Association for Information Systems
AIS Electronic Library (AISeL)

MCIS 2011 Proceedings

Mediterranean Conference on Information Systems (MCIS)

2011

MUTABILITY MATTERS: BASELINING THE CONSEQUENCES OF DESIGN

Jonas Sjöström
Uppsala University, jonas.sjostrom@im.uu.se

Pär J. Ågerfalk
Uppsala University, Sweden, par.agerfalk@im.uu.se

Ruth A. Lochan
Uppsala University, ruth.lochan@im.uu.se

MUTABILITY MATTERS: BASELINING THE CONSEQUENCES OF DESIGN

Sjöström, Jonas, Uppsala University, Sweden, jonas.sjostrom@im.uu.se

Ågerfalk, Pär J., Uppsala University, Sweden, par.agerfalk@im.uu.se

Lochan, A. Ruth, Uppsala University, Sweden, ruth.lochan@im.uu.se

Abstract

Artefact mutability has been proposed as an important component of design theory in information systems. Although initial work on establishing a solid foundation for discussing mutability has been reported, conceptual as well as practical uncertainty still prevails. This paper draws on empirical work in a design science research project in the health sector to explore the notion of mutability and provides a novel conceptualization based on four different types of mutability. The study shows that in order to embrace mutability, IS researchers need to establish a sound philosophy of mutability and be open to incorporate theory and best practices from neighbouring fields, perhaps primarily from software engineering.

Keywords: Mutability, design science research, design theory.

1 INTRODUCTION

Artefact *mutability* has been proposed as an important component of design theory in Information Systems (IS). It is highlighted by Gregor & Jones (2007) that any IS design theory should encompas how the artefact, within reason, can be adapted to a continuously changing use context and evolving technological infrastructures. When proposing design theories for IT-reliant work, mutability is at the core. As suggested by Orlikowski & Iacono (2001), we need to conceive of the emergent properties of both technology and the social world, while also recognizing the complex interplay between them (e.g. Orlikowski, 1991; El Sawy, 2003; Orlikowski, 2007; Orlikowski & Scott, 2008; Leonardi & Barley, 2008).

While Orlikowski & Iacono (2001) call for theorizing the IT artefact, Gregor & Iivari (2007) elaborate on the nature of artefacts v. natural/organic phenomena, and propose the notion of an natural-artificial continuum: ”We define semizoa as IS or IT artifacts that exhibit the characteristic of mutability to some degree, that is, they grow, change (or are changed), and exhibit adaptive behaviour. Further, semizoa have the potential to modify, transform or constrain their surrounding environment.” (p. 4). Instead of thinking of ‘artefacts’ being designed, Gregor & Iivari (2007) suggest that we understand IT artefacts as ‘half-living’ (semizoic) dynamic systems, and highlight that IS research has not paid enough attention to understanding design of systems from that perspective. Gregor & Iivari’s (2007) concept of *semizoa* highlights two important characteristics of design: (i) The need to relate design work to existing (and emerging) fusion of social and technological structures, and (ii) the socio-material insight that new (and existing) artefacts indeed transform the social world where they are put into play.

With respect to *technological* change, designers need to adapt technological solutions to an existing infrastructure. New technology is thus an artefact in a sense, but it is also a contribution to an evolving compound of technology. Technology thus undergoes an evolution over time rather than being deliberately designed for a well-specified purpose known in advance. Keen & Scott Morton (1978) early discussed the problems concerned with intended use versus ‘real’ use.

The aim of design is clearly to change the way of things in accordance with some design goals. However, the idea of *designing* the social world is problematic, given that IS researchers tend to emphasize the view of design as a way to induce change into a soft system (e.g. Checkland, 1981). The issues associated with changes to the social world in relation to IS development is well recognized in the IS field (e.g. Mumford & Weir, 1979; Checkland, 1981). IS development has, for example, been conceptualized as organizational change in which design activities and how they affect various stakeholders are taken into account (Lyytinen & Hirschheim, 1988; Kotter, 1996; Hayes, 2002; Sjöström, 2010). Ideas from the area of design thinking support this view of IS design as a complex socio-material activity (e.g. Krippendorff, 2006).

The notion of artefact mutability is also a core topic within the field of Software Engineering (SE), although SE terminology differs from that in IS. Developments in software design, such as modularisation (Parnas, 1972), abstract datatypes (Liskov, 1974), and object orientation (Rentsch, 1982) reflect incrementally sophisticated solutions to developing software that meet different ideals, such as *maintainability,* without compromising other aspects of software quality. Contemporary software projects typically embrace ideals of 'mutability'; for example, through the adoption of a service-oriented architecture to promote interoperability between applications, while at the same time allow for large degrees of freedom in the design of individual applications. Design patterns (Gamma et al. 1995), such as the model-view controller (MVC) pattern, are commonly used in software design to promote good architectues.

