HEALTH MODELLING FOR AGILITY IN SAFETY-CRITICAL SYSTEMS DEVELOPMENT

Z. R. Stephenson*, J. A. McDermid*, A. G. Ward†

*High-Integrity Systems Engineering Group, Department of Computer Science, University of York, Heslington, York YO10 5DD UK; zoe.stephenson@cs.york.ac.uk; Fax: +44 1904 432749
†Rolls-Royce plc., P.O. Box 31, Derby DE24 8BJ UK

Keywords: Agility, modelling, reuse, automation

Abstract

In the domain of software development, agile techniques are increasingly being used to improve the development process. Agile software development relies in part on rapid feedback of working software products to validate user requirements. There has been some effort to introduce agility in security-critical systems, using an explicit representation of security concerns known as an iterative security architecture. We propose a similar explicit representation of safety concerns in order to introduce agility into the safety-critical development process: the agile health model.

1 Introduction

Agile techniques have increasingly been seen as an important way for small, non-safety-critical software projects to deliver a working product to the customer while accommodating changes from rapidly advancing technology and volatile requirements. While aspects of agility have been present in many engineering disciplines for decades, it has only recently been the focus of explicit attention in the software community [8]. Agility is typically described using four comparative values [3], where "x over y", means that greater value is placed on x than y. This is not to say that y is not valued, but to state where weight or emphasis is placed:

- Individuals and interactions over processes and tool;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation, and
- Responding to change over following a plan.

Agility has generally been shown to work under "suitable circumstances", summarised [8, page *40*] as "non-safety-critical projects with volatile requirements, built by relatively small and skilled collocated teams". Research is continuing to find ways of embedding agility and agile practices into what are viewed as "unsuitable circumstances". In particular, recent work in the security domain has highlighted ways of developing security-critical systems using agile processes. We intend to learn from the security domain and apply a similar technique to allow for agility in the safety-critical domain.

2 Agility and security

The security domain is concerned with the protection of systems from malicious damage. The key concerns of the domain are captured in the Systems Security Engineering Capability Maturity Model [6]:

- Identify the organisational security risks.
- Define the security needs to counter identified risks.
- Transform the security needs into activities.
- Establish confidence and trustworthiness in correctness and effectiveness in a system.
- Determine that operational impacts due to residual security vulnerabilities in a system or its operation are tolerable (acceptable risks).
- Integrate the efforts of all engineering disciplines and specialties into a combined understanding of the trustworthiness of a system.

Several authors have examined the relationship between security and agility. Wäyrynen *et al* [7] assess eXtreme Programming from a security standpoint, taking key areas of the capability maturity model and examining their applicability to XP. They conclude that XP would need explicit representations of security requirements, proactive assessment of security risks, building of an assurance argument and verification through testing. This type of analysis is performed in the other direction by Beznosov [1], taking each XP practice and applying that to security engineering. In this work, the lack of support for incremental analysis and testing is highlighted as a key barrier to eXtreme security engineering.

To help resolve some of these issues, Chivers *et al* [2] recommend the use of an iterative security architecture that "remains true to agile principles by including only the essential features needed for the current system iteration". This provides the basis for the ongoing review of the system from a security perspective. It should be emphasised that an iterative security architecture represents the security viewpoint of the current iteration of the system as it is developed. It should not try to anticipate future needs, nor should it be used as a substitute for the overall security argument.

3 Agility and safety

The safety process is concerned with the explicit understanding of the failure behaviour of a system that is being designed. The conventional approach to safety is to take a snapshot of the design at a particular stage of development, produce a failure model that reflects the safety engineer's understanding of the design, and uses the failure model to inform subsequent design steps – typically with derived requirements. Paige *et al* [4] show how Beznosov's analysis technique applies to safety-critical systems. Their findings are of a similar nature – in the following "HIXP" means "High-Integrity eXtreme Programming", a proposed agile development process:

