

Heterogeneous Modelling of Evolution for Socio-technical Systems

Stuart Anderson

Massimo Felici

LFCS, School of Informatics, The University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
tel. +44-131-650 {5191, 5899}, fax. +44-131-6677209
soa@inf.ed.ac.uk, massimo.felici@ed.ac.uk

Abstract

This paper is concerned with the evolution of socio-technical systems. Although it is possible to identify an evolutionary space for socio-technical systems, the methodologies that address the evolution of socio-technical systems are still patchy. Moreover, it is still challenging to address multidisciplinary in modelling methodologies. This paper addresses the heterogeneous modelling of the evolution of socio-technical systems. The analysis of a case study highlights how the combination of diverse models allows the characterisation of requirements evolution.

1 On the Evolution of Socio-technical Systems

Modelling methodologies and languages advocate different design strategies. Although these strategies support different aspects of software development, they originate in a common *Systems Approach*¹ to solving complex problems and managing complex systems. Despite the fact that the system approach highlights that any system indeed consists of heterogeneous parts, the development (generally, the entire life cycle) of (socio-)technical systems is mainly technology driven. It is often the case that the technology development is explicitly planned, whereas the social aspects are implicitly or, worst, poorly considered. The system approach therefore emphasises a comprehensive viewpoint highlighting the interaction among heterogeneous parts.

¹“Practitioners and proponents embrace a holistic vision. They focus on the interconnections among subsystems and components, taking special note of the interfaces among various parts. What is significant is that system builders include heterogeneous components, such as mechanical, electrical, and organizational parts, in a single system. Organizational parts might be managerial structures, such as a military command, or political entities, such as a government bureau. Organizational components not only interact with technical ones but often reflect their characteristics. For instance, a management organization for presiding over the development of an intercontinental missile system might be divided into divisions that mirror the parts of the missile being designed.”, [12], Introduction, p. 3.

Heterogeneous engineering² stresses a holistic viewpoint that allows us to understand the underlying mechanisms of evolution of socio-technical systems. It is possible to identify a taxonomy of evolution, as a conceptual framework for the analysis of the evolution of socio-technical systems [7]. The evolutionary framework extends over two dimensions, from *Evolution in Design* to *Evolution in Use* and from *Hard Evolution* to *Soft Evolution*, that define an evolutionary space for socio-technical systems [7]. Evolution in Design and Evolution in Use capture the system life cycle perspective. Evolution in design identifies technological evolution mainly due to designers and engineers and driven by technology innovations and financial constraints. With respect to technical systems, evolution in use identifies the social evolution due to social learning. Social learning involves the process of fitting technological artefacts into existing socio-technical systems. Whereas, Hard Evolution and Soft Evolution capture different system viewpoints in which evolution takes place. Each viewpoint identifies different stakeholders. Hard³ evolution identifies the evolution of technological artefacts (e.g., hardware and software). Whereas, soft⁴ evolution identifies the social evolution (e.g.,

²“People had to be engineered, too - persuaded to suspend their doubts, induced to provide resources, trained and motivated to play their parts in a production process unprecedented in its demands. Successfully inventing the technology, turned out to be heterogeneous engineering, the engineering of the social as well as the physical world.”, [14], p. 28.

³“Hard systems viewpoints are basically those held by designers and engineers who are trying to create systems to meet an understood need in an effective and economic manner. Those in the soft camp caricature the approach as *head-down*, concerned with optimization, obsessed with quantitative metrics and highly pragmatic. So much so, in fact, that the term *system thinking* has been purloined by the soft camp as though they alone thought! The soft camp use the term *engineer’s philosophy*, not too endearingly, to describe the hard approach, in which the requirement is stated by a customer and the engineer satisfies the requirement without question.”, [11], p. 6.

⁴“Soft systems viewpoints are those held by behavioural, management, social anthropology, social psychology and other science students concerned with observing the living world, and in particular the human world. Human activity systems (HASs) are *messy*, in that they do not exhibit a clear need or purpose - if they can be said to exhibit purpose at all. Indeed,

organisational evolution) with respect to these technological artefacts. Soft evolution therefore captures the evolution of stakeholder perception of technical systems. Figure 1 shows the evolutionary space for socio-technical systems [7].

