Challenges in the Regulatory Approval of Medical Cyber-Physical Systems*

Oleg Sokolsky Insup Lee
Department of Computer and Info. Science
University of Pennsylvania
{sokolsky,lee}@cis.upenn.edu

Mats Heimdahl
Department of Computer Science
University of Minnesota
heimdahl@cs.umn.edu

ABSTRACT

We are considering the challenges that regulators face in approving modern medical devices, which are software intensive and increasingly network enabled. We then consider assurance cases, which offer the means of organizing the evidence into a coherent argument demonstrating the level of assurance provided by a system, and discuss research directions that promise to make construction and evaluation of assurance cases easier and more precise. Finally, we discuss some recent trends that will further complicate the regulatory approval of medical cyber-physical systems.

General Terms

Verification, Legal aspects

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*human safety,regulation*; J.3 [**Computer Applications**]: Life and medical sciences

Keywords

Medical cyber-physical systems, certification, assurance cases, virtual medical devices

1. INTRODUCTION

Modern clinical practice relies on a wide variety of medical devices to assist in the treatment of patients. Many of these devices perform functions that are critical to patients' lives. It is extremely important that these functions are performed in a safe manner. Like most other safety-critical systems, medical devices are regulated by government agencies. In the Unites States, for example, the Food and Drug Administration (FDA) is responsible for approving all medical devices for marketing.[1]

In this paper, we discuss challenges that regulatory agencies face in certifying modern medical devices. Medical devices and systems are a tough domain to regulate. On the one hand, there is a high degree of awareness that medical devices immediately affect patients' well-being and life, so they are expected to be very safe. At the same time, a device can be made "too safe" and interfere with the practice of medicine by the doctors. That is, if a doctor needs to induce a device to perform a potentially unsafe operation to save a patient's life, the safety functions of the device should not prevent him or her from performing this operation, even if this operation is considered unsafe under normal operating conditions. From the regulator's perspective, there are also conflicting perspectives to take into account. If a potentially unsafe device is approved for use, patients may be harmed by the device. On the other hand, if a device is not approved because of a safety concern that may never materialize in practice, many more patients may be harmed since an effective treatment will not be available.

Assessing safety of medical devices is particularly difficult because they typically operate in a highly unpredictable environment. The largest source of uncertainty are patients themselves. Even within the same target group of patients – such as children or the elderly – there is a tremendous variation of physiological parameters. Elderly patients, in particular, often have multiple conditions that interact with each other in unpredictable way. Accurately predicting possible interactions between a device and a patient is extremely challenging. Clinical trials, an accepted way of evaluating safety of drugs, are difficult and expensive to arrange, especially for implantable devices.

Another source of unpredictability in the environment comes from caregivers. Doctors and nurses operate under a lot of stress; their primary concern is patients, not devices. Treatment guidelines for the same condition differ significantly from hospital to hospital, and caregivers are highly creative in adapting the use of devices to the needs of their workflows and finding workarounds for restrictions imposed by a device. Predicting the ways in which a device will be used in the hospital is quite a challenge.

The challenges posed by the unpredictable operational environment of medical devices are, of course, not new and

*This research is supported in part by NSF grants CNS-0930647, CNS-1035715, and CNS-1042829 (FDA SIR)

[1]FDA policy is making a distinction between certification and approval, and the FDA mandate involves only approval of medical devices. In this paper, for brevity, we use certification to apply to both activities.

both device manufacturers and regulators are acutely aware of them. Recent technological advances, however, are introducing a host of additional challenges. Most of these challenges come from interactions between devices. More and more medical devices are network-enabled and can communicate during treatment with other devices, forming medical cyber-physical systems (MCPS). Communication allows MCPS to implement functionality, such as continuous care, that was not possible with stand-alone devices. At the same time, communication within an MCPS brings new hazards to patient safety and needs to be considered in certification. In addition to network failures, there are now security and privacy concerns to address. Furthermore, unlike systems in other safety-critical domains (e.g., aircraft), MCPS often need to be assembled at the patient's bedside using the devices required for a particular clinical scenario; devices that typically come from several vendors that may not have developed the devices to operate in concert. This requirement brings new regulatory challenges that seem to be unique to the medical domain.