However, the concept of mutability as part of IS design research is still not well understood (Gregor & Iivari, 2007) and does not sufficiently factor in progress made in the SE field. The aim of this paper, therefore, is to conceptualize further the concept of artefact mutability, and to explore the implications of such a conceptualization for IS design research. The paper draws on (i) an ongoing IS initiative in the Swedish e-Health sector, (ii) the contemporary meta-theoretical discourse in IS design research, and (iii) selected SE theories (as kernel theories for mutability).

2 IS DESIGN RESEARCH IN THE U-CARE PROGRAMME

This paper, although primarily adopting a conceptual/analytical approach (Järvinen, 2000), draws on design experiences from the U-CARE programme at Uppsala University. The overarching goal of the U-CARE programme is to promote psychosocial health among patients struck by somatic disease and their significant others, hopefully at a lower cost to the benefit of individuals and society. In order to achieve these goals, research is conducted in close collaboration between research groups within the fields of clinical psychology, information systems and economics at Uppsala University. Initial research activities are performed within the areas of paediatric oncology, adult oncology and cardiology in close collaboration with clinicians at Uppsala Akademiska Hospital. The idea of the U-CARE programme is to bridge scientific and organizational borders in order to offer a meeting place for researchers, clinicians, and students from different disciplinary backgrounds in a way that is unique within the field of care sciences.

During 2010-2011, research studies were designed, and a software platform for psychosocial care was developed. The platform will be applied within paediatric oncology, adult oncology, and cardiology to provide care and psychological treatment to patients and significant others, starting in the second half of 2011.

The development was conducted in close collaboration between IS researchers (who also acted as system analysts and software developers) and other stakeholders in the research context. For the IS researchers, this development process was part of a design-oriented research endeavour, meaning that theory as well as practice informed the development. The theoretical basis was drawn primarily from:

- Psychosocial care in general, and in particular experiences from Internet-based psychosocial care and cognitive behavioural therapy (e.g. Kraft et al., 2009)
- Software Engineering, including design patterns (e.g. Gamma et al., 1995)
- Interaction Design (e.g. Preece et al., 2002)

- Information Systems, particularly actability theory (Ågerfalk, 2003; Sjöström, 2010)

The multi-disciplinary approach in U-CARE resonates well with the ideals of rigorous evaluation in IS design research as put forth by Hevner et al. (2004). Contributions from psychology and economics, disciplines with a strong quantitative evaluation tradition, ingrain our research with evaluation methods that will be applied to evaluate (i) the clinical efficacy of psychosocial care delivered via the platform, and (ii) the health economics impact of delivering online psychosocial care. Evaluation of these aspects will be conducted through randomized controlled experiments that allow for statistically significant comparisons between treatment groups and control groups, with stratification for various variables.

Several types of evaluation of the platform are in progress. The DeLone and McLean (1992; 2003) model of IS success serves as a cornerstone for evaluation, including *organisational* impact, system *functionality* and *usability*, supplemented with input from the other disciplines, such as psychology (*individual* impact) and economics (*societal* impact). In doing this, we apply a combination of qualitative and quantitative approaches.

It is beyond the scope of this paper to go into details of the evaluation strategy. Instead, we focus on a few issues of evaluation work. A large subset of the evaluation work still remains to be done through controlled experiments. Nonetheless, the design efforts so far have rendered considerable experience worthy of reporting. The development followed a selection of agile development principles (ref). Development sprints lasted for 2–3 weeks, followed by sprint reviews where the 'customers' – various stakeholders (mainly researchers from other disciplines) – were exposed to the most current version of the platform. These review meetings rendered feedback and governed the continued development efforts. The feedback was documented and field notes were taken. E-mail correspondence has also been kept to provide additional corpus.

The stakeholder-centric (Sjöström, 2010) and iterative approach promotes a focus on value creation – a continuous assessment of the platform as a means to contribute to the overarching goals of the U-CARE programme. This allows for ideas and design principles to be gradually refined. A stakeholder-centric approach that combines Hevnerian evaluation ideals with action design research (Sein et al. (2011) thus forms the core of the research design. Essentially, in keeping with Sjöström & Ågerfalk (2009) we embrace rigorous evaluation methods (Hevner et al., 2004) to understand better mutability in design theory (Gregor & Jones, 2007; Gregor & Iivari, 2007).

3 EXPLORING FOUR TYPES OF MUTABILITY

The notions of design and use, and their relation, has been debated for a long time in IS and HCI research. In our conceptualization of mutability, we distinguish between the *process* of design and the *product* of that design process (Walls et al. 1992; Sjöström & Ågerfalk, 2009). We also conceive of the use of an artefact by emphasizing the process where it is used and the characteristics of the artefact as such. We thus identify four areas of relevance for mutability:

1. Mutability-in-Use (Process View)
2. Mutability-in-Use (Product View)
3. Mutability-in-Design (Product View)
4. Mutability-in-Design (Process View)

The four types of mutability are presented in the following subsections. Each type is presented through (i) a definition of the type of mutability, including a theoretical justification, (ii) illustrations from the U-CARE context, (iii) the implications of this type of mutability for the design process and the design product in the U-CARE programme, and (iv) a brief abstraction to raise the issue of the implications in a broader IS development context.

