ReSIST: Resilience for Survivability in IST

A European Network of Excellence

Contract Number: 026764

Deliverable D13: From Resilience-Building to Resilience-Scaling Technologies: Directions

Report Preparation Date: September 2007
Classification: Public
Contract Start Date: 1st January 2006
Contract Duration: 36 months
Project Co-ordinator: LAAS-CNRS
Partners: Budapest University of Technology and Economics
City University, London
Technische Universität Darmstadt
Deep Blue Srl
Institut Eurécom
France Telecom Recherche et Développement
IBM Research GmbH
Université de Rennes 1 – IRISA
Université de Toulouse III – IRIT
Vytautas Magnus University, Kaunas
Fundação da Faculdade de Ciencias da Universidade de Lisboa
University of Newcastle upon Tyne
Università di Pisa
QinetiQ Limited
Università degli studi di Roma "La Sapienza"
Universität Ulm
University of Southampton

Information Society Technologies

SIXTH FRAMEWORK PROGRAMME

Contents

Introduction to ReSIST deliverable D13

Michel Banâtre

IRISA/INRIA-Rennes

This document is the second product of workpackage WP2, "Resilience-building and -scaling technologies", in the programme of jointly executed research (JER) of the ReSIST Network of Excellence. The problem that ReSIST addresses is achieving sufficient resilience in the immense systems of ever evolving networks of computers and mobile devices, tightly integrated with human organisations and other technology, that are increasingly becoming a critical part of the information infrastructure of our society. These systems – large, networked, evolving systems constituting complex information infrastructures, perhaps involving everything from supercomputers and huge server "farms" to myriads of small mobile computers and tiny embedded devices – referred to in ReSIST as "ubiquitous computing systems", are essential to support and provide Ambient Intelligence (AmI) in the developing Information Society. Features of these ubiquitous systems are already present in embryo in existing infrastructures, and the risk of a "dependability and security gap" undermining the potential societal benefit from these systems is evident. A resilience approach to building and operating these systems is essential: they must be capable of surviving and delivering sufficiently dependable and secure service despite the inevitable residual development and physical faults, interaction mistakes, or malicious attacks and disruptions that their very scale, complexity and openness make more likely. ReSIST identifies the research challenge here in the need for *scalable* resilience of policies, algorithms and mechanisms. Quoting from ReSIST's Description of Work: "The current state-of-knowledge and state-of-the-art reasonably enable the construction and operation of critical systems, be they safety-critical (e.g., avionics, railway signalling, nuclear control) or availability-critical (e.g., back-end servers for transaction processing). The situation drastically worsens when considering large, networked, evolving, systems either fixed or mobile, with demanding requirements driven by their domain of application". Therefore, "The initial programme of jointly executed research (JER) on resilience-building technologies will focus on surveying and developing the expertise available within the network with a view to identifying gaps in the collective knowledge portfolio that need to be filled for these technologies to meet the scaling challenges of ubiquitous systems".

After a first phase devoted to an exchange of know-how between the partners, a first deliverable D12 summarised the current state of knowledge and ongoing research by the partners on methods and techniques for building resilient systems dealing with different aspects of resilience building and the corresponding sub disciplinary areas:

- Resilience architecting and implementation paradigms,
- Resilience algorithms and mechanisms,
- Resilient socio-technical systems,
- Resilience evaluation,
- Resilience verification.

This work has been the basis for the current phase reported in this D13 deliverable. It is focusing on "the scaling challenges identified by ReSIST, identifying a roadmap of integrated research for employing the current resilience-*building* technologies to develop the required resilience-*scaling* technologies, i.e.: evolvability, assessability, usability and diversity".

As detailed in the Description of Work, all of the various classes of threats have been considered in this pursuit of scalable resilience: development or physical accidental faults, malicious attacks, interaction mistakes. Those classes of threats are inter- related, e.g.: a) attacks are aimed directly at vulnerabilities, that are mostly of an accidental nature (either residual defects, including flaws in security enforcement, or physical malfunctions), and some vulnerabilities are unavoidable because of usability constraints, b) human-interaction mistakes can often be traced to development defects. However, interest in physical malfunctions in ReSIST is focused on problems that go beyond the current state-of-the-art, e.g., new modalities of malfunctioning in emerging technologies or applications (such as sensor networks or ambient computing services), or the role of physical malfunctions in creating vulnerability. It has to be stressed that the classes of threats are more than inputs for scalability issues, as the environmental and technological changes induce changing threats, e.g., accrued importance of a) malicious attacks that go along with openness, or of b) configuration mistakes that emerge as a major source of failures as system complexity increases.

This second deliverable D13 provides a detailed list of research gaps identified by experts from the four working groups related to assessability, evolvability, usability and diversity.

A work had been done between WG leaders to clearly identify to which resilience-*scaling* technology, gaps belong to. Moreover for each technology, a synopsis discusses in a general way about the characteristics of these future ubiquitous computing systems, identifies the challenges and shows how these challenges are covered by the identified gaps.

The structure of this deliverable is the following: a first part contains four sections each of them is dedicated to a synopsis of the each resilience scaling technology. The second part is the detailed list of the research gaps which have been identified; they are classified according to a particular resilience-*scaling* technology.

The deliverable thus presents the findings of ReSIST concerning research that needs to be pursued or undertaken on the resilience of computing systems and information infrastructures.

Each of the five working groups established for the deliverable D12 (architecture, algorithms, socio-technical issues, evaluation, verification) produced their views in terms of research gaps and challenges according to the four resilience-scaling technologies. The corresponding texts have constituted starting points for the newly formed working groups, according to the resilience-scaling technologies. Those texts have been reworked, augmented, and other texts have been produced. The working group leaders have produced syntheses out of the texts, where the various gaps and challenges have been clustered, and which constitute the first part of the deliverable.

Here is the brief structure of each reliance scaling technology section.

- Evolvability:

  Concerning this first resilient scaling technology, four mains groups of research challenges have been identified. The first one, *Resilient ubiquitous systems*, is focused on systems made of an extremely high number of components which interact in a ubiquitous way. The second is related to *Adaptation and self organisation*, it concerns the resilience of a ubiquitous system due to its ability to restructure or reorganize autonomously taking into account environment and/or users requirements. Another group puts in advance challenges behind *models for ubiquitous systems* mainly, models which provide a clear and correct specification of dynamic distributed systems. The last concerns *Resources and infrastructures for ubiquitous system*, it summarizes research problems related to architecture and data storage design, to virtualization, and to communication.

- Assessability

  Taking into account the scalability properties of future systems, assessability research gaps are grouped in three main topics: the impact of *large scale evolvable systems* on assessability research, *the methods and techniques* needed to be researched and developed in order to assess future large scale networked systems. The last one is about the acceptance of assessment methods and techniques as an *industrial engineering methodology.*

- Usability

  With respect to main stream research in Human-Computer Interaction, research gaps related to usability in resilient ubiquitous systems concern three main clusters addressed by research gaps: the *improvement of the development process* built around modeling activities; the *critical aspect of context* for addressing usability; the design of new methods, new techniques and tools to go *beyond standard usability*, taking into account user emotion or feelings which could have an impact on future systems and applications.

- Diversity

  Concerning this last technology, research challenges are divided in two main clusters according to the scale criteria. The *small-scale diversity* which is based on solutions and techniques already developed in previous research on diversity but taking into account the specificity of ubiquity such as implicit and/or spontaneous interactions. The second cluster concerns *large-scale diversity* in large scale ubiquitous systems.

The detailed texts for the gaps and challenges constitute the second part of the deliverable. The "gaps and challenges" texts have the same format: definition, application domains, current approaches, research challenges, and references.

Three drafts of the complete document have been produced, and each time underwent a reviewing process by all partners of ReSIST.

Part 1 - Syntheses

Evolvability

Co-ordinators: *András Pataricza*[1] *– András Kövi*[1]

Contributors: *Diola Abi Haidar*[6], *Roberto Baldoni*[8], *Sandra Basnyat*[9], *Christian Cachin*[7], *Miguel Correia*[10], *Marc Dacier*[5], *Jean-Charles Fabre*[3], *László Gönczy*[1], *Fabrizio Grandoni*[4], *Michael Harrisson*[11], *Marc-Olivier Killijian*[3], *Chidung Lac*[6], *David Navarre*[9], *Nuno F. Neves*[10], *Péter Pásztor*[1], *Gergely Pintér*[1], *Peter Popov*[2], *David Powell*[3], *HariGovind Ramasamy*[7], *Michel Raynal*[12], *Yves Roudier*[5], *Matthieu Roy*[3], *Paulo Sousa*[10], *Mark-Alexander Sujan*[13]

[1]University of Budapest, [2]City University, [3]LAAS-CNRS, [4]University of Pisa, [5]Eurecom, [6]France Telecom, [7]IBM, [8]University of Roma, [9]IRIT, [10]University of Lisbon, [11]University of Newcastle, [12]IRISA, [13]University of Warwick

1- Introduction

The notion of *evolvability* originates in biology, where it has two alternate and slightly complementary definitions: *"a biological system is evolvable if its properties show heritable genetic variation and if natural selection can thus change these properties";* alternatively *"if it can acquire novel functions through genetic change, functions that help the organism survive and reproduce"* [Wagner 2005]. Its core attributes in the original definition in the biology (structural variability and new functions originating as reaction to the influence from the environment) are still valid when generalized for IT systems, however, the definition has a broader coverage of phenomena in the context of information processing as evolution is not a spontaneous, but a purpose driven intentional process.

The design of **traditional IT systems** assumes that the system specification and resulting functional structure and implementation architecture are quasi-static. Changes in such systems aiming at an adaptation of the already running system to new requirements and technology improvements occur only by relatively rare updates. While these systems may function properly even during a long period of time, if their environment is well-predictable, the rigidity of the requirement-specification-functionality-implementation chain makes them vulnerable against to non-anticipated changes.

Nowadays, operational contexts evolve almost continuously, which necessitates an increasing level of *adaptation* to changes in the requirements, changes in the environment and in the implementation paradigms and technologies. These result not only in a demand for accelerated changes of the functionality and implementation, but also the development and application of new paradigms supporting adaptivity to the requirements and dynamic inter-application cooperation.

On the one hand, all the objectives, algorithms, and implementation paradigms **strongly depend on the application context**. The recent appearance of low-overhead virtualization drastically pushed "pure" IT infrastructures towards supervised utility computing performing resource allocation dynamically in a demand driven way. For instance, completely software based applications were traditionally deployed on distributed server farm frameworks relying on a fixed application-resource mapping, Similarly, service-oriented architectures are by their very nature adaptive and cooperative with a highly hidden architecture estimated on-demand from the functional architecture of the component services and their dynamically maintained list of available providers of related sub-functionalities (service offerings).

On the other hand, traditional embedded systems, especially safety-relevant ones **take a more conservative approach** to evolution by trying to confine fundamental system changes to gradual and hierarchical updates. However, even in the case of such traditional systems, satisfaction of a broader spectrum of user needs raises the need for a component-based product line approach. New embedded systems, however, often operate in a dynamic context, like mobile applications, that additionally require dynamic system configuration.

Thus, the broader notion of **evolvability** in systems and information infrastructures, which we consider here, targets the ability of the system to adequately accommodate changes in *user requirements* (and the corresponding functional and extra-functional specifications); or in the operational context (including short and long term changes in the *environment, the technology and threat scenarios),* or in the *resources* available at run-time for fulfilling the operational objective.

2- Challenges to dependability and security

Evolvability is a major means in the assurance of the efficient and dependable operation of IT systems. **Functional, structural adaptivity and integrative cooperation** are basic means to increase the matching of the requirements related to the quality, dependability and efficiency of the services delivered.

The core idea of *functional adaptivity* is the inclusion of the information related to the demands incoming from the environment ("outside world") into the control of the system, while *structural adaptivity* exploits the cooperation ability between intelligent resources and reconfigures them for a better fitting of the functional and extra-functional demand profile, eventually *integrating* new components into the operation.

The aspects of evolvability may take different forms during the different phases of the lifetime of an IT system.

- During **design and implementation** phases, evolvability concerns the ability to augment the system by adding new functions or accommodate to new technology.

  The rate of development of new applications and upgrades is increased by modern software technologies and by the efficient reuse of existing solutions, for instance through the use of model-based rapid application development and product-line oriented paradigms.

  However, this process has only a very limited support for the proof of correctness both from the point of view of functional and extra-functional requirements due to the limited reusability of (formal) proof methods. In addition, changes in the required functionality typically have a broad temporal range varying from a few days to a year, major technology changes have a characteristic period in the order of magnitude of a year, while new security threats arise with a rate rapidly accelerating beyond one day. *Design and proof methodologies* thus have to cope with security problems that did not exist at all at the time of creation, implementation and deployment of a system.

  A major challenge originates in the necessity of certifiable composition of existing systems having a long life span (like the majority of embedded systems) and new ones relying on latest technologies. Such an approach is typically used for augmentation by integration of add-ons, or integration of originally autonomous systems into a larger one in order to fulfil new requirements without compromising the built-in protective dependability assurance measures.

From the technical point of view, lack of interoperability and the limited scope of recent standards are major difficulties even in the recent industrial best practice. For example, rolling upgrade (i.e., runtime system modification and reconfiguration while still ensuring continuous operation) is still a major topic under discussion in SAForum (a major telco industry standardization body) for which there still does not exist a formal standard, despite years of practical experience (both positive and negative).

Another evolvability issue during design and implementation is that the methods and technologies employed must themselves be able to evolve to cope with new requirements and threat scenarios.

- During **operation**, evolvability concerns the ability to dynamically adjust system behaviour and potentially its functional architecture and deployment to cater for new operational contexts, including operational faults and attacks, new threats, and dynamic changes to the system environment.

  For instance, closed-loop solutions, like *autonomic and self-healing computing* aim to exploit spare resources in an optimal way. They adapt the system architecture to changing workloads, or to compensate faults by application migration. In addition, functional and structural adaptivity by means of intrusion detection triggered reconfiguration is a promising complement to intrusion-masking techniques for coping with the impacts of malicious attacks.

  Systems operating in environments where *physical mobility and/or infrastructure-related faults* may appear as frequently as every few seconds by the nature of the underlying mechanisms, have to comply with an extremely high rate of communication and membership faults compared to traditional systems, thus, fault tolerance mechanisms are fundamental in guaranteeing their correct functioning. An example for such environments is the mobile computing where loss and regain of connection of two units operating at a boundary of mutual communication is a quite frequent, thus necessitating a fast reaction.

While built-in intelligence is beneficial from the points of view of flexible creation of federated structures performing complex functionalities, resource exploitation and other performance related aspects, it imposes critical challenges from the point of view of dependability.

Traditional systems composed of isolated subsystems cooperate with their environment or other systems in a restricted way via predefined interfaces originally designed under a "closed world" assumption. Impacts of external influences are confined to have only restricted influence on their behaviour. Advanced methods exist for heuristic or formal proof of their correctness and run-time protection even in the case of anticipated faults.

The openness of context aware and structurally aware systems is a prerequisite for cooperation. However, it opens new possibilities for propagation of errors originating in external faults, as the fundamental "closed world" assumption on the operational environment is no longer valid. Neither proper modelling, nor formal analysis methods are currently able to cope with dynamically evolving architectures. Moreover, even the collection of critical situations in dynamically evolving systems is one of the generally unsolvable challenges due to the limited predictability of their behaviour.

As cooperation frequently relies on geographically widely distributed systems, proper protection against natural faults and cyber and physical criminality necessitates more and more the application of catastrophe planning techniques. For instance, open infrastructures became priority targets of malicious attacks as they were composed by integration of components originally designed under a closed-world assumption with priority concerns on safety and availability.

Moreover, as the complexity of systems grows more rapidly than the quality of their components and the reliability of the integration process, development faults, configuration management faults and inter-operational faults play an increasingly important role. The dominant design methodology for configuration design offered even by the best industrial solution providers is still rather ad-hoc and heuristic, and is thus in itself a major source of deployment/modification time design faults. One of the core problems is the lack of a proper model-based configuration design methodology.

Factors like the use of external services and infrastructure beyond the control of the final service provider and the necessity to co-exist with unavoidable faults result in additional problems related to the different dependability and security assurance means. One of the core problems is that such cooperative systems cannot rely on their components having been designed according to specific fault prevention and fault tolerance policies.

3- Research Challenges of Evolving Resilient Systems

The research challenges identified by the ReSIST contributors are clustered according to their main focus and target methodologies subject to research and development:

- **Resilient ubiquitous systems**: *Evolution Of Threats* (GE1), *Resilient Ambient Systems* (GE2), *Trustworthiness/intrusion Tolerance In WANs* (GE4)
- **Adaptation and self-organisation**: *Design for Adaptation: Framework and Programming Paradigms* (GE7), *Complexity & Self-Organisation* (GE9)
- **Models for ubiquitous systems**: *Distributed System Models* (GE3), *Service Oriented Architectures* (GE8), *Managing Multiple and Heterogeneous Models* (GE11)
- **Resources and infrastructures for ubiquitous systems**: *Resilient Data Storage* (GE5), *Critical Infrastructures* (GE6), *Virtualisation* (GE10)

In the following, we give a short overview on the different research gaps identified in the topic of *Evolvability* along the previously introduced aspects.

Resilient ubiquitous systems. One of the most active areas of research nowadays is focused on systems that contain an extremely high number of components and provide their services in a ubiquitous manner, so that they are available in extensive areas and not only in specific locations.

The huge number of components results in more and more complex emerging IT systems for which a fully extensive analysis is becoming less and less attainable, thus exposing them to unidentified technical and socio-technical (human-involved) threats. The "Evolution of Threats" research gap (GE1) focuses on the description of complex socio-technical systems, including human components in policies, and modelling the evolution of such systems.

The trend of decreasing size and cost of electronic devices enable their more widespread application in various domains. Among those, the application of ambient systems (wearable computing, personal area networking, ubiquitous computing, etc.) is one of the emerging fields. In these environments, malicious or faulty behaviour can cause serious problems, if such aspects are not considered deeply enough during the design of their protocols and interfaces. The contributors of the "Resilient Ambient Systems" research gap

(GE2) define two classes of systems — smart spaces and smart populations — as the two major research directions in this field.

The most well-known ubiquitous systems are Wide Area Networks (WANs), for example the Internet. Trustworthiness is especially important in these networks where a huge number of clients are connected to each other and communicate through very vulnerable communication channels. Fault tolerance techniques are used to prevent failures from manifesting themselves at the user’s level and to maximize availability by minimizing down time. Recently, similar techniques have been applied to security-related issues. The "Trustworthiness/Intrusion Tolerance in WAN" research gap (GE4) focuses on the design of evolvable trustworthy protocols and systems for Wide Area Networks.

Adaptation and self-organisation. Resiliency of a system is highly affected by its ability to adapt to new requirements of the environment and restructure, reorganize itself accordingly.

The adaptation capabilities of a system depend on early stages of its design and the software development technology that was selected. The contributors of the "Design for Adaptation: Framework and Programming Paradigms" research gap (GE7) identify the most important research challenges to set up an environment that offers system architects a high level of control over the adaptability of the final solution.

Centralised control can only be implemented in relatively small systems. However, today it is not rare that a huge number of entities cooperate to reach an objective, without an option for centralised control. The common attribute of these systems is the ability to organize themselves and thus overcome the complexity issues of the huge number of collaborating parties. The "Complexity and Self Organisation" research gap (GE9) addresses the questions related to the modelling and analysis of self-organisation.

Models for ubiquitous systems. Modelling and model based validation and verification have proven to be the only way to cope with the complexity of current systems. There are well-founded models for sequential, parallel and static distributed computations that are proven to correctly describe those types of systems. However, there is still no agreed model for dynamic distributed systems. The "Distributed System Models" research gap (GE3) raises the questions which may result in a clear definition of such systems.

The design of usable, reliable and fault-tolerant interactive safety-critical systems is based on a mass of information of multiple natures from multiple domains. This forces design complexities and dangers surrounding the gathering and refinement of this mass of information. The contributors of the "Managing Multiple and Heterogeneous Models" research gap (GE11) argue for formalisation on the gathering, refining and modelling of data to deal with the previously mentioned issues.

One of the most widely used paradigms nowadays for service orchestration and provision is the Service Oriented Architecture (SOA). The SOA refers to "loosely-coupled" heterogeneous systems where components expose only their functionality and interfaces and are integrated by their functionality, i.e., no strong connection between system modules is present. Essential work has already been done in this field including the development of modelling languages and methods for formal verification. However, there are still important aspects that have not been elaborated to a full extent, for instance, qualitative evaluation of service level agreements (SLAs) and componentization methods for large systems. Different aspects of these open questions are discussed in the "Service Oriented Architectures (SOA)" research gap (GE8).

Resources and infrastructures for ubiquitous systems. Currently, humanity is in an era where it appears to want to "permanently store every piece of information", which requires more and more storage space [Cooper 2007]. In the past few years, the amount of data stored and the additional requirements that emerged, such as catastrophe survival and compromise prevention, have raised new requirements on data storage mechanisms. The "Resilient Data Storage" research gap (GE5) argues for new data resilience and information evolution tracking mechanisms. Moreover, it considers issues of collaborating partners and disaster recovery for data storage.