In addition to the inherent problems with medical device systems, there are two industry trends that add additional challenges to an already challenging problem. First, in an effort to reduce development time and more cost effectively develop high-quality medical devices, there is a current trend in the medical device industry to adopt model-based development techniques and rely more on automation, for example, modeling, automated verification, code generation, and automated testing. The reliance on tools rather than people, however, introduces new and poorly understood sources of problems, such as the level of trust we can place in the results of such automation. Second, the move towards electronic health care records and the integration of medical devices and medical information systems is another source of concern. This integration, for example, integration of infusion pumps with medical records where the pump takes dosage information directly from the patient's electronic health record, makes the medical information system part of the medical device. Unfortunately, there is no established safety culture in the medical information systems domain. Incorrect or corrupted data provided to critical medical devices through this integration could have catastrophic and widespread consequences; techniques to assure the validity and integrity of the data provided by medical information systems are needed.

2. CHALLENGES IN SOFTWARE CERTIFICATION

Conventional notions of certification, applied to physical systems, are usually *product based*. By this we mean that safety of the product is assessed based on measurements performed on the product with desired level of confidence. By contrast, objective measurement-based evaluation technology for assessing software does not exist. Software does not fail like physical devices; when software "fails," it is a result of a design fault introduced somewhere during the software development process. Thus, the thinking has been that if we just improve our processes to eliminate design faults, the software will be of high quality. Therefore, most widely used standards, for example, IEC 61508 [9], IEC 62304 [10], DO-178B [26], and the former (British) Defence Standards 00-55 and 00-56 [24, 25], are focused on the development process and either recommend or require various development and assessment techniques. The IEC 62304 standard, in particular, defines the life cycle requirements for medical device software. It outlines processes, activities, and tasks in order to establish a common framework for the development of safe medical device software. Similarly, the regulatory approach for software-based devices taken by the FDA is based on design control provisions outlined in the Title 21 Code of Federal Regulations (CFR) Part 820 (21 CFR 820.30). These standards aim to provide good processes that help to introduce fewer problems during the development as well as detect and eliminate more problems in the process.

There are doubts, however, if there is really a correlation between the quality of the software produced and the practices required by a standard [22]. Although the observations are largely anecdotal, there are some indications that developing software to a higher "safety level" (in this case to the DO-178B Level-A classification) does not necessarily lead to lower failure rates [28] as compared to software developed to a lower classification (Level-B). Such findings casts doubt on the general approach of process oriented standards.

Furthermore, since process standards are prescriptive, adoption of new, potentially much more effective, techniques can be severely hindered; why would a manufacturer deviate from accepted practices when such deviation will carry a significant regulatory risk?

To address this problem we believe a move to an *evidence based* approach to certification and approval is in order. One such approach is to organize evidence using the concept of a *safety case* or, more generally, *assurance case*. The concept of a safety case is not new. In [2], a safety case is defined as "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment." An assurance case extends the argument to other system properties as well. For example, since FDA approves devices based on their safety and effectiveness, it may make sense for the assurance case for a device to cover both. Currently, however, this is not required. FDA is currently evaluating the use of assurance cases in the approval process. A draft guidance document has been issued as part of the Infusion Pump Improvement Initiative [30]. The document expresses the FDA expectation that applications for new infusion pumps will include an assurance case arguing that the new device is "substantially equivalent" to existing devices on the market.

2.1 Model-Based Development

Software development in critical medical device systems is a largely a manual process. Validation that we are building the right system has been achieved through requirements and design inspections and reviews. Verification that the system is developed to satisfy its requirements is archived through inspections of design artifacts and extensive testing of the implementations. This is a costly process and if we could devise techniques to help us reduce the cost of development and time to market, significant competitive advantages could be gained. One current trend of interest in the medical device domain is *model-based development* as an attempt to decrease cost and reduce development cycle time.

In model-based development, the development effort is centered around a formal or semi-formal model of the proposed software system. Through manual inspections, formal

verification, and simulation and testing we convince ourselves that the model possesses desired properties. Ideally, the implementation is then automatically and correctly generated from this model and little or no additional testing of the implementation is required.

There are several commercial and research tools that aim to provide part or all of these capabilities, for example, Simulink and Stateflow from Mathworks [20, 21], Esterel and SCADE from Esterel Technologies [4, 5], and various UML tools from a collection of vendors.

Reliance on modeling and automation gives rise to several new challenges for certification and approval. Naturally, it is now imperative that the model serving as the basis for the development is correct and the increased reliance on tools requires that they can be trusted so that the results can be used as evidence in certification.