3.1 Mutability-in-Use (Process View)

The process view of mutability-in-use relates to the influence on the design of continuous changes in the socio-material reality of various stakeholders. It mutually serves as both baseline and ultimate consequence of a design effort. The acknowledgement of the duality of technology (Orlikowski, 1992) and the emergent properties of the social world is what causes IS researchers to reason about mutability, and, the socio-material view is a premise to theorizing the IT artefact (in keeping with Orlikowski and Iacono, 2001). Users' appropriation of technology may lead to ways of use that were neither intended nor anticipated by designers. The phenomena of 'design in use' has been discussed by IS researchers in terms of, for example, *drift* (Ciborra, 1996), *tailoring* (Trigg and Bödker, 1994), *adaptation* (Majchrak et al., 2000), *reinvention* (Rogers, 1995) and *appropriation* (e.g. De Sanctis and Poole, 1994; Orlikowski, 2000). Thus, while we recognize that the distinction between 'design' and 'use' is not trivial, still we find it important for conceptual clarity to analytically distinguish between the two.

3.2 Mutability-in-Use (Product View)

A main concern for software developers is to make sense of customers' needs. This becomes evident when one looks at the plethora of methods and techniques for 'interaction design', 'requirements engineering', 'customer collaboration', *et cetera*. Different design approaches range from *conservative* (engineering-style) approaches, via *romantic* (creative and artistic) approaches, to *pragmatic* approaches, focusing social interaction between stakeholders (Fällman, 2003). A large number of stakeholders in the design process tends to increase the complexity in making sense of customer needs. In the U-CARE context, the researchers (i.e. "customers") intend to conduct nine different studies, structured into three work packages. After a number of sprint reviews, it became clear that the different work packages had different needs. The solution was to build a generic (and configurable) platform. The term 'generic' is used here to denote that the design intention is to make the platform applicable in new situations insofar as possible, without further design efforts other than user configurations. The term thus points towards an ideal, rather than denoting a claim to make an artefact that is applicable in all future situations.

This principle for mutability is to allow for the *users* to adapt the artefact to new situations through configuration. Configurable features (in contrast to hard-coded features) make the platform *mutable-in-use*. Our experience is that a strive for generic features increase the complexity of the artefact. This created a design tension – a trade-off situation between *simplicity* and *mutability-in-use*. Nevertheless, the efforts to create generic features appear to have pleased the customers. At the early stages of design, we could not get the customers to agree on the characteristics of the platform. However, once we presented fairly advanced implementations of generic character, the customers appeared more pleased with the design, since they realized it would satisfy the needs across all workpackages.

The U-CARE design process has rendered a number of generic features, such as:

- Creation and configuration of research studies with a number of options for inclusion criteria, randomization, activation of specific features in the platform (e.g. chat, forum, diary, library or FAQ), use of switchable user interface 'themes' to suit the target groups in the study (e.g. adolescents or elderly), and features to allow users' control of content.
- A questionnaire design tool that allows researchers and therapists to create custom questionnaires and results from calculations. The questionnaires can, among other things, be used to (i) collect research data at customizable observation points in the study, (ii) support the inclusion process in the studies, and (iii) allow for automated data analysis of collected data.
- A content management system that allows therapists to upload various types of items (e.g. pdf files and files of different audio and video format). Items are categorized using a triple-strategy consisting of (i) top-down classification using the dublin core (Duval et al., 2002), (ii) bottom-up classification using user-selected keywords ("tagging"), and (iii) on basis of user behaviour.

- Design of psychological interventions, following common principles in cognitive behavioural therapy, allowing the therapists to setup new interventions, which 'unfold' content to the patients on basis of rules defined by the therapist.
- Dynamic setup of actions, authentication, authorization, and menus. Action and menu setup is connected to dynamic roles. The actions and menus are specifically configured for each study, which further promotes the use of the platform in different research and treatment situations.
- Custom events to invoke 'configuration actions' at any given time, e.g. switching the role of a user at a given point in time, to allow for specific requirements in each known study, and to promote the usefulness of the artefact in possible future studies

The design and implementation of these features imply a rather complex system design. We agree with Folmer et al. (2006) that an important task for researchers is not only to propose the characteristics of features from a user interface perspective, but also to account for the architectural consequences of the proposed features. Aiming for a high mutability thus constitutes a risk. Every 'generic' feature adds to the architectural complexity of the artefact, and requires more resources to build than a situation-specific feature. In the U-CARE case, our design desicison were justified by an approximation of the desired 'use trajectory' of the design product, rather than merely the identifiable and current needs.

A final reflection is that the 'generic' features presented above are not really specific for the context of Internet-based psychosocial care. Although our evidence at this point is anecdotal, the ideas for mutability-in-use, as well as the concrete solutions, may be of value in other design contexts. But, to paraphrase Lee & Baskerville (2003), we are only able to make a generalization from empirical data to theory at this point.