In the recent years, there has been a considerable regain in interest regarding virtualization as a means for more efficient utilization of today's high performance hardware resources. Virtualisation is an architectural approach that allows the real hardware configuration of a system to be abstracted away. There seems to be a great potential in making our systems fault tolerant or more secure cost efficiently without a significant drop in the overall performance by using virtual machines. The contributors of the "Virtualisation" research gap (GE10) argue for exploration of new ways of leveraging virtualization for enhancing system dependability and scrutinize the effects of virtualisation on other system attributes like performance and security.

Additionally, computer networks are playing a vital role in today's society. Local area networks are used locally in every business and in an increasing number of homes. These networks are connected to the Internet that ties the whole world together and allows a user to easily access resources on a remote machine be it his neighbourhood or at some distant point on Earth. Large retailers (e.g. energy, gas, water, etc.) are no exception to this. Their systems are also connected to the Internet directly or indirectly, and this introduces vulnerabilities to these infrastructures that were not present before. Contributors of the Critical Infrastructures research gap (GE6) argue for a new reference architecture to create a common framework for unifying the access methods and structures used during system development. They also argue that the interdependencies between these systems have to be revealed in order to enable the creation of models to be used for fault prediction and simulation.

4- References

[Wagner 2005] Wagner, A. 2005. *Robustness and Evolvability in Living Systems* (Princeton Studies in Complexity). Princeton University Press. ISBN 0691122407. referred by wikipedia under the keyword "evolvability"

[Cooter 2007] M. L. Cooter, "Study: World needs more data storage space", Computeworld, Mar. 2007, URL: http://computerworld.co.nz/news.nsf/news/9162EF3CCF5D4092CC2572980014DC51

Assessability

Co-ordinator: *Aad van Moorsel*[4]

Contributors: *Cinzia Bernardeschi*[5]*, Peter Bokor*[1]*, Andrea Bondavalli*[5]*, Marc Dacier*[8]*, Colin O'Halloran*[6]*, Mohamed Kaâniche*[3]*, Karama Kanoun*[3]*, Marc-Olivier Killijian*[3]*, Jean-Claude Laprie*[3]*, Giuseppe Lettieri*[5]*, Bev Littlewood*[2]*, Paolo Lollini*[5]*, István Majzik*[1]*, Nick Moffat*[6]*, Alberto Pasquini*[10]*, Péter Pásztor*[1]*, Holger Pfeifer*[7]*, Peter Popov*[2]*, James Riordan*[11]*, Nicolas Rivière*[3]*, Yves Roudier*[8]*, Matthieu Roy*[3]*, Lorenzo Strigini*[2]*, Neeraj Suri*[9]*, Hélène Waeselynck*[3]

[1]University of Budapest, [2]City University, [3]LAAS-CNRS, [4]University of Newcastle, [5]University of Pisa, [6]Qinetiq, [7]University of Ulm, [8]Eurecom, [9]University of Darmstadt, [10]Deep Blue, [11]IBM

1- Synopsis

As indicated in the ReSIST work programme, the term **assessability** refers to the *ability to assess a system's ability to function properly and the quality of service it delivers*. The work programme observes that *current and future systems result from evolutions of pre-existing systems*, and that as a consequence, assessment should move *from off-line and pre-deployment, to continuous and automated operational assessment*. We note that assessment includes validation and verification approaches as well as quantitative and probabilistic approaches.

In the research gap analysis carried out within ReSIST the various authors and contributors have identified a set of research challenges that expand on this view by formulating in more detail the assessability challenges that are associated with the "evolvable systems" identified in the work programme. Moreover, the authors identify that assessment is not only carried out differently (in operational settings [GA1,2,9,10]), it is also significantly harder because of interacting systems, elements of chaos or self-similarity, and the complexity of inter-organisation and/or human interactions [GA7,8,9,15,16,17,18,19]. In addition, the contributors argue that these systems may require different or additional metrics to be assessed properly, to express notions related to security, trust, evolvement, business impact, and human perception and interaction [GA2,4,7,10,17].

In what follows we discuss assessability research gaps from three perspectives. First, we discuss the impact the characteristics of current and future large-scale *evolvable systems* will have on assessability research. Then we discuss the shortcomings of current assessment *methods and techniques* and identify the required set of tools needed to respond to the above-mentioned challenges. Finally, we discuss the issue of acceptance of assessment methods and techniques as *industrial engineering methodology*, in relation to issues such as benchmarking, resilience cases and model-based engineering.

2- Research Challenges of Assessing Evolvable Systems

The following research gap descriptions present a variety of systems and system characteristics that illustrate the increasing challenges we face in resilience assessment:

- GA7 *Metrics/models for evolution processes*
- GA8 *Evaluation of dynamic systems*
- GA9 *On-line assessment for resilience*
- GA10 *Trust and Cooperation*
- GA11 *Verification/testing of distributed mobile systems*
- GA13 *Test methods for aspect-oriented systems*
- GA15 *Emergent behaviours in socio-technical systems*
- GA18 *Inter-organisation boundary failures*

Current and future large-scale networked systems are typified by elements such as mobility, evolution, high interconnectivity, novel programming approaches, etc. All these elements introduce new assessment problems. It would go too far to attempt to be exhaustive in listing all these characteristic elements, but some of the individual gap descriptions illustrate such issues very well. For instance, increased mobility of system users challenges the traditional way of modelling traffic load, generating test cases and building test beds [GA8,11]. An increasingly popular and important programming approach such as aspect-oriented programming, itself an enabling technology for evolving systems, inherently requires operational assessment, as pointed out in [GA13]. Many other examples are discussed throughout the gap descriptions, but at this place we choose to elaborate on two particularly important overarching issues: (i) the role of the human, and (ii) the increased complexity of the systems (exhibiting emergent properties and complex (chaotic, self-similar, heavy-tailed, etc.) behaviour).

(i) human users. The human (be it a user, customer or IT personnel) plays a crucial role in the resilience of systems and applications. Humans make mistakes and cause failures, and thus influence resilience of socio-technical systems ([GA17] and also [GA3,5,15,16]). Vice versa, humans may be instrumental in maintaining system resilience [GA5,17]. To model human behaviour demands advances in techniques such as described in [GA17], and requires increased collaborative research with the social sciences. The creation of test beds or living labs, as suggested in [GA8] is also full of practical challenges.

(ii) increased complexity. Evolving IST systems exhibit emergent behaviour that is (or may be) significantly more complex than traditional computing systems. Multi-party game-theoretic scenarios and on-line adaptation scenarios are inherently more complex than traditional assessment problems, as discussed in [GA7,8,9,10]. Yet more dramatic is the increased complexity when one wants to assess emergent, highly variable and possibly chaotic behaviour. It has been known for over a decade that Internet traffic exhibits self-similarity, thus challenging (although not necessarily refuting) traditional simulation and modelling approaches (such as those based on Markov Chains). In [GA15,16] various resilience scenarios are discussed that may exhibit similar complex behaviour. If such effects can be demonstrated, it will further increase the need for modelling at higher order of complexity.

3- Research Challenges for Methods and Techniques

The following research gap descriptions concentrate on methods and techniques that need to be researched and developed in order to assess systems discussed in the previous section.

- GA2 *Data selection, collection, validation*
- GA4 *Security quantification*
- GA6 *Model complexity*
- GA12 *Abstraction*
- GA14 *Compositional reasoning*
- GA16 *Modelling effect of micro-decisions in the whole system*
- GA17 *Modelling human behaviour*

Traditional challenges in model-based or experimental evaluation of IST systems further increase when applied to evolving systems: models become more complex [GA15,GA16], abstraction is yet more critical [GA12] and compositional reasoning is a must [GA14]. In this summary we want to focus on three novel trends that one can identify across the various research gap contributions, namely (i) assessing for stakeholders (users, customers, enterprises, etc.), (ii) quantification of security, and (iii) methods and techniques to deal with increased complexity.

(i) stakeholders. The needs for and interest in assessment of an IST system depends on the interested stakeholder, which may be human users, customers, enterprises, government, etc. Not all gaps are discussed squarely from the perspective of stakeholders and their specific requirements, but [GA2,4,6,7] point to work that relates to recent research and funding efforts in enterprise trust economics and services sciences. These efforts explore resilience assessment from the perspective of enterprise decision makers, requiring the integration of user, system and economic models. Of interest in this respect are also the complex and sensitive issues that come into play when discussing benchmarking technologies from the perspective of system vendors [GA5]. Another case in point is the inherently subjective assessment of trust in dynamically established cooperation [GA10], thus demonstrating that assessment from the perspective of a human influences metrics and appropriate approaches.

(ii) security metrics. Quantification of security, as a specific subset of resilience, is receiving considerable attention among researchers, and is also highlighted in several of the gap descriptions ([GA4] in particular, but also [GA2,7,10]). The choice of metric as well as the modelling of aspects such as attacks, responses and the effort to break systems are among the challenges that need to be addressed.

(iii) dealing with complexity. We already discussed this issue in the section on evolvable systems, and the arguments put forward there directly translate in associated research challenges with respect to methods and techniques: fast on-line algorithms, continuous light-weight measurement, modelling emergent behaviour, representing chaotic and other complex interactions, etc.

4- Assessability as an Engineering Discipline

Several contributors discuss assessability from the perspective of its maturation and acceptance as an engineering discipline. In particular, the authors discuss model-based engineering, dependability cases and benchmarking:

- GA1 *Integration of modelling in the engineering process*
- GA3 *Dependability cases*

- GA5 *Benchmarking*

It should be noted that none of the contributed assessability research gaps questions the necessity or usefulness of assessment. However, there may be many legitimate reasons why (some types of) assessment cannot obtain the desired priority and time and effort investment when building, installing or configuring an IST system. One manner in which assessment can be conducted easier and more frequently is by integrating assessment in the engineering process. [GA1] discusses this in detail, especially in relation with model-based design as promoted by OMG. [GA3,5] advocates the use of dependability cases and systemic benchmarking. Each of these approaches becomes increasingly challenging, but possibly also increasingly urgent, in the domain of evolvable systems discussed in ReSIST.

Usability

Co-ordinator: *Philippe Palanque*[3]

Contributors: *, Sandra Basnyat*[3]*, Giorgio Faconti*[6]*, Jérémie Guiochet*[4]*, Michael Harrison*[5]*, Matthieu Roy*[4]*, Lorenzo Strigini*[2]*, Daniel Toth*[1]*, Marco Winckle*[3]

[1]University of Budapest, [2]City University, [3]IRIT, [4]LAAS-CNRS, [5]University of Newcastle, [6]University of Pisa

1- Introduction

A widely used definition of usability, provided by Jakob Neilsen [Nielsen, 1993] is as follows, "Usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process. Usability is defined by five quality components: a) learnability (how easy is it for users to accomplish basic tasks the first time they encounter the design?), b) efficiency (once users have learned the design, how quickly can they perform tasks?), c) memorability (when users return to the design after a period of not using it, how easily can they re-establish proficiency?), d) errors (how many errors do users make, how severe are these errors, and how easily can they recover from the errors?), e) satisfaction (how pleasant is it to use the design?).

We start this summary by reminding the importance of usability as a means of achieving a resilient Information Communication Technology (ICT) (section introduction). We then discuss research challenges related to usability that ReSIST partners have identified as of particular importance and that must be addressed in order to deal with the problem of designing with usability in mind for achieving resilience.

These research challenges are grouped into three broad clusters addressing research challenges in the field (or related to the field) of usability. With respect to main stream research in Human-Computer Interaction (HCI), (i.e. interface design, usability testing, participative design in order to improve the system-part of the whole system) the ReSIST view on usability takes a broader view, stressing the importance of **improved development processes** built around modelling activities, and putting usability at the centre of these processes. Another important ReSIST viewpoint is the **critical aspect of context** for addressing usability. Designing with context in mind raises new challenges such as how to address user confusion of system modes, how to avoid misunderstandings of procedures by operators and how to master an evolving context of use (related to the system's reconfiguration or emergent behaviours). The fact that systems may be used by a large number of different users, with implicit or explicit usage (with ubiquitous sensing systems), users on the move (using mobile technologies) and emotions or feelings that can have an impact on their efficiency (as in computer games) calls for methods techniques and tools that go **beyond standard usability**.

2- Research challenges to development processes

User Centred Design (UCD) and Usage Centred Design [Constantine & Lockwood 2002] approaches propose to centre the development processes on operators' work context (or end users in general which may not necessarily be operators) including activities, knowledge, experience, work environment, to gather information about such constraints. However, current practice in system development and research work in this area have only addressed usability as a marginal aspect. Despite some recent efforts, usability is still often addressed too little and too late in the development processes by means of usability evaluation activities when the entire (or most of the) system has been designed A good example of such a too little too late can be seen in the Rationale Unified Process connected to the widely spread UML set of notations and clearly pointed out in [Constantine & Lockwood 2002]. Current leading standards widely adopted in industry such as the Rationale Unified Process (RUP) [Kruchten 1998] built on top of UML still offer little to support the design and construction of usable systems. UCD approaches call for iterative development processes involving evaluations with users, based on preliminary versions of the system (or at least its user interface) thus getting early and detailed feedback that can be fed into other phases of the development process. *UCD & resilience engineering and development processes* [GU2] makes explicit the need to integrate User Centred Design processes and resilient engineering approaches to support the development of usable, safe and reliable systems.

One identified way of reaching such objective is to exploit, in a systematic way, models and modelling techniques. *Modelling aspects of HCI* [GU4] details the advantages as well as the challenges for bringing modelling techniques to current HCI practices. A need for models also appears in gaps that are highly related to usability (but not presented in this section as they are of primary interest to other resilience scaling technologies) but focusing on specific aspects: *Modelling Human Behaviour* [GA17], *Managing multiple and diverse models* [GE13], *Integration of modelling in the engineering process* [GA1]. With a non explicit relationship to usability, the issues raised by *Model complexity* [GA6] have also been identified (they address the management of large models by tolerant and avoidance approaches) and will become highly relevant to user centred approaches if modelling techniques are systematically introduced as argued. Modelling the role of humans is also one of the aspects in *Dependability case* [GA3] as their difficult-to-predict nature can play a critical role in the safety and dependability of the system.

Assessing the impact of the various proposed approaches on the overall usability of the system has been identified as a critical issue to be addressed in *Usability Metrics* gap [GU1] that focuses on assessment at design time while *On-line assessment for resilience* [GA9] deals explicitly with the assessment at runtime (with a system able to evolve).

Various authors of gap descriptions have also pointed out issues raised by the specificities of some application domains. Applications such as decision making systems (see research gap description *Modelling Effect of Micro-decisions In The Whole System* [GA16]) or ambient systems (*Resilient Ambient Systems* [GE2]) which also raise interesting research challenges in terms of development processes addressing usability explicitly. *Human Diversity and Human-machine Diversity* [GD4] addresses the broader perspective of development processes supporting the design of usable large scale systems with diverse users such as healthcare applications where "different people bring to their tasks different practices, background experiences, levels of expertise, strategies …".

3- Research challenges to contextual usability

Context aware and context sensitive applications have been the focus of attention for several years now mainly due to developments in the field of the ubiquitous computing which refer to the trend that we as humans no longer interact with one computer at a time, but rather with a dynamic set of small networked computers, often invisible and embodied in everyday objects in the environment.

The following research gap descriptions present a variety of ways to address the issues raised by the design and evaluation of usable context aware applications. These issues are related to the fact that such applications will change behaviour according to the context in which they are used (diverse users, diverse location, diverse environmental attributes …).

- GU1: *Usability metrics* – which metrics should be defined and used for assessing the usability of a context sensitive application?
- GU3: *Context confusion* – how to avoid misunderstanding and user confusion when applications change behaviour according to the context in which they are used?
- GD6: *Reconfiguration and contextual/environmental issues* – how to support design activities when reconfiguration of systems is required due to failure of sub-systems and more specifically how to assess the usability of the resulting reconfigured system?
- GA15: *Emergent behaviours in socio-technical systems* – how to predict the behaviour of large scale and evolvable systems and if some unexpected behaviour emerges how to assess and guarantee (if ever possible) their usability?
- GE6: *Critical infrastructures* – how to ensure the resilience of a critical infrastructure and more specifically to the usability domain, how to ensure that the procedures to manage and operate such infrastructures will be understandable and applicable in any context of use?

The next three gap descriptions specifically address the issue of resilience of context-aware and context-sensitive applications.

- GE2: *Resilient ambient systems* – how to ensure the resilience (with an emphasis on fault tolerance and dependability)?
- GA9: *On-line assessment for resilience* – how to assess at runtime the level of resilience of a system (issues such as how to inform the users of the evolution of that level belong to such research challenge)?
- GA8: *Evaluation of Dynamic Systems* – how to setup experiments for evaluating the resilience of such applications?

4- Research challenges requiring to go beyond standard usability

The fact that ReSIST addresses large scale, ubiquitous, dependable and evolvable systems raises issues that are not central to the field of usability engineering, in which research as mainly contributed to (for the last 15 years) desktop and mono user applications. The situation in the field is evolving, as there is a growing interest in the usability of non-standard systems such as mobile systems (which is the focus of the yearly Mobile HCI conference) or entertainment related systems (which is the focus of the yearly conference on Advance in Computing Entertainment for instance).

The following research gap descriptions relate to specific types of application that are of particular interest in ReSIST, making explicit research challenges in usability that are usually not entirely addressed in standard usability.

- GE2: *Resilient ambient systems* – addresses usability aspects of ambient systems
- GA8: *Evaluation of Dynamic Systems* – addresses usability aspects of mobile and ubiquitous systems
- GA16: *Modelling Effect of Micro-decisions In The Whole System* – addresses usability aspects of decision support systems

The following research gap descriptions relate to the complexity of the applications that are addressed by ReSIST. Indeed, such complex application usually involve various users, various organisations and various regulations;

- GE10: *Complexity and Self-organisation* – raises the issue of usability of systems that are capable of reorganisation according to external constraints
- GA18: *Inter-organisation boundary failures* – points out research challenges related to user behaviour when dealing with events involving system failures belonging to various organisations and how the procedures they have to follow (when they exist) can be usable when designed by several organisations. Similar considerations but more central to user behaviour are the focus of GD4: *Human Diversity and Human-machine Diversity* – which points out research challenges related to a vast number of users with large diversity.

Several contributors have identified research challenges directly targeting the usability research community by providing a research agenda made of three research gaps descriptions:

- GU1: *Usability metrics* – points out the usability research challenges related to definition and validation of usability metrics for large scale, ubiquitous, dependable and evolvable systems
- GU5: *Usable privacy* – points out the usability research challenge related to the notification of system privacy status to its users and to the definition (by the users) of a desired level of privacy
- GU6: *User Experience* –while usability focuses on the performance of activities by operators (and on ways of assessing and improving such performance) user experience focuses on the feelings of the operators (including flow, pleasure, satisfaction, …). User experience is a combination of established disciplines including psychology, anthropology, computer science, graphic design and industrial design. The research challenge of this gap lies here, in establishing user experience as a design discipline in the broader application context of ReSIST.

5- Summary of relationships between gap descriptions, clusters and usability

Figure 1 provides a graphical representation of the gaps with respect to the clusters highlighted in this summary: *development process*, *contextual usability* and *beyond standard usability*.

As can be seen in the figure, 21 of the gaps presented in this deliverable are directly or indirectly related to usability. The gaps directly related to usability [GU1 - GU6] are represented in white rectangles with dashed-lines. These gaps have been proposed by authors contributing directly to the research field of usability and cover the current burning issues in the field.

Gaps represented in dark grey with a plain border have been allocated in this deliverable to other resilient scaling technologies than usability but make explicit reference to usability issues in their descriptions. They are all directly connected to a specific cluster such as GA1, GA17 and GE13 to development process, with the exception of *Resilient Ambient System* [GE2] that raises issues related to the three clusters. Indeed, ambient systems make explicit use of context and their design and evaluation goes beyond current state of the art in usability engineering.

Gaps represented in light grey with doted-lines do not make explicit reference to usability in their descriptions but the research challenge they describe embeds research issues in the field of usability.

