmGen* and the code skeleton can be generated. The plugin will place all of the necessary 'JMS plumbing' into the source code that is generated. Primarily, this means that the connection logic for a 'JMS Sender' (*heartRateAlarmOUTSender*) and a specialized message handler (*heartRateINListener*) will be present in the new source file. The developer simply needs to flush out the sender with the relevent 'business logic.' See Listing 1 for an excerpt of this code including the implemented logic.

When all necessary module type signatures are defined the developer can use the scenario editor to specify the scenario. In this case, we place an instance of each of our component types into a fresh scenario. Next, connections between the ports need to be created. In this case, two *JMSChannel* are used. The first between the *HRMon* and the *HrAlrmGen* and the second between the *HrAlrmGen* and *AlrmDisplayPanel*. See Figure 8.

```
public class HrAlrmGen extends TransformerComponent{
...

    Sender heartRateAlarmOUTSender;
...
    class heartRateINListener implements MessageListener{
        public void onMessage(Message message){
            TextMessage tMsg = (TextMessage)message;
            try {
                String msg = tMsg.getText();
                int intMsg = Integer.parseInt(msg);
                //begin business logic
                if(intMsg < 20){
                    heartRateAlarmOUTSender.sendMessage("ALARM HR < 20");
                }
                //end business logic
            } catch (JMSException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
 ..
```

Listing 1: HrAlrmGen component source excerpt with 'business logic'

5 Experiments and Performance

In this section, we summarize the experiment results previously reported in [7]. These experiments aim to show how the MDCF might support the following clinical contexts in which device integration is used (Clinical Device Integration Contexts - CDICs).

Two categories of experiments were designed to evaluate the viability of the framework: baseline experiments and clinical scenario experiments. *Baseline performance* experiments use simple producer/consumer configurations to measure the raw performance of the framework as it propagates data representative of clinical contexts. *CDIC experiments* use device/display component configurations that correspond to the clinical integration contexts presented in [7] (e.g. operating room, ICU ward, and alarm forwarding) to assess the ability of the framework to support typical usage modes.

Three categories of data were considered in our experiments: device data (point data and streaming data from monitoring devices), alarm events (relatively infrequent anomaly events published by devices), and medical informatics data (relatively infrequent and large data sets corresponding, i.e., to patient record data, drug dosing information, and medical images). Parameter settings (i.e the rates at which device data are published) are set to account for perceived worst case assumptions (maximum system requirements). For example, given a source device such as an electrocardiograph, a data update rate of once every 50 ms is considered frequent enough for a physician's data display to appear as if the data arrive in real time, so the data transfer and display process will not affect the quality of the associated clinical assessment. Other types of sensor data (i.e. blood pressure, heart rate, or blood oxygen saturation) can arrive much more infrequently. In our experiments, we will simply assume that devices publish information at a minimum interval of once every 50 ms. Low latency is important for

device and alarm data, but less so for informatics data.

Tests were performed on a single server representing the anticipated minimum machine configuration likely to be encountered in an enterprise-grade hospital information system (HIS) setting. We used a Sun Fire X4150 server with dual 2.8 GHz quad-core Xeon processors, 8 GB of RAM, a local 250 GB hard disk, and a gigabit Ethernet connection to the network fileserver. The server runs Linux 2.6.23, Java 1.5.0_13-b05, and OpenJMS 0.7.7-beta-1. OpenJMS was configured for non-persistent messaging unless otherwise noted. We observed that the current openJMS internal software architecture produced strongly asymmetric results; we expect other JMS implementations to provide more balanced performance. All results were averaged over multiple runs.

5.1 Baseline Performance

These experiments were designed to measure the throughput of the framework for single-step propagation (from a data producer to data consumer) given different types and sizes of clinical data. Performance was measured as a function of the numbers of producers/consumers under different connection topologies (fan-in/fan-out of producer/consumer relations).

5.1.1 Data Types and Connection Topologies

Three types of data were considered: simple event notifications, Health Level 7 (HL7) messages, and DICOM data.

Simple Event Notifications: These support the alarm notification scenarios (little or no payload), control instructions such as the X-ray activation (workflow automation examples) as well as many forms of device data such as remote heart rate notification (small payload). To simulate messages of this type, we use JMS ByteMessages with a payload of 10 bytes.

HL7: Health Level 7 is a messaging standard for the electronic exchange of medical information. HL7 messages use a text format (frequently XML-based) to structure medical data, health record queries, and data from health records. Although theoretically unlimited in size, these message typically range between several hundred and several thousand bytes. Our base experiments use three sample HL7 messages from the CDC Immunization Record EXchange (iREX) project [5], where messages range in size from 313 bytes to 4312 bytes. The small 313-byte message is an HL7 patient vaccine record query message. The medium 2227-byte message contains a fragment of a patient record that notes adverse reactions to vaccinations (a VAERS record). The large 4312-byte message is also a VAERS record, but with more vaccination events noted.

DICOM: The DICOM image exchange and storage format supports high resolution digital images tightly coupled with patient information. For instance, a DICOM file or message will typically contain a digital image (JPEG or RLE/TIFF format), a header containing the patient name or identification, and other metadata such as image dimensions, format, color depth, manufacturer/software version, etc. [4, 6] For our experiments, we use sample DICOM data from [1]: "CR-MONO1-10-chest" (379 kB), "MR-MONO2-16-knee" (130 kB), and "MR-MONO2-12-shoulder" (70 kB).

Connection Topologies: The base experiments evaluate the framework with components in topologies likely to appear in real-world CDICs. These topologies consider that some devices, databases, or displays (i.e. a nurse's station display) may be shared within and across different scenarios. The topologies relating producers to consumers include 1 to 1, 1 to 50, 1

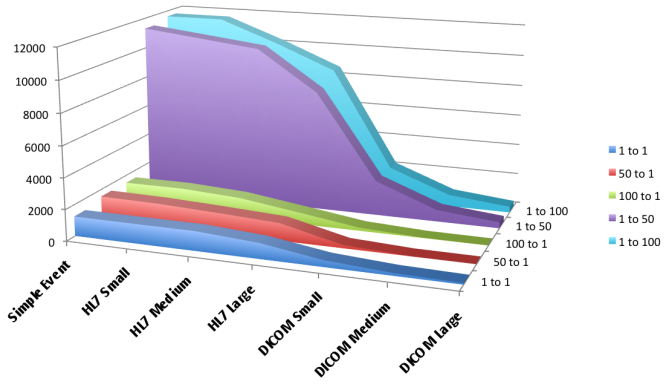

Figure 9: Message throughput

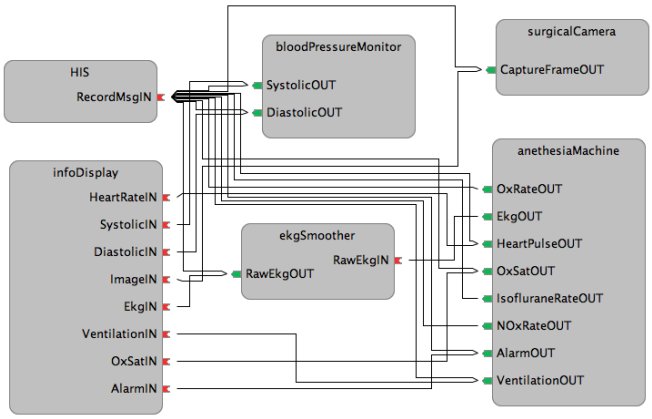

Figure 10: OR scenario components

to 100, 50 to 1, 100 to 1. In each topology, producers operate at "full throttle" – emitting messages in a loop as fast as the infrastructure can handle them.

5.1.2 Baseline Experimental Results

Both message size and connection topology affect the rate at which messages will move through the framework. Larger messages take longer to marshall/unmarshall, which reduces the rate at which the system can move messages. Interestingly, throughput is greatly affected by the connection topology. Increasing the number of producers will not increase the message throughput nearly as much as increasing the number of consumers. We suspect that this is because the JMS provider maintains a queue of pending messages that is shared between the provider's worker threads. In the case of many producers, many different messages can arrive at the message queue at the same time, and some resource contention can occur. When the number of consumers is scaled, the system merely has to remove one message from the queue and copy it to as many worker threads as system resources allow.

5.2 Critical Care Device Coordination

We begin the CDIC experiments with the Device Coordination Context discussed in [7]. Due to its safety-critical nature, this context has stronger real-time requirements. As discussed in [7], we expect that hospitals or critical care providers will use a dedicated server for each operating or critical care room, and the server will run one scenario instance at a time.

For this experiment, we imagine an operating room equipped with the following medical equipment networked to the MDCF: an anesthesia machine with an integrated ventilator and electrocardiograph (e.g. an Ohmeda Modulus CD/CV) plus a blood pressure cuff. The operating room is also equipped with a large heads-up display that renders device data streams. In this scenario, we also incorporate a software component, a `Transformer`, that preprocesses the electrocardiogram data stream prior to the stream's rendering on the physical display. See Figure 10 for a graphical depiction of this scenario's logical components.

Mean latencies of the informational messages are excellent - typically 1 ms. Each producer generates one data message on its output ports once every 50 ms (small numerical data messages that denote current sensor state, or a 50 ms subsection of a continuous waveform). Alarm events are updated once every 5 seconds.

Although this experiment does not represent an explicit coordination activity, it is clear from the performance discussion that our infrastructure would also be able to support critical care coordination activities such as those discussed in [7] when OpenJMS is used as the JMS provider and persistent messaging is disabled. Enabling persistent messaging increases the mean latency to 5 ms, but the peak latencies rise significantly, (in this case the peak latency was 7.42 seconds), indicating that OpenJMS may not be appropriate for some critical care scenarios when persistent messaging is enabled.

Mode	*Mean*	$\% < 50ms$	$\% > 2 \times mean$
Non-Pers.	1ms	99.99	1.0
Persistent	5ms	99.62	0.7

Table 1: Message latencies - OR scenario

5.3 Integrated Displays and Alarms

This experiment combines both the Room-Oriented Device Information Presentation and the Alarm Processing

CDICs. It demonstrates the ability of the MDCF to scale to ward level and still meet appropriate quality-of-service standards.

In this scenario, we imagine a large ICU ward with multiple rooms – each equipped with a blood pressure cuff, cardiac monitor, intravenous medicator, pulse oximeter, and ventilator. Each of these devices produces one or more data streams or alarm events (see Figure 6 for details). Each room is equipped with a configurable in-room, heads-up display that renders these data streams. The ward is equipped with a nurse's station display, which subscribes to all alarm events generated by any of the individual room's devices. This experiment replicates the scenario 1 - 100 times and aggregates all alarm messages to one nurse's station instance.

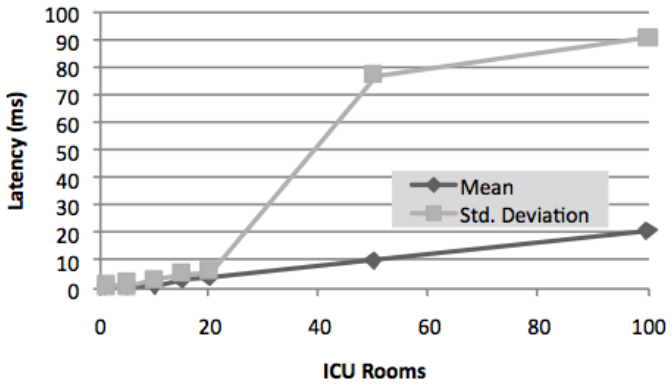

Figure 11: ICU latencies

As can be seen from Figure 11, the framework easily scales to 20 rooms. Even when managing 20 rooms, the maximum observed latency for any system message is 227 ms. The vast majority of the messages are transmitted much more quickly. At 50 rooms, the mean latency remains good, but the maximum observed latency has increased to 3 seconds (the spread of latencies has also increased, as can be seen by the increase in standard deviation). At 100 rooms, the maximum observed latency has grown to 4 seconds, but most latencies are still within allowable bounds.

6 Open Test Bed

The MDCF is part of a broader effort to build components which would be available to researchers for testing and experimentation of medical device integration. The MDCF described in this paper is a core component of this effort, as it supplies both the actual integration infrastructure and a tool with which formal methods and process oriented development can be excercised w.r.t. medical device integration systems.

We hope to build support for low cost or no cost (simulated) devices into the MDCF. Currently, work is underway to produce software ‘devices‘ which ‘simulate‘ Electrocardiograms by streaming pre-recorded data ([10]) into the MDCF. We hope that the availability of simulated devices will enabled researchers without the resources to obtain expensive medical equipment to use the MDCF as an experimental platform or test bed.

In addition to these virtual or simulated devices, we are working to build support for low-cost sensors into the MDCF. Such sensors include low-cost pulse-oximeters [12], thermometers, heart rate monitors, and multi-axis accelerometers available to us and currently used in vetinary telemedicine [13]. Lastly, we are exploring integrating pressure sensors found in entertainment oriented computer interfaces (such as a Dance Dance Revolution pad) as a way to provide a low cost version of a device which could be used to detect patient falls.

Finally, the MDCF programmers development environment is open; Researchers could further extend the tool to realize different and varied analysis capabilities for both the integration scenarios and components.