3.3 Mutability-in-Design (Product View)

As discussed in the previous section, one may attempt to design an IT artefact in a generic way to make it support different types of situations. However, it is improbable that such a strategy satisfies *all* future needs. An additional strategy is to take into account the need for future changes through the internal software architecture and design. We consider architectural issues to be product-focused *mutability-in-design*.

As early as in the 1970's, ideas on software architecture and interface-based modularisation of software appeared in the literature, promoting re-use of code and 'designing for change' (e.g. Parnas, 1976). Parnas was an early proponent of separation of concerns, and dependencies between interfaces rather than dependencies between concrete implementations. A common approach to architecture is based on the concept of the model-view controller (MVC) pattern. Business logic (models) is here separated from user interface logic (views) through a controller. These concept were developed by Reenskaug at Xerox Parc in the late 1970's. In contemporary software development environments, and in multiple programming languages, there are numerous available implementations of MVC frameworks for adoption in software development (At the time of writing, Wikipedia lists 95 MVC implementations, covering all major development languages). Doubtlessly, the MVC pattern as an architectural design approach is widely adopted in software development practice. By applying the MVC framework at an early stage, developers are forced to adhere to certain conventions (e.g. separating business logic from user interface design). Such conventions promote change, and support programmers who are familiar with the MVC ideas to more quickly interpret code and make adaptations to it.

The U-CARE platform is based on Microsoft's implementation of MVC (built on top of the .NET framework). While MVC promotes changes to the artefact, it does not cover every need for mutability-in-design. Several other design patterns from object oriented design have been applied, such as the factory pattern (Gamma et al., 1995), and the use of interfaces (e.g. Canning et al., 1989) to achieve a weak coupling between classes and allow for unit testing[1]. Further, use of inheritance and filter technology provides all controllers with the same basic functionality. An example is provided in Figure 1. The U-CARE architecture lets the developers quickly add new actions to the controllers. By

[1]The use of interfaces to facilitate unit testing is further discussed in the next section.

inheriting the class UcareController, the new actions will automatically be plugged into the architecture. The pre-action filter uses the action model to see if the new action has been configured. If not, it is automatically configured using default values, and an email is sent to the development team, urging them to specify the properties of the new action (e.g. meta-data about action and logging options).