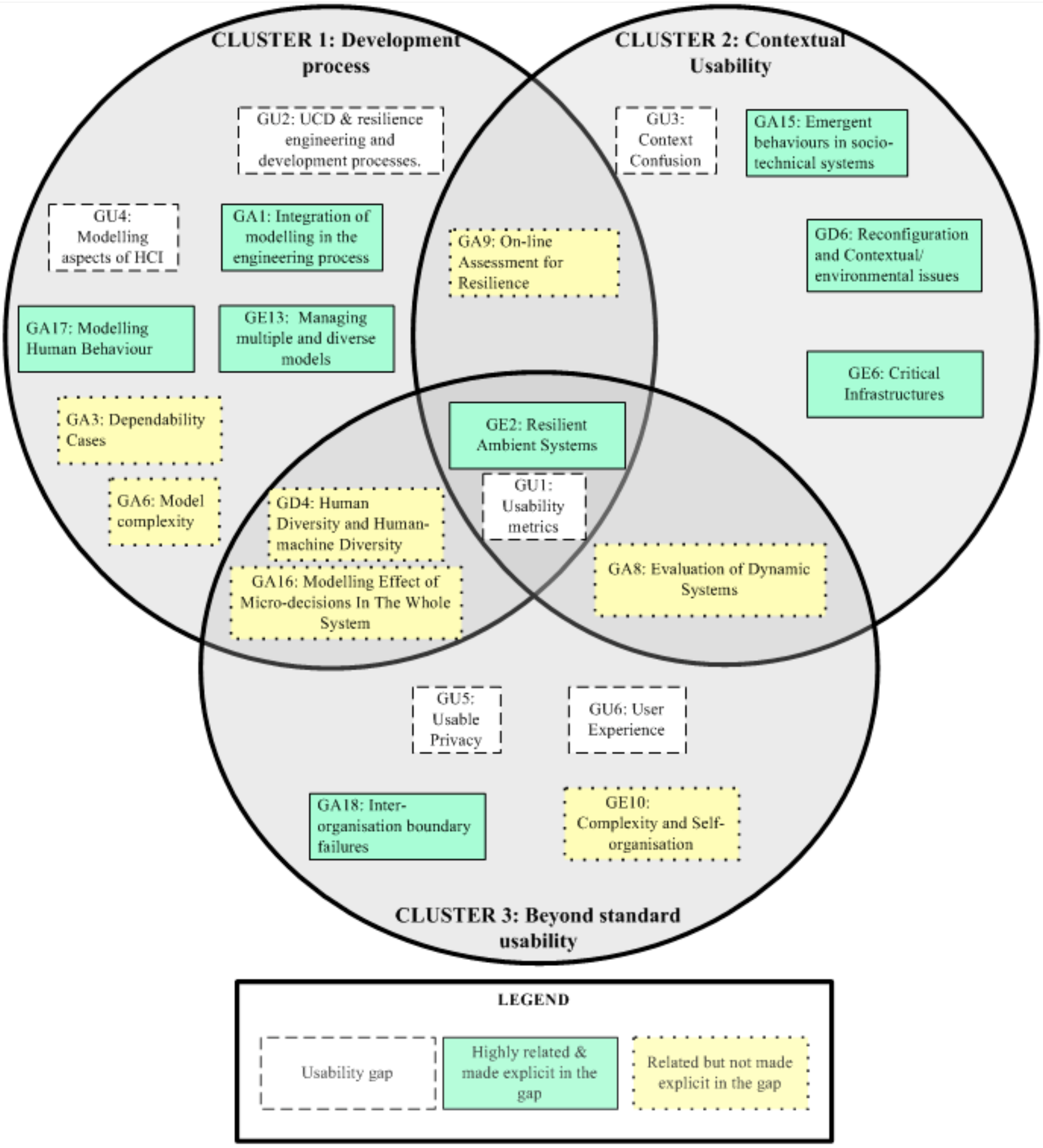
CLUSTER 1: Development process
CLUSTER 2: Contextual Usability
GU2: UCD & resilience engineering and development processes.
GU3: Context Confusion
GA15: Emergent behaviours in socio-technical systems
GU4: Modelling aspects of HCI
GA1: Integration of modelling in the engineering process
GA9: On-line Assessment for Resilience
GD6: Reconfiguration and Contextual/ environmental issues
GA17: Modelling Human Behaviour
GE13: Managing multiple and diverse models
GE6: Critical Infrastructures
GA3: Dependability Cases
GE2: Resilient Ambient Systems
GA6: Model complexity
GD4: Human Diversity and Human-machine Diversity
GU1: Usability metrics
GA8: Evaluation of Dynamic Systems
GA16: Modelling Effect of Micro-decisions In The Whole System
GU5: Usable Privacy
GU6: User Experience
GA18: Inter-organisation boundary failures
GE10: Complexity and Self-organisation
CLUSTER 3: Beyond standard usability
LEGEND
Usability gap
Highly related & made explicit in the gap
Related but not made explicit in the gap

Figure 1. Overlaps and complementarities between gap descriptions

6- References

[Constantine & Lockwood 2002] Constantine L., Lockwood L. Usage-Centered Engineering for Web Applications. IEEE Software, 19 (2), March/April 2002.

[Kruchten 1998] Kruchten P. The Rationale Unified Process – An Introduction. Addison Wesley, Longman Inc. Reading, Mass., USA.

[Nielsen, 1993] Nielsen, J. (1993) Usability Engineering. Morgan Kaufmann. Computer Bks / General Information. 358 pages. ISBN 0125184069

Diversity

Co-ordinator: *Lorenzo Strigini*[1]

Contributors: *Eugenio Alberdi*[1]*, Peter Ayton*[1]*, Christian Cachin*[3]*, Miguel Correia*[5]*, Marc Dacier*[6]*, Ilir Gashi*[1]*, Philippe Palanque*[2]*, Peter Popov*[1]*, Vladimir Stankovic*[1]

[1]City University, [2]IRIT, [3]IBM, [4]LAAS-CNRS, [5]University of Lisbon, [6]Eurecom

We start this summary by recalling the importance of diversity in making redundancy effective towards resilience and dependability (and the meaning of the word "diversity" in this context), and why this importance is bound to increase with the trend towards more "ubiquitous" ICT-intensive systems. We then discuss research challenges that have to be addressed for diversity to be used effectively towards resilience in ubiquitous systems. These challenges are described in more detail in the short papers in Part 2, "Gaps and Challenges", of this document.

1- Introduction: redundancy and diversity

Dependability and resilience normally require redundancy, to protect a system and the users (including other systems) that depend on its services against failures of its components (or at least the weaker components). If we understand "resilience" to include robustness against mishaps that were unforeseen at design time, redundancy is especially important for resilience: these "events outside the design base" must include that system components may fail, or be subverted, in ways that were intended to be prevented by their design. Redundancy offers the potential for tolerating broad sets of failure behaviours without a need for anticipating them in detail at component design time.

The effectiveness of redundancy is limited by how often the components or tasks that are supposed to provide protective redundancy for one another – for instance, performing the same task in parallel, or a component detecting errors in, or providing a fall-back function for, another one – fail instead together, so that the failure will affect the system of which they are part, i.e., so that their redundancy is ineffective in that particular event. When we take existing components and combine them in a redundant fashion, we are interested in the "diversity of failures" between them, i.e., in how unlikely they are to fail together in such a way. So, failure correlation is the enemy; "diversity" is an informal name for its opposite. "Diversity" is also (somewhat confusingly) the name used for the factors that tend to reduce failure correlation, and especially for those factors that a designer can control. Designers pursue "diversity of failures" by seeking "diversity" in the causes of failures; they may try to diversify:

- The stresses to which the components are subjected (e.g., geographical dispersion diversifies exposure to weather hazards, fires etc; making redundant embedded computers obtain data from separate sensors reduces the chance of their both reading the same sequence of inputs, which may be the one that triggers a software bug);

- The vulnerability of the components to such stresses (e.g., by using internally diverse components to perform similar functions in a redundant architecture). In turn, this may be pursued by diversifying the processes that produce the components and their vulnerabilities, e.g., the development processes of redundant software components.

Development methods and designs patterns for pursuing diversity in embedded, self-contained, closed, critical computer systems are well established, at least in some development organizations [Strigini, 2005]. Research under the heading "diversity" has focused on architectures based on component replication: those in which redundancy consists of few "redundant channels", i.e., components performing roughly functionally equivalent roles, with their outputs compared or voted or used as "failovers" for one another (in software engineering research on diversity, channels that are made intentionally diverse are called diverse "versions"). Diversity research has been motivated by the application of such architectures in safety-critical computers, e.g. for flight control, railway interlocking, emergency shutdown for nuclear reactors, because assuring sufficient "diversity of failures" diversity in these architecture has been seen as problematic. Of course, concerns about "diversity of failures" among redundant components apply to all redundant architectures: it is implicitly sought whenever, for instance, an error detection mechanism is selected; but mechanisms that are not based on replication have generally been uncontroversial and the study of their coverage factors has been seen as a different research area.

The main, still actively researched research problem with diversity has been how to link the (observable, and measurable) attempts to diversify with the resulting diversity of failures (predicted, and as such subject to uncertainty), and more generally with achieved system dependability. As is generally the case with designing for extreme dependability, this link between efforts and results is still difficult to describe precisely, although research has given useful insights [Littlewood et al., 2001; Popov and Strigini, 2001; Popov and Littlewood, 2004]. The benefits of diversity are also studied in the social sciences (e.g. in the area of aggregation of judgment by multiple "judges" [Budescu and Rantilla, 2000]), but bridges between these studies and those in the ICT area for which they are of interest are still incomplete.

2- Diversity for ubiquitous ICT systems

In this ReSIST project report, we focus on challenges opened for the research community by the "scaling up" of ICT-rich systems, along various dimensions, towards "ubiquitous" systems. There is little doubt that this scaling up demands attention to diversity. Indeed, large-scale networks of ICT resources offer large amounts of "redundant" resources (i.e., computing or communication capacity) that increase, or might be used to increase, resilience. But the scaling up also magnifies the importance of common-mode and common-cause failures: economics dictate that these abundant resources are largely made up of many essentially identical copies of few types of off-the-shelf hardware and software components (and market forces may make these few types very few indeed). Thus they all share vulnerabilities to common stresses and shocks, unintended as well as malicious. In addition, the "scaling up" of systems includes more extensive interconnection and faster communication, allowing those failures that have a potential for propagation to propagate farther and more rapidly.

This potential for common failure at some level in ubiquitous systems (in hardware, software, human management) both reduces the effectiveness of using spare resources for redundancy, and increases the risk of large scale outages or failures of services supported by ICT infrastructures, perhaps over large areas and sets of service types (for instance, a massive internet outage would affect disparate kinds of businesses and probably propagate to other societal infrastructures). This risk comes from unintentionally occurring common threats (like the Y2K problem and the various similar problems with clock counter overflows, or the occasional large-scale, software caused outages in telecommunication networks) as well as malicious ones like massive denial of service attacks. Thus, pursuing diversity among the components of ubiquitous systems has two related purposes: more effective redundancy between these components in delivering any one service, and reduced risk of large scale failures across multiple services[1].

As mentioned earlier, past research about defending against common-mode failure through diversification has mostly addressed embedded systems for safety critical applications. These typically required non-networked computer systems; redundancy consisted of few (two to five, in most applications) "channels" performing functionally equivalent roles, with rare and tightly controlled modifications. More recently [Popov et al., 2000; Reynolds et al., 2002; Gashi et al., in print] the area of interest has extended towards applications with less demanding dependability requirements, where using multiple off-the-shelf "versions" may offer cost-effective dependability and security improvements over a single off-the-shelf component, for networked applications. The evolution towards "ubiquitous" systems extends the area of interest in multiple directions, bringing up new questions.

First of all, we can divide the open issues according to the "scale of diversity": the number of diverse "versions", or generally, diverse components in redundant configurations. As potential redundancy levels increase from having 2, 3, 4 ... redundant channels to having perhaps 10^2, 10^3, 10^4..., so does the scale of potential diversification: with 100 redundant channels, possible architectures may perhaps involve 50 copies each of 2 diverse versions, or perhaps 10 copies each of 10 versions, or even 100 diverse versions; these options clearly pose different practical problems in development and evaluation. The research challenges identified here can be divided into two main clusters according to this scale factor:

- **challenges in "small-scale" diversity**, posed by subsystems (or by aspects of system properties) that are especially important in ubiquitous systems, but involving *few* diverse versions of the components involved (although a ubiquitous system may contain *many* such "small-scale diverse" subsystems). These challenges can be approached, at least initially, using the modelling and statistical apparatus developed in previous research on diversity [Littlewood et al., 2001]. These challenges include: GD1 (Diversity for security), GD3 (Interoperability for diversity), GD4 (Human diversity and human-machine diversity), GD6 (Reconfiguration and Contextual/environmental issues);
- **challenges in "large-scale" diversity**, where the challenge is created, to a large extent, by this novel potential for large-scale diversity in new systems, and we can expect to need new modelling and measurement approaches. These include: GD2 (Large-scale Diversity for Intrusion Tolerance) and GD5 (Spontaneous Redundancy in Large Systems).

3- The research gaps

Beside the "scale of diversity", evolution towards ubiquitous system extends the aspects of diversity to be investigated in multiple other dimensions:

[1] Collective failures are often costlier than if the same amounts of failed services or down-time were spread in a sparse pattern over space and time. Arguably this is just a larger-scale issue of common-mode failure defeating redundancy. If my bank is not working I can bank in another bank or obtain some bank-like services from other businesses; if a town's electricity distribution suffers a massive outage, engineers can be summoned from other towns to help. Massive outages overload these society-wide redundancy mechanisms.

- The threats of concern: bugs that are accidentally triggered remain a concern, but the importance of tolerating malicious attacks increases greatly due to the openness and interconnectedness of these systems;
- The types of components concerned, which extend to including people in addition to engineered components, since ubiquitous systems are tightly interwoven with organisations, with people connecting, and connected by, computers;
- The extent to which diversity is *designed* –pursued under the direct control of a designer or design manager– as opposed to *spontaneous* –available for free, but not necessarily suitable for exploitation. Over time, the trend has been towards less control but potentially more chances for exploiting spontaneous diversity. In fully purpose-built safety critical applications in the 20th century, system designers could dictate the specifications of each version and the way it should be developed; with a purpose design using off-the-shelf components, system designers can still select the components (from a market-determined set) in view of ensuring effective diversity; in ubiquitous systems, with peer-to-peer aggregations of web services (which are provided as components by independent organisations that control their evolution, while the aggregations themselves can undergo continuous changes and re-negotiations), the designers of a service may need to exploit the amount of diversity that makes itself spontaneously available, as little or perhaps large as it may be.

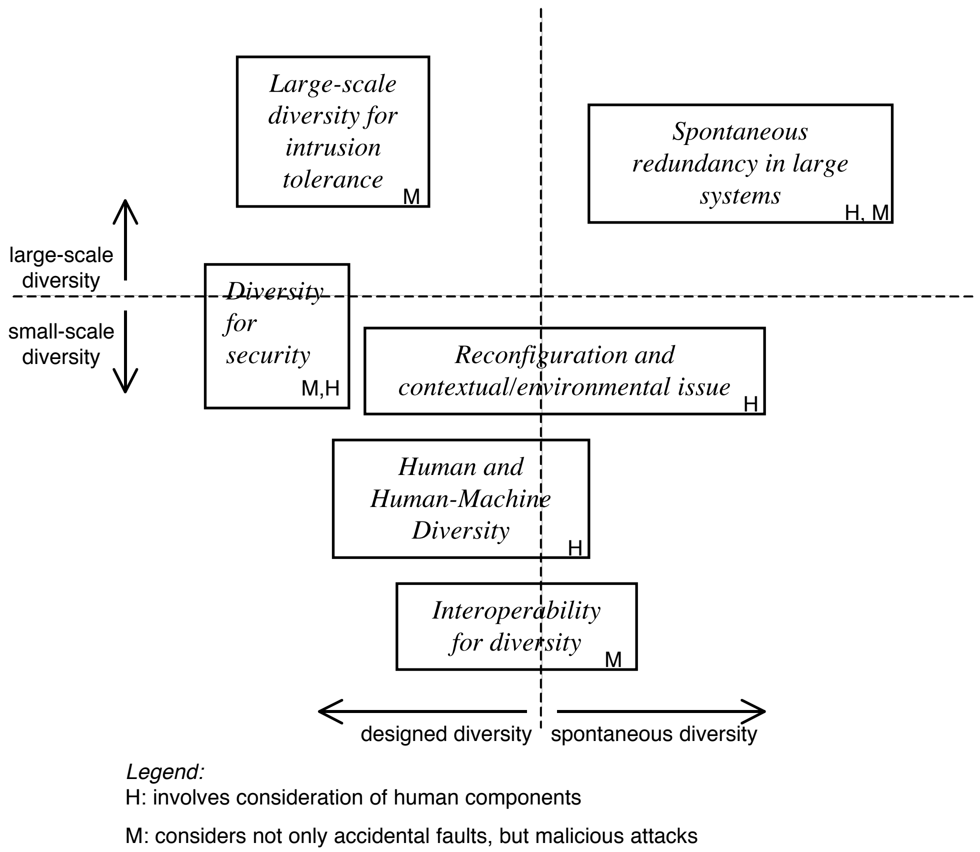
Large-scale diversity for intrusion tolerance M
Spontaneous redundancy in large systems H, M
large-scale diversity
small-scale diversity
Diversity for security M,H
Reconfiguration and contextual/environmental issue H
Human and Human-Machine Diversity H
Interoperability for diversity M
designed diversity
spontaneous diversity
Legend:
H: involves consideration of human components
M: considers not only accidental faults, but malicious attacks

Figure 2 Dimensions of expanding concerns for research in diversity

This set of dimensions of expanding concerns helps to clarify the relative position of the "research challenges" discussion essays by the various contributors in Part 2. If we consider each "dimension" as a binary variable (small scale vs large scale; not considering malicious attacks *vs.* considering them; etc), their combinations identify 16 research areas, of which only one (accidental only faults, computer-only, small-scale, designed diversity) has been intensively explored by previous research on diversity; the essays included here in Part 2 identify six subsets in this space, shown in the figure below according to the four dimensions just described (two represented in the Cartesian plane, and two by additional labels), each subset being characterised by a specific dominant "dimension of expansion", or interested research/application community, or pressing application problem.

Two of the seven essays are concerned with the *threat* dimension: extending concerns from accidental to malicious faults:

Diversity for security as a general topic has been discussed for at least 20 years, but still the contributions to the discussion and research have not created a coherent picture of the potential advantages, the suitable techniques and their effectiveness. The essay here identifies, as essential challenges that still exist even for dealing with only technical components and small-scale diversity, the need to understand (to take them into account in design and assessment): the effect of collusion and insider attacks; the effect of the various ways one can pursue diversity in the development of the diverse components.

Large-scale diversity for intrusion tolerance, starting from the same frame of concerns as the previous essay, addresses the specific problem of engineering "large-scale" diversity to contain the fraction of computers in a large co-ordinated set that a co-ordinated attack can compromise at once. Here the challenges identified concern both extending the set of automated, randomised diversity-creating methods (e.g. address randomisation, or more generally code obfuscation) now existing, and assessing the effectiveness of all (known as well as new) methods.

The other essays are less concerned with the origin of faults, but rather with the other dimensions above.

Interoperability for diversity addresses very concrete and immediate engineering problems found in designing systems to exploit existing diversity between off-the-shelf components: more "spontaneous" diversity than addressed by previous research, and limited, for the time being, to "small scale" diversity among non-human components. Development costs would recommend that designers choose their diverse "versions" mostly among existing off-the-shelf components; but these are not functionally equivalent, "plug-in replacements" for one another, as would be the case with purpose-built diverse versions of a component. The challenges concern supporting designers in their various decisions in the process of choosing, matching and integrating off-the-shelf diverse versions of components: choosing "compatible enough" components, and selecting appropriate "gluing" to achieve sufficient resulting dependability and performance

Human Diversity and Human-Machine Diversity is motivated by the need to consider human as well as technical components. While the mathematical modelling of small-scale diversity for technical components is well developed, considering humans brings in the challenges of considering their heterogeneity (two systems that are nominally implementations of the same design may behave very differently if they include human components who happen to be different people) and the fact that their failure behaviour evolves with these people's evolving experience of the machines' dependability. Thus the challenges are both in expanding the mathematical modelling apparatus, and in applying (and, to a large extent, creating anew) appropriate knowledge in the psychology and sociology of behaviour in the contexts of interest[2].

[2] An interesting twist and longer-term research need is created by research on “affective computing", the emerging, fast growing interdisciplinary field studying how to make computers recognize, express, synthesize and model emotions, to better interact with people, so that computers' behaviour and failure patterns may co-evolve with those of their human users.

Spontaneous redundancy in large systems focuses on the broader potential and problems with spontaneous, potentially large-scale redundancy, potentially including both technical and human components. The challenges identified pertain to extending the mathematical modelling of diversity towards using large-scale statistical parameters to characterise diversity among many components, evolving from the existing models that use detailed descriptions for few components; as well as exploratory and empirical research to characterise the range of forms of spontaneous diversity to be considered, and research in distributed algorithms to take opportunistic advantage of it when present.

Reconfiguration and contextual/environmental issues again deals with concrete and immediate engineering issues but with respect to an important class of fault-tolerant architectures involving technical and human components: that involving *multi-modal* interfaces between humans and machinery. These interfaces have potential for efficiency and resilience, but while their design has been studied in terms of generic usability issues, the diversity issues determining their potential for dependability and resilience are an open, current research area that needs addressing.

Last, it may be useful to outline links between diversity-related questions and other questions discussed in this document with respect to the other "scalability properties".