- What is a useful definition of increment in HIXP? This definition must satisfy the requirement of providing useful, rapid feedback to the multitude of customers in High Integrity System (HIS) engineering, as well as leading to a system that is eventually certifiable.
- What is a useful testing infrastructure that permits the different kinds of testing and simulation that occur in HIS, while still enabling rapid feedback? At which increments during HIXP can and should this infrastructure be used?
- What guidance and training must be provided to HIXP coaches in order to facilitate customer feedback and deal with the range of customers inherent in HIS engineering?
- Can the pair programming/modelling practice be used to enable independent assessment within pairs, for the purposes of leading to certification? This will require negotiation and discussion with the certification authorities, e.g., the Civil Aviation Authority.
- How will the extensions/additional (safety) practices be integrated with the typical [agile] approach?

In the previous section, a technique was outlined through which an iterative security architecture allows security to be considered as a part of a design that evolves through an agile process. In practice, a designer will include security features even without such an architecture, based on domain knowledge. This effect is also seen in safety-critical design, where a design will usually include features such as input redundancy, cross-checks, multiple lanes of control and watchdogs.

During safety analysis, the safety engineer produces a number of explicit safety models to confirm the intended behaviour of the system and systematically derive fault trees, Markov models and other safety models that demonstrate the relationship between events in the system and its safety. Some modelling techniques such as Cecilia/OCAS [9] and HiP-HOPS [5] represent the modular structure of the system. However, modular structure with safety information is not necessarily adequate for use as an agile safety model:

- Agility is based on communication. To adequately communicate safety concerns, there must be some shared understanding of the assumptions underlying the domain. This information can be made explicit in the agile safety model, improving communication so that the discussion is able to focus directly on the problem rather than continually reaffirming context.
- Similarly, it would be appropriate to capture design rationale from multiple points of view in an agile safety model. For example, there may be specific fault-accommodation schemes that have been selected because of the characteristics of the inputs being accommodated and the functions that use those inputs, and the safety engineer may view the reasoning behind the choice of scheme differently to the system engineer.
- Agility expects that development will be incremental; that is, that small changes have a localised effect and that functions can be considered in relative isolation. In safety-critical systems, the degree of integration and dependency means that this expectation will rarely be met. To identify when a change is incremental and when it has a wider effect, it must be possible to automatically trace the effect of changes in aspects such as fault accommodation, modes, maintenance procedures and shared computing resources.
- Safety analysis generally works on a representation of a complete system. Agility generally works on a representation of one area of functionality at a time, gradually improving the functionality so that it tends towards a complete design. To manage this difference, the agile safety model must allow the practitioners to include assumed details into the model to fill in the "blanks" in the design. These assumptions must be included in the automatic traceability so that later implementation in those areas can be assessed to show how it meets the existing assumptions.

To adequately support an agile process, an agile safety model must represent a "good enough" view of the engineers' assumptions about the behaviour of the model being constructed, it must be easy to maintain this model during development, and it must be possible to extract failure information from that model so that safety issues in the evolving design can be addressed as soon as they arise. Maintenance and extraction are a matter of providing adequate tool support and process infrastructure; in this paper we focus on the model content and its use within an agile safety-critical development process.

4 Model development

To develop the agile safety model, a number of sources were consulted:

- The original agile security architecture definition;
- Component-based safety modelling techniques such as Cecilia/OCAS [9] and HiP-HOPS [5];
- Recommended practice documentation ARP 4754 [10] and ARP 4761 [11];
- Expertise within the safety analysis community.