The model validation problem has received relatively little attention and the sufficiency of the validation activities has been largely determined through ad-hoc methods. The fundamental problem is how to evaluate model fidelity and the appropriate level of abstraction used in the model. Several questions have to be addressed to adequately address this issue: How do we know when we have provided enough validation? How can we catch unstated requirements that are not captured in the model? What techniques are acceptable for model validation? A natural first step to address this problem is to adapt the elicitation and validation techniques developed in the requirements engineering community [3] for the model-based domain.

We have taken initial steps and investigated how to use formal verification techniques to aid in the validation of natural language requirements, the formalized version of those requirements, and the models [23]. Whalen *et al.* have investigated if the notion of test-adequacy coverage criteria can be extended to apply directly to the requirements of a model as opposed to the model itself [32]. Nevertheless, the crucial validation problem necessitates further investigation.

With respect to the use of automated tools in the certification and approval process, the community has largely ignored addressing the issue of how we will be able to trust our powerful automated techniques enough to allow us to use them as a replacement for traditional testing and inspections.

Problems with automation can manifest itself in many places. For example,

- If the model execution environment and analysis tools misrepresent the semantics of the modeling language, all testing and verification done in the model domain is invalid.
- If the code generation is incorrect, the resulting implementation could be incorrect, and the problem may not be caught since we are now reducing testing in the code domain.
- If any of our analysis tools applied in the model domain provide false negatives (i.e., they fail to catch a faulty model), we may mistakenly accept a model as correct and use it for code generation (again, this problem is unlikely to be caught with the reduced code testing).

Solutions to such problems must be provided before we can confidently use extensive automation in the development of critical systems as well as evidence in certification.

A potential weakness of model-based development and increased use of automated techniques is the possible loss of *"collateral validation"*. In the current process, experienced professionals provide informal validation of the software system while designing, developing code, or defining test-cases; if there is a problem with the specified functionality of the system, they have a chance of noticing and taking corrective action. As we move towards more automated processes, there may be fewer opportunities for such informal validation. Currently, we do not have direct evidence on the loss of collateral validation. We have limited knowledge on how well the current manual processes work, or how error prone the current tools are. Without it, it is hard to be confident that model-based processes are improving the effectiveness of validation. Much analytical and empirical research is needed to help answer these questions.

3. MAKING ASSURANCE CASES WORK

We note that, by themselves, assurance cases do not solve the problem. Ensuring that the argument presented by an assurance case is valid remains a significant challenge. Existing languages for constructing assurance case, such as the Goal-Structuring Notation (GSN) [16], allow us to represent the argument as a graph of interdependent goal, strategy, and evidence nodes. Goals specify claims made in the argument, strategies describe how subgoals are tied together to demostrate larger claims, with evidence demonstrating satisfaction of claims. Such organization brings out the structure of the argument and makes it easier to evaluate it.

Nevetheless, goals and strategies are specified using the natural language, opening the door to inconsistencies in arguments. A study by Greenwell [7] has shown that logical fallacies are common in assurance cases. It is desirable to have means of checking logical consistency of an argument. Complete formalization of the claims in an assurance case is unlikely to be achieved. However, a more lightweight approach, which does not capture the meaning of the argument, but only its logical structure, may be possible. Such a lightweight formalization will be able to capture logical fallacies, that is, inconsistent arguments that cannot be true regardless of what is being argued. We are currently exploring the use of multi-sorted first-order logic for such formalization. The work is in its early stages, however, and much further research is needed.

3.1 Evaluation of Assurance Cases

A separate challenge lies in the evaluation of the level of confidence in the claims justified by the evidence used in the assurance case and matching this level of confidence against the safety risks posed by the device. There are several factors that affect the level of confidence. One factor is the confidence in the available evidence. Testing results, for example, may not give high enough confidence since testing typically addresses only a small subset of the system's behavior. Correct-by-construction development, on the other hand, may provide precise guarantees, but it usually relies on a set of assumptions and we may not be completely confident that these assumptions hold, or that all assumptions have been identified. Another factor is the confidence in strategies being used in the argument. That is, whether evidence is properly used to support a claim and whether claims are properly decomposed into sub-claims. Finally, there is always uncertainty in whether the argument is "good

enough." That is, whether the level of assurance delivered by the argument is commensurate with the risks.