Several of the issues discussed here have links to others under the "Evolvability" heading; indeed, all diversity topics have a dimension of "evolvability" problems: in the case of large-scale and spontaneous diversity, which depend on resources (and threats) whose autonomous evolution affects the degree of diversity available, but also of small-scale diversity, especially regarding changes of human behaviour. Looking now at Assessability, the study of diversity addresses a specific subset of concerns about both *constructing* and *assessing* systems, so challenges described in this chapter are special cases, with their own difficulties and possible solutions, of problems that also appear under "assessability". Thus, for instance, the difficulties in dealing with human components of systems, and with increased complexity, present here with both "small scale" and "large scale" diversity, are similar to those discussed under "Research Challenges of Assessing Evolvable Systems". The concerns about diversity for security relate to the general concerns about quantification of security. And diversity is both a technique used for, and a study object for, "Assessability as an Engineering Discipline". Last, the issues discussed here about the human components of systems are inevitably related to Usability, mostly to the issues raised in the "Context" cluster of usability topics.

4- References

[Budescu and Rantilla, 2000] Budescu, D.V. and Rantilla, A.K. Confidence in aggregation of expert opinions. Acta Psychologica, vol. 104, no. 3, 2000, pp.371-398.

[Gashi et al., in print] Gashi, I., Popov, P. and Strigini, L. Fault Tolerance via Diversity for Off-The-Shelf Products: a Study with SQL Database Servers. IEEE Transaction on Dependable and Secure Computing, in print. http://doi.ieeecomputersociety.org/10.1109/TDSC.2007.70208

[Littlewood, Popov et al., 2001] Littlewood, B., Popov, P. and Strigini, L. Modelling software design diversity - a review. ACM Computing Surveys, vol. 33, no. 2, 2001, pp.177-208.

[Popov, Strigini et al., 2000] Popov, P., Strigini, L. and Romanovsky, A. Diversity for off-the-Shelf Components. In Proc. DSN 2000, International Conference on Dependable Systems and Networks - Fast Abstracts supplement, New York, NY, USA, IEEE Computer Society Press, 2000, pp. B60-B61.

[Popov and Strigini, 2001] Popov, P. and Strigini, L. The Reliability of Diverse Systems: a Contribution using Modelling of the Fault Creation Process. In Proc. DSN 2001, International Conference on Dependable Systems and Networks, Goteborg, Sweden, 2001.

[Popov and Littlewood, 2004] Popov, P. and Littlewood, B. The effect of testing on the reliability of fault-tolerant software. In Proc. DSN 2004, International Conference on Dependable Systems and Networks, Florence, Italy, IEEE Computer Society, 2004, pp. 265-274.

[Reynolds, Just et al., 2002] Reynolds, J., Just, J., Lawson, E., Clough, L., Maglich, R. and Levitt, K. The Design and Implementation of an Intrusion Tolerant System. In Proc. DSN 2002, International Conference on Dependable Systems and Networks, Washington, D.C., USA, 2002, pp. 285-292.······[Strigini, 2005] Strigini, L. Fault Tolerance Against Design Faults. In Dependable Computing Systems: Paradigms, Performance Issues, and Applications,, (H. Diab and A. Zomaya, Eds.), pp. 213-241, J. Wiley & Sons, 2005

.

Part 2- Gaps & Challenges

Evolvability

GE1 - Evolution Of Threats

1- Definition

This gap is concerned with socio-technical aspects of system evolution and the potential for unpredicted and unexplored vulnerabilities. These vulnerabilities or threats may be a result of malicious attack or through system failure as a result of unexpected confluences of circumstances. There are two aspects to this problem. The first is concerned with the fact that threats themselves evolve because attackers are actively involved in the development of new techniques to inject and, or, activate latent faults in existing systems. In other words, they are discovering unknown vulnerabilities rapidly and are launching new types of attacks to take advantage of new, but also old, vulnerabilities. The other problem concerns the evolution of the system itself and the potential for combinations of circumstances that were unexpected and unpredicted. These situations emerge through the complexity of the system and derive from circumstances that may not have been envisaged in any analysis process based on decomposition and arguments about components. These situations may also arise as a result of the system's evolution - for example workarounds may lead to situations which were not envisaged initially. This gap relates to Objective ICT-2007.8.2: FET proactive 2: "Pervasive adaptation" (Dynamicity of Trust) "capabilities for establishing trust relationships between humans and/or machines that jointly act and interact with ad-hoc and changing configurations".

2- Examples / Application Domains

Internet security is a clear application domain of the first problem where a large population of people are discovering and launching new attacks on, almost, a daily basis.

Examples of this phenomenon occur in military contexts. The "friendly fire" situation may occur because the agents involved encounter a situation for which they were not trained or was unpredicted or may occur because opposing forces have devised techniques that confuse (see [Leveson et al 2002] for example).

Systemic problems can arise in temporary groupings where there are transient relationships of trust that may involve certain information but not all information. For a virtual organisation such as NATO that combines many national forces.

3- Current approaches

Threat evolution

Threats may be monitored as the system evolves thereby tracking and anticipating new techniques for exploiting vulnerabilities.

- Several initiatives exist to discover the manifestation of new attacks on the internet by monitoring various kinds of systems (darknets, internet telescopes, honeypots, honeynets, honeytoken, netflow monitors, intrusion detection consoles, etc.).
- Another approach tries to obtain some information about changing threats even before the existence of new attacks by closely following the activities and discussions within the so-called underground.

This enables observers to be aware of the fact that, e.g., some groups are actively trying to find new vulnerabilities against software X or Y, and that they are close to find something. This form of intelligence is sometimes carried out thanks to monetary incentives where observers are paying for information about, yet, not widely known attack or vulnerability.

Analysing the system

Specific work on understanding and analysing the coalitions and defining the policies that describe who is included in a coalition and how new parties can enter or leave is discussed in [Bryans et al, 2006].

4- Research challenges

As discussed before, threats faced by computing systems do evolve either because of the dynamics of the threats themselves or because of the dynamics of the system and of its environment. This is a complex system and cannot analysed completely by using decompositional techniques.

- What impact does this have on the way we do look at the resilience of our systems and how can we identify the effects of changes? What system models and argumentation techniques are required?
- Assuming we detect an evolution, what new mechanisms do we need to respond to these new threats? Anti virus propose regular, almost daily, updates of their signature files but how does this scale to complex systems? In many cases, you just can not have the system evolve as quickly as the threats without running the risk of creating unwanted instabilities. Is there a new avenue to deal with these issues?
- Security has been seen as a binary property: a system is secure or not secure. Security policies have classically being defined under this principle. If we accept the idea that there is a continuum between secure and insecure system, i.e., normal state vs. failure state then Dynamic policies have to be envisaged that can react to observed evolving threats, new requirements, or change in system configuration, or any other relevant contextual information. Such policies can only be possible if one can evaluate the efficiency of the protection mechanisms coverage with respect to these evolving parameters. How to correlate these evolutions with security policies? How to respect a minimal set of security properties? This implies that we are able to carry out the following tasks:
    - Describing the complex socio-technical system from this point of view
    - Describing the policies including the human components
    - Modelling the evolution of such systems
    - Demonstrating that the socio-technical system satisfies a policy and that this is invariant with respect to different notions of evolution

5- References

[Leveson et al. 2001] Leveson, N.A., Allen, P. & Storey, M.A. The Analysis of a Friendly Fire Accident using a Systems Model of Accidents Proceedings of the 20th International System Safety. 2001

[Bryans et al. 2006] Bryans, J. W., Fitzgerald, J. S., Jones, C. B., Mozolevsky, I. Formal Modelling of Dynamic Coalitions, with an Application in Chemical Engineering, Presented at the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA), Paphos, 15-19 November 2006.

GE2 - Resilient Ambient Systems

1- Definition

Decreasing size and costs of computing devices have enabled a steady transition from the many-people-per-computer situation of the 1970's to the several-computerized-devices-per-person situation of today. In the near future, it can be expected that there will be hundreds or thousands of computerized devices per person. This technology trend has led to a considerable amount of very active research into areas such as pervasive/ubiquitous computing, wearable computing, personal area networking, wireless sensor/actuator networks, etc., which we collective designate as ambient computing systems. The key characteristic of such systems is that computing and communication facilities are embedded in physical objects of everyday life. Objects so equipped or "devices" can communicate between themselves and/or with a fixed infrastructure.

This gap is thus concerned with dynamic distributed systems of intelligent objects. Evolvability in this context is essentially the ability of such systems to be resilient to mobility- and failure-induced changes to their composition and/or topology.

2- Examples / Application Domains

There are many existing or future applications for ambient computing systems, ranging from incremental extensions of nomadic computing and cellular phone systems, through domestic applications (home entertainment, games, smart homes, assisted living support systems for elderly or handicapped persons...), business applications (inventory control, remote maintenance...), transport applications (fleet management, toll collection and ticketing, contextual passenger information services...), ecological applications (waste recycling, environmental monitoring and control...), to futuristic distributed robotics and ambient intelligence scenarios [ISTAG 2001, OFTA 2007].

Ambient computing systems are not necessarily distributed over a wide geographical area. One can also imagine ambient computing systems built from hundreds or thousands of smart communicating objects distributed within a very small area, such as networked drug release microsystems (a foreseeable evolution of current work on non-networked drug release microsystems [Hu & Zengerle 2006]), or, why not, swarms of "nanorobots" or "nanites" [Stepney et al. 2004].

A socially-important application area of ambient systems is ambient assisted living, i.e., providing health care services to (typically elderly) people living in their own environment. Focusing on this aspect of ambient systems, the corresponding solutions should be human-oriented and their operation should induce as little inconvenience to their users as possible. Recently a European technology and innovation funding program "*Ambient Assisted Living*" (AAL) was initiated in this area (http://www.aal169.org/). The program is intended to address the needs of the ageing population, to reduce innovation barriers of forthcoming promising markets, but also to lower future social security costs. AAL aims - by the use of intelligent products and the provision of remote services including care services – at extending the time older people can live in their home environment by increasing their autonomy and assisting them in carrying out activities of daily living.

3- Current approaches

Whereas in the early days of ambient computing research, most of the challenges were related to technological issues such as displays, low-power processors, high-capacity memories and wireless standards, the current challenges are more at the system level. For example, [Want & Pering 2005] lists power management (or power-aware computing), wireless discovery techniques for semi-anonymous objects, dynamic adaptation of user interfaces, and location- or context-adaptation as major current research issues.

Another major current research issue is the dependability of ambient systems, which is indeed essential if such systems are to be "invisible" in Weiser's sense [Weiser 1993]. Users need to be confident that the confidentiality and integrity of critical data and services is preserved, that the technology is not abused to infringe on their privacy, and that the functionality they are intended to provide is available when and where needed. This latter point is also made in the final lines the conclusion of [Want & Pering 2005], which stress the importance of "automatic maintenance" if the computation must effectively be hidden from view of non-expert users. Here, we focus on resilience techniques for ensuring availability in ambient systems.

Since physical devices are potentially mobile, the composition of ambient systems can evolve continuously. Ambient systems are thus dynamic systems, consisting of a potentially large and evolving set of participants subject to frequent partitioning and changing membership. In dynamic systems, mobility and failure are often indistinguishable.

Research on resilience in ambient systems can be structured according to at least two broad types of system:

- *Smart spaces*: a smart space is a physical space equipped with sensors, actuators and a ubiquitous networked infrastructure with which and through which mobile devices can interact;
- *Smart populations*: a smart population is a set of mobile devices that can interact autonomously, without access, or with only intermittent access, to a fixed infrastructure.

Since a smart space possesses a static fixed infrastructure, critical services can be implemented within that infrastructure using classic fault-tolerance techniques. However, smart spaces must still deal with the dynamics of arrivals, departures (and failures) of mobile devices in the infrastructure's coverage area. Another characteristic of smart spaces is that devices considered can be very heterogeneous in their functionality, typically leading to resource scarcity rather than resource redundancy [Kindberg & Fox 2002]. In such a context, resilience techniques are relevant to:

- Ensuring that logical consistency of the smart space is maintained, and
- Attempting to carry out useful work,

despite the system dynamics and device heterogeneity.

Specific techniques include: event-based communication (publish/subscribe), data-centric organization (separation of data and function) and so-called "soft-state" data maintenance (refresh before deadline or invalidate) [Ponnekanti et al., 2003, Grimm et al. 2004].

A particularly original approach to ensuring that useful work is done despite system dynamics and device heterogeneity is that followed in the Gaia smart space [Ranganathan & Campbell 2004], which implements what can be seen as a generalized forward recovery scheme based on planning: users define goals (final states) that the smart space strives to meet by scheduling actions to be executed by the current but dynamically-changing set of available heterogeneous devices.

The notion of a smart population portrays ambient systems composed of a possibly large set of functionally similar devices (or nodes) that implement services cooperatively by direct (wireless) interaction in a fully decentralized manner. Smart populations may be dense or sparse, according to whether it is possible to assume the existence of a majority connected component or, on the contrary, the system is often, or even continuously, partitioned into minority components.

Cooperative services implemented by direct interaction between members of a smart population, without assistance from a fixed infrastructure, are of interest in applications where a fixed infrastructure is currently unavailable (e.g., overloaded due to a crisis situation) or infeasible (e.g., smart micro/nano-devices with very short-range communication, or battlefield scenarios). For example, the nodes of Wireless Ad Hoc Networks cooperate with each other in order to provide a meshed communication service that reaches beyond the range of any single node.

Resilient services other than communication have been studied in the smart population context. For example, there has been work on replicating data in mobile ad hoc networks [Wang & Li 2002] [Gianuzzi 2004] [Hara & Madria 2006]. Recent research by ReSIST members has proposed a cooperative data backup service for smart populations with intermittent infrastructure connectivity [Courtes et al. 2006]. However, going beyond data backup to implement truly replicated services in a smart population is a significant challenge. Unlike static distributed systems, where partitioning is rare, in a smart population, the set of replicas must continuously and dynamically adapt to the participants that are currently able to host replicas.

4- Research challenges

In the heterogeneous world of smart spaces, research is needed to help structure resilient services based on forward recovery. The goal-based approach of [Ranganathan & Campbell 2004] needs to be studied in other application contexts than the archetypical smart space considered in that paper (a smart conference room). Forward recovery techniques need to be considered in the context of domain-specific languages for engineering software for smart space applications (e.g., [Consel et al. 2007]).

In the homogeneous world of smart populations, there is much scope for research into implementing truly replicated services. This includes, for instance:

- How to migrate replicas to match the available participants;
- How to ensure atomic updates of a dynamic set of migrating replicas;
- How to define appropriate policies for dealing with momentary redundancy exhaustion (insufficient local participants in a sparse dynamic population).

An additional challenge is the provision of resilient "localized" services using participants passively subject to mobility (as opposed to participants whose mobility can be actively controlled). The idea is to be able to provide the service in a given place and a given time despite failures and uncontrolled mobility of participants (a sort of "virtual" smart space). When no participants are available in the right place, at least two policies can be envisaged:

- Priority to location/trajectory (the service is restarted from a default initial state as soon as there are potential hosts available in the right place) (cf. [Dolev et al. 2004]);
- Priority to service state (locality constraints are temporarily abandoned in favor of maintaining state of the service).

Ambient assisted living provides a potentially interesting application of both smart space and smart population concepts. In this area, the following issues are to be addressed:

- As an ambient assisted living solution may consist of a large number of small devices embedded in various objects without robust power sources, these devices should probably feature low power consumption and potentially should be capable of long term battery operation.
- As ambient devices are to be used in the original home environment of patients, the embedded systems may not rely on high bandwidth, reliable fixed network infrastructure; the necessary communication should be carried out using wireless facilities.
- As devices may be attached to the body of their user, the devices should be capable of operating in ad-hoc and possibly rapidly changing wireless network environment.
- In case of health care devices, the entire system should be capable of integrating the most modern devices to the already existing or legacy facilities (e.g., the integration of a new blood pressure meter should seamless to some already used sensor devices).
- The maintenance of the possibly large number of devices should be centralized and should not require explicit human interaction (e.g., the most recent driver software of medical appliances should be downloaded through the wireless network from the health care center without any interaction of the user).
- The user interface of these devices should be intuitive: users are neither experts of medical appliances nor IT specialists. The reliable and beneficial application of these devices should not require a major effort.

5- References

[Consel et al. 2007] C. Consel, W. Jouve, J. Lancia, and N. Palix. Ontology-Directed Generation of Frameworks For Pervasive Service Development. In Proceedings of The 4th IEEE Workshop on Middleware Support for Pervasive Computing (PerWare'07), March 2007.

[Courtes et al. 2006] L. Courtès, M.-O. Killijian and D. Powell, "Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices", in 6th European Dependable Computing Conference (EDCC-6), (18-20 October, Coimbra, Portugal), pp.129-38 IEEE CS Press, 2006. _ [Dolev et al. 2004] S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvartsman and J. Welch, "Virtual Mobile Nodes for Mobile Ad Hoc Networks (extended abstract)", in Distributed Computing (DISC), (October, Amsterdam, Netherlands), LNCS, 3274, pp.230-44, Springer, 2004.

[Gianuzzi 2004] V. Gianuzzi, "Data Replication Effectiveness in Mobile Ad-Hoc Networks", in 1st ACM Int. Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubitiquitous Networks (PE-WASUN'04), pp.17-22, ACM, 2004.

[Grimm et al. 2004] R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble and D. Wetherall, "System Support for Pervasive Applications", ACM Transactions on Computer Systems, 22 (4), pp.421-86, November 2004.

[Hara & Madria 2006] T. Hara and S. K. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks", IEEE Transactions on Mobile Computing, 5 (11), pp.1515-32, November 2006.

[Hu & Zengerle 2006] M. Hu and R. Zengerle, "Microfabricated Devices for Controlled Biochemical Release", in 10th Int. Conf. on New Actuators (ACTUATOR 2006), pp.263-71, 2006. http://www.imtek.de/content/pdf/public/2006/2006-06_controlled_biochemical_release.pdf

[ISTAG 2001] IST Advisory Group (ISTAG), Scenarios for Ambient Intelligence in 2010, ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf

[Kindberg & Fox 2002] T. Kindberg and A. Fox, "System Software for Ubiquitous Computing", IEEE Pervasive Computing Magazine, 1 (1), pp.70-81, January 2002.

[OFTA 2007] OFTA, Informatique diffuse, Arago, 31, Editions Tec & Doc, Paris, 2007.

[Ponnekanti et al. 2003] S. R. Ponnekanti, B. Johanson, E. Kiciman and A. Fox, "Portability, Extensibility and Robustness in iROS", in 1st IEEE Int. Conf. on Pervasive Computing and Communications (PerCom'03), pp.11-19, IEEE CS Press, 2003.

[Ranganathan & Campbell 2004] A. Ranganathan, R. H. Campbell, "Autonomic Pervasive Computing based on Planning", in Int. Conf. on Autonomic Computing (ICAC'04), pp.80-87, IEEE, 2004.

[Stepney et al. 2004] S. Stepney, S. L. Braunstein, J. A. Clark, A. Tyrrell, A. Adamatzky, R. E. Smith , T. Addis, C. Johnson, J. Timmis, P. Welch, R. Milner and D. Partridge, 2004, "Journeys in Non-Classical Computation - A Grand Challenge for Computing Research, http://www.cs.york.ac.uk/nature/gc7/journeys.pdf

[Wang & Li 2002] K. H. Wang and B. Li, "Group Mobility and Partition Prediction in Wireless Ad-Hoc Networks", in Int. Conf. on Communications (ICC), pp.1017-21, IEEE CS Press, 2002.

[Want & Pering 2005] R. Want and T. Pering, "System Challenges for Ubiquitous & Pervasive Computing", in 27th Int. Conf. on Software Engineering (ICSE), St Louis, MO, USA), pp.9-14, ACM, 2005.

[Weiser 1993] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, 36 (7), pp.75-84, July 1993.

GE3 - Distributed System Models

1- Definition

Recently researchers have been paying more and more attention to so-called dynamic systems [A04]. Basically named in the peer-to-peer literature, dynamic systems do not have an agreed and precise definition yet, but it is possible to recognize some features characterizing them, listed below [BBRT07, KLB02]:

- Full decentralization. A dynamic system is a fully decentralized system in which each entity plays the same role;
- Dynamicity. Dynamicity concerns a possible continually changing membership of the very entities defining the system;
- Locality. Locality concerns how entities interact in the system. Generally speaking, the system may show a possible arbitrary size while each entity has a possible arbitrary small knowledge about the system it is member of. It is also said that this knowledge defines the entity's neighbourhood, i.e., the other entities an entity will directly interact and communicate with.

Understanding the very nature of these systems can help in defining properties and then algorithms working in this constantly changing environment. Evolvability in this context is essentially the ability of building distributed applications embedding algorithms which are able to keep deterministic guarantee during its lifetime despite limited knowledge, arbitrary size (potentially infinite in an infinite run) and unbounded number of failures (even this number potentially infinite in an infinite run). To do that, we have to understand which are the fundamental problems of coordination, synchronization, agreement and communication in this setting and under which constraint we can provide algorithms that can solve these problems.