These sources help to define the model of information needed for all the parties involved in the development process to

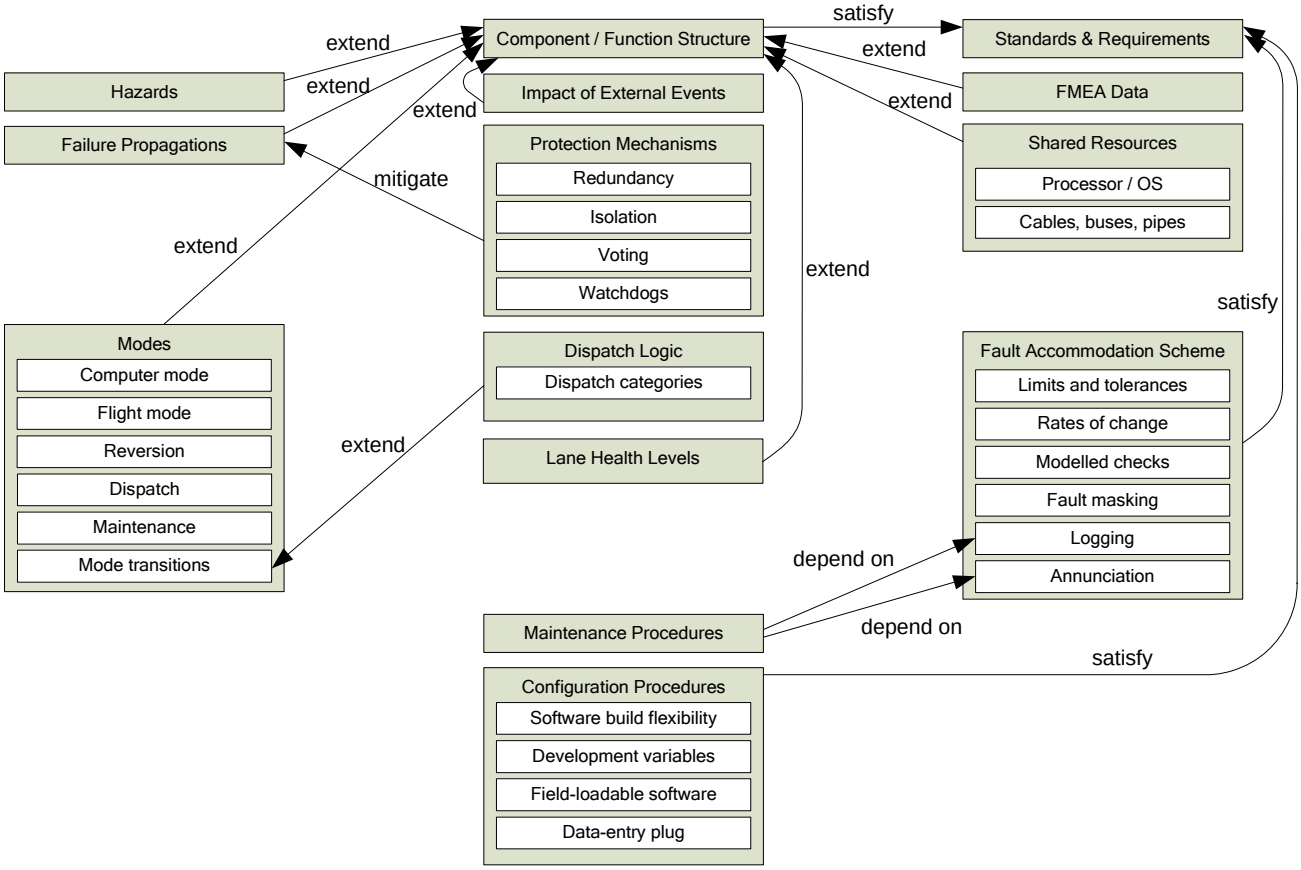

Figure 1 – Agile Health Model

understand the safety characteristics of the system. In this paper we call this model an "agile health model", although the term "agile dependability model" would also be appropriate in domains where "health model" already has a specific meaning. The overall approach to generating the agile health model can be summarised in 4 stages:

1. List processes creating and using information relating to safety behaviour. This includes safety analysis processes such as Preliminary System Safety Assessment (PSSA) and specific failure modelling techniques such as fault tree analysis; it also covers systems and software engineering processes such as architecture definition and validation. At this stage, it is relatively simple to validate that the list is complete.
2. List the information produced and used by those processes. For example, the safety analysis processes need information about events impacting on the system from outside (such as fuel line blockages) and the expected rates of occurrence. Some information is fed directly between processes, while other information is an external input or output of the development process.
3. Group information into categories. The categories come from the terminology of individual safety processes, safety-critical systems development and safety standards.
4. Layer related information together. For example, shared resources, failure propagation and protection mechanisms are all aspects of the system design. This produces the final model, shown in Figure 1.