7 Conclusion

We produced an open (source) Medical Device Coordination Framework (MDCF) with the ultimate hope that it will be used as a research artifact in the research community to explore issues related to automated medical device integration and coordination. The underlying technologies (JMS) are also open and there exist implementations of JMS that are freely available. Initial experiments indicate that the architecture is scalable enough to support many realistic device integration and coordination scenarios.

Available with the MDCF is an Eclipse plugin that provides the 'MDCF Programmer's Environment' - a model based development tool thats aids integration scenario developers by allowing the definition of MDCF component types, the assembly of MDCF components into workable integration scenarios, and provides code generation facility which auto-programs the low level JMS connection code for any component specified. This plugin is also open; researchers could potentially extend the tool to perform other analysis of the integration components and scenarios. The tool has already been used to rapidly prototype several different device coordination scenarios.

We see the MDCF as complementary to efforts like the MD PnP Integrated Clinical Environment. While the MDCF supports decentralized integration and coordination, it would be fairly straightforward to build centralized device coordination facility the MDCF. The internals of the MDCF have also been designed in a modular fashion in order to more easily allow developers to support the types of features which may require management of data at a lower level than what the programming model on its own provides. (e.g. ICE proposes functionality such as QoS and a 'device model')

8 Future Work

We plan to extend both the MDCF and the accompanying programmer's environment with more sophisticated analysis and verification technologies. In addition to the active type checking, we will extend the the programmer's environment to support more precise specification of functional properties (e.g. numerical behavior of transformer components) of a scenario. The scenario editor will be modifed to permit the developer to check the correctness of a given scenario vs. the scenario and component specifications (compositional reasoning). We hope to integrate other analysis tools such as Bogor [11] and Kiasan [3] so the programmer's environment plugin can be used to verify the 'business logic' integration scenario developers implement are correct w.r.t. to the specifications applied at the modeling level.

References

[1] S. Barre. DICOM images – Sebastian Barre respository. `http://www.barre.nom.fr/medical/samples/`.

[2] A. Childs, J. Greenwald, G. Jung, M. Hoosier, and J. Hatcliff. CALM and Cadena: Metamodeling for component-based product-line development. *Computer*, 39(2):42–50, February 2006.

[3] X. Deng, J. Lee, and Robby. Bogor/Kiasan: A k-bounded symbolic execution for checking strong heap properties of open systems. In *21st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 157–166, 2006.

[4] DICOM homepage. `http://medical.nema.org/`.

[5] CDC Immunization Record EXchange (irex) project. `http://www.dt7.com/cdc/`.

[6] W. B. Jr, S. Horii, F. Prior, and D. V. Syckle. Understanding and using DICOM, the data interchange standard for biomedical imaging. *Journal of American Medical Informatics Association*, 4(3):199–212, May 1997.

[7] A. King, S. Proctor, D. Andresen, S. Warren, J. Hatcliff, W. Spees, R. Jetley, P. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In *Proceedings of the 31st International Conference on Software Engineering (ICSE 09)*, 2009. To appear.

[8] Medical Device Coordination Framework (MDCF) – Kansas State University. `http://mdcf.projects.cis.ksu.edu/`.

[9] Medical device "plug-and-play" interoperability program. `http://mdpnp.org/`, 2008.

[10] Physiobank medical device data stream repository.

[11] Robby, M. B. Dwyer, and J. Hatcliff. Bogor: An extensible and highly-modular model checking framework. In *Proceedings of the 9th European Software Engineering Conference held jointly with the 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 267–276, 2003.

[12] D. Thompson and S. Warren. A small, high-fidelity reflectance pulse oximeter. In *2007 Annual Conference and Exposition, American Society for Engineering Education*, June 2007.

[13] S. Warren, D. Andresen, D. Wilson, and S. Hoskins. Embedded design considerations for a wearable cattle health monitoring system. In *2008 International Conference on Embedded Systems and Applications (ESA '08)*, July 2008.

[14] A. F. WK19878. New Specification for Equipment in the Integrated Clinical Environment - Part I: General Requirements for Integration., 2008.

A Modular Framework for Clinical Decision Support Systems: Medical Device Plug-and-Play is Critical

Williams M, Wu F, Kazanzides P, Brady K, Fackler J
Johns Hopkins University, Baltimore MD 21218

Abstract

This paper describes the design and initial implementation of a modular framework for Clinical Decision Support Systems and highlights the need for medical device plug-and-play standards. The software handles the tasks of data acquisition and validation, visualization, and treatment management in order to enable the development of protocol guideline modules as "plug-ins" to the framework. The system utilizes an asynchronous data-driven design to support real-time information flow and user interaction. All components of the framework are modular and easily extendible, allowing for new data sources, visualization methods, and protocols to be inserted. The system is configured by assigning each protocol a manager that handles decision communication with the rest of the framework. A set of classes has been created to allow communication between the different modules along with persistence of all data, decisions, and treatments to a database. The initial prototype is a Clinical Decision Support System focusing on the treatment of Traumatic Brain Injury.

Keywords

Clinical Decision Support System; modular framework; visualization; Traumatic Brain Injury; plug-and-play

Background

Clinical Decision Support Systems (CDSS) are fifty years old (Ledley and Lusted 1959). Examples of CDSS range from the laminated pocket cards that outline cardiopulmonary resuscitation guidelines but more typically are paper or computer-based order-sets as well as reminder systems (e.g, a reminder that gentamicin doses be trimmed in a patient with a rising

creatinine, or activated protein C be considered based on a multi-factorial score).

Relevant to medical devices and requirements for device interactivity, a specific form of CDSS, closed-loop control of medical devices, has an even older history. Almost 60 years ago, a paper was published describing the "robot anesthetist" describing earlier work using the electroencephalogram driving either ether or sodium thiopental (HAWARD 1952), (MAYO, BICKFORD and FAULCONER 1950). A comprehensive literature review of closed-loop anesthesia was published in 1992 (O'Hara, Bogen and Noordergraaf 1992) and literally hundreds of articles have been since published (particularly in the domain of mechanical ventilation (Tehrani and Roum 2008).

Two meta-analyses of CDSS recently reported remarkably similar findings. Garg et al searched multiple databases through September 2004 and found 100 studies of decision support systems meeting their rigorous entry criteria. Fifty-two of those studies measured one or more patient outcomes. Studies were evaluated for methodological quality and for study characteristics that predicted success (defined as at least a 50% improvement in the measured outcome) (Garg, et al. 2005). Only seven of 52 studies that attempted to examine patient outcomes showed an impact of CDSS. Overall, two decision support system characteristics were associated with an impact. First, studies more often showed an effect when the authors of the CDSS were authors of the report. The only extensible finding of this meta-analysis was decision support systems that automatically prompt users (compared to systems requiring user activation) were associated with improved outcomes. The second meta-analysis examined 70 studies for 22 CDSS features the authors believed important for successfully improving provider behavior (Kawamoto, et al. 2005). A multiple logistic regression analysis identified four of these features as independent predictors of improved provider behavior: 1) automatic provision of decision support as part of clinician workflow, 2) provision of recommendations rather than just assessment, 3) provision of decision support at the time and location of decision making, 4) and computer based decision support.

A major impediment to the extensibility of CDSS (beyond their largely unproved clinical efficacy (Garg, et al. 2005)), is the lack of widely implemented data standards. Even efforts to share CDSS between sites running the same major health care IT company's system can not simply "import" a rule set. In the domain of medical devices, CDSS require hard-coded connections between input signals and output controls.

The goal of the project described is to create a modular framework for a Clinical Decision Support System to facilitate the development of guideline protocol "plug-ins". Instead of focusing on creating a new method for defining guidelines or representing clinical knowledge such as in (Achour, et al. 2001), we address the issues of data acquisition and validation, visualization, and treatment management. The basic concept is to develop a software framework that allows users to concentrate on implementing guidelines and protocols, with our system handling the data, management, and display aspects. The initial prototype is focused on supporting the treatment of Traumatic Brain Injury (TBI) based on internationally agreed algorithms (Brain Trauma Foundation, American Association of Neurological Surgeons and Congress of Neurological Surgeons 2007) as a sample application of the framework. As much of the data inputs are necessarily from medical devices (e.g. physiological monitors) and in its future closed-loop form the therapeutic devices are also medical devices, this project presents a complex use-case for identification of medical device plug-and-play efforts.

One of the key components of the software framework is the acquisition and fusion of medical data from multiple sources. These sources include bedside monitoring devices, hospital electronic medical record databases, and manually entered information from clinical staff. In the development and testing of the TBI CDSS software, the data is acquired from the physiological monitors (e.g. heart rate and blood pressure), stand-alone monitors (e.g. oxygen saturation and end-tidal carbon dioxide), and therapeutic devices (e.g. ventilators and infusion pumps). Given the current lack of device data standards, we were forced to acquire real-time (5 second interval) data only from the devices on the monitoring network and it involved writing a module to continuously poll a set of text files generated by a third party solution (BedMaster, Excel Medical Electronics, Jupiter, FL) and then extract and reformat this data, finally sending it to the TBI

CDSS framework. If plug-and-play standards were implemented, such a convoluted (and therefore risky) set of work-arounds would not be necessary. The current lack of a universal interface for medical device I/O poses a significant difficulty to the continued development and use of our software framework. Every time a new device needs to be supported, a module designed for communication based on that device's protocol and standards must be written. This roadblock would be removed with the implementation of a common communication interface for all medical monitoring devices.

Besides monitoring device connectivity issues, access to lab results and the patient's medical record also pose an issue. Currently, the electronic medical record module in the framework must continuously query the hospital's databases for updated information. Each piece of information requested requires a separate query tailored specifically for the hospital's database. This makes deployment to medical centers using a different electronic medical record system difficult, as a new module specific to their implementation must be written. A common standard for electronic medical records, lab results, and methods to access this data would greatly enhance the range of environments into which our software could be deployed.

System Overview

The framework utilizes a modular design to enable expansion and scalability. Each module has an interface that defines which methods it supports. In addition, there are a number of classes that handle communication between the different modules (these are explained in the sections where they are relevant, and are loosely based on those used by the PROforma system (Sutton and Fox 2003)). The framework can be viewed as three different areas: data collection and processing, protocol and treatment management, and visualization (see Figure 1). The communication between these three areas could easily be extended to include wireless or ad-hoc network capabilities. Data collection and routing consists of DataSource and Validator modules, ProtocolManagers handle interaction with the Protocol modules that perform the medical data analysis and decision support, and the ProtocolDisplay and FeedbackManager modules interact with the user. As previously mentioned, the framework is designed to make any data necessary available to the guideline protocol modules, and then display the decisions of these protocol

modules through a graphical user interface that the user can interact with to provide feedback to the system.

We chose an asynchronous data-driven design to support the real-time goals of our requirements, similar to the approach taken in (Van Den Bossche, et al. 2008). This allows each piece of physiological data or user input to be handled and received by the protocol modules as soon as it is available, along with allowing the recommendations of the protocols to be available to the user as frequently as possible, which is a key component of CDSS design. The framework is event based, following a push methodology (as opposed to a pull) where data is pushed to the protocol modules, which in turn push their decision outputs back to the framework. This choice was made in an attempt to remove any timing or thread requirements from the protocol modules in an effort to simplify them.

Therefore, to add a protocol module to the framework, it only has to be configured to listen for data using the required interfaces, and then to output its decisions in the specified format, as the framework handles timing and initialization. The software is written in Java. The following subsections discuss the different modules and communications mechanisms that the framework is built upon.

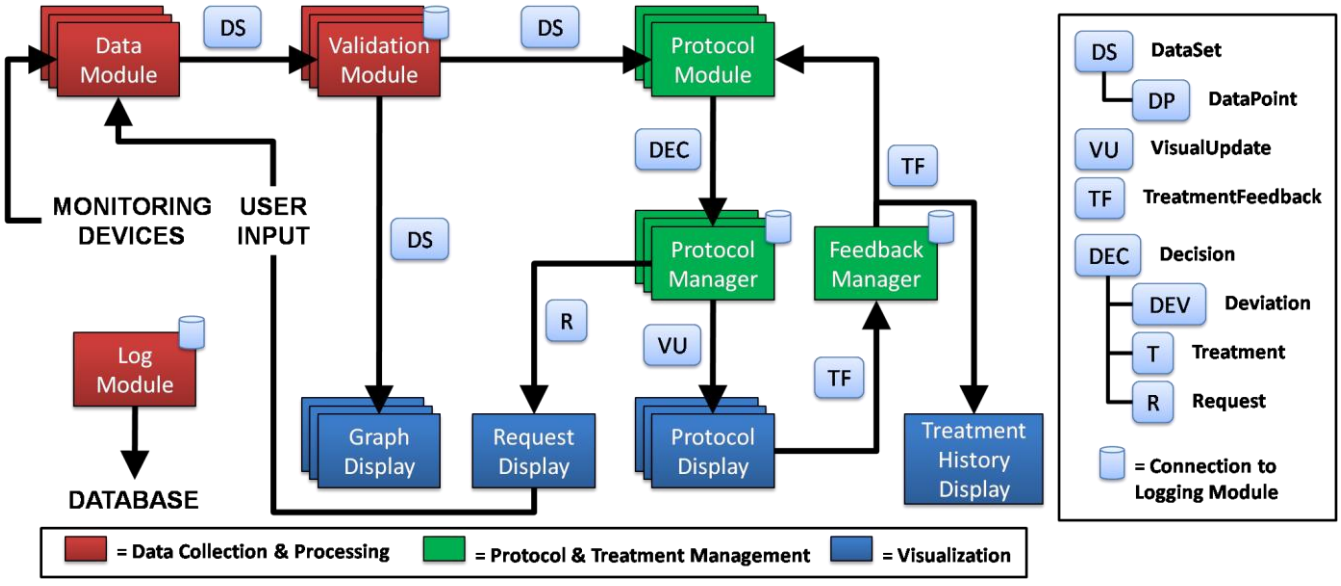

Figure 1: System Block Diagram

Data Sources & Validation

These modules interact with data sources, such as medical monitoring devices, patient record databases, and the clinical users of the system and transform the acquired data into formats that are recognizable by the rest of the framework. These classes are defined by the DataModule interface. DataModule objects generate DataSet objects, which are containers for the general DataPoint objects that hold different types of physiological data. The DataSet class is the primary mechanism for delivering information to the Protocol Modules, and it contains methods that allow its consumer to extract the different DataPoints that it contains. Each DataPoint consists of a source tag, a timestamp, the name of the data value it contains, and the actual data value itself. The relationship between DataSet and DataPoint is one-to-many (i.e., one DataSet can contain multiple DataPoints). Each DataModule can be configured to optionally forward the DataSet it generates to a Validator, which checks that the data it receives is within pre-configured boundaries to help prevent erroneous data from disrupting the rest of the system. The Validator then forwards this error-checked data to other modules in the framework.