2- Examples / Application Domains

The notion of dynamic distributed system given in the previous section abstracts quite well settings like P2P systems and sensors networks [BBRT07]. In such settings, a one-time computation (such as reliable broadcasting of an information, computing and aggregate one-time queries etc.) returns a valid result only if either the computation is restricted to a well defined subset of nodes of the system, as shown in [BGGM04], or, in the case the computation is carried out by all the nodes of the system, the valid outcome can only be returned in quiescent periods of the system i.e., when the dynamics of the system (joins, leaves and failures) ceases [TB07]. In both cases, it has been argued that a valid outcome of a computation depends on the capability of providing stable connections among the nodes involved in the computation either through all the time of the computation [BGGM04] or till a quiescent period of the system has been reached [TB07]. As an example, [BGGM04] shows that in large scale self-administered settings, the execution of a valid query can be done only for those nodes that during the interval of the query are both up and have a stable path (i.e., stable communication connection) with the source of the query.

3- Current approaches

Historically several models have been developed and their aim was to capture the very nature of the distributed system, each of these models is actually characterized by problems that can be solved (and which cannot) and by the constraints under which such problem find a solution.

Static reliable asynchronous distributed systems Static means that the number of entities is fixed. Reliable means that neither the entities nor the communication medium suffer failures. Asynchronous means that there is no particular assumption on the speed of the processes, or on message transfer delays. Moreover, the underlying network is usually considered as fully connected: any entity can send messages to, or receive messages from, any other entity (this means that the message routing is hidden at the abstraction level offered by this distributed computing model). An important result associated with this distributed computing model is the determination of a consistent global state (sometimes called a snapshot). It has been shown [CL85] that the "best" that can be done is the computation of a global state (of the upper layer distributed application) with the following consistency guarantees: the computed global state is such that (1) the application could have passed through it, but (2) has not necessarily passed through it. There is no way to know whether or not the actual execution passed through that global state. This is one of the fundamental facets of the uncertainty encountered in static distributed systems and that makes hard for example debugging distributed applications.

Static unreliable asynchronous distributed systems. The simplest static unreliable asynchronous model is characterized by the fact that processes may crash. The most famous result for this model is the impossibility to solve the consensus problem as soon as a process may crash [FLP85]. The impossibility to solve this problem comes from the net effect of asynchrony and failures. One way to solve consensus despite asynchrony and failures consists in enriching the asynchronous model with appropriate devices called failure detectors [CT96] (so, the resulting computing model is no longer fully asynchronous). Fortunately, problems simpler than consensus can be solved in this model. Let us consider the reliable broadcast problem [HT93] as an example. This problem consists in providing the processes with a broadcast primitive such that all the processes that do not crash deliver all the messages that are broadcast (while the faulty processes are allowed to deliver only a subset of these messages). Let a correct process be a process that never crash. This problem can easily be solved as soon as any two correct processes remain forever connected through a path made up of reliable channels and correct processes.

4- Research challenges

In the previous section we showed that when we proceed from the static reliable asynchronous distributed computing model to its unreliable counterpart, there are problems that can still be solved, while other problems become impossible to solve if asynchrony is not restricted. In the case of a dynamic distributed system, the varying size of the system (according to process joins and departures), and its "geography" captured by the notion of process neighbourhood pose additional problems, when someone wants to achieve reliable communication, agreement, coordination etc. The research challenge here is firstly to find an agreed definition of dynamic system model as the network and system communities did for synchronous/asynchronous reliable/unreliable distributed system. Assuming to have this agreed definition, an immediate macro difference between the dynamic model and the static ones is the communication underlying system. Static distributed system models implicitly assume the existence of a clique (every process is aware and can communicate directly with any other process). In a dynamic system the clique cannot be trivially kept due to concurrent join/leave in the systems, failures and locality. Based on this, all interaction paradigms have to be deeply revised and new solutions found for dependable and predictable computations. We believe that these solutions will be based on self organizing behaviours, however, how to formally define properties, algorithms, providing methodologies for proving their correctness is still a holy quest.

5- References

[A04] Aguilera M.K., A Pleasant Stroll Through the Land of Infinitely Many Creatures. ACM SIGACT News, Distributed Computing Column, 35(2):36-59, 2004.

[BBRT07] R. Baldoni, M. Bertier, M. Raynal, and S. Tucci Piergiovanni, Towards a Definition of Dynamic Distributed Systems, to appear in Proc. of 9th Int. Conference on Parallel Computing Technologies (PaCT'07), Springer Verlag LNCS, 2007.

[BGGM04] Bawa M., Gionis A., Garcia-Molina H. and Motwani R., The Price of Validity in Dynamic Networks. Proc. ACM Int'l Conference on Management of Data (SIGMOD), ACM Press, pp. 515-526, 2004.

[CT96] Chandra T. and Toueg S., Unreliable Failure Detectors for Reliable Distributed Systems. Journal of the ACM, 43(2):225-267, 1996.

[CL85] Chandy K.M. and Lamport L., Distributed Snapshots: Determining Global States of Distributed Systems. ACM Trans. on Computer Systems, 3(1):63-75, 1985.

[FLP85] Fischer M.J., Lynch N.A. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. Journal of the ACM, 32(2):374-382, 1985.

[HT93] Hadzilacos V. and Toueg S., Reliable Broadcast and Related Problems. In Distributed Systems, ACM Press, New-York, pp. 97-145, 1993.

[KLB02] D. Karger D. Liben-Nowell, H. Balakrishnan, Analysis of the Evolution of Peer-to-Peer Systems, Proceedings of the 21st ACM Annual Symposium on Principles of Distributed Computing (PODC02), pp. 233–242.

[TB07] S. Tucci Piergiovanni and R. Baldoni, Towards a Definition of Dynamic Distributed Systems, to appear in Proc. of 9th Int. Conference of Latin American Dependable Computing Conference (LADC'07), Springer Verlag LNCS, 2007.

GE4 - Trustworthiness/intrusion Tolerance In WANs

1- Definition

There has recently been a revival in the interest in applying the replication paradigm from fault-tolerance to the design of trustworthy, secure systems [Verissimo et al. 2006, Verissimo et al. 2003, Lala 2003]. This approach has been exploited under different names, ranging from trustworthiness, intrusion tolerance, survivability, to Byzantine fault tolerance and resilience-oriented computing. The general idea is to acknowledge that most computer systems are vulnerable to some extent and that some attacks will inevitably be successful, and to build in automatic mechanisms for preventing successful intrusions on components to destroy the overall system.

2- Examples / Application Domains

Critical Internet Services (e.g., Web, DNS, Certification Authorities).

3- Current approaches

Albeit this approach ultimately promises a new generation of large-scale, secure distributed systems, the research so far has mostly been focused on reasonably small-scale distributed systems. For instance, most protocols presented are optimized for local-area networks and few protocols scale well beyond a few dozen nodes, and are infeasible when run with tens of thousands of nodes.

4- Research challenges

We identify a gap in the design of evolvable scalable trustworthy protocols and systems, like those based on the Internet or, to put it in a more generic way, for wide-area networks (WANs). Some of the open research issues are as follows.

There is a clear need to better understand which applications or systems can benefit from the use of trustworthiness techniques, like state-machine replication [Schneider 1990] and Byzantine quorum systems [Malkhi and Reiter 1997]. Examples include services based on the client-server model, like DNS, Web and NFS, and collaborative applications, like computer-supported cooperative work. But there is a need to better generalize these examples to grasp in general which applications can be made trustworthy by using fault tolerance mechanisms.

Another aspect are Byzantine protocols, like atomic multicast and consensus, which have been presented but not optimized for large-scale systems with high and uncertain communication delays, large numbers of processes, temporary disconnections, crossing several administration domains, usually requiring quality of service properties like bounded delays.

The architectures proposed for enforcing trustworthiness are usually very simple. Most proposals are based on the client-server model and simply replicate the servers in several physical machines. For systems deployed on WANs, more complex architectures are needed, considering the above mentioned issues of high/uncertain delays, disconnections, etc., possibly supporting weaker consistency models that those normally considered.

Finally, systems on top of the Internet tend to be operational for long periods of time with a low acceptable downtime (think about Google, for instance). Therefore, there is a gap in research that allows trustworthy systems to evolve while operational, maintained their functional correctness and their trustworthiness.

5- References

[Verissimo et al. 2006] P. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, and I.Welch. Intrusion-tolerant middleware: The road to automatic security. IEEE Security & Privacy, 4(4):54–62, Jul./Aug. 2006.

[Verissimo et al. 2003] P. Verissimo, N. F. Neves, and M. Correia. Intrusion-tolerant architectures: Concepts and design. In R. Lemos, C. Gacek, and A. Romanovsky, editors, Architecting Dependable Systems, volume 2677, pages 3–36. 2003.

[Lala 2003] J. H. Lala (Ed.), Foundations of Intrusion Tolerant Systems, IEEE Computer Society Press, 2003

[Schneider 1990] F. B. Schneider. Implementing faul-tolerant services using the state machine approach: A tutorial. ACM Computing Surveys, 22(4):299–319, Dec. 1990.

[Malkhi and Reiter 1997] D. Malkhi and M. Reiter. Byzantine quorum systems. In Proceedings of the 29th ACM Symposium in Theory of Computing, pages 569–578, May 1997.

GE5 - Resilient Data Storage

1- Definition

The technologies and mechanisms to store data are changing. Data was previously stored on a storage system physically located near the processing system. Nowadays storage is a service often provided by remote entities, some without central administrative control. Information is a company asset, therefore its storage must comply with internal and regulatory requirements and cost optimization as well.

2- Examples / Application Domains

The most widespread solutions for storage systems are storage networks, which use specialized file systems that support distributed storage of data, and database systems. In addition to the traditional data storage functionality, the amount of data requires more complex and higher level services as well, such as content and information lifecycle management.

In the recent years, as connection speeds for Internet users were improving drastically, peer-to-peer networks have become more and more widespread and there is an increasing number of applications that use such systems to store and share data among a huge number of partners.

3- Current approaches

Today most data is stored in distributed systems and no longer on local, physically protected disks. Application service providers, outsourcing, and peer-to-peer systems have contributed to this trend going so far, as offering storage as a service. New mechanisms are needed for ensuring data resilience in this context. Data must be protected against loss, breach of confidentiality, and violation of integrity properties. Existing resilient storage services use a combination of distributed protocols and cryptographic mechanisms, but must be enhanced for new requirements and architectures.

4- Research challenges

New mechanisms for data resilience: A well-established technique to protect data against physical loss is to rely on redundancy at the device level, by using RAID mechanisms to tolerate disk failures. But the methods developed so far only apply to the physical level, and are inappropriate for application-layer storage in the networked environments of the future. Therefore, new tolerance mechanisms for data resilience in dynamic and evolving systems should be developed.

Tracing the evolution of information: Research should also focus on the lifecycle of information that is processed and changed, such that its evolution can be traced in an easily verifiable way. Today's digital signature technologies create every signature from scratch, and do not support structures to express that information represents the outcome of a process that refined existing knowledge. Methods for providing cryptographically verifiable integrity guarantees, for example, for information quoted in a different context than it was created should be addressed.

Collaboration of partners: The collaboration of a myriad of partners for offering storage, similarly to the functionality offered by most peer-to-peer file sharing infrastructures, is yet another domain that should be investigated. In particular, determining the actual availability of one's stored data in this context appears as a critical problem to be answered in order to provide large-scale dependable storage. Cryptographic as well as statistical techniques will be needed for this purpose.

Disaster recovery and business continuity metrics: Data storage becomes a key concept of disaster recovery: its objectives move towards the higher tiers; fixed costs, implementation costs, ongoing network costs, and maintenance costs grow exponentially. Higher tier disaster recovery solutions may require secondary sites with redundant operational equipment, media, software licensing, staff, and high bandwidth connections over long distances, coupling or clustering management software, and hardware that supports or provides point-in-time volume copies or remote data replication. The distinction between the tiers is how quickly you need to recover your data (recovery time objective), how quickly you need to recover the services provided by your environment, and how much data you cannot afford to loose (recovery point objective). It is important to understand that the cost of a solution must be in reasonable proportion to the business value of IT, and align the selected recovery tier to the business value of the service to be recovered. The measurement metrics of business continuity (BC) need to be redefined, to provide common sense of BC planning. A methodology required to define the service values and match them with the proposed BC solution. Another focus should be to develop an automated environment monitoring the consistency on platform, middleware, application and transactional level to establish secure points of recovery, allowing the initiation of snapshot based backups.

Data lifecycle and policy based data management: The business value of the information is changing during its lifecycle. Information is a company asset, therefore its management must comply with internal and regulatory requirements, but allowing cost optimization as well. Data protection schemes (RAID schemes, mirroring, replication, etc.) are used to protect data from disk failures and other hardware errors. However, if the data is mirrored or RAIDed, then the problem is accordingly multiplied. Some typical problems to solve include the following: (a) With direct-attached storage (whether internal or attached to a SAN), in some cases, only a very small percentage of available storage is used. (b) Applications are installed, but then are not used. No one tries to locate the unused files. Application upgrades can also leave unneeded files. (c) Many files are created once, used once, and never accessed or used again. This results in stale or obsolete files. (d) A duplicate file is no longer needed, but it is cheaper to leave the duplicate file as is rather than spend time to try to find it. (e) It is increasingly common to find illegal types of files and other personal data items placed onto expensive corporate storage. (f) The value of the information changes during its lifecycle. The storage areas where these data are stored must be aligned of the real information value.

The research challenge is to define policy based data management solution to data classification, on service, disks, storage volumes, file systems, files, and databases level. This solution should provide a central point of data lifecycle management, allowing automated processes to optimize data storage, assure compliance with data retention requirement, provide optimal storage area to store the data according to its changing value during its life cycle. Another challenge is to define data-lifecycle specific metrics to allow standardizing common policy definition, and easy compliance control.

5- References

[IEEE] IEEE Security in Storage Working Group (http://siswg.org/)

GE6 - Critical Infrastructures

1- Definition

Critical infrastructures like the power, water and gas distribution networks have a fundamental role in modern life. These infrastructures are essentially physical/mechanical processes controlled electronically. The control systems, usually called SCADA (Supervisory Control and Data Acquisition) or PCS (Process Control System), are composed by computers interconnected by communication networks.

In recent years, these systems evolved in several aspects that greatly increased their exposure to cyber-attacks coming from the Internet. Firstly, the computers, networks and protocols in those control systems are no longer proprietary but standard PCs interconnected through standardized technologies (Ethernet, WiFi, …), and the protocols are often encapsulated on top of TCP/IP. Secondly, these networks are usually connected to the Internet indirectly through the corporate network or to other networks using modems and data links. Thirdly, several infrastructures, possibly belonging to distinct organizations, are being interconnected creating a complexity that is hard to manage. Therefore, these infrastructures have a level of vulnerability similar to other systems connected to the Internet, but the socio-economic impact of their failures can be huge.

In addition, the interdependencies between operators can cause major problems. These interdependencies can be related to operators of the same field (e.g., a shortage of electricity in one country can lead to a demand sent to other countries' networks), but cascading events may result in a butterfly (or domino) effect, where failures initiated in one field can propagate through infrastructure's interconnection to other areas (e.g., a lasting power supply outage inevitably uses up all spare batteries of a Telco). Under-dimensioning and absence of protecting barriers often cause huge infrastructure outages in presence of abnormal requests. Socio-technical aspects constitute another facet of the sources of failure. Human misunderstanding of emergency procedures have lead to most of the past disasters in several industrial systems (Chernobyl, ...); also, lack of proven confinement requires that field operators can not ignore all potential consequences of domino effects following their actions (e.g., the European black-out in November 2006 [Europe 2007]).

2- Examples / Application Domains

The mentioned critical infrastructures such as general distribution networks (e.g., power, water, gas) serve today's society. High availability of these systems is crucial for the society and failures can easily have catastrophic consequences. As a result, they have to follow the humanity development. Maybe one of the best examples for this requirement is the electric system. Consumers are buying more and more electric devices. Nowadays, it has become clear that power suppliers will not be able to serve these growing demands in the future in huge cities without continuous development of the supply system. Newer and newer power plants are built and the most recent technologies are used to maintain a continuous service.

Such phenomena are perceivable in any kind of critical infrastructures. It results in ever growing complexity and unforeseen challenges that need to be overcome and means the continuous and necessary evolution of these systems.

Besides the distribution networks, telecommunications systems have also become crucial for the whole society. Telco networks provide nowadays a wide scale of services. For example, the emergency telephone services, inquiry office services, business conferencing, chatting with other people, audiovisual entertainment, etc. are all available in public infrastructures (composed of Internet and Telco switched networks). The appearance of mobile devices has further broadened the areas of usage. Today's society has come to a point where the very existence of a blooming, successful economy is impossible without a highly-developed communication infrastructure.

3- Current approaches

There have been efforts to handle the increasing complexity of huge systems. One of the approaches is that large systems can be divided in two classes: i) those systems whose goal is a centralised system that overcomes the capabilities of each component that it consists of; ii) and those, composed of loosely coupled individual subsystems, each able to pursue its own goal, but with the necessary interchange of information with others.

In the first class, a hierarchical model is the natural way to represent the real structure. For this kind of view, widely used in most scientific disciplines, the challenge is to devise analysis/design models and tools that take advantage of the structure, avoiding the state explosion stumbling block. In the second class, more or less isolated nodes, connected by communication lines, have to be organized in a scalable fashion. Several options can be examined; as a first suggestion: i) grouping into "islands" those nodes whose information interchange is significantly over the average, and the total I/O flow off the island is "limited"; ii) a hierarchical-distributed configuration, where it is possible to partition the nodes in rings, such that the interaction between rings is limited, and a "ring interface" could be univocally defined for all nodes external to a given ring.

From the evolvability perspective, both above mentioned approaches support scalability. Another critical infrastructures perspective is formed of portability and manageability. Among others, these properties are addressed by standardization efforts like the Application Interface Specification (AIS) of the Service Availability Forum (SA Forum) [Saf], or specifications of the Distributed Management Task Force [Dmtf].

Interdependencies of systems are problematic because failures in one system can lead to failures in other dependent systems. In [Rinaldi et al. 2001], the authors identified some of the most important questions related to critical infrastructures, and, based on this work and many other sources, [Laprie et al. 2007] elaborated a model for the qualitative analysis of these dependencies. By using such models, the expectable risks are easier to estimate throughout the system lifetime, and counteractions in case of failures can be defined.

4- Research challenges

As discussed before, critical infrastructures are in need of continuous evolution since they serve the ever changing society. To overcome the newer and newer difficulties, research needs to emphasize the following topics:

- **Creation of a new reference architecture.** There is a need to devise a new reference architecture to protect critical infrastructures that takes into consideration not only the aspects mentioned above, but also the existing legacy control sub-networks, which cannot be modified and will continue to exist in the years to come. Moreover, certain security mechanisms might introduce difficulties to the normal operation of these networks, which are amplified during emergency periods. Especially, security policy(ies) should be developed, adapted to the architecture, and accounting for the various organizations involved in the infrastructure and the various roles of the users belonging to these organizations.

- **Implementation of the architecture.** This will also introduce very significant challenges because the problem being addressed is complex since it involves aspects of large-scale distribution, dependability and real-time operation.
- **Interdependency** has hardly been addressed in the past, but as systems evolve, inter-system relationships are getting more complex. Solving this problem requires new approaches. Sharing information, possibly confidential, between the operators in different infrastructures, is an issue which goes far beyond technical aspects/solutions, and which clearly requires trust between the parties to benefit from addressing the interdependency concerns. Studying the mechanisms, via which anomalies/disturbances in one infrastructure may propagate to another infrastructure, possibly leading to failures in it, is badly needed.
- **Interdependent failures** are one of the main concerns within critical infrastructures. Although lot of work has been done in this field, modelling of such phenomena adequately is still in its infancy and correctness of the different approaches still needs to be proved.
- **Mastering the complexity.** As the current systems evolve, they inevitably tend to grow bigger, both as single subsystems, and in the number of interconnections. While it is tempting just to put together more components already available off-the-shelf, by means of available connection techniques, this way the need for "insulation" is easily neglected. But, even though the "insulation" way is pursued, it is not enough to evolve smoothly towards really larger resilient systems: the problem of mastering the increasing complexity must be tackled beforehand.

5- References

[Europe 2007] Europe Press release, Blackout of November 2006: important lessons to be drawn, Reference IP/07/110, Brussels, 30 January 2007

[Saf] Service Availability Forum, http://saforum.org

[Dmtf] Distributed Management Task Force, http://dmtf.org

[Rinaldi et al. 2001] S.M. Rinaldi, J.P. Peerenboom, and T.K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies", IEEE Control Systems Magazine, Vol. 21, N° 6, Dec 2001, pp. 11-25.

[Laprie et al. 2007] J.-C. Laprie, K. Kanoun, and M. Kaâniche, "Modelling Interdependencies between the Electricity and Information Infrastructures", Proc. SAFECOMP 2007, Erlangen, Sept. 18-21, pp. 54-67.

GE7 - Design for Adaptation: Framework and Programming Paradigms

1- Definition

The adaptation capabilities of a system depend on early stages of its design and the software development technology that was selected. At one extreme of the spectrum we find systems composed of black box components glued together in a static manner. Clearly this kind of systems has limited adaptation features. The ability to master system components behaviour and their inter-relation at runtime is necessary to cope with evolvability requirements. To this aim the architecture and the software technology selected must provide observation and control features to dynamically master the structure and behaviour.