A recent approach proposed in [8] partially addresses this problem. There, an assurance case is complemented by a separate confidence case. A confidence case argues, separately for each claim in the assurance case, that there is sufficient confidence in support of the claim. Note that this approach addresses only the first two factors identified above. The "good enough" aspect is still left for the evaluator to assess. Moreover, it seems that the evaluator now needs to make separate decisions on whether each claim in the confidence case is indeed sufficient. We believe that there should be a more systematic, and hopefully more quantitative, way of evaluating the adequacy of claims in an assurance case. However, much more research is needed to achieve this goal.

3.2 Assurance Case Patterns

Developing a high-quality assurance case from scratch is a challenging task. At the same time, similar claims and argument strategies appear in many assurance cases over and over again [15]. A pattern can reduce the effort to develop an assurance case and create a record of successful arguments for future applications. At the same time, it is important that developers use the pattern that fits the argument and do not attempt to twist the argument to fit a chosen pattern [14]. We believe that additional research is needed to develop guidance for the appropriate selection and interpretation of assurance case patterns.

Pacemaker project. In our prior work, we have developed a partial assurance case for a software controller for an implantable pacemaker [11]. The case study was motivated by the Pacemaker Challenge, the certification challenge problem issued by the Software Certification Consortium [29]. The challenge involves the development of pacemaker controller software that is formally verified for compliance with the timing requirements released by Boston Scientific. Since our case study started with the requirements and not with hazard analysis, our assurance case did not include the argument that requirements are adequate for mitigating all the hazards. However, we believe that the safety argument in this case would decompose nicely around the requirements: the argument about requirements adequacy would form a separate branch in the assurance case.

One of the goals of the project was to develop an assurance case pattern for system constructed through the model-based development process. The process we followed involved formal modeling of the pacemaker controller using timed automata, followed by formal verification by the UPPAAL model checker and code generation by the TIMES tool (see [12] for more detail). Our premise was that systems developed using similar process would rely on a similar safety argument. The proposed assurance case template was making the argument by requirements satisfaction. We claimed that the model was correct with respect to every requirement, using the verification result as evidence. Then, we claimed that the platform-independent code generated by the TIMES tool was preserving behavioral properties of the model, using correctess proof of the code generation algorithm as evidence. In addition, we claimed that the glue code added to port the code to the chosen platform had no side effects and thus did not interfere with the properties of the generated code. The evidence was the outcome of the glue code inspection. Finally, we claimed that every requirement was satisfied with the prescribed tolerance by appealing to the test results.

The resulting assurance case captured our understanding of the guarantees provided by the model-based process we used. However, we had no clear way of evaluating, how convincing the assurance case was, in particular because we had no frame of reference. We therefore felt the need to explore other ways to structure assurance cases, such as the ALARP ("as low as reasonably practicable") pattern [15], for a system constructed through a similar development process. We are currently in the pursuing this direction as part of the GPCA project [17]. The project aims to develop a generic patient-controlled analgesic (PCA) infusion pump based on the safety requirements developed in conjunction with researchers at the FDA [1]. We followed a similar model-driven process based on timed-automata modeling and code generation. While this is a work in progress, we believe that by comparing the resulting assurance cases we would be able to evaluate the relative effectiveness of the patterns.

4. CERTIFICATION OF VIRTUAL DEVICES

Medical devices are increasingly capable of interacting with each other by exchanging data and, possibly, control commands. A collection of devices that is used together to treat a patient, from a regulatory perspective, constitutes a medical device that needs to be certified or approved as a whole.

A distinguishing aspect of safety-critical systems in the medical domain is the dynamic nature of medical device systems. Given a patient with a complicated condition, caregivers need to enact a clinical scenario that fits the condition. For this, they utilize medical devices from the set of available equipment in the hospital. The resulting *ad hoc* system cannot be realistically certified in advance because of the large number of possible combinations of devices. Moreover, if the devices used in this collection are built by different manufacturers, it is not clear who is the "manufacturer" of the system to bring the case up to the regulatory authorities. Yet, a poor choice of devices can adversely affect patient safety, even though each device may be individually approved for use.