Protocol Managers

Each ProtocolModule forwards the Decision objects it generates to its assigned ProtocolManager. These managers are responsible for handling the Decision and then forwarding its contents to the rest of the framework, allowing there to be a single point of communication between a Protocol Module and the rest of the framework. Each Decision object contains Deviations, Conditions, Treatments, and Requests. A Deviation is generated by a protocol when a physiological variable is outside of the guideline range (e.g. Intracranial Pressure > 20 mmHg). The Deviation contains the name of the physiological variable, the actual value, the guideline value that was violated, and a timestamp. A Condition is the medical condition that is indicated by one or more Deviations (e.g. Intracranial Hypertension). A Treatment is the medical procedure that should be taken to alleviate a given Condition. Therefore, each Condition has one or more Deviations that triggered it, and one or more Treatments that will remedy it. The ProtocolManager internally keeps track of what Conditions, Deviations, and Treatments are currently suggested, along with any pending Requests. The Request class is used by a ProtocolModule to request any additional information

that cannot be determined from the currently available data. A Request contains a source tag, a timestamp, the question to present to the user, a list of response options, and the name of the variable to fill with the user's response. An example of a Request would be "Does the patient have a skull fracture?" with responses being either yes or no. If the response list is left empty, then a text field is provided for the user. The ProtocolManager forwards any active Requests to the RequestDisplay, and a VisualUpdate is generated to tell the corresponding ProtocolDisplay module what to update. The VisualUpdate class contains lists of which Conditions, Deviations, and Treatments to add and/or remove from the ProtocolDisplay (making it easier to develop displays).

Visual Modules

There are several different types of visual modules that allow the user and the system to interact. The first is the ProtocolDisplay, which provides a visual representation of the Deviations, Conditions, and Treatments that a Protocol generates, along with allowing the user to input their feedback about the suggested Treatments, specifically whether they were performed or not (see Figure 2). This feedback is sent in the form of a TreatmentFeedback object, which is handled by the FeedbackManager discussed in the next section. The default ProtocolDisplay can be extended to provide more in depth features, such as a flowchart representation of a specific protocol. Each ProtocolDisplay is tied to the corresponding ProtocolManager of the ProtocolModule that it represents. Another visual module is the TreatmentHistoryDisplay, which is a timeline that gives the user a quick and simple overview of all the Treatments that have been performed on the patient during this session. The third type of visual module is the GraphModule, which is a real time dynamic graph of one or more physiological vitals. (In the future we plan to incorporate guideline, raw, and average values into the graph also). The GraphModule provides the user a history of important physiological values to reference when viewing the suggested treatments. Finally, the RequestDisplay visual module presents any pending information requests to the user. The user provided data is treated as another DataSource.

Feedback Manager

The FeedbackManager collects and records all the treatment feedback from the user, such as if/when a treatment was performed. The ProtocolDisplay modules send TreatmentFeedback objects to the manager based on user input. The TreatmentFeedback class contains the Treatment in question, a timestamp, a tag indicating if the treatment was performed or skipped, and an optional explanation section the user can fill in. The FeedbackManager forwards these feedback results to the ProtocolModules and the TreatmentHistoryDisplay, along with keeping an internal record.

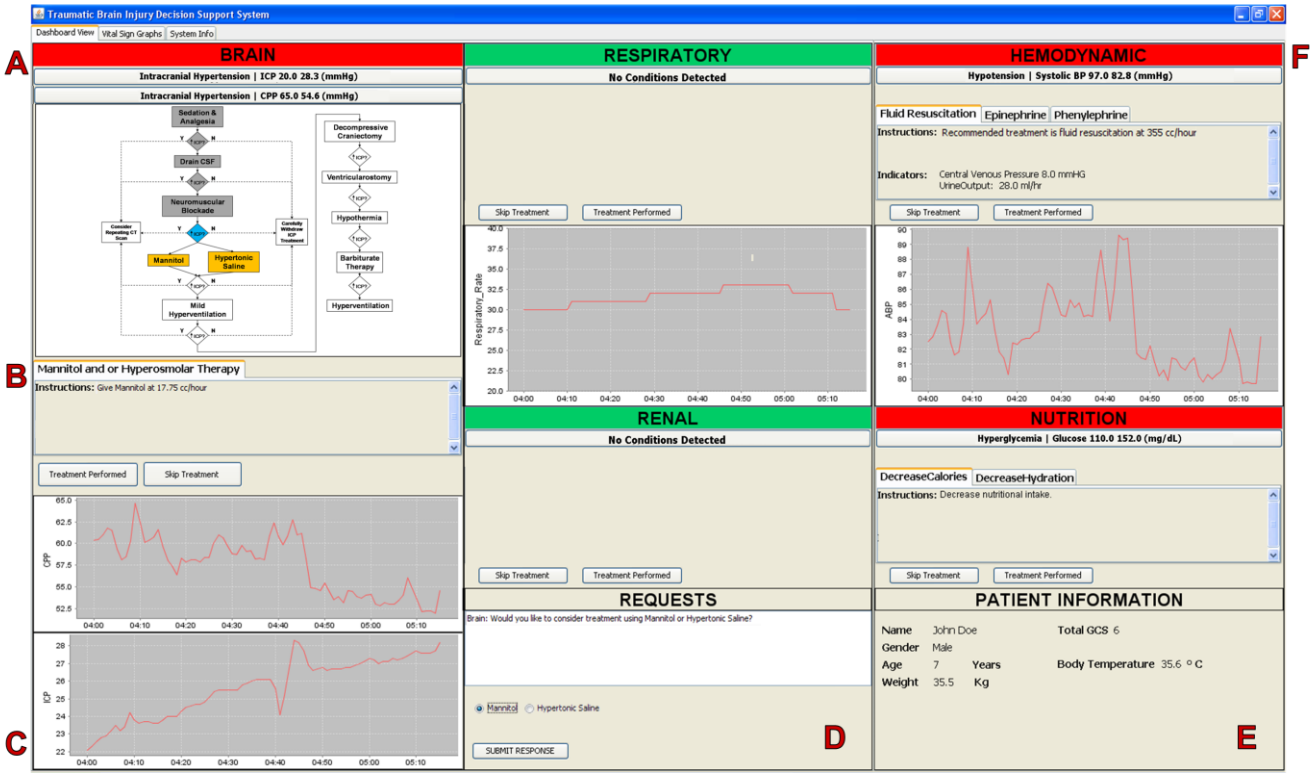

Figure 2: Graphical User Interface

This figure highlights some of the visualization and user interaction features of the framework. The title bars for each protocol are green if no conditions are detected and red otherwise. The buttons directly below each title bar show currently detected conditions. (A) The custom Brain ProtocolDisplay including the treatment flowchart. (B) The currently suggested treatment based on the Brain protocol, along with buttons to allow the user to choose direct the treatment flow. (C) Vital sign graphs. (D) The RequestDisplay allowing the user to choose which treatment path to take. (E) General patient information display area. (F) The default ProtocolDisplay for the Hemodynamic protocol.

Logging Subsystem

The primary function of the LogModule is to persist all necessary values for record keeping purposes. Each ValidationModule uses the log to save any DataPoints that it determines are erroneous, and can be configured to save all DataPoints that pass through it in the event that the real time physiological data is not being simultaneously recorded on another system. All of the ProtocolManagers record every new Deviation, Condition, Treatment, and Request that they receive in a Decision. The FeedbackManager records all TreatmentFeedback objects it receives. The logging subsystem is implemented using Hibernate and a MySQL database. A future goal is to be able to replay the treatment history of a patient using this framework for both analytical and training purposes, i.e. "flight data recorder" concept.

Traumatic Brain Injury Application

Two additional DataModules were written besides the standard user input module: one to query data from the hospital's Electronic Medical Record database, and one to interface with the BedMaster software that integrates output from multiple bedside monitoring devices. The BedMaster software provides real time physiological value monitoring by pulling data at five second intervals. The traumatic brain injury guidelines (Brain Trauma Foundation, American Association of Neurological Surgeons and Congress of Neurological Surgeons 2007) were adapted into five separate ProtocolModules, focused on the brain, hemodynamic, respiratory, renal, and nutrition aspects of a patient. The development and functioning of these protocol modules was previously presented in depth (Wu, et al. 2009). The base ProtocolDisplay module was extended to support the sequenced treatment pattern from the brain protocol, while the default display was used for the other four protocols. The screenshot in Figure 2 details some of the important elements of the visualization modules and how they interact with the user and the rest of the framework.

Conclusions and Future Work

This paper presented the initial progress on a framework for developing a Clinical Decision Support System, with an example application focused on treating Traumatic Brain Injury. The

focus of this framework is to enable guideline protocols to be written as plug-ins that take advantage of the data, treatment, and visual management provided by the framework. The framework is built on a modular design that allows for easy expansion to include new data sources, protocol guidelines, and visual interfaces. Programmers perform the initial configuration of connecting DataSources to provide information to ProtocolModules, connecting ProtocolManagers to handle the decision outputs of each guideline protocol module, and connecting the available VisualDisplay modules to interact with the user. Future work includes continued testing on a larger variety of patient data and medical conditions, along with clinical trials with studies of patient outcomes. Another goal is the development of additional protocol modules and to expand into areas other than TBI.

However, without implementation of medical device plug-and-play standards, the ability for other investigators to replicate our efforts is impossible. As such, progress in the development of CDSS and deployment will be impossible. This project provides the community a robust and challenging use-case as these standards efforts continue.

Reference

Achour, S L, M Dojat, C Rieux, P Bierling, and E Lepage. "A UMLS-based knowledge acquisition tool for rule-based clinical decision support system development." *Journal of the American Medical Informatics Association.* 8, no. 4 (Jan 2001): 351-60.

Brain Trauma Foundation, American Association of Neurological Surgeons, and Congress of Neurological Surgeons. "Guidelines for the management of severe traumatic brain injury." *Journal of neurotrauma* 24 Suppl 1 (Dec 2007): S1-106.

Garg, Amit X, et al. "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review."*Journal of the American Medical Association* 293, no. 10 (Mar 2005): 1223-38.

Hayward, L R C. "The robot anaesthetist; an introduction to the automatic control of anaesthesia by means of an electro-encephalographic intermediary." *Medical world* 76, no. 23 (Aug 1952): 624-6.

Kawamoto, Kensaku, Caitlin A Houlihan, E Andrew Balas, and David F Lobach. "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success." *BMJ (Clinical research ed)* 330, no. 7494 (Apr 2005): 765.

Ledley, R, and L Lusted. "The Use of Electronic Computers to Aid in Medical Diagnosis." *Proceedings of the IRE*, Dec 1959.

Mayo, C W, R G Bickford, and A Faulconer. "Electroencephalographically controlled anesthesia in abdominal surgery." *Journal of the American Medical Association* 144, no. 13 (Nov 1950): 1081-3.

O'Hara, D A, D K Bogen, and A Noordergraaf. "The use of computers for controlling the delivery of anesthesia." *Anesthesiology* 77, no. 3 (Aug 1992): 563-81.

Sutton, David R, and John Fox. "The syntax and semantics of the PROforma guideline modeling language." *Journal of the American Medical Informatics Association : JAMIA* 10, no. 5 (Jan 2003): 433-43.

Tehrani, Fleur T, and James H Roum. "Intelligent decision support systems for mechanical ventilation." *Artificial Intelligence in Medicine* 44, no. 3 (Nov 2008): 171-82.

Van Den Bossche, Bruno, Sofie Van Hoecke, Chris Danneels, Johan Decruyenaere, Bart Dhoedt, and Filip De Turck. "Design of a JAIN SLEE/ESB-based platform for routing medical data in the ICU." *Computer methods and programs in biomedicine* 91, no. 3 (Sep 2008): 265-77.

Wu, F, M, Kazanzides, P Williams, K Brady, and J Fackler. "A Modular Clinical Decision Support System: Clinical Prototype Extensible into Multiple Clinical Settings ." *Pervasive Health.* London 2009.