The candidate technologies must provide separation of concerns and sufficient reflective features to master adaptation. There is a trade-off however between, on one side, the observation and control features and, on the other side, the real needs for adaptation. A good balance is required indeed because the requirements for adaptation correspond to a given set of observation and control features. Too much meta-information is a penalty and so there is clearly a trade-off here.

2- Examples / Application Domains

Open component technology (like OpenCom [Coulson et al. 2002], Fractal [Bruneton et al. 2006], etc.), Aspect oriented software development (see. [AOSD]), and other reflective computing related technologies are possible candidates. However, in their present form they only provide structural views of components and bindings. They do not provide enough internal meta-information to master the execution from a resilience viewpoint.

3- Current approaches

As soon as systems can be designed with open components then adaptation can be performed with a recursive approach that separates several abstraction layers: the application (level 1), the resilience strategy (level 2) and the adaptation strategy (level 3). This reflective tower can be extended, for instance, by adding a consistency layer on top of the adaptation layer, showing the great benefit of this approach. The adaptation decision can simply depend on resources thresholds (e.g. for long lifetime embedded systems) but also on real changes in the system specifications (e.g. for large scale applications).

The benefit of such system architectural frameworks is to clearly identify a place for adaptation with appropriate software sensors and actuators to dynamically perform the update of resilience strategies in a consistent manner.

Two main example implementations of such a resiliency framework are the case where an application is put into the context of a specialized resiliency enforcing middleware (like the standard AIS of SA Forum), or when the sensors and actuators and the resiliency policy are all implemented in a separate, general purpose component (like IBM Tivoli or HP OpenView).

4- Research challenges

In summary we can identify three challenges:

- Frameworks, programming paradigms, algorithms must be defined along these lines to master resilience adaptation.
- Resilience techniques for adaptive systems have to be developed using open components and AOSD technologies and their assessment.
- How supervised environment can be used as development testbeds for tightly integrated resiliency mechanisms. The assessment of the design paradigms, frameworks, and environments is obviously an important facet of the research work to be carried out. There is thus a strong link with other sections such as on-line assessment and design for assessability.

5- References

[Coulson et al. 2002] Coulson, G., Blair, G.S., Clarke, M., Parlavantzas, N., "The Design of a Highly Configurable and Reconfigurable Middleware Platform", ACM/ Springer Distributed Computing Journal, Vol. 15, No. 2, pp 109-126, April 2002.

[Bruneton et al. 2006] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, and J.-B. Stefani. The Fractal Component Model and Its Support in Java. Software Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems. 36(11-12), 2006.

[AOSD] Aspect Oriented Software Development – AOSD-Europe NoE, Coordinator: University of Lancaster (UK), http://www.ctit.utwente.nl/research/projects/international/noe/aosd-e.doc/

GE8 - Service Oriented Architectures (SOA)

1- Definition

Service Oriented Architectures (SOA) refer to "loosely-coupled" heterogeneous systems where components are given only by their interfaces and are integrated by their functionality, i.e., no strong connection between system modules is present. Usually, parties of such architecture can play three roles: Service providers offer services via standard well-defined protocols, where conversation format is defined by service interfaces. These interface descriptions are published to service registries where service consumers can query them. Having the description of the desired services, service consumers connect to providers directly. Note that these roles can overlap as in a composite service a single component can be both a provider and a consumer as well.

XML-based web services are a technological platform for implementing SOA. Standard technologies include WSDL (Web Service Description Language) for service description, SOAP for messaging, UDDI (Universal Description, Discovery and Integration) for service publication, etc. A lot of emerging WS-* standards aim at providing support for designing and enforcing policies for service behavior and non-functional characteristics of services (security, transactionality, logging, communication, policies, etc.). All of these are XML-based and have been implemented at different levels (from beta-version trials to mature products). However, a lot of modeling, design, analysis and monitoring issues are open. Due to its potential business value, SOA is one of the most frequent buzzwords used both by industrial and academic players.

2- Examples / Application Domains

Currently, SOA as a modeling/implementation paradigm is used in most business sectors to implement large-scale distributed systems. One of its main application areas is reengineering existing legacy systems (banking and insurance applications) to modularize and reuse functionality.

Besides the above described functional design, more and more tools support BPEL [BPEL 2007] which is a composition/orchestration language, now a standard of OASIS. System management tools are now getting closer to monitor Service Level Agreements of Web services which increases the accountability of such applications.

Enterprise Service Bus (ESB) products aim at supporting integration of extra-and intraorganizational services by providing enterprise-wide solutions for policies, logging, messaging, etc. Most ESB products also try to reduce the overhead of XML messaging by integrating other communication protocols as well.

Looking at the modeling front-end in the engineering process, although several UML profiles exist to describe Web services, their composition, etc. (e.g., [UMLBPEL, 2005], [UMLSS, 2005]), none of these can be considered as a complete solution.

3- Current approaches

As there is an increasing number of technical implementations of Service Oriented Architecture available, the importance of a common model-based analysis, development and deployment mechanisms are needed.

Therefore, one of the current research trends is to use standard visual modeling languages and derive formal analysis models (e.g. to find adequate model checking mechanisms for BPEL). Another important research step is made to the direction of model driven deployment to industrial platforms, such as reliable messaging middleware (see [Gönczy, 2006a]). Related work in [Alwagait& Ghandeharizadeh, 2004] aims at to develop a dependable web services framework, which relies on extended proxies, however, without consulting standards.

Another solution is suggested in [Wirsing, 2006] where qualitative analysis description (to be basis of verification of service compensations) and quantitative evaluation models for SLAs are derived from high level visual models. Verification of fault-tolerant services and middleware is described in [Gönczy, 2006b]. These works are connected to the ongoing Sensoria EU IP [Sensoria, 2005].

One of the most interesting recent technological SOA paradigms is SCA combined with SDO. Service Component Architecture (SCA) tries to give a framework for building composite services, which can be implemented as standard Web services, BPEL processes, Java classes or even PHP code. It can be combined with Standard Data Objects (SDO) technology, which provides a graph-like representation and a uniform manipulation API for multiple data sources (relational databases, XML files, etc.).

From the more scientific point of view, current research topics of SOA include

- Formal models for service choreography
- Verification/validation methods for service oriented system models
- Modelling and evaluation of security and trust
- SLA modelling and evaluation of services
- Model-based development and deployment technologies for SOA

4- Research challenges

From the more business oriented aspect, componentization of large enterprise processes enables the definition of precise SLA metrics as the basis of monitoring the non-functional characteristics of such services, giving the possibility of executing runtime reconfigurations to provide resilient services at the business level as the system evolves.

This raises the following questions:

- How can the functional equivalency of services proved runtime?
- How can the correct operation of the evolving system proved in the presence of new components?
- How can the dependability of evolving service compositions be evaluated? How to insert dependability patterns to systems where the equivalency of services/variants is not known a priori?
- How can new requirements be decomposed to service goals?
- How to combine existing verification/testing methods of distributed systems to facilitate runtime system evolvement?
- How to check the coherency of different views of service models (orchestration, non-functional descriptions, dynamic behaviour)? How to generate service deployment and orchestration code considering these aspects?

- How to orchestrate business services in a content-based manner?

Services may fail sometimes in a way that compensation of previous executions is needed. Although modelling of such compensation action is currently targeted, it is an open question how to include feedback and preventive measures in the runtime system to minimize the effect of compensations and optimize the workflow of composite services to weaken the effect of a failed service.

The large scale deployment of services makes it necessary to resort to service discovery as illustrated by UDDI for Web Services. This mechanism, generally envisioned as an enabler in most service-oriented middleware, also brings vulnerabilities. Because most services may pertain to diverse organizations without a priori trust, they do not share secrets. In addition, performing a lookup may leak private information regarding the activity of a client service. Finally, in ubiquitous systems, the presence of a service should not necessarily be disclosed to any querying party since it also exposes it to denial of service attacks. Cryptographic and policy based mechanisms are therefore in order to mitigate these threats.

5- References

[Alwagait& Ghandeharizadeh, 2004] Alwagait, E. & Ghandeharizadeh, S. (2004) Dew: A dependable web services framework. RIDE, vol. 01:pp. 111–118.

[BPEL 2007] BPEL (2007). OASIS Web Services Business Process Execution Language (WS-BPEL) 2.0. Retrieved from docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html

[Gönczy, 2006a] Gönczy, L, Ávéd, J., Varró, D. Model-based deployment of web services to standards-compliant middleware. Proc. of ICWI2006. Iadis Press. 2006.

[Gönczy, 2006b] Gönczy, L., Kovács, M, Varró, D. Modeling and verification of reliable messaging by graph transformation systems. In Proc. of (GTVC2006). Elsevier, 2006.

[SCA 2007] SCA (2007), Service Component Architecture Specifications v 1.0. Retrieved from http//osoa.org. 2007.

[Sensoria, 2005] Sensoria (2005), Software Engineering for Service-Oriented Overlay Computers, http://www.sensoria-ist.eu/

[UMLBPEL, 2005] UMLBPEL (2005).UML 2.0 Profile for WS-BPEL, IBM, 2005. Retrieved from www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/.

[UMLSS, 2005] UMLSS (2005) UML 2.0 Profile for Software Services. Retrieved from www-128.ibm.com/developerworks/rational/library/05/419_soa/.

[Wirsing, 2006] Wirsing, M., Clark,A., Gilmore,S., Hölzl,M., Knapp, A., Koch, N, Schroeder. A. (2006) Semantic-Based Development of Service-Oriented Systems. Proc. FORTE'06, pp. 24-45. LNCS 4229. Springer-Verlag. 2006

GE9 - Complexity & Self-Organisation

1- Definition

Within ReSIST evolvability has been described as the capacity of systems to adapt to changes in the user requirements, threats or in the operational environment. This gap is concerned with the problem of understanding the underlying mechanisms of adaptation in ubiquitous and distributed systems and the challenges this poses for approaches to modelling, designing, predicting and measuring the evolvability (i.e. the capacity for successful adaptation) of such systems.

Ubiquitous and distributed systems are complex in the sense that they comprise a large number of heterogeneous and interacting agents. Within complex systems, the local interactions of these agents may result in emergent forms of system behaviour that were not planned or even anticipated. The computer based components within these systems will be expected to operate under conditions that cannot be predicted using conventional analytical techniques. Order may be created in such systems without the need for an overall plan, or it may break down in case of ill-adaptation. The former characterises an aspect of resilience of complex systems. In this sense, resilience can be regarded as a capability that implicates a process of self-organisation in order to cope with the complexity and the dynamics of the environment. Important aspects such as quality of production and safety emerge from this process or manifest themselves as failures in case of ill-adaptation. Understanding self-organisation is essential for modelling ubiquitous systems and predicting the effects that changes will introduce. Approaches for enhancing the capacity for successful self-organisation will be key elements of resilience engineering.

2- Examples / Application Domains

Many of our future distributed and ubiquitous systems can be regarded as complex systems. The concepts apply to safety-critical domains such as air traffic management within a free flight air space or military applications (dynamic, heterogeneous teams of humans and technological autonomous agents), as well as to non safety domains such as public administration and domestic applications (see GE 2), as well as networks of sensors, autonomous robots, or mobile ad-hoc networks that have a dynamic population. For example, in manufacturing autonomous robots can dynamically form coalitions to fulfil jobs and can re-distribute individual activities in case of failures or shifting priorities (smart factory). In some of these systems the agents will be entirely technological artefacts (e.g. sensor networks), while in other applications the human element plays a key role in the adaptation process. The latter can be observed, for example, in an Emergency Care Department: as more and more unexpected patients arrive, procedures may be abbreviated, diagnostics may be requested in anticipation, patients may be moved into corridors, and other departments may phone and offer extra resources. These behaviours support resilient adaptation to variations [Anders et al., 2006][Wears et al., 2006]. The effects of these adaptation processes may spread throughout the hospital. In the future technology plays an essential role in providing means for monitoring variations and facilitating the adaptive resource allocation by collecting information and making this available to a network of relevant actors. Self-organisation can occur both short-term and on a local level in response to operational breakdowns as well as longer-term and on an organisational level in response to, for example, a changing IT infrastructure or market environment.

3- Current approaches

There is a large and growing body of work within complexity science that provides insights into the characteristics and dynamics of complex systems that may be useful as an analogy for the study of resilience in ubiquitous systems. Work in complexity science could be described as falling into two complementary, yet very distinct classes [Parvard & Dugdale, 2000]. The first class originated from the study of non-linear deterministic systems, and comprises theories such as Chaos Theory [Gleick, 1988] and the work on dissipative structures [Prigogine & Stengers, 1984]. Of greater direct relevance is the second class of work, which is concerned with distributed self-organising systems. Complex Adaptive Systems (CAS) theory models systems as consisting of a large number of agents that interact with each other based on some local rules. Utilising computer simulations it is possible to model, for example, complex behaviours such as the flocking behaviour of birds [Reynolds, 1987] or food collection of ants [Langton, 1996] based exclusively on such local interactions without the need for an overall blueprint.

The work of Kauffman [Kauffman, 1993] and Goodwin [Goodwin, 1994] emphasises emergent novelty due to self-organisation (as opposed to merely moving to different attractors already embedded within the system). Following such a position challenges current management and safety engineering thinking that assumes that desirable system states or attractors can be determined and that subsequently the system can be induced to self-organise towards those attractors by externally manipulating them [Stacey, 2002].

4- Research challenges

There are a number of gaps in the current state of knowledge that need to be addressed in the future:

- How and to what extent can this self-adaptation be modelled and be accounted for during design?
- What factors is the successful self-adaptation dependent on, when does it break down, and how can it be restored?
- What impact do these notions of self-adaptation have on the development of appropriate computer based technologies?
- To what extent can the outcome of self-adaptation be predicted and guided? How should system engineers take a whole system view?
- How can the capacity for successful self-adaptation (as an aspect of resilience) be measured or expressed?
- How and to what extent can the propagation of effects be predicted and assessed?
- How can an overarching system model and argument for resilience be developed that integrates data from the various heterogeneous agents?
- How can the assessment of resilience be communicated to stakeholders of different backgrounds?

5- References

[Anders et al., 2006] S. Anders et al. (2006) Limits on adaptation: Modelling resilience and brittleness in hospital emergency departments. In Proc. 2nd Resilience Engineering Symposium

[Wears et al., 2006] R.L. Wears et al. (2006) "Free Fall" – A case study of resilience, its degradation, and recovery in an emergency department. In Proc. 2nd Resilience Engineering Symposium

[Parvard & Dugdale, 2000] Pavard, B. and Dugdale, J (2000) The contribution of complexity theory to the study of complex socio-technical systems. InterJournal of Complex Systems, 335, New England Complex Systems Institute

[Gleick, 1988] Gleick, J. (1996) Chaos: Making a new science. Vintage

[Prigogine & Stengers, 1984] Prigogine, I. and Stengers, I. (1984) Order out of Chaos. Random House

[Reynolds, 1987] Reynolds, C.W. (1987) Flocks, Herds and Schools: A Distributed Behavioural Model. ACM SIGGRAPH Computer Graphics, 21(4)

[Langton, 1996] Langton, C.G. (1996) Artificial Life. In Boden, M.A. (Ed.) The Philosophy of Artificial Life, pp.39-94, Oxford University Press

[Kauffman, 1993] Kauffman, S.A. (1993) The origins of order: self-organisation and selection in evolution. Oxford University Press

[Goodwin, 1994] Goodwin, B. (1996) How the leopard changed its spots. Pocket Books

[Stacey, 2002] Stacey, R. (2002) Strategic Management and Organisational Dynamics (4th Ed.). Prentice Hall

GE10 - Virtualisation

1- Definition

Virtualization is an architectural approach that allows the real hardware configuration of a system to be abstracted away. One method of virtualizing the hardware resources of a computer involves using a layer of software called the Virtual Machine Monitor (VMM) to provide the illusion of real hardware for multiple virtual machines (VMs). Each VM runs its own operating system (often called the guest OS) and applications, and is isolated from other VMs running on the same physical machine. The guest OS and applications inside a VM run on the VM's own virtual resources, such as virtual CPU, virtual network card, virtual RAM, and virtual disks. A VMM can be hosted directly on the computer hardware (e.g., Xen [Barham et al. 2003]) or in a host operating system (e.g., VMware).

2- Examples / Application Domains

Virtualization of storage, networking, and computing resources enables improved efficiency, flexibility, and cost savings in data centres where a common physical infrastructure is shared among multiple virtual infrastructures belonging to different (perhaps, competing) customers. Both the efficiency and cost saving aspects and dependability can be improved by using virtualisation, even in almost transparent, mostly automatic ways – resembling evolution.

Virtualisation can also be used to enhance the resilience of the systems by tracking anomalous activities, and utilizing the attacked, to-be-compromised systems as honeypots for the intruders, delaying or even completely "trapping" the attack, while observing the behaviour of the attacker and making steps to stop the attacker (for example by disabling the compromised user account, stopping the hijacked service, etc).

Adapting to new environmental situations is not the only aspect virtualisation can be used to enhance the operation of a system, but based on accumulated historical load data, and past attack tendencies, anticipatory actions could be taken to ensure the seamless operation of the system. Both periodical patterns and trends like tendencies could be observed this way, helping the optimization of costs and utilization while leveraging dependability.

It could also be possible to use diverse implementations of virtualisation hosts to automatically compensate for unknown exploitable flaws, since with the detection of several anomalous activities of the same kind against the same type of systems, the whole system can automatically evolve towards a state where the "species" of systems attacked become "extinct" – at least until the attacks are stopped, or the aforementioned systems are "immunized" against them by security patches.

3- Current approaches

While virtualization has a long history of related work spanning more than three decades, we restrict our focus to the use of virtualization for enhancing system dependability. Bressoud and Schneider [Bressoud & Schneider 1996] implemented a primary-backup replication protocol tolerant to benign faults at the VMM level. The protocol resolves non-determinism by logging the results of all non-deterministic actions taken by the primary and then applying the same results at the backups to maintain state consistency. Double-Take [VMWARE] uses real-time byte-level replication to replicate data on multiple VMs, on the same or on different physical machines, so that the application can automatically fail over to a spare VM in case of an outage. Dunlap et al. describe ReVirt [Dunlap et al. 2005] for VM logging and replay. ReVirt encapsulates the OS as a VM, logs nondeterministic events that affect the VM's execution, and uses the logged data to replay the VM's execution later. Joshi et al. [Joshi et al. 2005] combine VM introspection with VM replay to analyze whether a vulnerability was activated in a VM before a patch was applied. The analysis is based on vulnerability-specific predicates provided by the patch writer. Backtracker [King & Chen 2003] can be used to identify which application running inside a VM was exploited on a given host. An extension of Backtracker [King et al. 2005] has been used to track attacks from a single host at which an infection has been detected to the originator of the attack and to other hosts that were compromised from that host.

4- Research challenges

New ways of leveraging virtualization for enhancing system dependability must be explored. We have taken some steps in this direction, and have proposed ways of using virtualization for coping with load-induced failures, applying patches for high-availability services, enforcing fail-safe behavior, proactive rejuvenation, and improving diversity in fault-tolerant replication [Ramasamy & Schunter 2007]. While these techniques may enhance dependability, we do not know how they affect other important system attributes, such as performance and security. Future work should include actually implementing some of the proposed techniques and studying their effect on other system attributes.

- The effect of using historical load, performance, and anomalous activities' data in the automatic evolution of the systems should be investigated
- The area of utilizing virtualised layers to leverage resilience is still full of unknown potentials, and has opportunities to explore, tracking the attackers and utilizing virtual honeypot systems, etc.
- The effect of having a security flaw in the virtualisation system should be also investigated, with special regards to the error propagation and security flaw exploitation possibilities between virtual machines, and between the host and guest machines, in each direction.
- Using the same virtualisation solution over the whole system might pose an enormous security threat by having the same – possibly exploitable – fault in every system, while utilising different systems hinders performance and is not generally supported.
- The possibilities of automatic evolution using historical data and current environmental observations between competing solutions should also be investigated from the aspects including resilience, performance, operational costs, and efficiency as well.

5- References

[Barham et al. 2003] P. T. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In Proc. 19th ACM Symposium on Operating Systems Principles (SOSP-2003), pages 164–177, October 2003.

[Bressoud & Schneider 1996] T. C. Bressoud and F. B. Schneider. Hypervisor-Based Fault Tolerance. ACM Trans. Comput. Syst., 14(1):80–107, 1996.

[VMWARE] VMWare Double-Take. http://www.vmware.com/pdf/vmware_doubletake.pdf.

[Dunlap et al. 2005] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. SIGOPS Operating System Review, 36(SI):211–224, 2002.

[Joshi et al. 2005] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen. Detecting Past and Present Intrusions through Vulnerability-Specific Predicates. In Proc. 20th ACM Symposium on Operating Systems Principles (SOSP-2005), pages 91–104, 2005.

[King & Chen 2003] S. T. King and P. M. Chen. Backtracking Intrusions. In Proc. 19th ACM Symposium on Operating Systems Principles (SOSP-2003), pages 223–236, 2003.

