Limits in Modelling Evolving Computer-based Systems

Massimo Felici[*]
LFCS, Division of Informatics
The University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ, UK
mas@dcs.ed.ac.uk

Juliana Küster Filipe
LFCS, Division of Informatics
The University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ, UK
jkf@dcs.ed.ac.uk

ABSTRACT
This paper explores the limitations of one technique for modelling computer-based systems with evolving requirements. A case study is introduced which highlights the importance of taking a multi-perspective on dependable computer-based systems. This should be reflected in the modelling technique. Such considerations motivate our ongoing research agenda.

Keywords
Computer-based Systems, Modelling, Evolution, Design

1. INTRODUCTION
Design of computer-based systems requires different knowledge, activities, methodologies and tools reflecting the inner complexity of the modern electronic-mediated society. The massive usage of computer-based systems in our daily life has become unavoidable. Computers do not only interact continuously with our activities, but have became essential to carry out most of them. Applications of computer-based systems range from simple support to non critical activities (e.g., word-processing) to safety-critical applications (e.g., process control, medical systems and avionics). Design issues of computer-based systems can trigger system degradation and malfunctioning up to critical loss in safety-critical cases [8, 11]. The analyses of accidents point out that some undependable failures are due to miss-modelling human aspects related to the system in its context and underestimation of system evolution [8, 11].

This paper describes a computer-based system involving hybrid distributed resources interacting with each other. The analysis points out some of the limits in modelling evolving computer-based systems. The case study illustrates several issues arising in the design of dependable, service-level, human-process based systems. The identified limits in modelling and evolving computer-based systems motivate our ongoing research agenda.

The paper is structured as follows. Section 2 describes the case study emphasising general aspects characterising computer-based systems. Section 3 analyses the limits in modelling and evolving such a system. The review of these limits identifies our research agenda described in Section 4. Finally, Section 5 summarises our conclusions.

2. CASE STUDY: PARCELCALL
ParcelCall[1] is a project looking at creating a *parcel localisation system*: an open distributed system which is to be integrated with the legacy systems of transport and logistic companies (referred to as carriers). The project consortium includes hardware providers, integrators and carriers. ParcelCall, as described below, is an example of how computer-based (or, software) systems are changing in the modern electronic-mediated society. ParcelCall is useful to reveal engineering aspects of computer-based systems as well as general operational failures related to the nature of computer-based systems.

Current demands have transformed the transport and logistics process of today into an increasingly complex process requiring intelligent systems for sorting, planning, and routing; enabling faster and more reliable transportation, while supporting additional services such as time-sensitive deliveries and tracing of products. While many larger companies have developed solutions to provide their customers with such services, high costs impede smaller companies to do so as well. Current services provided to customers include notifications of product delivery or dispatch but are not commonly very precise. The ParcelCall project explores the development of a new low cost information infrastructure that enables the continuous information about the exact geographic position of parcels at any time. Transportation companies will thus be able to offer an additional valuable service to customers: the position and status of transportation goods can be queried at any time.

[*]This work has been partially funded by a grant of the Italian National Research Council (CNR) within the thematic Science and Information Technology, Bando n. 203.15.11. Research Programme: "Requirements Evolution: Understanding Formally Engineering Processes within Industrial Contexts".

[1]ParcelCall, EU project within the IST programme of the Fifth Framework. Project publications and description can be found at `http://www.parcelcall.com`.

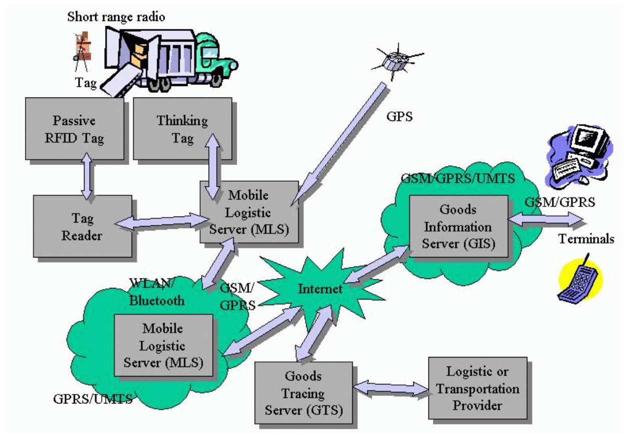
Short range radio
Tag
Passive RFID Tag
Thinking Tag
GPS
Tag Reader
Mobile Logistic Server (MLS)
GSM/GPRS/UMTS
Goods Information Server (GIS)
GSM/GPRS
Terminals
WLAN/ Bluetooth
GSM/ GPRS
Internet
Mobile Logistic Server (MLS)
GPRS/UMTS
Goods Tracing Server (GTS)
Logistic or Transportation Provider

Figure 1: The ParcelCall Architecture.