Standards for Physiological Data Transmission and Archiving for the Support of the Service of Critical Care

J. Mikael Eklund and Carolyn McGregor

University of Ontario Institute of Technology

2000 Simcoe Street North, Oshawa ON L1H 7K4 Canada

{mikael.eklund,carolyn.mcgregor}@uoit.ca

Abstract—Physiological data is monitored and displayed on medical devices around the world every day, and the volume of this data is steadily increasing and newer monitoring devices enter the clinical setting. However, the vast majority of this data is lost since it is most often displayed once as it is recorded, perhaps replayed one or more times while it exists in the device's volatile memory. What little data that is permanently recorded is most commonly saved through hand written annotations, in paper records and in some limited samples stored on hospital clinical information systems. Meanwhile, current methods of data analysis provide opportunities to utilize this data for improved care of these same critical care patients. A major inhibitor to this becoming reality is the lack of standards for the representation, transmission and storage of physiological data. HL7, for example, does not include definitions for time series data. Research into the use of these data will soon be reaching the clinical setting and the need for such standards to be defined is becoming urgent.

Introduction

Our research is focused on the collection and analysis of clinical physiological data streams both in real time and for later offline analysis [1]. The goal of this research is to use these streams for computer aided diagnostics using evidence based rules. However, the lack of standardized formats for transmission and archiving of these data limits access to these data -- and those collected by others -- and prevents significant amounts of analysis for the identification of such rules from being carried out.

We are developing just such standards and exploring opportunities with Canada Health Infoway (www.infoway-inforoute.ca) and the Canadian Neonatal Network (www.canadianneonatalnetwork.org) to develop such standards through a demonstration project with IBM Research's TJ Watson Research Center, NY and the Hospital for Sick Children in Toronto [2].

Physiological Data Streams: A Case Study

Within Intensive Care Units (ICUs), there are typically found several types of devices that are collecting physiological patient data rates of up one thousand sampled per second per sensor. In the example of monitoring brain electrical activity (electroencephalograms or EEG), this might include as many as 14 streams of EEG at 1 kHz per stream. Simultaneous collection for other devices might produce another 12 channels of heart activity monitoring (electrocardiographs or ECG) and 3 streams of intravenous blood pressure (iBP) monitoring, each also at 1 kHz. Additionally, these and other devices produce raw or derived streams of data (e.g. heart rate [HR] information at 1 Hz) also in real-time. Modern medical devices typically allow for storage of these data for review by physicians, and, while many allow for transfer of these data for backup or inclusion in clinical information systems, it is very unusual to electronically archive more than a limited number of samples, statistics and/or snapshots of these data. On the other hand many studies have shown that there is significant information contained in these data which could be exploited for diagnostic purposes.

In the Artemis project, we are working with IBM's TJ Watson Research Center, NY to develop a system for real-time event processing of precisely these data types, as well as offline storage for analysis and development of novel, evidence-based diagnostic routines. While much of the data that is being produced can readily be stored in standard Health Level 7 (HL7) format, for example as snippets and snapshots, it is unclear how large amounts of continuous data will be stored. As an illustrative example of the challenges, a prematurely born baby might be in an NICU for months, while being continually

monitored for nearly the entire time. Current medical data transmission, e.g. HL7 of Digital Imaging and Communications in Medicine (DICOM), do not provide clear methods to transmit or store such continually monitored data, and current clinical information systems do now support storage of these, partially as a result. However, research and clinical experience show that such data can be very informative for immediate diagnostics and for development of new diagnostic methods.

Within our research we have previously proposed two alternatives for the transmission of physiological data. Firstly, utilising the service oriented architecture based web services in [3] and secondly by extending DICOM principles in [6].

Integration with Electronic Medical Records

While the methods to use these types of data for diagnostic purposed are still at the early stages of development, the ability to record and store them is there and the potential benefit of doing so is significant [4, 5]. In the example of prematurely born babies, the ability to help them survive to maturity has improved drastically over the past several decades. However, the long term effects of both the premature birth itself and of the treatments required to help them survive are unclear [6]. For these babies, if more complete medical records of their birth and during their stay at the NICU can be stored, there is enormous potential in using these records for follow-up treatment and studies to better understand how to treat these people as they grow to maturity and to improve treatment of future prematurely born infants.

In order to allow for these data to be made compatible with electronic health record systems and with clinical research repositories, standards need to be selected and/or developed. Many different standards exist and several of them support aspects of continuous physiological data streams. As proposed previously, the DICOM standard can be used for these data if they are treated as 1-dimensional images, instead of the usual use of DICOM which is for 2- or 3-dimensional images. DICOM

furthermore supports “mosaics”, which is intended to allow for images which overlap and to allow them to be used to create a more complete image from several limited perspective images. The same idea can be used to store and transmit multiple segments of 1-dimensional data streams, and to construct the larger stream from them [7]. However, it is not clear that this is a good approach to continuous, multi-rate and heterogenous data streams.

Some other notable efforts to enable parts of this are, however, in progress. For example the Rosetta Terminology Mapping (RTM) profile aims to harmonize the use of existing ISO/IEEE 11073-10101 nomenclature terms for systems compliant with the Integrate Health Enterprise (IHE) Patient Care Device (PCD) profiles [8]. The RTM profile would facilitate safe and interoperable communication between devices and systems, an important step forward.

The issues that exist in storing these data streams in EMRs also include questions of data compression [9] and methods of temporal abstraction [10, 11]. On top of these are issues of how these compression and abstraction methods might affect the quality of the data stored and how it can later be used. And so it is important to consider also the transmission of archiving of details on the methods used to accomplish these so that, if necessary, the effect of compression or abstraction on diagnoses can also be evaluated.

In the Artemis project, NICU patient data is being collected and stored and will be used to develop new, evidence-based clinical rules for diagnostics. The data is recorded and analysed in real-time and through archiving of the data will provide a rich repository of data for clinical research. These data will also be made available to patient EMRs as soon as the standards can be determined on how to do so.

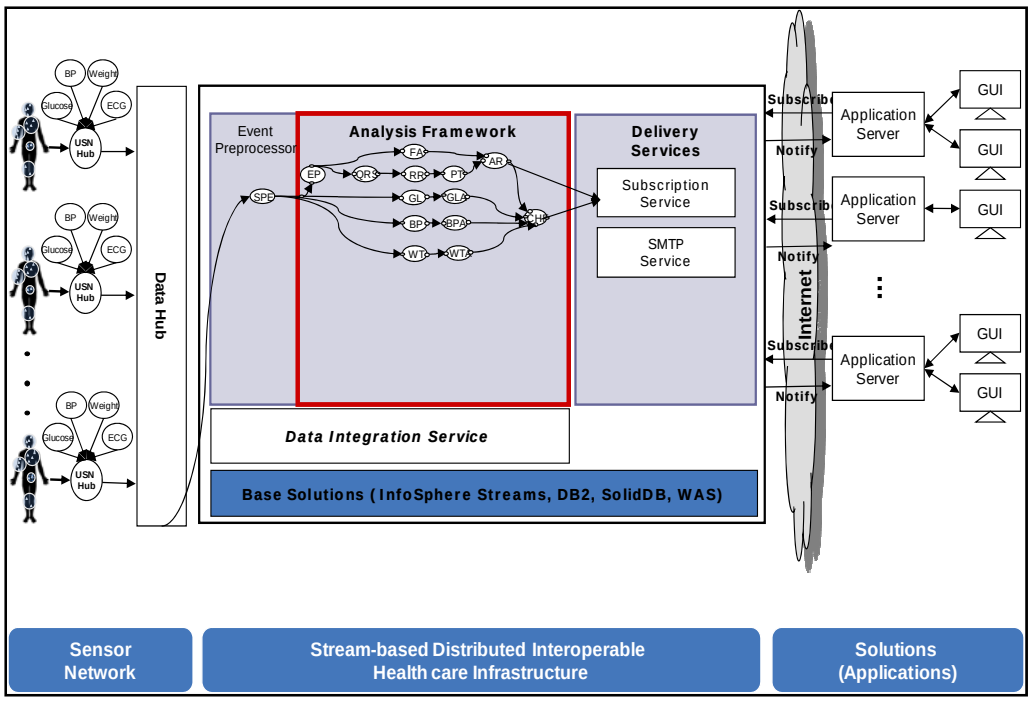

Figure 1: Artemis system for real-time event stream processing and archiving. Courtesy of IBM Research.

Through the Artemis project a demonstration project is proposed with the Canadian Neonatal Network and Canada Health Infoways to show how physiological data streams can be include in EMRs in the Canadian health system. Existing standards will be leveraged and specifications will be developed on how to use those standards to allow for transmission and storage of these records in EMRs.

Conclusions

The potential for the analysis of physiological data streams to support real-time clinical management and historical clinical research is significant. The classical nature of the electronic health record, supporting patient care by many providers in varied locations over their lifetime drives the need for standards to support this new area of medical support. Our research is working to develop standards for integration within the Canadian EHR commencing at birth through Neonatal Intensive Care Units.

References

1 M. Stacey, C. McGregor, and M. Tracy, "An architecture for multi-dimensional abstraction and its application to support neonatal intensive care," in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC07)*, Lyon, France, 2007, pp. 3752-3756

2 "First-of-a-Kind Technology to Help Doctors Care for Premature Babies", IBM Press release 23 July 2008, http://www-03.ibm.com/press/us/en/pressrelease/24694.wss

3 McGregor, C., Heath, J., & Wei, M. (2005), "A Web Service Based Framework for the Transmission of Physiological Data for Local and Remote Neonatal Intensive Care*", Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, Hong Kong, IEEE pp 496-501

4 Griffin, P. and R. Moorman, Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. Pediatrics, vol. 107, no. 1, 2001

5 McIntosh, N., J.-C. Becher, S. Cunningham, B. Stenson, i. A. Laing, A. J. Lyon, and P. Badger, "Clinical diagnosis of pnneumothorax is late: Use of trend data and decision support might allow preclinical detection," Pediatric Research, vol. 48, no. 3, pp. 408–415, 2000

6 Shankaran, S., J. C. Langer, N. Kazzi, A. R. Laptook, and M. Walsh, "Cumulative index of exposure to hypocarbia and hyperoxia as risk factors for periventricular leukomalacia in low birth weight infants," Pediatrics, vol. 118, no. 4, pp. 1654–1659, 2006

7 J. Mikael Eklund, Carolyn McGregor, Kathleen P. Smith, A Method for Physiological Data Transmission and Archiving to Support the Service of Critical Care Using DICOM and HL7, in *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC08)*, Vancouver, Canada, 2008

8 Integrated Health Enterprise (IHE) Technical Frameworks, Available online http://www.ihe.net/Technical_Framework/

9 McGregor, C., Purdy, M., & Kneale, B. (2005), "Compression of XML Physiological Data Streams to Support Neonatal Intensive Care Unit Web Services", *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, Hong Kong, IEEE pp 486-489

10 Stacey, M., McGregor, C., Tracy, M., (2007), "An architecture for multi-dimensional temporal abstraction and its application to support neonatal intensive care", 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC07), Lyon, France, pp 3752-3756

11 McGregor, C., Stacey, M, (2007), "High Frequency Distributed Data Stream Event Correlation to Improve Neonatal Clinical Management", *Inaugural International Conference on Distributed Event-Based Systems (DEBS 07)*, Toronto, Canada, CD-ROM, 6 pages

Technical Session III

Medical Networks

Session Chair:

Oleg Sokolsky

University of Pennsylvania

Monitoring and Diagnosis of Networked Medical Hardware and Software for the Integrated Operating Room

Stefan Bohn, Michael Lessnau, Oliver Burgert
Innovation Center Computer Assisted Surgery (ICCAS), Medical Faculty, University of Leipzig, Germany
stefan.bohn@medizin.uni-leipzig.de

1. Purpose

Interoperability of heterogeneous medical devices, clinical information systems and components of computer assisted surgery (CAS) has been recognized for its potential to improve the overall clinical workflow as well as ergonomic conditions by centralized access and control of the integrated system. However, the installation of an integrated IT infrastructure with additional computer hardware, software, and network components heavily increases the overall technical complexity within the operating room (OR). The life critical domain within the OR demands safe and reliable operation of the integrated OR components. Therefore, an appropriate technical supervision framework is required that supports high confident functionality by facilitating systems monitoring and diagnosis of the networked hardware and software. System failures, network bottlenecks or unstable conditions should be detected to enable appropriate interventions and mitigation strategies.

2. Methods

The structural design of our modular OR integration infrastructure follows the Therapy Imaging and Model Management System (TIMMS) meta-architecture, which was published by Lemke and Vannier in 2006 [1]. Our prototype TIMMS implementation interconnects standard CAS components such as tracking, PACS, display and video routing as well as navigation, patient modeller, workflow software, and the central surgical display. In contrast to existing proprietary integration solutions, we are focussing on the development of an open architecture using standard communication protocols (e.g. DICOM, RTP, SNMP, ZeroConf, TCP/IP) and standard network technologies such as Ethernet. The integrated system has a central management unit, the TIMMS Component Controller (TCC), which facilitates service discovery, session management, time synchronization and component control. The

TCC also implements the supervisory control and data acquisition (SCADA) module (Figure 1), which realizes systems monitoring and supervision functionality for the entire OR network on three levels: 1. Network Backbone Hardware, 2. Computer Hardware, and 3. Software Applications. The diagnostic information elements from the supervised systems are maintained in the form of Management Information Base (MIB) (RFC 1450) objects. The MIB tree index structure describes the hierarchical order of all monitored variables that can be obtained or modified, their data types and by which operations they can be accessed. Management agents handle the access to the MIB objects between managed components and the SCADA module using the Simple Network Management Protocol (SNMP).