[King et al. 2005] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen. Enriching Intrusion Alerts Through Multi-Host Causality. In Proc. Network and Distributed System Security Symposium (NDSS-2005), 2005.

[Ramasamy & Schunter 2007] H. V. Ramasamy and M. Schunter. Architecting Dependable Systems Using Virtualization. Workshop on Architecting Dependable Systems: Supplemental Volume of the 2007 International Conference on Dependable Systems and Networks (DSN-2007).

GE11 - Managing Multiple and Heterogeneous Models

1- Definition

The design of a usable, reliable and fault-tolerant interactive system is based on a mass of information of multiple natures from multiple domains. This forces design complexities and dangers surrounding the gathering and refinement of this mass of information. This complex and currently mostly informal process can be supported using models that allow handling data to be handled at a high level of abstraction. However, not all relevant information can be embedded in a single model. There are models for the requirements, functional aspects, the interaction part, models to describe human operators, operator tasks, and so on. All these models represent several views of the same world that will evolve and co-evolve during the whole design process and so, must and stay compatible in order to fully and clearly describe a system. The gaps in current research are the following:

- A first point is that for each kind of modelling technique, the corresponding modelling process iteratively produces models that evolve from a first version to the next one, implying a possible gap in verification with the need to ensure that every version meets a set of requirements.
- A second point is that each time a model is produced or improved there is a need to verify its compatibility with other models involved in the system description.

Thus, the various models ought to be consistent and coherent with one another.

One of the problems associated with interactive safety-critical design is the lack of coherence between multiple viewpoints and therefore multiple design models of the same world. We believe there should be coherence between these design models to increase a system's resilience. Some work on model-based approaches has tried to address these issues but there is still a lot to do before design methods actually provide a framework to support this critical activity. Indeed, it is still not commonly agreed that there should be a framework for multiple models as some current research argues that models of one kind could be generated from models of the other kind. For instance [Paternò et al., 1999] proposes to generate user interfaces from task models while [Lu et al. 1999] proposes to generate task models from system models. More recent work (not dealing with the generation from models) promotes the use of several models describing different views on the human-computer system such as [Navarre et al, 2006] that combines data flows and behavioural models and [Basnyat et al., 2005] that combines incidents and accidents investigation techniques and behavioural model for both systems and humans.

Another critical aspect of the design of interactive systems is that designing for usable systems calls for user centred design approaches that are highly iterative involving end users at each stage. Ensuring a convergence point with these diverse (and generally heterogeneous) models on continuous evolution is the main topic of this gap description.

2- Examples / Application Domains

The research problems discussed in this section can be applied to most interactive systems, but since the resources required to perform the cross-model coherence checking are costly, we believe it is particularly relevant to critical applications such as satellite command and control room, interactive cockpits for military and civilian aircrafts, command and control rooms for drones and air traffic control workstations are more applicable.

3- Current approaches

Most research on model coherence relates to the user/task model and the system/architecture models. For example, Paternò [Paternò, 1997] takes a task model which is used to produce a model similar to a dialog model and then uses temporal logic to verify the coherence between the task model and the dialog model. Furthermore, in [Bourget-Rouger, 1988] the authors study the behavioural compatibility between two Petri nets as a means of version checking of model evolution.

Work has been done on the compatibility of data represented in a user model and a system model as a means of performing fault diagnostics in complex systems [James et al., 1996].

Some of the RESIST partners are currently studying methods for integrating the necessary models for safety critical interactive systems design. To date, two approaches have been devised for integrating the task model and system model while taking into account human errors. One approach uses scenarios as a bridge between the two [Navarre et al. 2001]. The second approach uses task patterns as a means of cross-checking properties between the two models. Furthermore, a PhD Thesis has been written [Basnyat, 2006] proposes a generic integrated modelling framework for the analysis design and verification safety-critical and resilient systems, based on the above two approaches. This work is part of more ambitious long-term plan to deal with multiple models for safety critical interactive systems.

4- Research challenges

Due to the large number of domains involved, it is highly unlikely that the data gathered, analyzed and documented will be represented in the same way. For example, it is unlikely that the system engineers will take into account all information provided by human factors analysts (for instance about work practice and users). This is not only because of time constraints and the amount of data involved, but also and mainly, because the usual kind of notation cannot record that information efficiently. This can have serious effects on the reliability, efficiency and error-tolerance of a system.

It is clear that there is a need for formalizing not only the process of gathering this mass of data, but also for refining and modelling it when necessary in order to provide valuable input to the system design.

When considering typical design iterative design processes, there are in general phases for gathering information, sharing and embedding information and formalizing information via modelling.

Furthermore, there is the issue of the various types of information involved in the design process of a safety-critical system that must be dependable and usable; there is a pre-deployment and post-deployment type. For pre-deployment data, we refer to data that can be obtained throughout the design process before the system has been deployed. Of course, much of this data can be made available and used for evaluation purposes, once a system has been deployed. By post-deployment data, we mean data that can only be obtained once the system in mind has been deployed. Examples of such are usability analysis, incident and accident reports or the use of metrics for risk analysis [Fenton and Neil, 1999]. Of course, the data gathered will evolve over time as the systems will evolve when updates, improvements or corrections are deployed in turn.

Additionally, the data gathered and analyzed for input into a safety-critical interactive system design is collected by multiple specialists of a wide-array of domains. This is due to the nature of safety-critical systems that range from cockpits to surgical equipment to mining instruments to name just a few but also to the variety of information that has to be gathered and the fact that this information stems from multiple domains of expertise. This combination of diverse specialists and diverse domains adds to the complexity of design of a safety-critical system.

5- References

[Basnyat, 2006] Basnyat, S (2006) A generic integrated modelling framework for the analysis, design and validation of interactive safety-critical and error-tolerant systems. PhD Dissertation. December 2006, University Paul Sabatier, Toulouse, France

[Basnyat et al., 2005] Basnyat, S., Chozos, N., Johnson, C., Palanque, P. (2005) Redesigning an Interactive Safety-Critical System to Prevent an Accident from Reoccurring. 24th European Annual Conference on Human Decision Making and Manual Control. (EAM) Organised by the Institute of Communication and Computer Systems. 17-19 October 2005. Athens, Greece

[Bourget-Rouger, 1988] Bourguet-Rouger, A (1988). "External Behaviour Equivalence Between two Petri Nets". Concurrency 88, LNCS 335, Springer-Verlag, 1988, pp. 237-258

[Fenton and Neil, 1999] Fenton N. E. and Neil M, (1999). Software Metrics and Risk, Proc 2nd European Software Measurement Conference (FESMA'99), TI-KVIV, Amsterdam, ISBN 90-76019-07-X, 39-55, 1999.

[James et al., 1996] James T. Sawyer, Brian Minsk, Ann M. Bisantz (1996) Coupling User Models and System Models: A Modeling Framework for Fault Diagnosis in Complex Systems Interacting with computer 1996

[Navarre et al. 2001] Navarre, D., Palanque, P., Bastide, R., Paternó, F., and Santoro, C. (2001). "A tool suite for integrating task and system models through scenarios." In 8th Eurographics workshop on Design, Specification and Verification of Interactive Systems, DSV-IS'2001; June 13-15. Glasgow, Scotland: Lecture notes in computer science, no. 2220. Springer

[Navarre, et al., 2006] David Navarre, Philippe Palanque, Pierre Dragicevic & Rémi Bastide. An Approach Integrating two Complementary Model-based Environments for the Construction of Multimodal Interactive Applications. Interacting with Computers, vol. 18, n°5, 2006, pp. 910-941.

[Lu et al. 1999] Lu, S., Paris, C., Vander Linden, K. (1999) Toward the automatic construction of task models from objectoriented diagrams. Engineering for Human-Computer Interaction. Kluwer Academic Publishers. 169-180.

[Paterno, 1997] Paternò, F. Formal Reasoning about Dialogue Properties with Automatic Support Interacting with Computers, August 1997

[Paternò et al., 1999] Paternò, F., Breedvelt-Schouten and N. de Koning. (1999). Deriving presentations from task models. In Engineering for Human-Computer Interaction. Kluwer Academic Pub. pp. 319-338.

Assessability

GA1 - Integration of modelling in the engineering process

1- Definition

Model-driven engineering refers to the systematic use of models as primary artefacts throughout the engineering lifecycle. Model-driven engineering technologies offer a promising approach to address the complexity of computing platforms and express domain concepts effectively [Schmidt, 2006]. Models (especially models with formal semantics) allow not only an unambiguous specification and design but also open the way to automated, mathematically precise resilience verification and evaluation as well.

2- Examples / Application Domains

The approach of model based design, and particularly the Model-Driven Architecture (MDA) [Mellor et al., 2004] initiative of the Object Management Group (OMG) leads to novel design techniques and supporting tools like model based code generation [Samek, 2003][Mellor & Balcer, 2002], model based formal verification [Clarke & Wing, 1996], and model based testing [Broy et al., 2005]. The adoption of platform-independent models (PIMs) for specifying functionality and behaviour, and platform-specific models (PSMs) to deal with the implementation technology has been used successfully in the design processes of, among others, distributed e-business applications [Frankel, 2005], critical web services, and safety-related embedded real-time systems [Herzner et al., 2006]. Moreover, modelling deserves attention not only in the design phases but also in the operational phases like system management [Lanfranchi et al., 2003] and run-time optimization. One of the emerging trends is the application of Service Level Agreements (SLAs) that would benefit from model based quantitative evaluation to facilitate adaptability and evolvability and to improve the automation of IT infrastructure management.

3- Current approaches

The OMG's Unified Modelling Language (UML [Booch et al., 2005]) together with its profiles (e.g. QoS properties [OMG, 2006]) is the most known modelling language that forms the basis of MDA. Recently, the emergence of domain-specific languages (like AADL, the Architecture Analysis and Design Language [SAE, 2005]) can be observed. The core technology of model based design is model transformation. Transformations are used to map PIMs to PSMs, apply design patterns, project design models to various mathematical analysis domains and back-annotate the results [Czarnecki & Helsen, 2003].

Several model based techniques can be mentioned that can partially support the assessment of resilient systems, for example the verification of fault handling using model checking, formal reasoning on fault coverage, deriving dependability models for the purpose of quantitative evaluation. The set of techniques is far from being complete and it is continuously extended. From modelling point of view, some drawbacks and limitations of the existing solutions can also be mentioned. Typically only the core aspects of static, fault-free functionality are described using standardized modelling languages. The specification of resilience properties and the modelling of the notions addressed in the design of resilient systems (e.g., faults and threats, adaptive behaviour) are ad-hoc, informal and incomplete. The cooperation of these techniques, the seamless extension of them (e.g., to include new types of faults) and the adaptation to new application domains is difficult or often impossible.

4- Research challenges

Formalizing resilience properties. Languages shall be developed to allow the specification of the required characteristics of resilient systems, often referred to as quality of service (QoS) properties and used in SLAs. The languages shall have formal background (to allow automated verification and evaluation) and shall seamlessly fit (both by their structure and notation) into system models that use a standard modelling notation. QoS models should have a well-defined common metamodel which enables comparison of properties (e.g., to check offered and required QoS on the interfaces of system components) and automated translation to specific formats used in existing tools (e.g., in SLA monitoring tools).

Modelling extensions for resilient systems. Modelling extensions are needed to describe application-platform interaction, adaptive behaviour, evolving functionality and usability. Specifically, the means for systematic fault modelling (e.g., by introducing the notion of faults at the metamodel level, which covers incidental and malicious faults) is one of the prerequisites for the model-based design and analysis of resilience techniques (e.g., augmentive and adaptive maintenance). Formal background of these extensions is needed to allow mathematically precise verification of the design decisions and application of model-based run-time decision making in the operation phase.

Domain-specific modelling languages and model transformations. Domain-specific modelling languages have to be developed to help designers and system operators. Then the existing verification and validation techniques shall be adapted to these models, which calls for property-preserving model transformations towards the analysis models. For example, system management and the corresponding infrastructure is one of the domains that needs the development of an efficient modelling methodology.

Multi-view modelling. The complexity of IT systems and resilient infrastructures necessitates the development of hierarchical and multi-view system modelling techniques, which immediately raises the problem of consistency and management of multiple models with potentially different formalisms. The semantic mapping and proof of consistency (although profited from new concepts like metamodels and the use of ontologies) is still an open research area. Specifically in resilient systems targeted by ReSIST, the interaction between the traditionally separated views of availability, security and safety (all with their own modelling and verification techniques) is a crucial point in the modelling process.

5- References

[Schmidt, 2006] Schmidt, D.C. (2006): Model-Driven Engineering. IEEE Computer 39 (2), February 2006.

[Mellor et al., 2004] Mellor, S., Scott, K., Uhl, A., Weise, D. (2004): MDA Distilled, Principles of Model Driven Architecture. Addison-Wesley Professional, 2004.

[Samek, 2003] Samek, M. (2003): Practical Statecharts in C/C++. CMP Books, 2003.

[Mellor & Balcer, 2002] Mellor, S., Balcer, M. (2002): Executable UML: A foundation for model-driven architecture, Addison Wesley, 2002

[Clarke & Wing, 1996] Clarke, E. M., Wing, J. M. (1996): Formal Methods: State of the Art and Future Directions. ACM Computing Surveys, 28 (4), December 1996.

[Broy et al., 2005] Broy, M., Jonsson, B., Katoen, J-P., Leucker, M., Pretschner, A. (eds.) (2005): Model-Based Testing of Reactive Systems. Advanced Lecture Series, LNCS 3472, Springer-Verlag, 2005.

[Frankel, 2005] Frankel, D. S. (2005): Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley & Sons.

[Herzner et al., 2006] Herzner, W., Huber, B., Balogh, A., Csertán, Gy. (2006): The DECOS Tool-Chain: Model-Based Development of Distributed Embedded Safety-critical Real-time Systems. DECOS/ERCIM Workshop, SAFECOMP 2006, Gdansk, Sept. 26, 2006.

[Lanfranchi et al., 2003] Lanfranchi, G., Della Peruta, P., Perrone, A., Calvanese, D. (2003): Toward a new landscape of systems management in an autonomic computing environment. IBM Systems Journal, Vol 42, No 1, 2003.

[Booch et al., 2005] Booch, G., Rumbaugh, J., Jacobson, I. (2005): Unified Modeling Language User Guide (2nd Edition), Addison-Wesley Object Technology Series, 2005.

[OMG, 2006] OMG (2006): UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms v1.0, http://www.omg.org/cgi-bin/doc?formal/06-05-02.

[SAE, 2005] SAE (Society of Automotive Engineers) (2005): Architecture Analysis and Design Language. SAE-AS5506, Warrendale, PA, 2004. http://www.aadl.info/

[Czarnecki & Helsen, 2003] Czarnecki, K., Helsen, S. (2003): Classification of Model Transformation Approaches. In Proc. of the OOPSLA'03 Workshop on the Generative Techniques in the Context Of Model-Driven Architecture, Anaheim, California, USA, 2003.

GA2 - Data Selection, Collection, Validation

1- Definition

We need more empirical data to drive quantitative modelling in all areas. This lack of data impedes the development of research aiming at calibrating and/or validating models. The collection of such unbiased, representative and useful data is an open issue. It is particularly evident in the domain of computer security where the lack of data on ongoing attacks does not enable the security actors to i) identify the most important problems to work on first, ii) assess the adequacy of adopted solutions with respect to existing threats, iii) compare the merits of various solutions with respect to sound fault assumptions.

2- Examples/Application Domains

Business decisions can be directly impacted by an underestimation of computer security risks. Unfortunately, the identification, understanding, and modelling of the threats represent unexplored research territories. Since 2003, the nature of attacks seems to have shifted from servers to clients and from fast spreading worms to commercially more interesting activities like theft, fraud, phishing, online gambling, extortion, etc. Nobody knows to what extent this is true and, if true, how prevalent the problem is. The answers to such questions are extremely important, as they help decision makers to invest in the appropriate security measures. Without such knowledge, security decisions tend to be stabs in the dark.

3- Current Approaches

A very large number of initiatives have been taken to collect all kinds of data related to computer security threats and incidents. Unfortunately, very few efforts are made to share the collected information. Some approaches aim at collecting and sharing various types of security logs (e.g., [SANS, Dshield, mynetwatchman]). Similarly, several anti-virus software producers have deployed their own data collection environment (e.g., Symantec with its DeepSight Early Warning Services and IBM with its Xforce threats alert system). Darknets and Internet Telescopes, also collect information about observed attacks on the Internet [Caida, Internet motion sensor, atlas, Nicter]. Last but not least, a new class of honeypot-based data collection tools aims at capturing malware binaries (i.e. mostly worms) [Nepenthes, Mwcollect, Werner, Riordan et al. 2006]. From an analytical point of view, several attempts have been made to find models of worm propagation [Serazzi & Zanero 2004] and malware activities, but the modeling techniques are usually limited to old, well known, threats and to some very popular worms (e.g. [Chen et al. 2003, Weaver et al. 2003, Zou et al. 2002]).

4- Research Challenges

As we can see, the problem is not so much the absence of data as the unavailability of "good" data with respect to the announced objectives. First of all, one must have a clear understanding of what data are required and what properties they must satisfy in order to be usable by those who need them. Does each model, each approach require its own dataset provided by a specific environment or, at the contrary, can we imagine and develop some generic monitoring and data collection tools to cope with our needs? How can we assess the quality of the data captured? Does time impact their quality, i.e., do we need to use "fresh" data or can we think of producing some stable "standard" dataset like the ones used in benchmarking approaches? Quite likely, capturing information about attacks is not enough. What is really needed by models is some meta information about the observed threats. How can we process data obtained in order to extract useful information from them? Do we need to instrument the systems we are designing in order to obtain information to be used for evaluation purposes? If yes, how?

Assuming we can obtain such data, sharing them with interested people may be problematic if the data contain sensible or confidential information. We need sanitization methods that preserve the properties of the data that are needed by the "consumers". Is it possible? Always?

Assuming systems do not evolve, how can we assess at a given point in time that its defenses are "good enough"? How do we define the dependability and security objectives (i.e., the failures), the most likely successful attack processes (i.e., the failure modes), the most likely vulnerabilities exploited (i.e., the injected or activated faults, according to defined fault models). More importantly, what reasonable fault models do we have at hand for malicious faults? We need to investigate the feasibility of using observed attacks in "representative" (?) environments to create "benchmarks" to assess the resilience of our systems.

5- References

[SANS] SANS. The sans internet storm center - http://isc.sans.org.

[Dshield] Dshield distributed intrusion detection system, www.dshield.org.

[mynetwatchman] mynetwatchman - www.mynetwatchman.com.

[Caida] Caida project, www.caida.org.

[Internet motion sensor] Internet motion sensor, Univ. of Michigan, ims.eecs.umich.edu

[atlas] Active threat level analysis system (atlas). http://atlas.arbor.net.

[Nicter] Nicter project. https://www2.nict.go.jp/y/y211/e index.html.

[Nepenthes] Nepenthes malware collection, nepenthes.mwcollect.org.

[Mwcollect] Mwcollect. The mwcollect alliance - alliance.mwcollect.org

[Werner] T. Werner. Honeytrap - http://honeytrap.sourceforge.net/.

[Riordan et al. 2006] J. Riordan, D. Zamboni, and Y. Duponchel, "Building and deploying billy goat, a wormdetection system", Proc of the 18th FIRST Conference, 2006

[Serazzi & Zanero 2004] G. Serazzi and S. Zanero. Computer virus propagation models. In M. Calzarossa and E. Gelenbe, editors, Performance Tools and Applications to Networked Systems, Revised Tutorial Lectures, volume 2965 of Lecture Notes in Computer Science, pages 26–50. Springer, 2004.

[Chen et al. 2003] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In Proceedings of IEEE INFOCOM, 2003

[Weaver et al. 2003] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms", Proc. of WORM '03, ACM 2003.

[Zou et al. 2002] C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis", Proc of the 9th ACM CCS , 2002.

GA3 - Dependability Cases

1- Definition

A dependability case is a structured argument based on assumptions and evidence, which supports a claim that a system meets its specified dependability requirements at a particular level of confidence.

2- Examples/Application Domains

It is now commonplace that developers of safety critical systems are required to produce a corresponding safety case – communicating an argument, supported by evidence that a system is acceptably safe to operate. Safety is an attribute of dependability, hence a safety case is a special form of a dependability case. Examples of application domains for such systems include the defence and railway sectors. In such domains the description of requirements for the safety case product, as well as the processes for development, are described by the respective safety standards such as the U.K. Defence Standard 00-56 [b] and the U.K. Health and Safety Executive Railway Safety Case Regulations [Health 2000]. Dependability Arguments are much less mature, but a notable recent use of a Dependability Argument was to assess the dependability of the Irish Electronic Voting machines carried out by QinetiQ on behalf of the Irish Independent Commission on Electronic Voting [c].

3- Current Approaches

Having existed as a core requirement of safety standards for many years, safety case development practice has significantly matured over the last decade and is now widespread in safety engineering. Today, safety cases are well supported by development methodologies and notations such as the Goal Structuring Notation (GSN) [Kelly 1998].