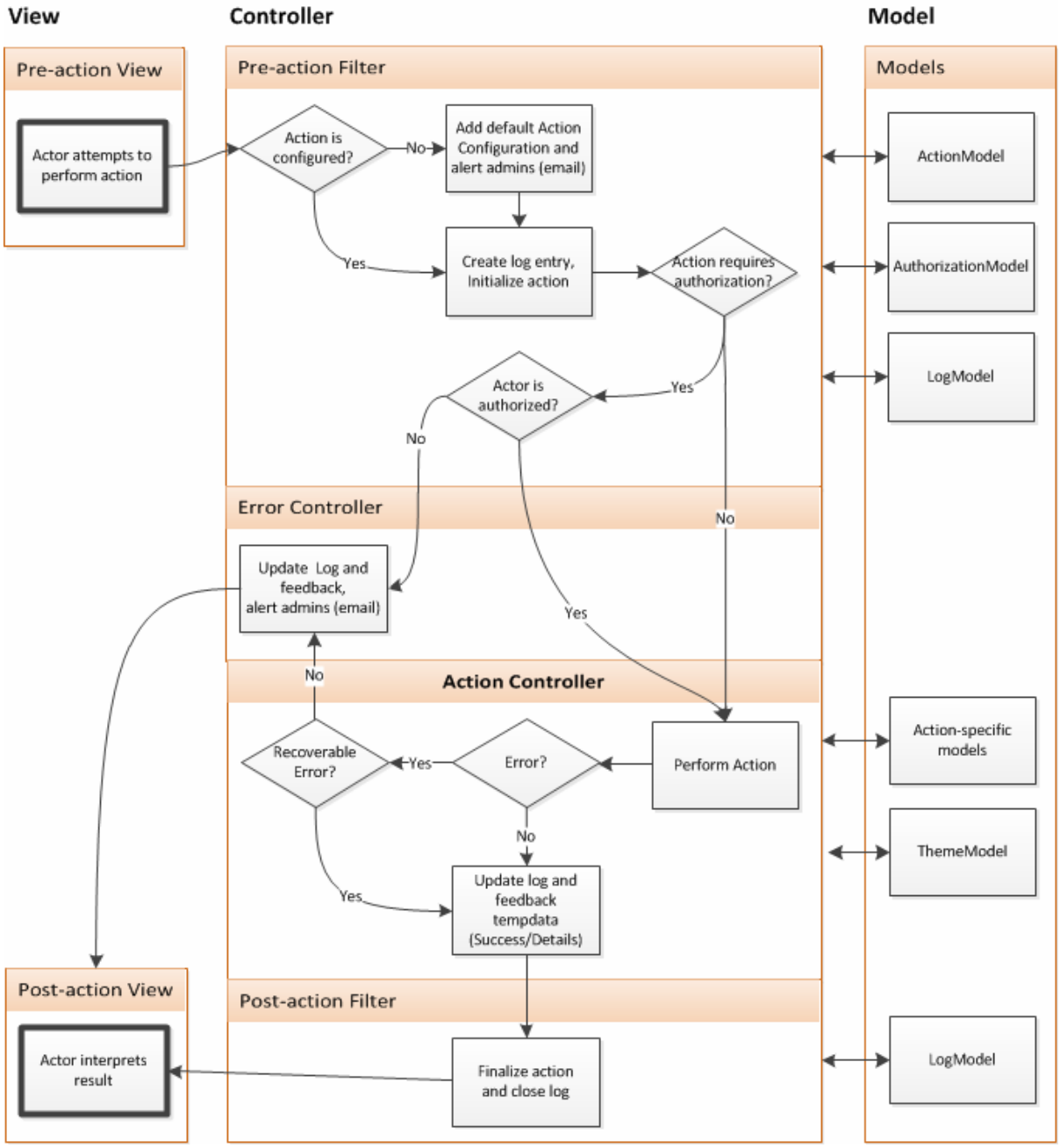

Figure 1: The U-CARE action framework

Figure 1 shows how the controllers use the services of the model classes to complete their work, such as authorization, logging, user feedback, and user interface themes. While this strategy minimizes the effort to add new actions to the artefact, it also provides an opportunity to make a change in a single place (e.g. in the pre-action filter) that affects every action in the artefact. The adoption of MVC and other object oriented design principles facilitates mutability-in-design, allowing for the *developers* to adapt the artefact to new situations. Experiences from the U-CARE development setting show that

new features are easily added to the existing ones, and seamlessly integrated into the overall architecture.

3.4 Mutability-in-Design (Process View)

At an early stage of development, the platform was re-factored from a regular .NET web solution into an MVC-based one, based on the idea that it would have a positive long-term impact on productivity. At that point, two out of three programmers were inadequately skilled in MVC, which hampered productivity in the short-term. Customer requirements are still changing, but the large number of existing model classes makes it a relatively easy task to adapt or create new features. The database consists of 50 tables, and the code base is approximately 20 KLOC at this stage. Changes to the database are followed by a rendering of so-called LINQ classes (supported by Visual Studio), and automated rendering of interfaces for data access. The interfaces are rendered using the DAL-automation script designed in the project. Figure 2 shows relation between the MVC framework and data access in the U-CARE project. Sometimes there is a need to manually revise the model classes. However, since the models have clear responsibilities, changes to the database lead to a manageable and limited set of changes in the source code.

Developers can change *any* piece of software (provided they have access to the source code). The challenge is to preserve robustness of the software whenever revisions are made. Automated testing is a common practice in contemporary software development. The idea is that a change of the source codes is followed by an execution of all tests. To make the testing efficient, test cases are automated through software implementation. The tools and techniques for automated testing have grown more sophisticated over time, one can for instance:

- Implement unit tests as code
- Test units in isolation without depending on other units (mock objects)
- Create re-runnable tests through automation of other IT artefacts, e.g. a web browser

In brief, automated testing reduces the risk of creating new errors when the existing software is adapted to new circumstances, and it is supported through various testing frameworks in different development environments.