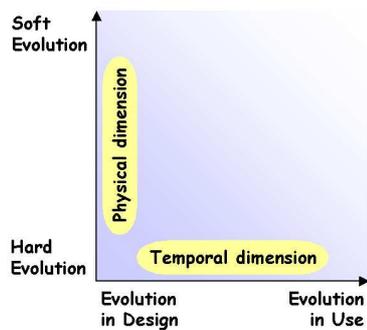


Figure 1. The evolutionary space.

The identification of a broad spectrum of evolutions in socio-technical systems points out strong contingencies between system evolution and dependability [7]. Although the evolution of socio-technical systems is a desirable feature. On the one hand evolution allows the capture of emerging social needs. On the other hand evolution allows the mitigation or the propagation of socio-technical failures. Socio-technical failures are often failures to evolve. This paper argues the better our understanding of socio-technical evolution, the better system dependability. Unfortunately, current methodologies still have severe limitations with respect to the evolution of socio-technical systems. This paper highlights heterogeneous modelling of the evolution of socio-technical systems. This paper is structured as follows. Section 2 describes a case study. Section 3 reviews related modelling in requirements engineering. The review highlights the modelling limitations with respect to evolution and multidisciplinary. Section 4 describes how the combination of diverse models allows the characterisation and analysis of requirements evolution for socio-technical systems. Section 5 draws the conclusions.

2 A Case Study in Healthcare

This section describes a socio-technical system drawn from the healthcare context [1]. The case study consists of a patient information management system, which presents design and deployment issues due to lack of support for system evolution. Moreover, the case study highlights how the introduction of a new system (or the evolution from an

so complex is the real world of people that the idea of driving towards optimal solutions may be a non-starter - perhaps we should see if we can simply understand and concern ourselves with improving the situation.”, [11], p. 7.

old system to a new one) may affect work practice. Patient information management systems are enabling technology at the heart of healthcare strategy aimed at generating new knowledge to guide a variety of healthcare processes (e.g., clinical decision making, resource allocation and clinical governance). At the heart of most patient information systems, users are asked to take classification decisions about a variety of different situations. The quality of the classification and of the classification activity are key to the quality of the knowledge being generated by the system.

The investigation of the introduction of a new patient information management system within a large primary healthcare highlights design and deployment issues due to lack of support for system evolution [1]. The system has been introduced with the aim of integrating and standardising patient administration practices within numerous different healthcare service units. The new system aims to provide various benefits (e.g., higher quality and uniform data about healthcare activities). Thereby, it enables to make accurate resource allocation decisions, together with timely, accessible and detailed clinical data to inform clinical and nursing users care of patients, and to support evidence-based practice and research activity. Unfortunately, the new system represents an instance of how unsupported (socio-)technical evolution may affect work practice and give rise to dependability hazards.

One critical part of the new system has been the contact purpose menu. Previously, each of the healthcare services relied upon its own locally meaningful classifications of contact purpose. The new system introduced a standard contact purpose classification and built this into the system’s user interface menus. During configuration prior to roll-out, the contact purpose menu was originated from previous existing systems. The new integrated system was intended to replace numerous separate systems developed and used in different services. Many of the contact options had been placed in the contact purpose menu, which only allowed one purpose to be selected. By contrast, previous systems had allowed users to select more than one option, which they had found useful. System users made unfavourable comparisons with the functionalities of the previous systems, which provided little constraints in work practice. Unfortunately, users struggled to make sense of the new menu categories. Despite the issues with the new contact purpose menu, system users rarely selected the “unspecified” option. Although this option aims to take into account situations when users are unable to match the situation at hand with the newly standardised categories, system users preferred to select from several seemingly similar options the category that seems nearest.

The initial development assumption that the contact purpose menu was unproblematic failed to capture user expectations. Moreover, the contact purpose menu appears to

need maintenance for the system to be useful. The result is that users were unhappy how the system poorly reflected their work practices. Moreover, management is concerned that the quality of data captured is too poor to support its strategic goals.