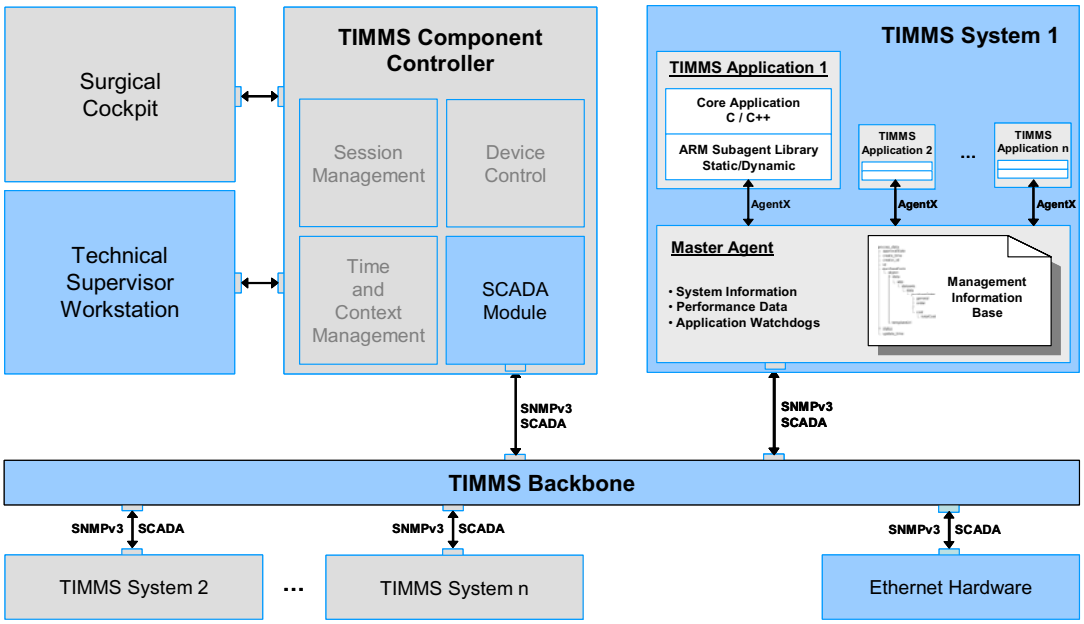

Figure 1: Supervisory control and data acquisition (SCADA) architecture for the integrated TIMMS OR network.

2.1 Simple Network Management Protocol

The transfer of diagnostic and management information among management agents and the SCADA module is based on the SNMP protocol standard (RFC 3411 – 3418), which is an application layer protocol within the OSI model. The SNMP network protocol is based on TCP/IP and thus directly applicable within the TIMMS network environment. SNMP provides few operations on MIB items such as SNMP GET and SNMP SET for retrieval or change of a MIB variable as well as SNMP TRAP for unsolicited event notifications sent by an agent to a management application.

2.2 Monitoring of Hardware Components

Diagnostic information from network and computer systems is gathered by hardware management agents, which reside in the supervised components. Most of today's standard network hardware such as routers or bridges already implement SNMP agents, which maintain access to configuration information and network traffic parameters (Level 1). Using the *Agent++* Library [2], we developed a master agent for personal computers (PC) that acquires system and surveillance information from each TIMMS PC connected to the network (Level 2). These master agents obtain performance indicators such as CPU usage, hard disc load, physical and virtual memory load as well as network interface card traffic and translate these into the corresponding SNMP MIB objects (Figure 1).

2.3 Monitoring of Software Applications

Monitoring at code implementation level (Level 3) is facilitated using the Open Group Standard "Application Response Measurement - ARM 4.1" [3]. The ARM interface standard accommodates bindings for the programming languages C and JAVA. ARM enables the measurement of application performance by introducing transactions as "units of work". The application calls the ARM API before a transaction starts, optionally an update during processing, and after it ends. We defined two custom transaction types for the SCADA framework: 1st) Transactions for measuring the duration of (critical) code sections and 2nd) Transactions that periodically report update events for the monitoring of the application's alive status. To integrate the ARM functionality into the SCADA framework, we implemented an ARM compliant subagent library that communicates with the corresponding Level 2 master agent using the Agent Extensibility (AgentX) Protocol (RFC 2741). Upon application start, the subagent registers the application and all transactions at the master agent, which creates the corresponding information items within its MIB. Watchdog alive heart beats from the application are processed by sub- and master agent and deployed using SNMP TRAP events to the SCADA module, which supervises the alive status of TIMMS applications.

2.4 TIMMS Component Controller & SCADA Module

Automatic Configuration and Plug-and-Play Service Discovery of TIMMS components are realized using the ZeroConf protocol [4]. Whenever a TIMMS component joins the network, the TCC connects

to the component's master agent and retrieves the corresponding management information base object. By passing through the MIB tree elements, the SCADA module periodically queries the particular diagnostic information elements from the master- and subagent. The diagnostic information is processed and analyzed, e.g. to raise alarms if a previously defined threshold for a given element is exceeded. The TCC also provides a database logger, which stores all alarms and events for documentation purposes.

The resulting information from the SCADA module is visualized on different workstations according to the particular user group. Clinical users obtain an intuitive view of the overall system status at the surgical cockpit while the technical supervisor has access to all in depth information and control over configurable elements (Figure 2).

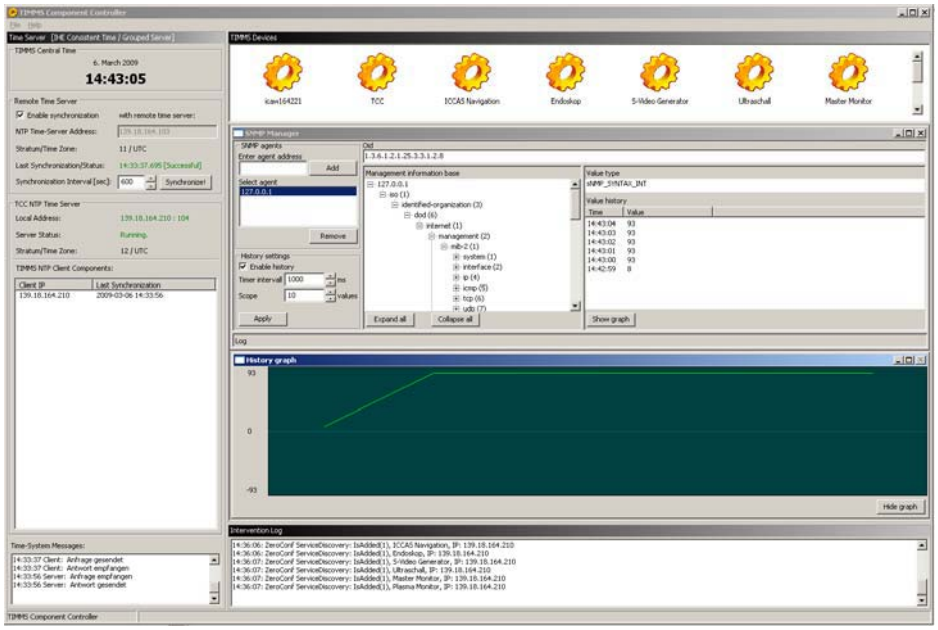

Figure 2: TIMMS technical supervisor SCADA application displaying the networked TIMMS components (top right), the management information base of the selected component (middle right) and time course of the monitored variable (bottom right).

3. Results

We designed and implemented a technical supervisory control and data acquisition (SCADA) framework for the monitoring of networked medical hardware and software components. Information about the overall system status and controlling access is designed at different abstraction levels for

different user groups (clinical/technical) with separate user interfaces. The master and sub-agents are implemented as C++ class library and are fully compliant with SNMP Versions 1 to 3. The first prototype of the SCADA module is able to retrieve diagnostic information from Ethernet network devices, computer hardware and TIMMS software applications. The user interface for the technical supervisor (Figure 2) comprises simple numerical values of performance measurements as well as graphical trend views for time-dependent values (e.g. network load). Methods of auto-configuration facilitate a highly automated monitoring process without the need for manual interaction.

4. Conclusion

The life critical environment within the operation room requires reliable and safe operation of medical device hardware and software, especially when a large number of different technologies are applied. The proposed SCADA framework is based on standard protocols and encounters these requirements by introducing technical means for the acquisition of performance indicators at hardware and software levels. The framework provides information to detect system anomalies such as network bottlenecks, cache and hard disc space exceeds or CPU consuming software processes and announces these using appropriate alarms to the corresponding user groups. The combination of AgentX subagents with ARM enables the assessment of software performance as well as the detection of hanging or crashed applications with the SCADA watchdog functionality. Further developments focus on automatic reasoning of the overall system status as well as appropriate user interface feedback for the clinical users at the surgical cockpit.

5. References

[1] Lemke H, Vannier M, "*The operating room and the need for an IT infrastructure and standards*", International Journal of Computer Assisted Radiology and Surgery, Vol. 1, No. 3, pp. 117 - 121, 2006.

[2] Fock F, "*Agent++, An Object Oriented Application Programmers Interface for Development of SNMP Agents Using C++ and SNMP++.*", http://www.agentpp.com, Last visited 03/06/2009.

[3] The Open Group, "*Application Response Measurement (ARM) Issue 4.0 V2 - C Binding*", ISBN 1931624380, 2004; available at http://www.opengroup.org/bookstore/catalog/c041.htm, Last visited 03/06/2008.

[4] E. Guttman, "*Zero Configuration Networking*" Proc. INET 2000, Internet Society, Reston, VA; available at http://www.isoc.org/inet2000/cdproceedings/3c/3c_3.htm, Last visited 03/06/2009.

Wireless Health and the Smart Phone Conundrum

Jonathan Woodbridge*, Ani Nahapetian*, Hyduke Noshadi*‡,
Majid Sarrafzadeh*, William Kaiser†

*Computer Science Department, University of California, Los Angeles
{jwoodbri,ani,hyduke,majid}@cs.ucla.edu

†Electrical Engineering Department, University of California, Los Angeles
{kaiser}@ee.ucla.edu

‡Google, Inc. 1600 Amphitheater Parkway, Mountain View, CA

ABSTRACT

This paper presents a study of the five best selling Smart Phones in terms of their applicability to Wireless Health. Smart Phones are generally used as central controlling units in Wireless Health applications. We carried out our investigation by implementing a wireless health application that performs sensor communication, data processing, and data visualization. Our overarching goal is to develop a plug-and-play Wireless Health software platform. Our task begins with an in depth study of Smart Phones: the central controller of Wireless health applications.

KEYWORDS

Wireless Health, Smart Phones, Bluetooth

1. INTRODUCTION

Over the past few years, researchers developed several wireless health platforms such as [3][14][16][17][22][23]. These platforms often include a mobile device (such as a PDA or cell phone) as the central controlling, processing, and visualization unit. Figure 1 depicts one such architecture. Previous work is predominantly focused on overall architectures and lacks focus on the central processing unit. Our goal is to analyze several commercial Smart Phones to determine the best targets for wireless health.

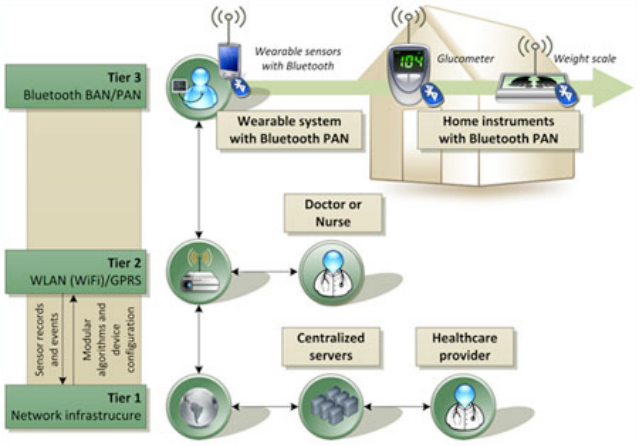

Figure 1. A standard architecture for Wireless Health Applications. The wearable system acts as a central processing unit for the patient.

Our comparison is based on a set of libraries developed for wireless health. We feel that extending current wireless health platforms by offering an additional software component (set of libraries) for mobile devices would add tremendous value. Reliability, code reuse, and decreased development times are just a few of the many benefits offered by such software.

Implementing a set of libraries requires intimate knowledge of the target devices. This paper presents an examination of popular Smart Phone platforms, based off of the design of a simplified application that uses a few basic components of a wireless health library. We design our libraries based on a few active research projects here in our labs at UCLA. We describe these projects in section 2. Next, we discuss the feasibility of implementing such an application (including libraries) on several commercial platforms. Finally, we present the development of our simplified application and libraries to prove the feasibility of such a software platform.

This paper does not present a complete wireless health software library. Our library is only representative of a complete implementation. We use this representative as a basis of comparison between Smart Phone platforms and prove the feasibility of a complete wireless health software library.