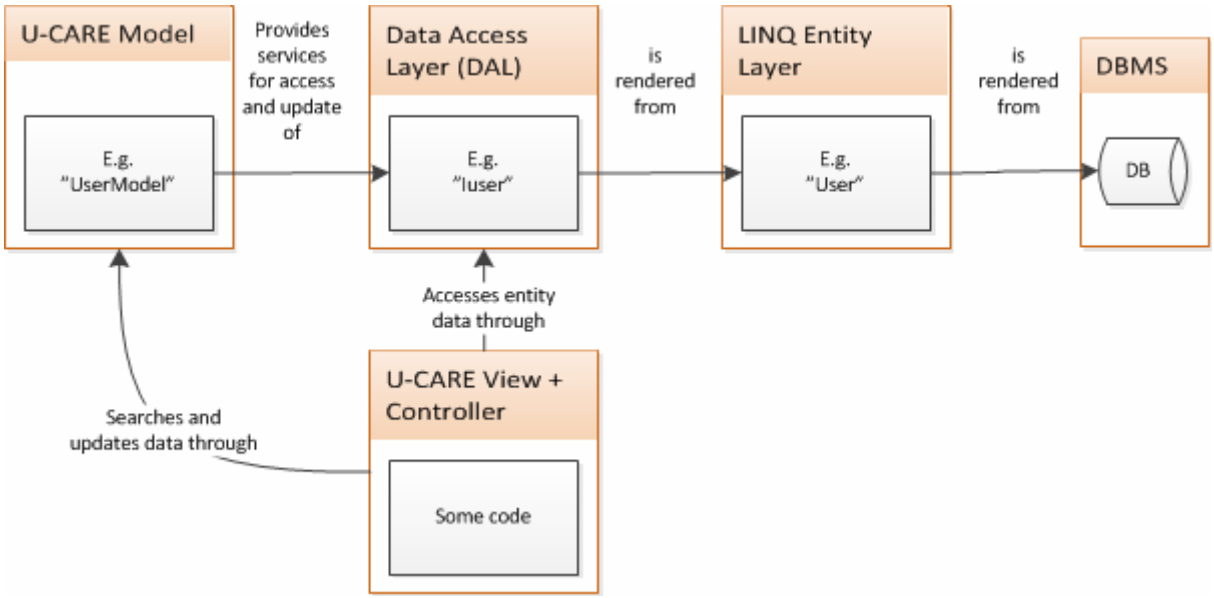

Figure 2: The U-CARE data access model

Two basic design patterns are important pre-requisites for automated testing: The Law of Demeter (LoD) and the Inversion of Control (IoC) pattern. The Law of Demeter dictates that an object should only speak to its closest neighbors. Such a design minimises the number of unnecessarily complex

connections between different system parts. The Inversion of Control Pattern (Gamma et al., 1995; Johnson & Foote, 1998), similar to Parnas (1972; 1976) ideas, advocates that we need to de-couple the execution of a task from its implementation, i.e. only to create dependencies between interfaces. This allows for flexible provision of one or more implementations of the interface at any point in time. It also allows for 'mock objects' for testing purposes; i.e. to create units without actually implementing other parts of the system in advance. The idea of only creating dependencies between interfaces is the rationale behind the extra DAL layer (Figure 2) and the DAL automation script mentioned above. In summary, the architecture and code requires certain architectural characteristics to allow for automated unit testing. The initial cost to setup automated testing is high due to several factors: (i) the architectural requirements, (ii) design of test cases and extra code to implement those as software and (iii) the need to employ developers focused on testing and the resources needed to educate the development team on the topic.

4 CONCLUSION

In this paper, we have proposed a further developed concept of IT artifact mutability and shown a number of empirical examples of how to achieve artifact mutability while at the same time maintaining stability of the artifact. This dual purpose affects both the process of design, and the characteristics of the design product. Based on these experiences, we conclude that 'mutability' as a design ideal affects design greatly. We have identified four types of mutability:

1. Mutability-in-Use (Process View)
2. Mutability-in-Use (Product View)
3. Mutability-in-Design (Product View)
4. Mutability-in-Design (Process View)

These types are an abstraction from one empirical context. Based on the notion of generalizability as suggested by Lee & Baskerville (2003), we cannot claim that these types are valid also in other development contexts. However, there is a bulk of existing theory that supports the categories, as discussed above.

The four aspects of mutability we propose support design researchers in thinking about the long-term impact of IT artifacts. The process view of mutability-in-use tells us to acknowledge the emergent properties of the human world and recognize that technology is appropriated, rather than 'used'. Gill & Hevner (2011) suggest that "the fitness of a design artifact must be estimated using a utility function that considers the full range of characteristics that can impact the likelihood that the artifact will further be reproduced and evolve" (p. 247). As design researchers, aiming at producing relevant and useful artifacts that impact society, it makes sense to factor in the likelihood of future use and evolution of the artifact, not only its aptness to solve a particular problem in a particular situation. The product view of mutability-in-use highlights the need for feature flexibility: IT artifacts that the users themselves configure for different situations. As designers, this means that the features of the artifact need to be based on a thorough understanding of current needs as well as elaborate ideas on future use scenarios. The process view of mutability-in-design highlights that design work by designers/developers need to be governed, e.g. through automated testing to promote stability when revising the artifact. The product view of mutability-in design informs us that mutability impacts the conceptual and technical software architecture of the artifact.