3 Requirements (Evolution) Modelling

Modelling has attracted a substantial effort from research and practice in requirements engineering. In spite of quality and effective development processes, many faults in software systems are traced back to high level requirements. This has motivated the increasing use of modelling in requirements engineering. The overall goal of modelling is mainly to support development activities (e.g., testing) and to reduce the gap between system requirements and design. Although this gap is one of the sources of requirements changes, research on requirements evolution clearly points out other origins of changes [17].

Modelling incorporates design concepts and formalities into requirements specifications. This enhances our ability to assess requirements correctness and completeness. For instance, the *Software Cost Reduction (SCR)* consists of a set of techniques for designing software systems [10]. The SCR techniques use formal design techniques, like tabular notation and information hiding, in order to specify and verify requirements. According to information hiding principles, separate system modules have to implement those system features that are likely to change. Although module decomposition reduces the cost of software development and maintenance, it provides limited mechanisms to deal with requests of requirements changes [20], hence requirements evolution.

Intent Specifications [13] further support the analysis and design of evolving systems. Intent Specifications extend over three dimensions. The vertical dimension consists of various hierarchical levels that represent the intent. Along the horizontal dimension, Intent Specifications decompose the whole system in heterogeneous parts: Environment, Operator, System and Components. The third dimension, *Refinement*, further breaks down both the Intent and Decomposition dimensions into details. Each level (along the vertical dimension) provides rationale (i.e., the intent or “why”) about the level below. Each level has mappings that relate the appropriate parts to the levels above and below it. These mappings provide traceability of high-level system requirements and constraints down to physical representation level (or code) and vice versa. In general, the mappings between Intent levels are many-to-many relationships. In accordance with the notion of *semantic coupling*, Intent Specifications support strategies to reduce the cascade effect of changes [19]. Although these strategies support the analysis and design of evolving systems, they provide limited support

to understand the evolution of high-level system requirements⁵. The better our understanding of requirements evolution, the more effective design strategies.

Although these models give different representations, the Systems Approach represents a common origin for all of them. A common aspect is that models identify the relations between the different system parts. On one hand these relations constrain the system behaviour (e.g., by defining environmental dependencies). On the other hand they are very important for system management and design. Looking at requirements from a heterogeneous engineering [4] perspective further explains the complex interaction between system (specification) and environment. The most common understanding in requirements engineering considers requirements as goals to be discovered and (design) solutions as separate technical elements. Hence, requirements engineering is reduced to be an activity where technical solutions are documented for given goals or problems. Contrastingly according to heterogeneous engineering, requirements specify mappings between problem and solution spaces [2]. Both spaces are socially constructed and negotiated through sequences of mappings between solution spaces and problem spaces. These mappings define the *Functional Ecology* model, which implies that requirements emerge as a set of consecutive solution spaces justified by a problem space of concerns to stakeholders [3]. Figure 2 shows a representation of the Functional Ecology model⁶ [3].

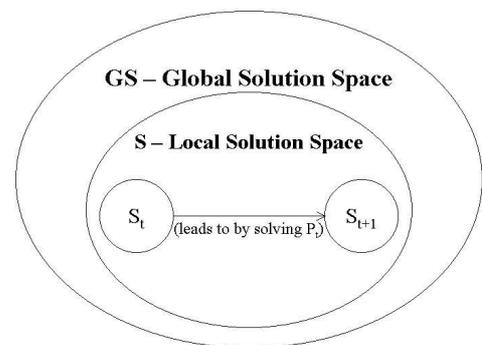


Figure 2. Functional Ecology model.

⁵Leveson reports the problem caused by “Reversals” in TCAS (Traffic Alert and Collision Avoidance System) [13]: “About four years later the original TCAS specification was written, experts discovered that it did not adequately cover requirements involving the case where the pilot of an intruder aircraft does not follow his or her TCAS advisory and thus TCAS must change the advisory to its own pilot. This change in basic requirements caused extensive changes in the TCAS design, some of which introduced additional subtle problems and errors that took years to discover and rectify.”

⁶Local Solution Space (S), Global Solution Space (GS), Current Solution Space (S_t), Proposed System Problem Space (P_t) and Proposed Solution Space (S_{t+1}).