All platforms compared in this paper support cellular connectivity (such as CDMA and UMTS). Such devices offer far greater network coverage than devices that only support technologies such as Bluetooth and WiFi. In a large number of wireless health applications, this constant connectivity is required, especially in the case of applications that require high mobility.

During our comparison of platforms, we analyze feature availability as well as emulation and debugging environments. We hope this comparison serves as a guide for those wishing to develop wireless health applications for deployment on smart phone platforms.

The key contributes of this paper are three fold. First, we provide an assessment of the five best selling Smart Phones platforms and their applicability towards wireless health. Second, we determine the best software runtime environment in applicable to our five Smart Phone platforms. Finally, we developed a wireless health application to prove the correctness of our assessments.

2. WIRELESS HEALTH APPLICATIONS

The following section presents a few wireless health projects under development at UCLA. We use these projects as inspiration for a set of wireless health libraries. These projects by no means represent all wireless health applications. However, they do establish a baseline for comparison.

2.1 SmartCane

Falls are the leading cause of death in the elderly. To mitigate this phenomenon, The Wireless Health Institute at UCLA has developed the SmartCane System [27]. This system performs a series of signal processing algorithms to assess the users current state. These algorithms assess various attributes such as improper cane usage, high-risk behaviors, and potential injuries (such as falling). Once these attributes are detected, the SmartCane can propagate these attributes to patients, care givers, clinicians, as well as emergency services. To accomplish signal processing and network

connectivity, the SmartCane connects to a PDA, Cell Phone, or tablet PC via Bluetooth. This central controlling unit can predict hazards, store behavioral data, notify health care professionals, as well as display visual feedback to the user [27].

3. Smart Shoe

Smart Shoe is an orthotic shoe developed in our labs at UCLA [8][21]. Through the use of gyroscopes, accelerometers, and a few well-placed pressure sensors, Smart Shoe is able to monitor feet motion and pressure distribution to evaluate the state of a patient. The Smart Shoe can currently detect the formation of foot ulcers in patients with diabetes. Similar work has been done in other labs. For instance, [19] have developed a shoe-integrated sensor system for gait analysis. Like the SmartCane, Smart Shoe wirelessly connects (via Bluetooth) to a cell phone or PDA for data processing, visualization, and network connectivity.

4. Developing Wireless Health Applications

Projects presented in section 2 share many of the same requirements. The following lists a series of features required by the aforementioned projects:

- Short Range Connectivity (such as Bluetooth or Zigbee)
- Internet Connectivity (through Wifi, CDMA, etc…)
- Visualization (such as OpenGL ES)
- Data Storage (such as SQL)
- GPS Services

Usability is an important addition to our technical requirements. End users range from health care professionals to patients (who are often elderly). Therefore, we can expect a large percentage of non-technical users who are not computer savvy. A successful wireless health application must provide a usable experience that integrates seamlessly into a patient's life. Otherwise, we risk low adoption rates.

4.1 Sample Application

To provide a basis of comparison, we created a wireless health application that interfaces with UCLA's Smart Shoe. This application connects to wireless sensors via Bluetooth and displays their output graphically. This application requires four main modules:

- Connectivity module for Bluetooth (Zigbee was not supported by any of our Smart Phones)
- Graphical module for displaying data
- Data storage module for archiving sensor data
- GPS module for associating locations with data

Our comparison is limited to Smart Phones to account for Internet connectivity. Other mobile platforms such as PDAs typically connect to the Internet through WiFi. Wireless Health Applications require a constant level of connectivity regardless of locality. This requirement cannot be satisfied with WiFi alone. Therefore, we require a more ubiquitous network such as CDMA or GSM. With such networks, patients are constantly connected regardless of their locations. However, even cell coverage has its limitations in remote locations and may experience dead spots in urban areas. However, we feel that there is no technology that offers a higher level of connectivity. Also, dead spots could often be mitigated through WiFi (supported by many Smart Phone platforms).

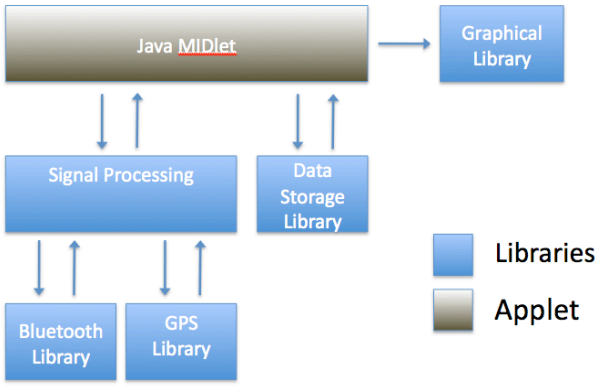

Figure 2. Our Simplified Library Architecture

5. Mobile Devices

The list of potential wireless devices is practically endless. In order to provide a useful survey, we limited our set of devices to five. Our devices include Symbian, Rim Blackberry, Windows Mobile, Android, and iPhone. Symbian, RIM Blackberry, and Windows Mobile have the three highest world market shares at 57.1%, 17.4%, and 12% respectively [1]. iPhone holds the fifth largest market share at 2.8% and offers a unique user interface paradigm unlike the four preceding platforms (Linux being number 4 at 7.3%). We chose Android to represent the Linux platform. There are several Linux platforms in the mobile market. Android was chosen due to its openness, support, and standardization offered by Google; thereby lending itself as an attractive research platform. Android also offers a similar user experience to the iPhone and serves as a suitable comparison. Android was also chosen due to Google's past record and market penetration by their other products.

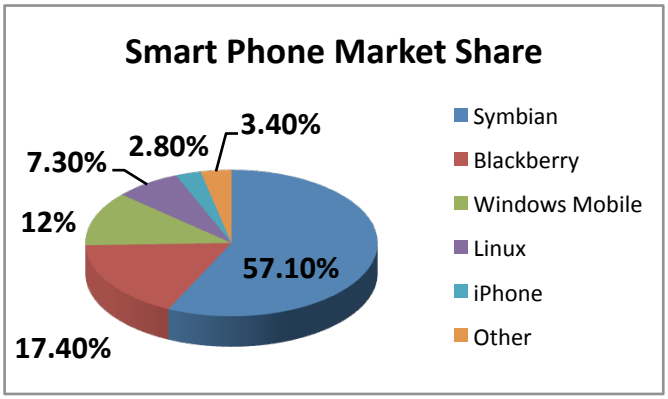

Figure 3. Smartphone market share

5.1 BlackBerry/Symbian/Windows Mobile

Blackberry, Symbian and Windows Mobile support a standard J2ME port. Applications can be compiled to a jar file and loaded to all three devices without the need for recompilation. However, while developing for such platforms, we must verify support for required JSRs for each device. Many JSRs are optional such as JSR 82 for Bluetooth support. However, we found that many of the latest devices (such as Nokia's N95) advertise support for all JSRs listed above.

5.2 Android

Android is built upon an open Linux Kernel and consists of a virtual machine optimized for mobile environments. Android uses the Java

programming language. However, their JAVA port is for the Dalvik JVM. This port consists of a mix of standard JAVA and Android specific APIs. Therefore, JAVA applets compiled against standard ports, such as J2ME, are not compatible with Android.

Android is open source lending itself nicely to research and industry. Developers can run their custom Android builds on unlocked hardware available through Google. Android also makes no distinction between applications; all applications, whether static core applications or dynamic third-party applications, are treated identically and have equal access to the device's functionality [2].

5.3 iPhone

iPhone OS offers a similar platform to MAC OS X. IPhone OS runs a variant of the same Mach kernel used by MAC OS X. This enables iPhone to support standard core services such as BSD Sockets and POSIX Threads. Also, objective C is a superset of C and C++. For these reasons, open source projects can be trivially ported to the iPhone. While working with this device we ported a simple JSON interpreter [12]. The port consisted of importing the necessary headers and source files and compiling. We found this functionality an attractive feature of iPhone. Unfortunately, it was the only platform in our selection to offer standard core OS services.

6. Platform Comparison

To host our sample application, our platforms must provide several APIs including Security, Bluetooth, GPS, storage APIs (such as SQL), standard networking APIs, and graphical APIs. Platforms must also provide a user-friendly interaction model. While not required, a large touch screen is highly attractive for such applications. Large touch screens allow us to display large text, graphics and controllers (such as buttons and lists). Much of our work is often targeted towards the elderly where vision and finger acuity is diminished. Large touch screen displays afford an experience much better suited to such users.

Both iPhone and Android had no programmatic support for any low power wireless protocols (such as Bluetooth and Zigbee). Wireless sensors are typically connected via Bluetooth or Zigbee. Support for such APIs is a strong requirement for wireless health. Without such APIs, wireless platforms are severely limited. Unfortunately, Zigbee was supported by none of our Smart Phone platforms. This is quite a drawback since Zigbee provides an extremely energy efficient wireless alternative to Bluetooth [6]. However, it is important to note that Google has announced Bluetooth support in future SDKs.

Our initial goal was to choose a single mobile device to serve our research interests. However, as we researched several devices, we found that many of these devices are extremely similar and creating a library that extends several platforms was possible.

For these reasons, we targeted Symbian, BlackBerry, and Windows Mobile. Each of these mobile platforms supports J2ME. This allows us to create libraries that we can share across all 3 platforms without the need for recompilation. Also, J2ME offers JSRs that support security (JSR 219), Open GL ES (JSR 177/239), GPS (JSR 179) and Bluetooth (JSR-82) [15]. By choosing J2ME as our target, we include a much larger set of mobile devices than those we present in this paper. These include mobile devices mot considered Smart Phones (such as low end cell phones and other embedded devices).

Figure 2 lists our APIs of interest and their respective support by our five Smart Phones. (Symbian, Windows Mobile, and BlackBerry were merged to J2ME).

Table 1: List of supported APIs

	J2ME	iPhone	Android
Bluetooth	√	-	-
WiFi	-	√	√
ZigBee	-	-	-
GPS	√	√	√
Open GL ES	√	√	√
Security Suite	√	√	√
SQL	√	√	√
Touch Screen	√	√	√

6.1 Ease of Deployment

Ease of deployment refers to a developers‘ ability to deploy applications to a handset. For our purposes, applications are often cable loaded. We found loading applications to J2ME and Android trivial. Neither platform required any licensing. Tools were free and easy to access. iPhone, on the other hand, requires developers to join their Developer Program. This process required an application as well as a nominal fee. For us, the entire process took several weeks. While we understand the business justifications for such a process, we feel that it quite a deterrent to the academic community.

6.2 Emulation and Debugging

We found Androids debug environment impressive. First, Android emulates several features such as GPS coordinates, network speeds (such as UMTS and GSM), SMS, and voice calls. Android also contains a debugger for stepping through code while monitoring various attributes such as thread states, heap usage (with manual garbage collection), and file system state. All of this is accomplished through the Eclipse IDE with no external tools with a minimal startup time. We were able to load an application and start using all these features in about 45 minutes with a little help from Androids documentation.

However, we found two areas of improvement for Android. First, we would like to see a graphical CPU monitoring that works on the device as well as the emulator. CPU monitoring only worked on the emulator. Also, the output was a small line drawn on the upper part of the screen. A graphical display that allows us to save info and correlate CPU usage to its respective code segments would be quite useful. Second, GPS simulations only worked for the emulator (as for all devices presented by this paper). We would like to see GPS simulation supported by the device as well.

For J2ME, on device debugging is quite device specific. Several manufacturers do release their own device specific debugging tools that integrate with common IDEs such as EclipseME [9] and Netbeans [20]. However, we found this fragmented experience to be quite a limitation of J2ME. While the other two platforms provided a uniform and robust debugging framework, J2ME's on device debugging support relied solely on the device manufacturer (granted, iPhone is developed for only Apple hardware).

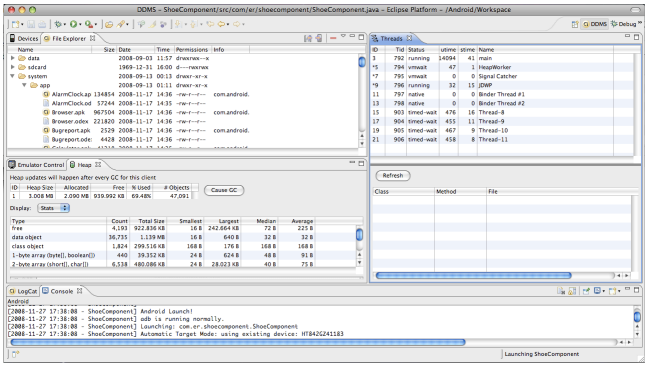

Figure 4. Android's Debugging Environment

We used the Java Debugger (jdb) from sun while debugging on the Simulator. We ran this tool from the command line as well as NetBeans and found the debugger to be quite rudimentary. We were able to set break point, check variables, and step through code as with most debuggers. However, we felt that jdb lacked the advanced features offered by Android and iPhone. The emulator supported standard simulated features, such as GPS and telephony features

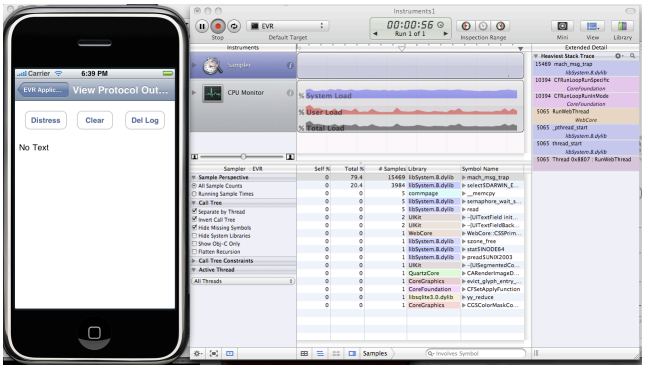

Figure 5. Apple's Instruments Tool

iPhone had the most impressive statistical tools out of the mobile devices we compared. Apple offers their Instruments tool that allows developers to graphically monitor several features such as memory usage, CPU usage, frame rate, etc. Instruments also allows developers to save traces to later analyze or send to fellow developers. Instruments not only worked for the simulator, but for the device as well. For intensive applications that require a high level of optimizations, we rank iPhone as the clear winner for support tools.