Figure 1 shows the architecture of the ParcelCall system with three main components:

- a *Mobile Logistic Server* (MLS): is an exchange point or a transport unit (container, trailer, freight wagon, etc). The transport units carry the parcels. A parcel has a tag identifying it: a passive radio-frequency identifier (RFID) tag or an active "thinking" tag. A thinking tag is able, for instance, to monitor environment conditions and emit alarms in case certain constraints are exceeded. Since containers can be inside other containers MLSs form a hierarchy. MLSs always know their current location via the GPS satellite positioning system.
- a *Goods Tracing Server* (GTS): comprises several databases one of which contains MLS hierarchies. Moreover, it keeps track of all the parcels registered in the ParcelCall system. GTS is integrated with the legacy system of transport or logistic companies.
- the *Goods Information Server* (GIS): interacts with the customers, provides the authorised customer the current location of her/his parcels, and keeps her/him informed in case of delivery delays. Interactions between the customers and the GIS are bidirectional and done via a terminal or a mobile phone.

All components interact through common open interfaces on top of standardised communication protocols (e.g., TCP/IP, GMS, GPRS, Bluetooth). The legacy system of a carrier is a computer-based system including both human carriers and machines.

The ParcelCall architecture is an example of how the modern electronic-mediated society has evolved since the utilisation of software as a means to perform computations. The original role of software was to automatically execute computations in order to simplify human workload (e.g, mathematical computations). Software consisted of single-platform (almost single-user) code developed and executed in the same environment. Nowadays computer-based systems like ParcelCall are heterogeneous multi-user, multi-platforms and multi-environments distributed systems supporting a wide range of human activities (e.g., computation, safety and security assurance, monitoring, etc.). The wide utilisation of software and its integration in many different applications, ranging from commercial to safety-critical systems, has changed the nature of software. This role revolution has changed the perception of design of computer-based systems [9]. Computer-based systems are not only a means to perform computational activities but also where the computations take place. The computational activity is distributed over the heterogenous resources (e.g., Software, Hardware and Liveware [5]) of a system and may be human as well. Hence cognition is not limited neither to the human cognition nor to the computer computation. The human is part of a computer-based system, and computations are the result of interactions among computer-based activities creating artefacts.

For instance, once integrated with the carrier system, the parcel localisation system will affect and influence the carrier's organisation in several different ways: economically, sociologically, and so on. Moreover, the impact on the work performed by human carriers is particularly important. Notice that in the carrier system computer-based activities consist of entwined human and machine actions, which will necessarily change even if the result of one such activity (e.g., delivery of a parcel) might apparently be the same. The ParcelCall system provides further artefacts, like the response to an alarm emitted by a parcel with a "thinking" tag. This artefact is the resulting outcome of combining human and machine artefacts.

A further implication of the ParcelCall's localisation system concerns quality of service (QoS) requirements like, for instance, timeliness, capacity, predictability, reliability, safety and security. One example of such a requirement in the ParcelCall system corresponds to the time required *normally* for accessing the status and localisation information of an item. According to ParcelCall documents, this should not exceed 15 seconds. This type of requirement is difficult to assess during design (e.g., how to test the system's performance during design?), as it may be sensitive to changeable human and technical factors. Other issues arise with conflicting requirements. For instance, information may be delayed for security in order to protect the parcel from malicious intentions. This delay is a trade-off between performance requirements related to information retrieval (to the customer) and security constrains (to protect the customers' goods). This shows how complex could be the identification and definition of non-functional requirements (e.g., quality, dependability, survivability, etc.) for computer-based systems.

3. LIMITS

In this section we discuss some issues arising in the specification and design of ParcelCall. We focus on the limits experienced in modelling ParcelCall and in analysing its evolution.

3.1 Modelling

As described in the previous section, ParcelCall is a computer-based system consisting of distributed and interacting heterogeneous resources. To model the ParcelCall system and its integration with the carrier system we take a component-based approach. We essentially use a pragmatic extension of UML [10, 13] for component-based design as given in [4]. We have used it to specify some aspects of ParcelCall in [7].