This model defines evolutionary cycles of iterations in the form: *solution* → *problem* → *solution*. It implies that requirements engineering processes consist of solutions searching for problems, rather than the other way around (that is, problems searching for solutions).

One important difference between the solution space transformation and other requirements engineering models is its emphasis on stakeholder interactions. This heterogeneous account of requirements is convenient to capture requirements evolution. Despite the existence of many modelling methodologies and languages, very few directly address requirements evolution. The PROTEUS Project [17] proposes a formal *goal-structure framework* for representing and reasoning about requirements changes. The PROTEUS goal-structure framework represents requirements and their interactions with respect to requirements changes. Most importantly, the framework captures the interactions between system and the environment in which it operates. These interactions (i.e., between requirements, and between system and environment) form a basis for sensitivity and impact analyses. In order to be effective, these analyses have to take into account information about requirements volatility and rationale for design decisions. The combination of sensitivity and impact then provides a measure of risk [17].

Another approach is directly to model requirements evolution. One way of modelling requirements evolution is by ordered sequences of requirements releases. Requirements therefore evolve by changes from one release to the successive one. The formalisation of requirements releases provides a logical framework for modelling and reasoning about requirements evolution [22]. There is little coherence among the different models that address requirements evolution. On one hand this is due to the complexity of requirements evolution. On the other hand requirements evolution has received little attention. Although the evolutionary models capture diverse aspects, it is possible to identify stakeholder interactions as an important driver of evolution.

4 Heterogeneous Modelling of Evolution

Heterogeneous engineering stresses a holistic viewpoint that allows us to understand the underlying mechanisms of evolution of socio-technical systems. Requirements, as mappings between socio-technical solutions and problems, represent an account of the history of socio-technical issues arising and being solved within industrial settings [2, 3]. The formal extension of solution space transformation (i.e., the Functional Ecology model) provides a framework to model and capture requirements evolution [8]. The basic idea is to provide a formal representation of solutions and problems. The aim of a formal representation is twofold. On the one hand the formalisation of solutions and problems

supports model-driven development. On the other hand it allows us formally to capture the solution space transformation, hence requirements evolution. The formalisation represents solutions and problems in terms of modal logic [9]. Intuitively, a solution space is just a collection of solutions, which represent the organisational knowledge acquired by the social shaping⁷ of technical systems. Solutions therefore are *accessible possibilities* or *possible worlds* in solution spaces available in the production environment. The resulting framework is sufficient to interpret requirements changes [8]. Hence, it is possible to define requirements evolution in terms of sequential solution space transformations. Requirements evolution consists of the requirements specification evolution and the requirements changes evolution. Hence, requirements evolution is a co-evolutionary process [8]. The formally augmented solution space transformation captures evolutionary requirements dependencies [8]. It is important to capture these dependencies in order further to understand requirements evolution. The modelling of evolutionary dependency highlights that the formal extension of the solution space transformation enables the gathering of evolutionary information at different abstraction levels [8]. Hence, the formally extended solution space transformation allows the reasoning of ripple effects of requirements changes at different abstraction levels.

The underlying heterogeneity of the functional ecology solution space transformation supports stakeholders during the requirements specification activity. Although understanding stakeholder interactions highlights requirements evolution, poor understanding of the mechanisms of requirements evolution affects stakeholder interactions. This often results in poor requirements baselines that affect system production as well as system features. *Activity Theory* [18] describes how social interactions influence human cognition. Figure 3 shows the whole structure of human activity [6], which extends the structure of mediated act [18].

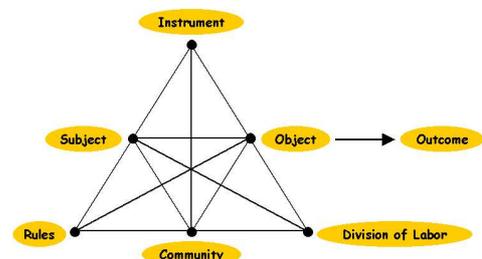


Figure 3. The structure of human activity.