However, we were unable to emulate GPS coordinates from the simulator. While we were able to create an instance of their Location API, the API consistently returned the same coordinates at Cupertino, Ca.

7. ANALYSIS

While comparing mobile platforms, we quickly realized that development of Wireless Health libraries could extend across multiple mobile platforms using Java. With Java's compile once run anywhere environment, we gain a level of standardization that not only improves our code portability, but also provides a set of standard APIs shared across mobile devices. Of the five platforms compared, three supported a standard Java implementation (specifically J2ME). We found these three platforms tended to provide necessary APIs we need for Wireless Health such as security, Open GL ES, GPS and Bluetooth. We also found that functional requirements such as threads and background applications were supported. Through providing a standard environment, Java has not only provided a portable platform, but an environment inclusive of our requirements for Wireless Health.

Interesting to note, we also ran our initial libraries on standard Windows XP and Mac OSx laptops. Porting to such platforms involved a simple integration with the BlueCove library provided by [5] and some changes to our visual APIs. This exercise provided us with a much higher level of optimism for our Wireless Health libraries. As Java moves across many new embedded platforms, our libraries can be leveraged not only in mobile devices (such as mobile phones and PDAs) and laptops, but also by any embedded device supporting a standard implementation of Java.

With these observations, it was abundantly clear that J2ME offers the best runtime environment for wireless health applications. This does leave out both Android and iPhone in the interim. Since iPhone runs a similar kernel (Mach) as standard OSX, it is possible for apple to include a standard Java virtual machine (J2ME or J2SE). As for Android, the Dalvik JVM is not a far reach from Sun's J2ME standard. In fact, much of our code on both J2ME implementations and Android could be shared. We feel that our Wireless Health libraries could extend to Android with increased API support from Android as well as some ingenuity from us.

Android and iPhone currently have limitations that severely hinder wireless health applications. The most noticeable limitation was the lack of a low power networking APIs for technologies such as Zigbee or Bluetooth. While these devices do support WiFi, we feel this technology is too power hungry for wireless health applications. However, both of these devices do offer hardware support for Bluetooth 2.0 and could very well provide API support in the future. iPhone also lacks the ability to run background applications (without using discouraged means such as jail breaking). Wireless Health applications are often required to constantly monitor their environment. However, when using a mobile device such as a mobile phone, we must remember that these devices are intended for multiple purposes. While our medical monitoring is extremely important we cannot completely control the device rendering other functionalities inaccessible. As stated earlier, one of our requirements is the ability to seamlessly (as possible) add our applications into patients' lives.

7.1 User Experience

As noted earlier, user experience is critical to wireless heath applications. End users may or may not be technically savvy. Therefore, applications should be intuitive. In addition, many wireless health applications are intended for the elderly, such as SmartCane and Smart Shoe (described in section 2). In general, eyesight and finger dexterity decrease with age. Enhanced features such as larger attributes (such as fonts, images, and inputs) and better color contrasts are necessary to address the needs of the elderly [10][11]. Large touch screens cater to these attributes quite well. Fortunately, all five platforms presented in this paper support such displays. Hardware support was the largest limiting factor when considering user experience. We found that both Android and iPhone excelled in the area of user experience. While Blackberry, Windows Mobile, and Symbian support similar displays, there are few commercial products that utilize such a display.

8. SECURITY

This paper has purposely left out an in depth comparison of security

related features. We feel that security is extremely important and includes such a broad area, that this topic deserves its own dedicated analysis. For the purpose of this paper, we only compared whether each application supports a suite of security APIs. It is important to note that security in wireless health as well as mobile/embedded devices is an area of ongoing research. Authors in [4][13] have noted several areas of security concerns in protecting health information (wireless health info for [13]) such as authentication, confidentiality, secure links for data exchange, data integrity, and access protection for stored data. Authors in [26] have noted several potential solutions for these issues. Authors in [7][18][24] have discussed how resource constrained embedded platforms offer a new set of requirements for security measures beyond their "wired" counterparts. We feel that future work should include a survey of all these aspects and how current mobile platforms and their respective security suites help alleviate issues in security for wireless health.

9. IMPLEMENTATION

Our Analysis was completed with an implementation of a simplified library on the J2ME platform. J2ME lends its self quite nicely to developing wireless health libraries. As noted earlier, Bluetooth, GPS, SQL, OpenGL, and a security suite are all supported by J2ME.

9.1 Implementation Details

We used a similar configuration as [26] in our implementation. Our shoe consists of one MicroLeap [3] processor for both the left and right shoe. This processor connects two pressure sensors (one in the heel and toe) as well as an accelerometer and gyroscope in all three, X, Y, and Z, axes. Figure 9 shows our application running on the Nokia N95.

Figure 9. J2ME Application running on the Nokia N95

Our JAVA library consists of MicroLeap, data storage, graphic, and data processing abstractions. Each of these abstractions hides the intimate details of their respective functionalities. We also retrieve and store GPS data through the data storage APIs.

Once activated, the application retrieves sensor input from the pressure sensors, accelerometer and gyroscope. We sampled data at about 50Hz (although, much higher sampling rates are possible). This data is processed by the data processing API and stored by the data storage API. We performed basic analysis of the shoe's sensory data at runtime to determine the person's current balance. We used a basic algorithm that compared data from the left and right shoe to determine symmetry. This was accomplished by comparing the standard deviation of the X, Y, and Z accelerometer data for each foot. This rudimentary algorithm was able to determine if a person was walking abnormally (such as limping, stumbling, or shuffling).

Data stored on the device was later transferred to a PC where we could do more complex data processing. We implemented a playback mechanism for our PC using the same Java libraries. This mechanism allowed us to visually replay data retrieved by our Smart Phones

9.2 Implementation Analysis

We found some discrepancies while deploying our libraries on Windows Mobile, BlackBerry, and Nokia devices. For Windows Mobile, we used an HP iPaq. While the device has Bluetooth and GPS support, JSR 82 and JSR 179 were not a part of their JVM. For Blackberry, we used the Blackberry Pearl. This device fortunately does support JSR 82 and JSR 179. However, some minor features were not supported. For example, when using Bluetooth serial port profile (btspp) we could not set the server as the master node. This issue was also present on the Nokia N95. Fortunately, we were able to work around this dilemma with our implementation. However, this could be an issue for other implementations.

Several mobile devices and laptops require a passkey when pairing with a Bluetooth device. Typically embedded devices account for this with a hard coded passkey in their firmware. However, we found lacking support in some of our prototype hardware. While this is an issue of the embedded hardware, developers should be aware of such technicalities.

Overall, we still feel that J2ME is the best target for developing wireless health libraries. However, we found that some devices had no support for GPS and/or Bluetooth (even when hardware support was present).

We also feel that J2ME's debugging utilities are lacking. While the current jdb is sufficient for debugging issues (such as the Bluetooth discrepancies we described earlier), the experience is quite fragmented. iPhone and Android, on the other hand, offer a completely seamless debug experience with a series of tools for optimization. We hope to see similar support in J2ME in the future.

iPhone and Android are severely limited by their lack of low power connectivity APIs (such as Bluetooth and Zigbee). However, their platforms may suite intensive data processing quite well due to their optimization tools (especially iPhone). However, iPhone also lacks the ability to run background processes; a requirement necessary for wireless heath applications similar to those described in section 2.

10. CONCLUSIONS

Our initial intent was to find the best mobile platform for wireless health applications. We based our comparison on the development of a simplified wireless health library. The requirements for this library were based on several ongoing research projects in our labs at UCLA.

We developed these libraries (where possible) on five mobile platforms (Windows Mobile, Blackberry, Symbian, iPhone, and Android). Through this process, we became quite familiar with all five environments. During this process, we noticed several advantages afforded by J2ME such as a unified runtime

environment across a large number of devices. These advantages, along with the lack of necessary functionality by both iPhone and Andoird, led us to a J2ME implementation.

This exercise proved that J2ME was indeed a prime candidate for developing wireless health applications. However, we found support for GPS and Bluetooth varied across devices. Therefore, the number of devices that can actually host such applications is smaller than preferred. We hope to see a more unified support for Bluetooth and GPS on J2ME devices. We also hope for a better debugging environment, similar to that of Android and iPhone.

11. REFERENCES

[1] 2008 Press Releases, Gartner, Inc., 2008. http://www.gartner.com/it/page.jsp?id=754112

[2] Android, Open Handset Alliance, 2008. http://www.openhandsetalliance.com/android_overview.html

[3] AU, L. K., WU, W. H., BATALIN, M. A., MCINTIRE D. H., KAISER, W. J. 2007. MicroLEAP: Energy-aware Wireless Sensor Platform for Biomedical Sensing Applications. Biomedical Circuits and Systems Conference, BIOCAS 2007. Montreal, Canada, November 2007, 158–162

[4] BlackBerry Developer Zone, Research In Motion Limited. 2008. http://na.blackberry.com/eng/developers/

[5] BlueCove, BlueCove.org. 2008. http://www.bluecove.org

[6] CALLAWAY, E., GORDAY, P., HESTER, L., GUTIERREZ, J. A., NAEVE, M., HEILE, B., BAHL, V. 2002. Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks. IEEE Communications Magazine, 40(8), August 2002, 70–77.

[7] CLAYTON P. D. 1997. For the Record: Protecting Electronic Health Information. National Research Council, National Academy Press. Washington DC. 1997.

[8] DABIRI, F., VAHDATPOUR, A., NOSHADI, H., HAGOPIAN, H., SARRAFZADEH M. 2008. Ubiquitous Personal Assistive System for Neuropathy. The 2nd International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet), in conjunction with ACM MobiSys, Breckenridge, Colorado, July 2008.

[9] EclipseME, EclipseME.org. 2008. http://eclipseme.org/

[10] HANSON, V. L. 200. Web access for elderly citizens, in Proceedings of the Workshop on Universal on Accessibility of Ubiquitous Computing, WUAUC'01, Alccer do Sal, Portugal, May 2000, 17 – 24.

[11] HANSON, V. L. 2004. The user experience: designs and adaptations. In W4A: Proceedings of the international cross-disciplinary workshop on Web accessibility, New York, NY, USA. May 2004. 1-11.

[12] Introducing JSON, JSON. 2008. www.json.org

[13] iPhone Developer Center, Apple. 2008. http://developer.apple.com/iphone/

[14] JAFARI, R., BAJCSY, S., GLASER, B. GNADE, M. SGROI, and SASTRY, S. 2007. Platform design for health-care monitoring applications. In Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, Boston, MA, USA, June 2007, pp. 88–94.

[15] Java ME at a Glance, Sun MicroSystems. 2008. http://java.sun.com/javame/index.jsp

[16] JOVANOV E., PRICE J., RASKOVIC D., KAVI K., MARTIN T., and ADHAMI R. 2008. Wireless Personal Area Networks in Telemedical Environment. In Proc. 3rd Int. Conf. Inf. Technol. Biomed. Arlington, VA, Nov. 2000, 22-27.

[17] LO, B., THIEMJARUS, S., KING R., and YANG, G.Z. 2005. Body Sensor Network - A Wireless Sensor Platform for Pervasive Healthcare Monitoring. In Adjunct Proceedings of the 3rd International Conference on Pervasive Computing, May 2005, Munich, Germany, 77-80.

[18] MILLER, S. K. 2001. Facing the Challenges of Wireless Security. In IEEE Transactions on Computers, July 2001, 46–48.

[19] MORRIS, S.J. and PARADISO, J.A. 2002. A compact wearable sensor package for clinical gait monitoring. Motorola Journal, 2002. 7-15

[20] NetBeans, NetBeans.org. 2008. http://www.netbeans.org/

[21] NOSHADI, H., AHMADIAN, S., DABIRI, F., NAHAPETIAN A., STATHOPOULUS, T., BATALIN, M., KAISER, W., SARRAFZADEH, M. 2008. Smart Shoe for Balance, Fall Risk Assessment and Applications in Wireless Health. Microsoft eScience Workshop Indianapolis, IN, December 2008.

[22] OTTOAND, C., MILENKOVIC, A., et al. 2006. System Architecture of a Wireless Body Area Sensor Network for Ubiquitous Health Monitoring. Journal of Mobile Multimedia. 1(4), 307–326.

[23] PENTLAND A. 2004. Healthware: Medical Technology Becomes Wearable. IEEE Computer, 37(5), May 2004, 42-49.

[24] POTLAPALLY, N., RAVI, S., RAGHUNATHAN, A., and LAKSHMINARAYANA G. 2002. Algorithm exploration for efficient public-key security processing on wireless handsets. In Design, Automation and Test in Europe (DATE), Nice, France, Mar. 2002, 42-46.