Using UML we assume that the computer-based system under development can be modelled according to the object-oriented paradigm, that is, in terms of objects, relationships between objects, and so on. Furthermore, a component-based approach for specification allows us to model ParcelCall's architecture at a high level of abstraction focusing on the main components and their interactions. UML is mainly a diagrammatic language offering several diagrams to capture different aspects of a system. It includes the Object Constraint Language (OCL), a textual notation for representing static constraints on the model which cannot be given through the diagrams. To adopt the UML modelling language, UML tools and methodologies for the analysis and design of dependable computer-based systems, it is necessary to adapt these to an inter-disciplinary approach.

UML is mainly used by software engineers or computer scientists, i.e., by people who can be easily trained to use it. To develop complex systems which involve a team of people of different disciplines, UML is not adequate as a common language. Even though UML as used in the requirement analysis relies mainly on simpler diagrams (like use case diagrams, sequence diagrams and possibly also statechart diagrams) that should be understandable enough for discussions with non-technical people this is far from being the case. These diagrams have been developed by software engineers (widely used to thinking in object-oriented terms) for software engineers. The diagrams do not reflect different perspectives or understandings of a system or its requirements, ways of working or thinking. Hence using UML implies adopting the design viewpoints represented by the UML syntax. Making available new alternative diagrams developed by people from other disciplines for capturing their understanding of the requirements, and combining their use with the UML diagrams would be a partial solution. A need for integrating these diagrams would follow. Alternatively, more powerful and different kinds of tools should be made available for a mixed team of people including tools for non-technical as well as for technical people.

From the software engineer's point of view, dependable systems raise further issues which need to be dealt with. In particular, how to capture QoS requirements. UML does not provide an adequate solution to this problem. In the ParcelCall example, for modelling adequately the dependable integration and interaction of the parcel localisation system and carrier system we need to:

- model QoS requirements that may cut across several components in the system;
- model part of the environment of the system as well;
- understand human behaviour and their understanding of the computer-based system.

UML describes the interactions between the software system and its environment (which includes the human) through use cases (which can later be refined into other types of diagrams). There is, however, a strong distinction between the software system and its external environment. There is no intention of integrating the environment, or part of it, into the model. This clearly limits the expressiveness of modelling languages like UML for the computer-based systems that we are interested in; capturing at least certain environmental features when modelling computer-based systems is essential. Moreover, it is crucial to understand and capture human behaviour and their interactions with the system in order to be able to assess or even predict possible failures.

One approach that considers the human in systems where human-computer interactions are highly critical is the work by J. Rushby as described in [14], among others. To try to analyse how errors can result from human-computer interaction, the approach compares what is called the *mental* model of an operator (system user) and the *system* model. The mental model corresponds to the model the operator believes to be the real model of the system. Both the system model and the mental model are described as finite state transition systems and checked for consistency using a mechanised formal method. The outcome of such a check suggests places where design should be improved.

In any case, to model human behaviour there is a need to borrow concepts and models from other disciplines like Cognitive Science and/or Artificial Intelligence. A combined formal approach can then be used to describe software systems and part of their environment, as well as their interactions. Verification tools based on such combined formalisms would make it possible to verify for instance dependable systems with human-computer interaction.

The design of computer-based systems like ParcelCall requires expertise from different disciplines. Human factors and organisational knowledge are important skills to define the systems requirements and how to deploy the system into a particular context. Engineering skills become crucial in designing, developing and testing. Most of the expected outcomes overlap different kinds of expertise. Hence a production environment requires all these different kinds of expertise and productive cooperation. This is a major issue in organising and managing multi-disciplinary development environments. Solving some issues in the presence of multi-disciplinarity can imply to turn into inter-disciplinarity, that is, the development environment has evolved in such a way to create its own inter-disciplinary settings. The mechanisms behind this evolution from multi-disciplinarity to inter-disciplinarity are still vaguely understood. Moreover, how to translate the above concepts into design, i.e., how to move from a multi-disciplinary to an inter-disciplinary design of computer-based systems, is challenging for future research.

3.2 Design for Evolution

Evolution is one of the most critical issues in the design of computer-based systems. Systems are already obsolete when they are delivered either because the environment has changed or stakeholders have changed their understanding about the system and its requirements. One of the reasons for this is that systems are engineered following a *static* paradigm. Even iterative development processes like the spiral model only capture evolution to a limited extent.