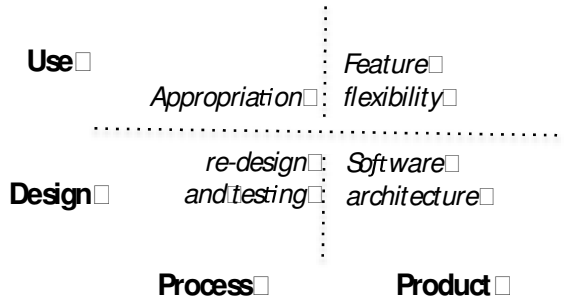

Figure 3: Different meanings of mutability

In brief, there are different meanings of mutability; depending on what aspect we focus. Figure 1 shows examples of how mutability may be interpreted given the proposed aspects of the concept. The idea of viewing mutability from these four aspects allows us to elaborate on how they are interrelated. Clearly, the ultimate goal is that the IT artifacts we design bring value to the emergent human world. Through the U-CARE case, we have shown a number of ways this is accomplished through the three other aspects of mutability. First, we attempted to understand a likely 'use trajectory' for the IT artifact, and build generic and configurable features in order to support a variety of anticipated immediate and future use situations. Gregor & Iivari (2007) use the phrase "re-design as a response to externally initiated change" as a way to enable artifact mutability. In a similar way, we see that there is a need for learning between the process of use and the process of design. Re-design efforts need to be based on what designers learn from users, while at the same time, designers play an important role in proposing the use of new technologies and solutions. We have shown that the design process and the software architecture are interdependent on one another - an architecture may enable or constrain design activities. Finally, every re-design effort affects the IT artifact (i) at an architectural level (e.g. by reinforcing or breaking an established architectural pattern, or introducing a new one), and (ii) at a feature level. For instance, an old feature may be re-factored to become more generic, or an entirely new feature may be added to solve new user needs.

Another type of interrelation between the mutability aspects is that the IT artifact itself may be built to support learning. In the U-CARE case, there are built-in features that allow users to provide feedback to designers (e.g. bug reports or wishes for changes). On top of this, there is a feature to enable rigorous logging of activities on the web site. The logs may be mined to identify re-design ideas, usability problems, and frequent technical problems on the web site. They are also used to filter out contextually relevant information for the users. All in all, the proposed conceptualization of mutability promotes a systemic thinking about how each aspect of mutability can inform the other ones.

While Gregor & Iivari (2007) focus on techniques to make IT artifacts adapt over time, e.g. through learning or context-aware IT artifacts, we have explicitly added a dimension of process – and thereby the social use context of the IT artifact – to explain mutability. Our empirical work has highlighted that in order to promote mutability, it needs to be acknowledged throughout the design process. Furthermore, IS research needs to build upon the knowledge base from SE, which to a large extent focuses on methods and frameworks to promote mutability.

Given the strong influence of mutability as a design ideal, it may be slightly misleading to regard 'mutability' a characteristic of a design theory (see Gregor & Jones, 2007). A 'mutable' solution to some problem class implies that good SE practices, such as the ones established in the U- CARE project and outlined above, are applied as a complement to any specific design theory. Any IS design theory based on the recognition of a continually evolving social world and technological context should thus suggest the use of state-of-the-art SE practices, and (if applicable) address principles for mutability that are specific to the particular theory (i.e. deviations from state-of-the-art practices that may be required in the context of the specific theory). In the case of U-CARE, so far, we have not identified any such deviations; still we have shown that mutability is a core issue in the design process.

Our findings signal that mutability, while having an impact on both the design process and the design product, qualify in its own right as an important knowledge base in design theorizing. In order to move

forward our understanding of mutability in design-oriented research IS researchers need to embrace a philosophical foundation that embraces it (as initiated by Gregor & Iivari, 2007) and factors in relevant theories from neighbouring fields, such as SE.

Acknowledgements

We are grateful to the U-CARE programme at Uppsala University for providing funding and empirical context for this research. Part of this research has been funded by the Swedish National Research School on Management and Information Technology through a postdoc grant.

References

Ågerfalk P J (2003) *Information Systems Actability: Understanding Information Technology as a Tool for Business Action and Communication*. Doctoral Dissertation. Department of Computer and Information Science, Linköping University, 2003

Avison, D.E. and Fitzgerald, G. (1995). Information Systems Development: Methodologies, Techniques and Tools. 2nd Edition. McGraw-Hill, London.

Canning, P.S., Cook, W.R., Hill, W. L. & Olthoff, W. G. (1989) Interfaces for strongly typed object-oriented programming. ACM SigPlan Notices, 24(10), pp. 25–35

Checkland, P. B. (1981). *Systems Thinking, Systems Practice,* Chichester: Wiley.

Ciborra, C.U. (1996). Groupware and teamwork. New York: John Wiley & Sons.

De Sanctis, G. and Poole, M.S. (1994). Capturing the complexity in advanced technology use: adaptive structuration theory, Organization Science, 5(2), 121-147.

DeLone, W. and McLean, E. (1992). Information systems success: the quest for the dependent variable. *Information systems research*, 3(1), pp. 60–95.

Delone, W. and McLean, E. (2003). The DeLone and McLean model of in- formation systems success: A ten-year update. Journal of Management Information Systems, 19(4):9–30.

Duval, E., Hodgins, W., Sutton, S., Weibel, S.L. (2002) "Metadata principles and practicalities", *D-Lib Magazine*, 8(4).

El Sawy, O. (2003). The IS Core IX: The 3 Faces of IS Identity: Connection, Immersion, and Fusion. Communications of the Association for Informa- tion Systems, 12(1):39.

Fällman, D. (2003). Design-oriented human-computer interaction. In Proceedings of the SIGCHI conference on Human factors in computing systems, page 232. ACM.

Folmer, E., van Welie, M., Bosch, J. (2006) Bridging Patterns: An Approach to Bridge Gaps between SE and HCI. Journal of Information and Software Technology. Volume 48, Issue 2, pages 69-89.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software,* Reading, MA: Addison-Wesley.