[25] SHNAYDER, V., CHEN, B., LORINEZ, K., FULFORD-JONES, T. R. F., WELSCH, M. 2005. Sensor Networks for Medical Care. In Harvard University Technical Report TR-08-05, 2005.

[26] WARREN, S., LEBAK, J., YAO, J., CREEKMORE, J., MILENKOVIC, A., and JOVANOV, E. 2005. Interoperability and Security in Wireless Body Area Network Infrastructures. In Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Shanghai, China, 2005. 3837-3840.

[27] WU, W., AU, L., JORDAN, B., STATHOPOULOS T., BATALIN M., KAISER, W., VAHDATPOUR, A., SARRAFZADEH, M., FANG M., CHODOSH, J. 2008. The Smart-Cane System: An Assistive Device for Geriatrics. Third International Conference on Body Area Networks (BodyNets 2008), Florence, Italy, March

Flexible RFID Location System Based on Artificial Neural Networks for Medical Care Facilities

Hao-Ju Wu, Yi-Hsin Chang, Min-Shiang Hwang, Iuon-Chang Lin

g9729007@mail.nchu.edu.tw, mika830@gmail.com, mshwang@nchu.edu.tw, iclin@nchu.edu.tw

Department of Management Information Systems, National Chung Hsing University, 250 Kuo Kuang Road, 402 Taichung, Taiwan

Abstract

RFID location systems are often used in real-time location systems that come up with the problems like multipath phenomenon and layout changing. These make locating difficult because most of the location systems are based on fixed mathematical calculation that cannot take these situations into account. Using artificial neural network, our location scheme can learn the geography features to adapt to the real world. It could avoid multipath phenomenon effect and be flexibly applied to any environment. The experimental processes and result are shown in the end of the paper.

Keywords: Real-time location system (RTLS), Radio frequency identification (RFID), Back propagation network (BPN), Received signal strength indicator (RSSI).

1. Introduction

Radio Frequency Identification (RFID) is a fast growing automatic data retrieval technology that has become very popular in supply chain, retail logistics, and other applications [1]. Nowadays, RFID location and tracking application is also important that can be helpful to support the asset tracking and equipment management.

RFID location systems are often be used in Real-time location systems (RTLSs). Location systems come up with the problems that signal reflection of walls, ground, and objects are received from various directions over a multiplicity of paths, called multipath phenomenon [1][2]. Moreover, the layout of objects is likely to be changed in many cases. These make locating difficult because that most of the location systems [3][4][5] are based on fixed mathematical calculation that the calculation model should be reconstructed when the layout of objects changing.

Artificial neural network is a learning algorithm that can automatically learn the features of input and create appropriate output. In this paper, we locate the user's position by applying the Back Propagation Network (BPN).

The rest of the paper is organizes as follows. In section 2, the brief introduction of Received Signal Strength Indicator (RSSI) and the Artificial Neural Networks (ANNs) will be given. Section 3 describes our proposed scheme. The experimental processes and result are discussed in section 4. Finally, we provide some conclusions in the last section.

2. Related Works

In this section we brief introduce the Received Signal Strength Indicator (RSSI) and the Artificial neural networks (ANNs).

2.1 Received Signal Strength Indicator (RSSI)

Many location systems use the Received signal strength indicator (RSSI) to calculate the distance between user and reader. RSSI is the signal strength received from the reader antenna [1]. RSSI decrease by the distance between the user and reader according to the path loss model. But the path loss model is not fixed, it impacted by geography condition, reflection of walls, ground, and even layout of objects like barriers or a big desk. That is, maybe two RSSIs are the same, but indeed their distance to reader are different. These features make the fixed mathematical model difficult to construct. Moreover, if we use fixed mathematical model to locating the user's position, we may have to reconstruct a new model for location when the geography condition changing manually.

2.2 Artificial neural networks (ANNs)

Artificial Neural Networks (ANNs) are information processing tools inspired by the learning ability of the human brain. About the theories and functions we can find in Hecht-Nielsen's paper [6]. ANNs can automatically learn the features of inputs and create appropriate outputs that users don't need to know the hidden processes between them.

There are three layers in the ANNs: the input layers, the hidden layer, and the output layer. In this paper, the ANN used is the Back propagation network (BPN). There are two phases in BPN, the training phase and the predicting phase. When we get the training data set, we define the input and the corresponded expected output. BPN would automatically create the model that satisfies the training data set as much as it can, calls the training phase. After the model is created, we can use it to predict the outputs corresponded to the new inputs, calls the predicting phase.

Using this feature, we collect the RSSIs of RFID readers as the inputs of BPN, and let the corresponded position be the expected outputs to train the collecting data. After the model is created, we apply it to predict the positions by giving new RSSIs. Therefore, our scheme doesn't compute the mathematical model and virtually take the geography condition into account because that the RSSIs in the specific zone is the result of multipath phenomenon and other condition effect.

3. Proposed Scheme

Proposed scheme locating the user's position by using BPN modeling that can real-time locate which zone the user is. Proposed scheme can be divided into three phases: the data collection and pre-processing phase, the neural network training phase, and the neural network predicting phase. The three phases are described in the following paragraphs.

3.1 The Data Collection and Pre-processing Phase

We put three RFID readers in the location area, and all of them can sense the signals in the whole location area. Firstly, we divide the location area to predefined *n* zones, calls Z_1 to Z_n. For example, we divide the location area to 2x3 zones, shown in Figure 1. The marked numbers 1 to 6 are the dividing zones of the location area Z_1 to Z_6, and R_1 to R_3 are RFID readers.

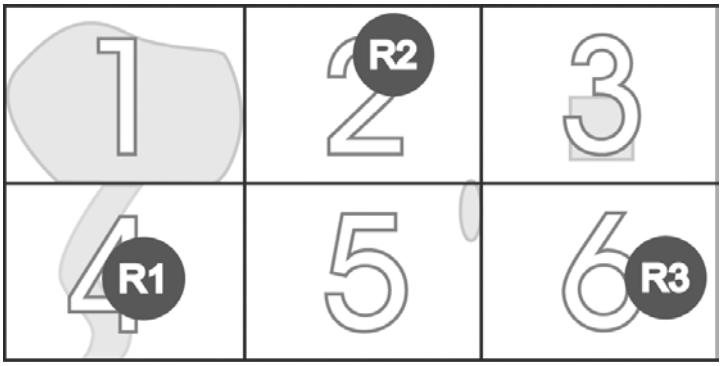

Figure 1: Example of the location area map

Then we record the RSSIs of each reader in every zone. There are two ways to collect the training data: one is going around in the whole zone to collect real data, and another is stay in the center of zone to get more intensive data. In our experiment, stay in the center of zone make the location more accurate than the another one.

We should normalize the collecting data because that data input and output in BPN are in the range of 0 to 1. We perform the normalization to the received RSSIs according to the following Equation (1). The variable x_i is the original received RSSI, and x_i' is the normalized RSSI. $x_{\max}$ and $x_{\min}$ are the maximum and minimum of all the received RSSIs in the whole location area.

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

3.2 The Neural Network Training Phase

BPN is a learning model that consists of three layers: the input layers, the hidden layer, and the output layer. In this paper, the input units are the received RSSIs of the readers R_1, R_2 and R_3. The output units represent the user's position. We use the normalized RSSIs calculated in previous phase as the input unit so that there are 3 units in the input layer, corresponded to the three readers.

We use the zone number of the location area as the output of BPN. If the location area is divided to *n* zones, there *n* units in the output layer, corresponded to the *n* zones. The value 1 represents that the user's position is the corresponded zone, whereas the value 0 means that the user is not in the corresponded zone. For example, if the zone number is 1 and the location area is divided into 6 zones, the output units should be 100000; if the zone number is 2, the output units should be 010000.

The number of hidden layer unit is generally defined by the following two approaches:

$$Nhidden = \frac{Ninput + Noutput}{2} \tag{2}$$

$$Nhidden = \sqrt{Ninput \times Noutput} \tag{3}$$

In our scheme, the number of hidden layer unit is defended according to the Equation (3).

In this example, there are 3 units in the input layer, 4 units in the hidden layer, and 6 units in the output layer. The structure of BPN is shown in Figure 2.

Then we can use the data set collected in the previous phase to train the BPN, and after training we would get the locating model of this location area.

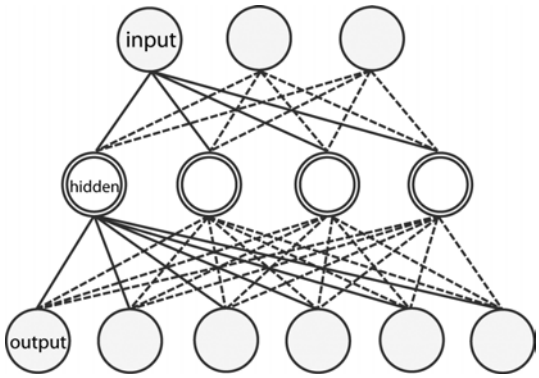

Figure 2: Example of Neural Network Structure

3.3 The Neural Network Predicting Phase

After the locating model created, we can use the model to predict the user's position. Firstly, we load the parameters of the model to BPN, and then normalize the newly received RSSIs as the input. The output is the prediction of the user's position.

When the geography or layout of objects is changed, we can simply retrain the BPN to get the new model, and then load the new model to locate the user's position. These processes can be automatically done so that we don't need to reconstruct the mathematical model manual.

4. Experimentation

Our experimental location area is in the outdoor lawn, the ground has dimension of 9 m by 18.3 m. The location area is divided into 6 zones that each zone has dimension of 4.5 m by 6.1 m, as Figure 1 shown. The gray marked regions are trees, stones and other big barriers, and R_1 to R_3 are RFID readers. The users take the RFID tags in the hand and stay in the center of zone to record the RSSIs. In each zone, we record 10 sets of RSSIs and then go to the next zone. The specification of RFID readers and tags we use are shown in Figure 3, Table 2 and Table 2.

The 60 sets of RSSIs are used to train the BPN. There are 3 units in the input layer, 4 units in the hidden layer, and 6 units in the output layer. The structure of BPN is shown in Figure 2.

In our experimentation, the correct rate is instable, generally between 60% and 90%. We find out that the accuracy is decreased when the weather change. For example, the model created in a dry and hot day performs well in the sunny days whereas performs poor in the raining days. The temperature and humidity would be important features in our experimentation that affect the accuracy of model. In the future work, the temperature and humidity should be taken into account. The inputs of BPN should be the RSSIs, temperature and humidity.

Figure 3: RFID readers and tags in experimentation

Table 1: Specification of experimental RFID readers

Communication	2.45 GHz Support read and write
Frequency	2.40~2.48 GHz
Channel	255
Address	65536
RSSI	0-255
LQI	0-255
Programmable	Set Parameter
LED	Reader action or R/W status
Ethernet	10BASE-T/100BASE-TX port, 10/100Mbps auto-sensing
RS232	RX,TX
RS485	+,-
Protocols	ICMP, ARP, IP, TCP(Server/Client), UDP, DHCP, HTTP
Baud Rate	2,400 bps ~ 115,200 bps
Power Input	7.5 VDC ~ 28 VDC
Action Current	500 mA @ 9 VDC MAX
Operating Temperature	-20 °C to 65 °C, 5 to 95%RH
Storage Temperature	-30 °C to 85 °C, 5 to 95%RH
Dimension	107W x 138H x 30D (mm)

Table 2: Specification of experimental RFID tags

Frequency	2.45GHz support read/write
Channel	256
Address	65536
RSSI	0-255
ID	64 bit
LED	Indicates the status of the signal transmission and battery power
Replaceable Batteries	CR2032 3VDC * 2
Operational Life	1 ~ 3 years
Operational Temperature	-20 °C to 60 °C, 5 to 95%RH
Storage Temperature	-30 °C to 70 °C, 5 to 95%RH
Dimension	86W*54H*6D (mm)

5. Conclusion

Using artificial neural network, our location scheme can learn the geography features to adapt to the real world. It would take the geography and reflection of walls, ground, and layout of objects into account. Therefore, it could avoid multipath phenomenon effect and be flexibly applied to any environment. If the geography or layout of objects is changed, we can simply retrain the BPN to get the new model to locate the user's position. In the experimentation, the accuracy of scheme is generally between 60% and 90%. We find out that the temperature and humidity would be important features that should be taken into account. In the future work, the inputs of BPN should be the RSSIs, temperature and humidity.

Reference

[1] Aman Bhatia, "Analysis of Different Localization Techniques in Indoor Location Sensing using Passive RFID," *Term Report of Department of Electrical Engineering, IIT Kanpur*, 2007.

[2] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Processing Magazine*, vol. 22, no. 4, 2005.

[3] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System", *IEEE Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings (IEEE INFOCOM'00)*, pp. 775-784, 2000.

[4] L. M. Ni, Y. Liu and A. P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *Wireless Networks*, vol. 10, no. 6, pp. 701-710, 2004.

[5] Guang-yao Jin, Xiao-yi Lu, and Myong-Soon Park, "An Indoor Localization Mechanism Using Active RFID Tag," *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, pp. 40-43, 2006.

[6] R. Hecht-Nielsen, "Theory of the backpropagation neural network," *International Joint Conference on Neural Networks (IJCNN'89),* pp. 593-605, 1989.