⁷The mechanisms underlying the social design and implementation of technology systems are referred to as the *Social Shaping of Technology* (SST) [15, 21].

The model allows the analysis of a multitude of relations within the triangular structure of activity. However, the important aspect is to grasp human activity as a whole. The systemic model allows us to take into account all the connections as a whole, as oppose to separate connections. On the one hand activity theory allows the interpretation of productive activities. Therefore, it is possible to instantiate the activity model in the context of the healthcare cases study. On the other hand activity theory allows the analysis of system design activities (e.g., requirements phase). Hence, it is also possible to instantiate the activity model with respect to the functional ecology model. For instance, each subject (e.g., a software engineer) negotiates within the community (e.g., the design context) and with the division of labour (e.g., business stakeholders) the objectives (e.g., desired solutions) according to specific rules (e.g., change management process) using specific instruments (e.g., current solutions). Thus, the outcome consists of the requirements, as mappings between solution and problem spaces.

In the healthcare case study, there were unclear and conflicting objectives that were negotiated using vague rules. This resulted in rough negotiation of the system requirements [1]: *“During the early phases of roll-out, users in one particular, large specialist service asked for several additions to the menu, as they felt that the options supplied failed to reflect their teams activities. The development team granted this request because the service concerned was large and vocal (hence powerful), and had not previously kept good records, in the expectation that this would help achieve user buy-in and ensure compliance in use of the system.... Other smaller, less prestigious teams, unhappy with the contact purpose menu options, had also discussed proposed additions to reflect their activities. These had been sent to the development team, but had been lost without trace.”*

Although the requirements process in the case study was vaguely specified, it would be still possible to explain (capture) the requirements evolution in terms of solution space transformation. The delivery of the initial solution gave rise to user feedback reporting specific problems. Thus, the proposed future solution would address (some of) these problems. However, this always requires further commitments in order to achieve the future objectives. Although the structure of human activity allows us to capture the diverse interactions that contribute to the outcome, it provides limited support to characterise the evolution of these interactions. In particular, it fails to capture the new *configuration* due to changed objectives (or other basic model entities). Therefore, the combination of the activity model with the functional ecology model captures how requirements evolve due to social interactions.

It is also useful to characterise socio-technical solutions. *Distributed Cognition* [16] focuses just on the interaction

between representational resources, which can be located within human mind as well as external artefacts. A holistic view of socio-technical systems may explain the nature of socio-technical systems. For instance, the SHEL model [5] defines any productive process as performed by a combination of Hardware, Software and Liveware resources embedded in a given Environment. Although the SHEL model supports a systemic view, it provides limited support to capture how resource interactions evolve, may be, due to localised changes (e.g., software changes). The healthcare case study clearly highlights the interaction of heterogeneous resources [1]: *“System users made unfavourable comparison with the affordances of the previous systems, which did not represent practice in quite such a reduced way.”*

5 Conclusions

This paper is concerned with the evolution of socio-technical systems. Although it is possible to identify an evolutionary space for socio-technical systems, the methodologies that address the evolution of socio-technical systems are still patchy. Moreover, it is still challenging to address multidisciplinary in modelling methodologies. This paper addresses the heterogeneous modelling of the evolution of socio-technical systems. This paper takes a multidisciplinary account of socio-technical systems. The analysis of a case study highlights how the combination of diverse models allows the characterisation of requirements evolution. Moreover, it is possible further to analyse requirements evolution with respect to activity theory and distributed cognition.

Heterogeneous engineering stresses a different role for requirements. The shift from the paradigm of problems searching for solutions to the one of solutions searching for problems points out a new role for requirements with respect to (design) solutions and problems. Heterogeneous engineering therefore points out that requirements link (design) solutions and given problems observed (e.g., by coding, testing, usage, etc.) in the system implementation. The modelling of requirements evolution highlights how requirements evolve due to the social shaping of technical systems. The new role of requirements, with respect to solutions and problems, points out new scenarios of use for requirements engineering tools. Requirements engineering tools should also support the mapping of solutions to observed problems. That is, requirements engineering tools should support the analysis of observed problems in order to narrow the solution space. Future work should further address the use and integration of heterogeneous models of socio-technical systems. This will increase the current understanding of the evolution of socio-technical systems. Moreover, it will allow the characterisation of the strong contingencies between evolution and dependability.