The model has been organised so that elements higher up in the diagram are those that are produced earlier in the overall development process. Standards, requirements and the system architecture form the basis of the model. Following on from those, hazards are associated with functions, and FMEA data with components; these underpin the safety analysis processes. Design aspects come in the next layer; in particular the shared resources description is used during common-cause analysis. The next layer considers functional aspects, and makes explicit information such as fault accommodation schemes. The final categories are process-based, defining the maintenance and configuration of the system.

While there is a related discipline of agile modelling [12], the agile health model presented in this paper represents a different use of modelling. In agile modelling, agile methods are used to construct a development model from which the system implementation is produced. An agile health model is a companion model; it reflects the safety-related information and assumptions that are not necessarily captured in the development model. The agile health model could model system designs that are produced using agile modelling, as well as software implementations produced using an agile

process; it could also work with designs that are produced using more conventional processes.

5 Model usage

With the model defined, it is now possible to address the five findings introduced at the beginning of section 3.

5.1 Increments

A typical agile development process delivers functionality in predefined increments; core functionality first, with performance, interoperability and full functionality reserved for subsequent increments. The architecture is often refactored to accommodate performance and advanced functionality. For this to work in the high-integrity environment, the safety analysis process is divided into two parts – one "success-directed" part to ensure that correct, safe behaviour is being introduced, and one "failure-directed" part to ensure that failures, dependencies and common causes (of failures) are properly addressed. The agile health model coordinates these two processes:

1. In the "success-directed" part, the contribution of a function to overall system safety is continually assessed by the developers of that function using information in the agile health model. This stage may make use of automated safety analysis techniques if the cost of encoding the design in an appropriate representation and performing the analysis is acceptable for use as rapid design feedback. This all requires that the development team include appropriate safety expertise.
2. In the "failure-directed" part, a snapshot of the design is taken and automated safety analysis techniques are used to produce safety models such as minimal cut-sets. Where functionality and components are yet to be prototyped, the model is extended with explicit assumptions. The safety engineer would add justification around these models to identify the basis for acceptance as a valid representation of the final design, and this may impose constraints on any refactoring of the architecture.

This aspect of agile development is promising for safety-critical development, as it brings safety analysis into direct contact with the iterative design cycle. The use of two separate processes coordinated over a common model provides independence in the safety assessment, and it is expected that the development environment would help to facilitate independent review.

5.2 Testing infrastructure

In the development of a safety-critical system, there are many opportunities for feedback. In the engine controller domain, for example:

- The control system structure can be simulated in a modern control simulation tool, providing feedback on the control algorithms and their parameters;
- The state-based control behaviour can be modelled in a state-based representation, and scenarios can be exercised;
- Code can be downloaded onto the target platform to test conformance with operating system, memory and timing resource limits.

For these activities to benefit from the agile environment, the tools must be amenable to incremental model construction. For example, when additions are made in the models, it must be obvious when a vital property that was previously guaranteed has been violated by those additions. None of the tools that manage models or code provide this type of service, but some research has been undertaken recently on the topic of explicit assumptions for Simulink and Stateflow components [13]. This type of technology would be vital for agile development of safety-critical systems. The explicit health model may help in framing the problem of recording appropriate assumptions for the tools to check.