A new and somewhat extreme idea is to certify common scenarios based on types of devices that are included in the scenario. A scenario would act as a *virtual medical device* (VMD). We believe that a VMD can be approved instead of every possible instantiation of the VMD using different physical devices. Assuming certain capabilities of the devices and properties of interconnections between them, a safety of the scenario can be assessed. Then, when devices for a scenario are assembled together at the hospital, checking that the selected devices satisfy the assumptions made for the scenario, we will ensure that patient safety is guaranteed. The approach advocated by the Medical Device Plug-and-Play (MD PnP) interoperability initiative [6] would automate this assumption checking, further improving patient safety. The MD PnP architecture incorporates a centralized supervisor module in an implementation of any clinical scenario. The supervisor includes a specification of the scenario and its assumptions. Devices connecting to the supervisor would announce their capabilities, allowing the supervisor to check the assumptions before allowing the scenario to proceed.

Of course, for this procedure to be sound, we need to ensure that instantiation of a VMD is performed in a safe

manner. In particular, a VMD instantiation needs to be supported by a network infrastructure that will manage devices used in the scenario, ensure that flows of data between devices are as prescribed by the scenario, and failures of devices and communication links are detected and brought to the attention of caregivers. The infrastructure, together with the supervisor, needs to be approved as a medical device, since its operation directly affects patient's safety. We are designing a prototype of such a network infrastructure [19, 18] and expect to use it in developing approaches to VMD certification.

Foundations for this approach can be traced to the notion of modular assurance cases [13]. An assurance case module has an interface that specifies which claims made inside the module are public, that is, can be used in other modules and, conversely, which claims are assumed to be argued in other modules. Similarly, an assurance case for a VMD will rely on claims made in assurance cases for the individual devices. However, a significant difference is that modular assurance cases are intended to argue about concrete system implementations. By contrast, the composition of assurance claims for a VMD occurs when an instantiation of the VMD is assembled. One can imagine that a device carries a record of its public claims. These claims will be delivered to the supervisor for assessing the VMD instantiation. A similar vision has been put forth by Rushby [27]. However, much research is needed before this vision can be realized, and the notion of dynamic certification remains controversial.

The notion of approving a VMD separately from its physical realization may sound extreme. However, we note that medical "apps" that are intended to be executed on mobile platforms such as cell phones are gaining popularity with physicians. These apps are software applications that are able to utilize the sensors available on a mobile platform to perform functions that, until now, were performed only by medical devices. For example, a microphone can be used to turn a cell phone into a stethoscope. A recent draft guidance issued by the FDA [31] identifies many such apps as medical devices and discusses the regulatory approach for such virtual devices. Given that mobile platforms are rapidly evolving and an app is likely to run on multiple versions of a platform, we believe that approval of mobile apps will encounter similar challenges to the ones facing VMDs.

5. CONCLUSIONS

We have presented a set of challenges that face the regulatory approval of MCPS. Some of these challenges are similar to other safety-critical industries, while others appear to be specific to the medical device industry. Regulators are well aware of these challenges and are interested in finding scientifically sound solutions to them. This offers an opportunity to the research community to get involved in this intellectually deep and societally important problem.

Acknowledgements. The authors thank Paul Jones of the FDA for his insightful comments on an early draft of this paper. Shaohui (Vincent) Wang and Anaheed Ayoub participate in assurance case research at the University of Pennsylvania and contributed in the discussion of future research directions.

6. REFERENCES

[1] D. Arney, R. Jetley, P. Jones, I. Lee, A. Ray, O. Sokolsky, and Y. Zhang. Generic infusion pump hazard analysis and safety requirements version 1.0. Technical report, University of Pennsylvania, February 2009. Department of Computer and Information Science Technical Report No. MS-CIS-08-31.

[2] P. G. Bishop and R. E. Bloomfield. A methodology for safety case development. In *Safety-critical Systems Symposium*, February 1998.

[3] B. Cheng and J. Atlee. Research directions in requirements engineering. In L. Briand and A. Wolf, editors, *Future of Software Engineering 2007*. IEEE-CS Press, 20007.

[4] Esterel-Technologies. Corporate web page. http://www.esterel-technologies.com, 2004.

[5] Esterel-Technologies. SCADE Suite product description. http://www.esterel-technologies.com/v2/scadeSuiteForSafetyCriticalSoftwareDevelopment/index.html, 2004.

[6] J.M. Goldman, R.A. Schrenker, J.L. Jackson, and S.F. Whitehead. Plug-and-play in the operating room of the future. *Biomedical Instrumentation and Technology*, 39(3):194–199, 2005.

[7] W. S. Greenwell. A taxonomy of fallacies in system safety arguments. In *Proceedings of the 2006 International System Safety Conference*, 2006.