Experience shows [1, 2, 3, 12, 15] that it is not possible to freeze requirements at any stage of the life cycle. Designing phases are organised in terms of beginnings, ends, deadlines, that often developing environments fail to meet. Systems evolution is mainly considered within maintenance. But the evolution of computer-based systems is a concept broader than maintenance. Evolution actually starts as soon as a business case identifies a particular system, that is, evolution starts even before the system exists. Unfortunately most of the methodologies provide little support to evolution. Hence the need to develop new methodologies supporting system evolution. We started in different contexts to analyse requirements evolution [1, 2, 3]. The analysis of the taxonomy of requirements changes and product features has improved our understanding of the specific case study. Our previous experience should be taken into consideration while designing ParcelCall. The design context should register symptoms of evolution, otherwise it will not be possible to understand evolutionary information. Evolution stresses a different way of interpreting system design. Future research should investigate how methodologies can support *design for evolution*. In this new perspective, design and evolution are at the same level. Systems do not evolve, if we do not design their evolution.

4. A RESEARCH AGENDA

The previous section points out some limits related to modelling and evolving ParcelCall. As more changes to the system are necessary, the complexity of the system increases. The management of the system's complexity without any methodological support for understanding and bounding the side effects all over the life cycle will collapse triggering subsequent complexity explosions. There are still few methods to analyse and design scalable complexity. Our analysis of the ParcelCall case study suggests some limits in modelling such systems due to their inner complexity. The modelling issues and the evolutionary perspectives point out the crucial importance of analysing, modelling and evolving computer-based systems. It emphasises how development depends on these three aspects and how which they be understood as orthogonal to the entire life cycle. Design of computer-based systems should be based on an integration of analysis, modelling and evolution. Failing in taking into account one of these aspects will affect our ability to deploy evolvable computer-base systems.

A mature development environment should be able to analyse its own multi-disciplinarity in order to understand which inter-disciplinarity it is able to deploy. In other words, this inter-disciplinarity should match the one required by the specific computer-based system that the development environment aims to produce. To analyse multi-disciplinarity in a context and understand the process to deploy inter-disciplinarity into the development of computer-based systems is a key factor to success.

Experience shows that computer-based systems do fail. Consequently, these systems should be designed in such a way that considers faults. Evolution provides new insights in how to handle faults and should thus be considered in the design of such systems. This implies a new strategy in designing computer-based systems. The deployment of quality (dependable or survivable) computer-based systems can be obtained by designing its evolution to reach continuous quality (dependability or survivability). Quality (dependable or survivable) computer-based systems depend on our ability to react to failures by evolution.

Our analysis of ParcelCall points out research directions to improve our ability in modelling and evolving computer-based systems. In particular, in order to deal with evolution of computer-based systems we are currently devising an empirical framework to analyse rough data [3]. Industry collects a massive quantity of data, which is not analysed. This is because there is little support to analysing and structuring life cycle data. A systematic and methodological data analysis should be part of the corporate culture and not considered as an extra and expensive activity. Analysis is the only way to provide feedback to design and organisation.

The empirical framework represents the basis for understanding how requirements of computer-based systems evolve. This will allow to identify the stable and changeable parts of a computer-based system and making this information available for future developments. In contrast to the widely held perception about evolution, we consider evolution a paradigm for designing computer-based systems. Hence we intend to investigate evolutionary tools and methodologies for designing. Our analysis aims to take into account a multi-disciplinary perspective in order to link and understand socio and technical evolutions. The evolutionary perspective aims moreover to understand how evolution influences undependability and may influence dependability of computer-based systems.

ParcelCall made clear the current limits in expressing multi-perspective modelling. In order to improve our ability to model computer-based systems we need to understand how to incorporate human-factors information. Several questions can be formulated on our case study providing essential information which is, however, not further taken into account in current modelling techniques. Such questions include: how does the localisation system influence the carrier system in terms of organisation and culture; how do people view and cope with the information provided by the localisation system; how do different groups within a carrier organisation conceptualise and represent temporal issues; what is the impact of temporal validity of information on the working procedure of the humans in the carrier system; how are planning and routing decisions changed; how is status and localisation information represented for the different parties involved in the transportation network from carriers or subcarriers to customers; how accurate should provided information be; and so on.