5.3 Guidance and training

The agile health model provides a powerful mechanism for the documentation of assumptions and communication among different disciplines. This helps to solve the problem of managing different customers with different needs, by ensuring that everything is explicitly documented. In addition to this, therefore, training and guidance must be carefully tailored:

- Guidance should be given on the parts of the model appropriate for each stakeholder at each stage of the process. If possible, this should be reinforced with model viewpoints and domain-specific model presentation.
- An agile process typically uses an on-site customer representative to act as the day-to-day requirements stakeholder. This may be a member of the customer organisation, or it could be a designated proxy. The FAA system of designated engineering representatives (DERs) and EASA's compliance verification engineers (CVEs) operate in a similar manner, indicating that the practice is broadly compatible with the domain. Training should include the use of the agile health model to identify situations in which a customer representative can be effective.
- Using a radically different approach such as the agile health model means a serious change in culture. Training should be focused on managing this culture change and mitigating the risk of failure to change.

5.4 Independence

With the increase in automation for safety analysis – especially with newer techniques such as AltaRica [14] and HiP-HOPS [5] – there is a concern that the role of the safety analyst could be subsumed by the system and software engineers and the automated tools. In this situation, there is a loss of independence; the same stakeholders are responsible for both creating the system and demonstrating that it is safe. There may be many more opportunities to miss important dependencies and assumptions.

Independence is already found in several agile methodologies. In extreme programming, for example, the pair programming practice requires independent agreement on the production of correct code. It was suggested in the original assessment [4] that a form of "pair analysis" could be used for high-integrity systems development. With an agile health model in place, this analysis can operate in two different ways:

- Safety analysts and development engineers work in front of a single workstation. The systems engineer or software engineer makes design choices and updates the design model, which is reflected in the agile health model. Automated consistency checking and keen observation by the safety analyst indicate where changes to the design model have an effect on the current safety analysis. When this occurs, the safety analyst takes over, switches to the safety view, and assesses the impact of the changes. This could lead to identification of derived (safety) requirements, to be fed into the current development and also collected together for formal review. Having created new requirements or advised on undesirable consequences, the parties switch places and development continues.
- Safety analysts and development engineers work at separate workstations viewing the same agile health model. As the system or software engineer makes changes to the model, automated consistency checks and observation identify possible effects on the safety analysis. The safety analyst takes a snapshot of the design and subjects it to appropriate manual and automatic analyses to determine the effect of the changes. These are fed back in real time to the designer as potential derived (safety) requirements and observations, as noted in the previous paragraph. The parties may also confer using instant message technology to query changes and to resolve design issues; the logs of these conversations help to justify the eventual result of the changes.

Both processes require a significant investment in culture change and tool support to be effective. While the arrangement does provide a measure of independence, certification representatives would have to be consulted to determine whether the independence of process sufficiently offsets the close interaction of the models.

5.5 Integration

The Agile manifesto outlines four general trends that govern agility [3], presented in the introduction to this paper. These are phrased as values; one facet of development is valued over another because of its positive effect on the delivery of software projects. For each of these four statements, the agile health model also provides a way for safety-critical development to fulfil the requirements of agility:

- *Individuals and interactions over processes and tools.*

The agile health model provides specific information required by different individuals. By consolidating the various types of information needed for safety-critical systems development in a single model, it facilitates communication between those individuals. The model also plays a role in shaping the domain-specific language of each participant, improving their overall communication bandwidth.

- *Working software over comprehensive documentation.*

Rapid updates to the derived safety requirements help the software engineer to ensure that the functionality deployed incrementally in the software is representative of the final state of the system. The key difference with the majority of agile development is that the software is immediately validated in the customer environment whenever a function is delivered; in the safety-critical domain a suitable proxy environment would be needed (e.g. an engine simulator instead of an actual installed engine).

- *Customer collaboration over contract negotiation.*

The agile health model does not directly influence this particular value. However, a case can be made for its ability to facilitate discussion with the customer, especially if used in conjunction with a simulation environment for validation.

- *Responding to change over following a plan.*

For certification, there must always be an overall development plan; in agile methods, a distinction is made between the overall programme of phased function delivery and the day-to-day iterative, incremental development process. The scenarios outlined in section 5.4 identify exactly how the agile health model can be used in responding to change – fulfilling the process in the small – while also supporting the certification requirements by feeding separate analysis and validation processes.