[8] R. Hawkins, T. Kelly, J. Knight, and P. Graydon. A new approach to creating clear safety arguments. In *Proceedings of Safety-critical Systems Symposium (SSS '11)*, February 2011.

[9] *IEC-61508: Functional Safety of Electrical/ Electronic/ Programmable Electronic Safety-Related Systems*. International Electrotechnical Commission (IEC), 1999.

[10] International Organization for Standardization. IEC 62304:2006, Medical device software – Software life cycle processes, 2006.

[11] E. Jee, I. Lee, and O. Sokolsky. Assurance cases in model-driven development of the pacemaker software. In *Proceedings of International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2010)*, volume 6416 of *LNCS*, pages 343–356, October 2010.

[12] E. Jee, S. Wang, J. K. Kim, J. Lee, O. Sokolsky, and I. Lee. A safety-assured development approach for real-time software. In *Proceedings of the 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 133–142, August 2010.

[13] T. Kelly. Using software architecture techniques to support the modular certification of safety-critical systems. In *Proceedings of the 11^{th} Australian Workshop on Safety-Related Programmable Systems*, August 2005.

[14] T. Kelly and S. Bates. The costs, benefits, and risks associated with pattern-based and modular safety case development. In *Proceedings of the UK MoD Equipment Safety Assurance Symposium*, October 2005.

[15] T. Kelly and J. McDermid. Safety case patterns –

reusing successful arguments. In *Proceedings of IEE Colloquium on Understanding Patterns and Their Application to System Engineering*, April 1998.

[16] T. Kelly and R. Weaver. The goal structuring notation – A safety argument notation. In *Proceedings of the Dependable Systems and Networks Workshop on Assurance Cases*, 2004.

[17] B. Kim, A. Ayoub, O. Sokolsky, P. Jones, , Y. Zhang, R. Jetley, and I. Lee. Safety-assured development of the gpca infusion pump software. In *this volume*, October 2011.

[18] A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter. Prototyping closed loop physiologic control with the medical device coordination framework. In *ICSE Companion*. ACM, 2010.

[19] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. L. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In *ICSE Companion*, pages 141–151. IEEE, 2009.

[20] MathWorks. The MathWorks Inc. corporate web page. http://www.mathworks.com, 2004.

[21] Mathworks Inc. Simulink product web site. Via the world-wide-web: http://www.mathworks.com/products/simulink.

[22] J. A. McDermid. Software safety: where's the evidence? In *SCS '01: Proceedings of the Sixth Australian workshop on Safety critical systems and software*, pages 1–6, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[23] S. P. Miller, A. C. Tribble, M. Whalen, and M. P. E. Heimdahl. Proving the shalls: Early validation of requirements through formal methods. *Int. J. Softw. Tools Technol. Transf.*, 8(4):303–319, 2006.

[24] *Requirements for Safety Related Software in Defence Equipment, Issue 2*. UK Ministry of Defence, 1997.

[25] *Safety Management Requirements for Defence Systems, Issue 2*. UK Ministry of Defence, 1996.

[26] Radio Technical Committee for Aeronautics. RTCA/DO-178B, Software Considerations in Airborne Systems and Equipment Certification, December 1992.

[27] J. Rushby. Runtime certification. In *Proceedings of the 8^{th} Workshop on Runtime Verification (RV 08)*, volume 5289 of *LNCS*, pages 21–35, April 2008.

[28] M. L. Shooman. Avionics software problem occurrence rates. In *Proceedings of the Seventh International Symposium on Software Reliability Engineering*, pages 55–64, 1996.

[29] Software Quality Research Laboratory, McMaster Univeristy. Pacemaker formal methods challenge. `http://sqrl.mcmaster.ca/pacemaker.htm`, accessed July 28, 2011.

[30] U.S. Food and Drug Administration, Center for Devices and Radiological Health. *White Paper: Infusion Pump Improvement Initiative*, April 2010.

[31] U.S. Food and Drug Administration, Center for Devices and Radiological Health. *Draft Guidance for Industry and Food and Drug Administration Staff - Mobile Medical Applications*, July 2011.

[32] M. Whalen, A. Rajan, M. P. E. Heimdahl, and S. P. Miller. Coverage metrics for requirements-based testing. In *Proceedings of International Symposium on Software Testing and Analysis*, July 2006.