Gill, T.G. & Hevner, A. R. (2011) A Fitness-Utility Model for Design Science Research. In H. Jain, A.P. Sinha, and P. Vitharana (Eds.): DESRIST 2011, LNCS 6629, pp. 237–252.

Gregor, S. D. & Iivari, J. (2007) Designing for Mutability in Information Systems Artifacts. In Hart, D. N. & Gregor, S. D. (Eds.) *Information Systems Foundations: Theory, Representation and Reality*. Canberra: ANU E Press.

Gregor, S. & Jones, D. (2007). The Anatomy of a Design Theory, *Journal of the Association for Information Systems*, 8(5), pp. 312–335.

Gregor, S. D. & Jones, D. (2007) The Anatomy of a Design Theory.

Hayes, J. (2002). The theory and practice of change management. Palgrave Houndmills.

Hevner, A.R., March, S.T., Park, J., and Ram, S. (2004) Design Science in Information Systems Research," *MIS Quarterly* (28):1, pp. 75–105.

Järvinen, P. (2000) Research Questions Guiding Selection of an Appropriate Research Method, in Hansen, Bichler and Mahrer (eds.), *Proceedings of the 8th European Conference on Information Systems*, 3–5 July, 2000, Vienna, pp. 124–131.

Johnson, R. E. & Foote, B. (1988) Designing Reusable Classes. *Journal of Object-Oriented Programming*, 1(2), pp. 22–35.

Kautz, K. and Mcmaster, T. (1994). The failure to introduce systems development methods: A factor-based analysis. In Proceedings of the IFIP TC8 Working Conference on Diffusion, Transfer and Implementation of Information Technology (Levine, L. Ed.), p. 275, IFIP Transactions A-45, North-Holland, Amsterdam.

Kotter, J. (1996). Leading change. Harvard Business School Press.

Kraft, P., Drozd, F. & Olsen, E. (2009) ePsychology: Designing Theory-Based Health Promotion Interventions. *Communicaions of the AIS:* Vol. 24, Article 24, pp. 400 - 426

Lee, A.S. and Baskerville, R. L. (2003). Generalizing generalizability in information systems research. *Information Systems Research,* 14(3) 221-243.

Leonardi, P. and Barley, S. (2008). Materiality and change: Challenges to building better theory about technology and organizing. Information and Organization, 18(3), pp. 159–176.

Liskov, B. (1974). Programming with Abstract Data Types, in *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*, pp. 50–59, 1974, Santa Monica, California.

Lyytinen, K. and Hirschheim, R. (1988). Information systems failures – a survey and classification of the empirical literature. In Oxford surveys in information technology, page 309. Oxford University Press, Inc.

Majchrzak, A., Rice, R., Malhotra, A., King, N. and Ba., S. (2000). Technology adaptation: the case of a computer-supported inter-organizational virtual team. MIS Quarterly. 24(4), 569-600.

Markus, M.L. and Robey, D. (1988). Information technology and organizational change: Causal structure in theory and research. Management Science, 34 (5), 583-598.

Mumford, E. and Weir, M. (1979). Computer systems in work design–the ETHICS method: effective technical and human implementation of com- puter systems: a work design exercise book for individuals and groups. John Wiley & Sons.

Orlikowski, W.J. (1992). The duality of technology: rethinking the concept of technology in organizations. Organization Science, 3(3), 398-427.

Orlikowski, W.J. (2000). Using technology and constituting structures: a practice lens for studying technology in organizations. Organization Science, 11(4), 404-428.

Orlikowski, W. (2007). Sociomaterial practices: exploring technology at work. Organization Studies, 28(9), pp. 1435–1448.

Orlikowski W. and Scott, S.V. (2008). Sociomateriality: Challenging the Separation of Technology. Work and Organization, The Academy of Management Annals, 2(1), pp. 433–474.

Parnas, D.L. (1972). On the Criteria to Be Used in Decomposing Systems into Modules, *Communications of the ACM,* 15(12).

Parnas, D. (1976) On the design and development of program families. *IEEE Transactions on Software Engineering,* Vol. SE-2, pp.1-9, Mar. 1976

Preece, J., Rogers, Y., and Sharp, H. (2002). Interaction design: beyond human-computer interaction. John Wiley & Sons, Inc. New York, NY, USA.

Rentsch, T. (1982). Object Oriented Programming, *ACM SIGPLAN Notices*, 17(9), pp. 51–57.

Sjöström, J. (2010). Designing Information Systems: A Pragmatic Account. PhD Dissertation. Uppsala University, Sweden.

Sjöström, J. & Ågerfalk, P.J. (2009) An Analytic Framework for Design-oriented Research Concepts. Proceedings of AMCIS 2009.

Trigg, R.H. and Bodker, S. (1994). From implementation to design: tailoring and the emergence of systemarization in CSCW. Proceedings of CSCW 94, 45-54.