6 Conclusion

We have investigated the relationship between safety and agility by comparing it with existing successes in the area of security and agility. We have shown that while there are some issues that must be addressed if an agile safety process is to be attempted, none of those issues will completely prevent agility and safety from working together. The particular issues that will drive further work in this area are as follows:

- There must be consultation with certification authorities to demonstrate that the process provides at least the same levels of confidence and independence as existing development processes.
- A suitable collaborative working, simulation and validation environment must be developed to facilitate the process within a complex safety-critical domain.
- A system of designated representatives must be devised to allow the consideration of needs from disparate organisations. For example, an engine controller development may involve computing hardware suppliers,

sensor and actuator manufacturers, thrust reverser manufacturers, performance engineers, mechanical engineers, airframe manufacturers, certification authorities and airlines. A single representative is unlikely to suffice.

- The agile methodology is a significant change to the established organisational culture of a safety-critical systems vendor. There are likely to be many specific issues that must be addressed when considering agility in the context of a safety-critical organisation.

The bottom line for the adoption of an agile methodology is to demonstrate that there will be a return on investment. In the world of safety-critical projects, where development cycles last well over a year, some thought must be given to the way in which the process is demonstrated, with confidence, to provide significant savings.

Acknowledgements

The work reported in this paper is part of a research programme at the Rolls-Royce UTC in Systems and Software Engineering, University of York. We are grateful to Rolls-Royce plc. for their support.

References

[1] K. Beznosov. "Extreme Security Engineering: On Employing XP Practices to Achieve "Good Enough Security" without Defining It", *First ACM Workshop on Business Driven Security Engineering (BizSec)*, Fairfax VA (2003)

[2] H. Chivers, R.F. Paige, and X. Ge. "Agile Security via an Incremental Security Architecture", *LNCS* **3556**, (June 2005)

[3] R. C. Martin. "Agile Software Development: Principles, Patterns and Practices", *Prentice Hall* (October 2002)

[4] R. F. Paige, H. Chivers, J. A. McDermid, Z R Stephenson. "High-Integrity Extreme Programming", *Symposium on Applied Computing*, Santa Fe (March 2005)

[5] Y. Papadopoulos, M. Maruhn. "Model-based Automated Synthesis of Fault Trees from Matlab-Simulink Models", *International Conference on Dependable Systems and Networks*, Gothenburg (July 2001)

[6] SSE-CMM. "Systems Security Engineering Capability Maturity Model, Model Description Document Version 3.0", http://www.sse-cmm.org/docs/ssecmmv3final.pdf, (acc. October 2005)

[7] J. Wäyrynen, M. Bodén, G. Boström. "Security Engineering and eXtreme Programming: An Impossible Marriage?", *LNCS* **3134**, pp117-128 (2004)

[8] L. Williams, A. Cockburn. "Agile Software Development: It's about Feedback and Change", *IEEE Computer*, **36(6)**, pp 39-43 (June 2003)

[9] A. Bozzano, A. Villafiorita, O. Åkerlund *et al*. "ESACS: An Integrated Methodology for Design and Safety Analysis of Complex Systems", Proceedings of ESREL 2003 (June 2003)

[10] SAE. "Certification Considerations for Highly Integrated or Complex Aircraft Systems", ARP4754 (November 1996)

[11] SAE. "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment", ARP4761 (December 1996)

[12] S. W. Ambler, R. Jeffries. "Agile Modeling: Effective Practices for Extreme Programming and the Unified Process", Wiley (March 2002)

[13] A. Galloway, I. Toyn, F. Iwu, J. A. McDermid. "The Simulink/Stateflow Analyser", FAA and Embry-Riddle Aeronautical University (ERAU) Software Tools Workshop, Florida, USA (May 2004)

[14] P. Bieber, C. Bougnol, C. Castel *et al*. "Safety Assessment with AltaRica – Lessons Learnt Based on Two Aircraft System Studies", Proceedings of the 18th IFIP World Computer Congress Topical Day on New Methods for Avionics Certification (August 2004)