Furthermore, current methodologies provide little guidance (or none) in how to integrate non-functional requirements into modelling. Simulation or verification tools could provide some feedback on the satisfiability of non-functional requirements which cut across different components of the system. The development of tools requires a formal modelling approach. In [7] we provide a logical framework to formalise the pragmatic extension of UML for component-based specification from [4]. Our framework consists of a distributed temporal logic MDTL which allows us to describe for instance local and global properties of components and component interactions [6]. With our framework we can already describe interactions between the several components within the ParcelCall system; between the customers and ParcelCall (in this case the component GIS); and between the legacy system of the carrier and ParcelCall (in this case component GTS). In the last case, we can also deal with the interactions caused by human carriers and ParcelCall. We can, however, not describe the human carrier perception of ParcelCall. We are considering an extension of our distributed logic to incorporate agent logics (essentially logics of knowledge of belief) for describing relevant aspects of human behaviour. How non-functional requirements can be captured in our logic or how feasible such an approach is for verification needs to be investigated.

5. CONCLUSIONS

In this paper we focus on multi-disciplinarity in design of computer-based systems with the aim of clarifying current gaps and problems. We outline the need to obtain an interdisciplinary approach in design and discussed how modelling languages like UML offer a limited support in dealing with dependable computer-based systems. In order to model computer-based systems, as in our case the legacy system of transport and logistic companies and its integration with the localisation system offered by ParcelCall, an approach like UML is not satisfactory. Several non-functional requirements arising from diverse perspectives on the system including those of disciplines like cognitive science, artificial intelligence, sociology and psychology, cannot be captured. A more powerful framework emerging from combined efforts of the different disciplines is required. Furthermore, the analysis of the case study points out limits in modelling *evolving* computer-based systems. Such considerations motivate our ongoing and future research. Finally, we believe that considerations made in this paper can be beneficial to other people approaching or dealing with the design of dependable computer-based systems.

6. ACKNOWLEDGEMENTS

We thank the ParcelCall consortium for providing the case study and in particular Robin Williams. Work reported here has been conducted within DIRC[2] (Interdisciplinary research collaboration in dependability of computer-based systems), a UK EPSRC (Engineering and Physical Sciences Research Council) funded project.

7. REFERENCES

[1] S. Anderson and M. Felici. Controlling requirements evolution: An avionics case study. In *Proceedings of SAFECOMP 2000, 19th International Conference on Computer Safety, Reliability and Security*, LNCS 1943, pages 361–370, Rotterdam, The Netherlands, Oct. 2000. Springer-Verlag.

[2] S. Anderson and M. Felici. Requirements changes risk/cost analyses: An avionics case study. In M. Cottam, D. Harvey, R. Pape, and J. Tait, editors, *Foresight and Precaution, Proceedings of ESREL 2000, SARS and SRA-EUROPE Annual Conference*, volume 2, pages 921–925, Edinburgh, Scotland, United Kingdom, May 2000.

[3] S. Anderson and M. Felici. Requirements evolution: From process to product oriented management. In *Proceedings of Profes 2001, 3rd International Conference on Product Focused Software Process Improvement*, LNCS 2188, pages 27–41, Kaiserslautern, Germany, Sept. 2001. Springer-Verlag.

[4] J. Cheesman and J. Daniels. *UML Components.* Component Software Series. Addison-Wesley, 2001.

[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] J. Küster Filipe. Fundamentals of a Module Logic for Distributed Object Systems. *Journal of Functional and Logic Programming*, 2000(3), March 2000.

[7] J. Küster Filipe. A logic-based formalization for component specification. Submitted for publication.

[8] N. G. Leveson. *SAFEWARE: System Safety and Computers.* Addison-Wesley, 1995.

[9] D. A. Norman. *The Invisible Computer.* The MIT Press, 1998.

[10] OMG Unified Modeling Language Revision Task Force. *OMG Unified Modeling Language Specification*, Version 1.4 draft, February 2001.

[11] C. Perrow. *Normal Accidents: Living with High-Risk Technologies.* Princenton University Press, 1999.

[12] PROTEUS. Meeting the challenge of changing requirements. Deliverable 1.3, Centre for Software Reliability, University of Newcastle upon Tyne, June 1996.

[13] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley, 1999.

[14] J. Rushby. Modeling the human in human factors. In U. Voges, editor, *Proceedings of Safecomp 2001, The 20th International Conference on Computer, Safety and Reliability*, LNCS 2187, pages 86–91, Budapest, Hungary, Sept. 2001. Springer-Verlag.

[15] G. Stark, A. Skillicorn, and R. Ameele. An examination of the effects of requirements changes on software releases. *CROSSTALK The Journal of Defense Software Engineering*, pages 11–16, Dec. 1998.

[2]http://www.dirc.org.uk/