Acknowledgements. We thank the colleagues who have been working on the healthcare case study, in particular, Gillian Hardstone, Rob Procter and Robin Williams. This work has been supported by the UK EPSRC DIRC⁸ project, Interdisciplinary Research Collaboration in Dependability of Computer-Based Systems, grant GR/N13999.

References

- [1] Stuart Anderson, Gillian Hardstone, Rob Procter, and Robin Williams. Supporting the evolution of organisational information systems. In M. Ackerman, T. Erickson, and C. Helverson, editors, *Evolving Information Artefacts*. Kluwer, to be published.
- [2] Mark Bergman, John Leslie King, and Kalle Lyytinen. Large-scale requirements analysis as heterogeneous engineering. *Social Thinking - Software Practice*, pages 357–386, 2002.
- [3] Mark Bergman, John Leslie King, and Kalle Lyytinen. Large-scale requirements analysis revisited: The need for understanding the political ecology of requirements engineering. *Requirements Engineering*, 7:152–171, 2002.
- [4] Wiebe E. Bijker, Thomas P. Hughes, and Trevor J. Pinch, editors. *The Social Construction of Technology Systems: New Directions in the Sociology and History of Technology*. The MIT Press, 1989.
- [5] E. Edwards. Man and machine: Systems for safety. In *Proceedings of British Airline Pilots Associations Technical Symposium*, pages 21–36, London, 1972. British Airline Pilots Associations.
- [6] Y. Engeström. Learning by expanding: An activity-theoretical approach to developmental research, 1987.
- [7] Massimo Felici. Taxonomy of evolution and dependability. In *Proceedings of the Second International Workshop on Unanticipated Software Evolution, USE 2003*, pages 95–104, Warsaw, Poland, April 2003.
- [8] Massimo Felici. Observational models of requirements evolution, 2004.
- [9] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer Academic Publishers, 1998.
- [10] Constance L. Heitmeyer, James Kirby, Bruce G. Labaw, and Ramesh Bharadwaj. SCR*: A toolset for specifying and analyzing software requirements. In Alan J. Hu and Moshe Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification, CAV '98*, volume 1427 of LNCS, pages 526–531, 1998.
- [11] Derek K. Hitchins. *Putting Systems to Work*. John Wiley & Sons, 1992.
- [12] Agatha C. Hughes and Thomas P. Hughes, editors. *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and After*. The MIT Press, 2000.
- [13] Nancy G. Leveson. Intent specifications: An approach to building human-centered specifications. *IEEE Transactions on Software Engineering*, 26(1):15–35, January 2000.
- [14] Donald A. MacKenzie. *Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance*. The MIT Press, 1990.
- [15] Donald A. MacKenzie and Judy Wajcman, editors. *The Social Shaping of Technology*. Open University Press, 2nd edition, 1999.
- [16] Donald A. Norman. *The Invisible Computer*. The MIT Press Cambridge, Massachusetts, 1998.
- [17] PROTEUS. Meeting the challenge of changing requirements. Deliverable 1.3, Centre for Software Reliability, University of Newcastle upon Tyne, June 1996.
- [18] L. Vygotsky. *Mind in Society*. Harvard University Press, 1978.
- [19] Kathryn Anne Weiss, Elwin C. Ong, and Nancy G. Leveson. Reusable specification components for model-driven development. In *Proceedings of the International Conference on System Engineering, IN-COSE 2003*, 2003.
- [20] Virginie Wiels and Steve Easterbrook. Formal modeling of space shuttle software change requests using SCR. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering, RE'99*, pages 114–122. IEEE Computer Society, 1999.
- [21] Robin Williams and David Edge. The social shaping of technology. *Research Policy*, 25(6):865–899, 1996.
- [22] Didar Zowghi and Ray Offen. A logical framework for modeling and reasoning about the evolution of requirements. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 247–257, Annapolis, Maryland, USA, January 1997. IEEE Computer Society Press.

⁸<http://www.dirc.org.uk/>