In overall terms a safety case should provide an argument, accompanied by evidence, that all safety concerns and risks have been correctly identified and mitigated. However, a safety case ultimately addresses only one attribute – safety. In addition to safety, there are also other important system attributes that need to be addressed, such as maintainability and security. A security assurance case [a] also uses a structured set of arguments and a corresponding body of evidence to demonstrate that a system satisfies specific claims with respect to its security properties.

Dependability is a system property that is perceived to encompass these wider attributes. The idea of arguing more formally about dependability is a logical extension of the practice of safety cases.

4- Research Challenges

As we remarked in ReSIST's earlier deliverable, there are several reasons why it is difficult to build convincing dependability cases for computer-based systems. One of these is that dependability claims are inevitably dependent upon very disparate evidence: this can range from informed expert judgment to hard empirical evidence. Another is the ubiquity of uncertainty (e.g., in the "strength" of evidence, in the correctness of assumptions and in the modes of reasoning). Understanding how such uncertainty propagates through a case is hard, particularly when we want to do this quantitatively via probabilities, which is necessary when the case is supporting some wider evaluation, e.g., for risk assessment. In the following we discuss some of the important issues that we believe would benefit from research in the next few years.

4.1 Claim decomposition issues

The first problem faced by someone building a dependability case is, of course, to decide exactly what is to be claimed, and the detail of how such claims should be expressed. This issue is closely related to the composition of dependability properties. Claims for a particular system will be derived from the requirements of a wider system, and the fulfilment of the claims for a system will depend, in turn, upon the satisfaction of claims at a lower level. Important issues needing further work include:

- Relationship between compositional reasoning and dependability decomposition of claims.
- Trade-offs between what is needed, and what is feasible.

4.2 Confidence

A key idea, as we said in an earlier ReSIST deliverable (D12: Part Eval - Methods and Tools for Resilience Evaluation), is that there is inherent uncertainty that needs to be handled in dependability cases: we cannot claim that dependability claims are true with certainty. Such uncertainty arises from many sources: examples include doubts about assumptions upon which a dependability case will be founded (e.g. correctness of an oracle for a case based upon testing); strength of evidence (e.g. number of test cases, and number of these that are correct), and so on. A particular problem concerns the epistemic nature of much of this uncertainty – i.e. it concerns uncertainty of "knowledge about the world". Typically this involves an inevitable subjectivity and is harder to handle than aleatory uncertainty – i.e. "natural" uncertainty or randomness (such as that involved, for example, in a dependability claim itself). Particular issues that require addressing are:

- Need for a formal, rigorous (claim, confidence) calculus that supports quantitative and qualitative dependability arguments.
- Need for case studies to try out these new formalisms.
- Issues of validation: what does it mean to say "I trust this dependability case?" Validation of expert belief versus real-world validity..

4.3 Diverse arguments

A particularly intriguing approach to gaining confidence in dependability claims mimics the fault tolerance approach used to gain reliability in systems. This multi-legged argument approach [Bloomfield & Littlewood 2003, Bloomfield & Littlewood 2006] has been proposed as a way forward in a couple of standards. Not surprisingly, some of the difficulties experienced in system fault tolerance – e.g. the implausibility of claims for independence between subsystem failures – also have their counterparts here. For example, a dependability claim supported by two argument legs, each of which singly incurs 10% doubt in the truth of the claim, will not induce 99% confidence from the two-legged argument.

In fact, early work on this problem suggests that there may be some unexpected subtleties here. For example, it has been shown that in certain circumstances it may be possible for the addition of a second supportive argument leg to reduce confidence in a dependability claim. Particular issues that require addressing are:

- Better formal models and empirical evidence are needed (e.g., in practice, when real experts are involved, can we expect the counter-intuitive results not to apply?) [Littlewood & Wright 2007].
- Need for much more realistic examples.
- Scalability of models to support diverse arguments.

4.4 Modeling the role of humans

There is a growing awareness that (almost) all systems are socio-technical in nature. The role of humans in the overall dependability of systems needs much more sophisticated modelling than it has received so far, especially in the area of confidence in claims that are made. Early results suggest that there may be interesting aspects of diversity between humans and "computers" – e.g. they may differ usefully in what they find "difficult" in certain problems.

- Need for empirical input from psychologists (and sociologists?)
- Better understanding of the different nature of uncertainty about human behaviour and (technical) system behaviour, e.g. variation between humans and nature of human failures
- How do dependability cases for socio-technical systems differ from those for technical systems? Are there fundamental limitations here?

4.5 Evolution

Shifting the focus from only one attribute (i.e., safety) and attempting to address all attributes equally, poses two main obstacles that need to be overcome: the evolution of a dependability argument, and the resolution of conflicts between evolving attribute objectives. To evolve or maintain the dependability argument the rationale for trade-off between conflicts between, for example, security and safety has to be recorded. Without the rationale for a trade-off it becomes difficult to evolve or maintain a dependability argument because the analysis that led to the original trade-off has to be repeated from scratch.

5- References

[Kelly 1998] T. Kelly, Arguing Safety A Systematic Approach to Managing Safety Cases (1998). PhD Thesis, available at http://www.cs.york.ac.uk/ftpdir/reports/YCST-99-05.ps.gz

[a] https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/assurance.html

[b] http://www.dstan.mod.uk/data/00/056/01000300.pdf

[Health 2000] Health and Safety Executive. Railway Safety Case Regulations – Guidance on Regulations, HSE Books, 2000.

[c] http://www.cev.ie/htm/report/index.htm

[Bloomfield & Littlewood 2003] R. E. Bloomfield and B. Littlewood, "Multi-legged arguments: the impact of diversity upon confidence in dependability arguments," presented at International Conference on Dependable Systems and Networks (DSN), San Francisco, 2003. http://doi.ieeecomputersociety.org/10.1109/DSN.2003.1209913

[Bloomfield & Littlewood 2006] R. E. Bloomfield and B. Littlewood, "On the use of diverse arguments to increase confidence in dependability claims" in Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective", (editors) D. Besnard, C. Gacek and C. B. Jones, pp 254-268, Springer, 2006, ISBN 1-84628-110-5

[Littlewood & Wright 2007] B. Littlewood and D. Wright, "The use of multi-legged arguments to increase confidence in safety claims for software-based systems: a study based on a BBN analysis of an idealized example", IEEE Trans Software Engineering, vol 33, no 5, pp 347-365, 2007.

GA4 - Security quantification

1- Definition

The objective of security quantification is to define quantitative metrics and appropriate methods to evaluate them. These metrics can be used to assess the intensity of malicious threats faced by operational systems and the capacity of these systems to resist to potential attacks. Also, the objective is to use such metrics to support tradeoffs and the decision process during the system operation phase as well as during the development.

2- Examples/Application Domains

Security evaluation is currently qualitative, based more on the development process than on the developed product(s). However, there is a need for quantitative evaluation that includes the definition of effective metrics to aid decision makers, taking into account technical, business and economic concerns. Work has been performed for at least two decades, but no agreement currently exists.

3- Current Approaches

Four main classes of evaluation methods can be distinguished to analyse and assess the impact of malicious threats on system resilience: 1) security evaluation criteria, 2) risk assessment, 3) model-based quantitative evaluation, and 4) experimental evaluation.

The most commonly used approach is based on the assumption that a proper design and development process would be sufficient to prevent damage from malicious threats. Accordingly, several security evaluation criteria have been developed to assess the security of computer based systems and compare their ability to cope with malicious faults, e.g., TCSEC, ITSEC and the common criteria. These criteria are qualitative and take into account the functional aspects of the system and the methods used in the development process. As regards risk assessment approaches, the goal is to analyse and determine the likelihood that identifiable threats will affect the target system security, together with the severity of the damage they might cause. Damage is generally assessed by evaluating the average loss of money an attack might cause.

Security evaluation criteria and risk assessment methodologies are widely recognized to be insufficient: a) to assess in operation the impact of malicious attacks and vulnerabilities on the security of systems or b) to support architecture design tradeoffs based on quantitative criteria. Particular attention has been paid during the last decade to exploring new approaches for security evaluation aimed at addressing these objectives based on probabilistic models similar to those used in computer dependability (see e.g., the ReSIST state of knowledge deliverable for a survey of such approaches [ReSIST 2006]). We can mention in particular the pioneering studies carried out in the 90's by some ReSIST partners in an attempt to develop a) representative quantitative "measures" characterizing the capacity of a system to resist to attacks during operation and b) probabilistic methods for security evaluation analogous to those used for the evaluation of reliability (see e.g., [Littlewood et al. 1993] [Dacier et al. 1996] and [Ortalo et al. 1999] which introduced the concept of effort to derive quantitative security metrics and proposed stochastic models for their evaluation). Other approaches have emerged more recently (see e.g., the survey presented in [Nicol et al. 2004]). We can mention in particular the state-based models proposed in [Madan et al. 2002, Gupta et al. 2003] targeting intrusion-tolerant systems. Other types of stochastic models have been also explored in the context of security. These include, for example, a) epidemiology models or interactive Markov chains for studying malware propagations (see e.g., [Staniford et al. 2002, Zou et al. 2005], b) models based on complex network theory to analyse the vulnerability of networked systems to cascading attacks [Crucitti et al. 2004], and c) game theory-based models for characterizing attackers behaviours and for analyzing impact on security [Lye & Wing 2005]. Additionally, we can mention other recent trends that address the problem of cyber insurance [Böhme 2005] and security evaluation and analysis from economic [Anderson & Moore 2007] and business perspectives [Knorr & Röhrig 2000].

In addition to the model-based evaluation approaches mentioned above, experimental studies have also emerged for collecting and analyzing real data characterizing attacks and malicious activities on the Internet (see for example [Bailey et al. 2005], [CAIDA], [DShield], [Leurré.com]). Such data are useful for understanding attack behaviours and for elaborating realistic assumptions about malicious threats and attacker behaviours.

4- Research Challenges

Among difficulties is the representation of the mix of deterministic and uncertain phenomena that come into play when addressing the behaviour of systems faced with malicious attacks.

The central question is thus the very possibility of defining probability-theoretic metrics that could be associated to models for prediction. A related question is the definition of metrics obtained from measurements (either from deployed systems or from controlled experimentations) for helping improve the understanding of the behaviour of systems faced with malicious attacks. These two facets of metrication (models, measurements) should of course be consistent.

Another challenging issue concerns the development of analysis and modelling methodologies that 1) allow security evaluation to be used for driving design decisions, and 2) support the development of efficient early warning and intrusion detection systems that will enable system administrators to react efficiently to attacks. Methodologies are also needed to carry out trade-off analyses between security mechanisms and ensuing quality-of-service penalties. In particular, to support self-managing systems, such trade-offs must be made increasingly frequent in an on-line, automated fashion. This requires techniques to deal explicitly and objectively (that is, using quantitative metrics) with trade-off issues that would otherwise be based on intuition or subjective arguments. The development of such methodologies requires the definition of realistic assumptions about system vulnerabilities and attacker behaviours. Such assumptions could be derived based on the observations of real attacks using, for instance, widely deployed honeypots or intrusion detection systems.

So far the evaluation of dependability properties and security properties have generally been addressed separately. However, what we need is to evaluate the resilience of the services delivered to users taking into account the variety of threats that might affect these services. Clearly, there is a need for a comprehensive modelling framework that can be used to assess the impact of accidental faults as well as malicious threats in an integrated way.

5- References

[Anderson & Moore 2007] R. Anderson and T. Moore, "The Economics of Information Security: A Survey and Open Questions", in Fourth Bi-Annual Conference on the Economics of the Software Internet Industries, (Toulouse, France), 2007.

[Bailey et al. 2005] M. Bailey, E. Cooke, F. Jahanian and J. Nazario, "The Internet Motion Sensor - A Distributed Blackhole Monitoring System", in Network and Distributed Systems Security Symposium (NDSS-2005), (San Diego, CA, USA), 2005.

[Böhme 2005] R. Böhme, "Cyber-Insurance Revisited", in Workshop of the Economics of Information Security (WEIS) 2005, (Cambridge, MA, USA), 2005.

[CAIDA] CAIDA, "Home Page of the CAIDA project: http://www.caida.org".

[Crucitti et al. 2004] P. Crucitti, V. Latora, M. Marchiori and A. Rapisarda, "Error and Attack Tolerance of Complex Networks", Physical A, 340 (1-3), pp.388-394, 2004.

[Dacier et al. 1996] M. Dacier, Y. Deswarte and M. Kaâniche, "Models and Tools for Quantitative Assessment of Operational Security", in 12th International Information Security Conference (IFIP/SEC'96), (S. K. Katsikas and D. Gritzalis, Eds.), (Samos, Greece), pp.177-186, Chapman & Hall, 1996.

[DShield] DShield, "Home page of the DShield.org Distributed Intrusion Detection System: http://www.dshield.org. "

[Gupta et al. 2003] V. Gupta, V. V. Lam, H. V. Ramasamy, W. H. Sanders and S. Singh, "Dependability and Performance Evaluation of Intrusion Tolerant-Server Architectures", in First Latin-American Symposium on Dependable Computing (LADC 2003), (Sao-Paulo, Brazil), pp.81-101, IEEE Computer Society, 2003.

[Knorr & Röhrig 2000] K. Knorr and S. Röhrig, "Security of Electronic Business Applications: Structure and Quantification", in 1st International Conference on Electronic Commerce and Web Technologies EC-Web 2000, 2000.

[Leurré.com] Leurré.com, "Leurré.com project. Publications web page : http://www.leurrecom.org/paper.htm".

[Littlewood et al. 1993] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid and D. Gollmann, "Towards Operational Measures of Computer Security", Journal of Computer Security, 2, pp.211-229, 1993.

[Lye & Wing 2005] K.-w. Lye and J. M. Wing, "Game strategies in network security", International Journal on Information Security, 4 (1-2), pp.71-86, 2005.

[Madan et al. 2002] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan and K. Trivedi, "Modeling and Quantification of Security Attributes of Software Systems", in IEEE International Conference on Dependable Systems and Networks (DSN 2002), (Washington, DC, USA), pp.505-514, IEEE computer Society, 2002.

[Nicol et al. 2004] D. M. Nicol, W. H. Sanders and K. S. Trivedi, "Model-based Evaluation: From Dependability to Security", IEEE Transactions on Dependable and Secure Computing, 1 (1), pp.48-65, 2004.

[Ortalo et al. 1999] R. Ortalo, Y. Deswarte and M. Kaâniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security", IEEE Transactions on Software Engineering, 25 (5), pp.633-650, 1999.

[ReSIST 2006] ReSIST Deliverable D12 "Resilience-building Technologies: State of Knowledge". Available at http://www.resist-noe.eu/Publications/Deliverables/D12-StateKnowledge.pdf

[Staniford et al. 2002] S. Staniford, V. Pawson and N. Weaver, "How to Own the Internet in your Spare Time", in USENIX Security Symposium, pp.149-167, 2002.

[Zou et al. 2005] C. C. Zou, W. Gong, D. Towsley and L. Gao, "The Monitoring and Early Detection of Internet Worms", IEEE/ACM Transactions on Networking, 13(5), pp.961-974, 2005.

GA5 - Benchmarking

1- Definition

A dependability benchmark is intended to characterize system behaviour in the presence of faults. Potential faults include component failures, hardware or software design flaws, faults in other systems interacting with the benchmarked systems, malicious attacks, operator errors, and perturbations in the environment. Benchmarking the dependability of a system consists of evaluating dependability or dependability-and-performance-related measures in the presence of faults, in a well-structured and standardized way. Measures may characterize i) the system in a comprehensive way (that is, they may address the service delivery level and take into account the occurrence of various events impacting its behaviour and their consequences), or ii) specific features of the system such as coverage provided by fault tolerance mechanisms, time to system restart, or time to system backup.

2- Examples/Application Domains

Dependability benchmarks are intended for objective assessment of system dependability. Therefore, they provide a good means for fair comparison between alternative systems. They can be used by system purchasers and system integrators for supporting acquisition and design choices, and also by system constructors for guiding development efforts.

3- Current Approaches

Pioneer work on dependability benchmarking has been published in [Tsai et al. 1996, Mukherjee and Siewiorek 1997, Koopman and DeVale 1999]. Since then, a great variety of benchmarks, for several application domains, have been defined and implemented in the last decade.

DBench, a European project on Dependability Benchmarking [DBench] developed a framework for defining dependability benchmarks for computer systems, with emphasis on Off-the-Shelf components and on systems based on Off-the-Shelf components, via experimentation and modelling. To exemplify how the benchmarking issues can actually be handled in different application domains, a set of benchmarks and their associated implementations has been developed. They concern general-purpose operating systems [Albinet et al. 2004, Kalakech et al. 2004], embedded systems (automotive [Ruiz et al. 2004] and space applications [Moreira et al. 2004]), and transactional systems [Vieira and Madeira 2003]. The work of the Special Interest Group on Dependability Benchmarking [SIGDeB], of the IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance, started with an open-ended mission of exploration [Koopman and Madeira 1999], and evolved to consider a system vendor point of view [Wilson et al. 2002].

The University of Berkeley has defined a dependability benchmark to assess human-assisted recovery processes [Brown et al. 2002]. IBM developed benchmarks to quantify system autonomic capability [Lightstone et al. 2003], while Sun Microsystems defined a framework dedicated to availability benchmarking [Zhu et al. 2003] and to system recovery [Mauro et al. 2004].

4- Research Challenges

In general, dependability benchmarking is still a developing art and it is reasonable to expect that dependability benchmarking will take many years to reach maturity.

The development and application of benchmarks are still at an early stage in spite of major efforts and progress made in recent years. Indeed, well established and widely accepted dependability benchmarks, approved by recognized consortiums as in the case of performance benchmarks, do not really exist. Definition of standard dependability benchmarks, taking into account important benchmarking properties such as representativeness, repeatability and portability, are still needed.

It is worth noting that most of the advances made to date in the development of dependability benchmarks have concentrated on moderate size target systems. There is a need to extend these approaches to large-scale systems and infrastructures (i.e., there is a need for scalability). Also, it will be worth connecting dependability modelling and experimentation (to date no full scale approaches have been identified).

Another technical challenge comes from the fact that these large scale systems and infrastructures are continually evolving during operation, due i) to runtime changes in the environment, and ii) to demands of online deployment of services, and of reconfiguration. Dependability benchmarks will need to be adapted to cope with the changing nature of the real world.

Finally, so far, only accidental faults have been taken into account for dependability benchmarking. However, malicious faults are becoming one of the major sources of system failures. There is thus a need to develop benchmarks with respect to malicious faults as well. The main difficulties are related to the definition i) of meaningful security measures (see GA4 on security quantification), and ii) of realistic attack models to solicit the system and observe its behaviour. Such models can be based on collection and analysis of real attacks on real-life systems.

5- References

[Albinet et al. 2004] A. Albinet, J. Arlat and J.-C. Fabre, "Characterization of the Impact of Faulty Drivers on the Robustness of the Linux Kernel", Proc. Int. Conf. on Dependable Systems and Networks (DSN 2004), (Florence, Italy), pp.867-876, 2004.

[Brown et al. 2002] A. Brown, L. C. Chung and D. A. Patterson, "Including the Human Factor in Dependability Benchmarks", Proc. 2002 DSN Workshop on Dependability Benchmarking, (Washington, DC, USA), 2002.

[DBench] http://www.laas.fr/DBench

[Kalakech et al. 2004] A. Kalakech, K. Kanoun, Y. Crouzet and J. Arlat, "Benchmarking the Dependability of Windows NT, 2000 and XP", Proc. Int. Conf. on Dependable Systems and Networks (DSN-2004), (Florence, Italy), pp.681-686, 2004.

[Koopman and DeVale 1999] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems", Proc. 29th Int. Symp. on Fault-Tolerant Computing (FTCS-29), (Madison, WI, USA), pp.30-37, 1999.

[Koopman and Madeira 1999] P. Koopman and H. Madeira, Dependability Benchmarking & Prediction: A Grand Challenge Technology Problem", 1st International Workshop on Real-Time Mission-Critical Systems: Grand Challenge Problems, November 1999, 4 pages, Phoenix, AZ, USA.

[Lightstone et al. 2003] S. Lightstone, J. Hellerstein, W. Tetzlaff, P. Janson, E. Lassettre, C. Norton, B. Rajaraman, and L. Spainhower. "Towards Benchmarking Autonomic Computing Maturity." Proc. First IEEE Conference on Industrial Automatics (INDIN-2003), Banff, Canada, August 2003.

[Mauro et al. 2004] J. Mauro, J. Zhu and I. Pramanick. "The System Recovery Benchmark," in Proc. 2004 Pacific Rim International Symposium on Dependable Computing (PRDC 2004), pp. 271-280, Papeete, FrenchPolynesia, IEEE CS Press, 2004.

[Moreira et al. 2004] F. Moreira, D. Costa, M. Rodriquez, "Dependability Benchmarking of Real-Time Kernels for Onboard Space Systems", Proc. 15th International Symposium on Software Reliability Engineering (ISSRE 2004), Saint-Malo, France, November 2-5, 2004.

[Mukherjee and Siewiorek 1997] A. Mukherjee and D. P. Siewiorek, "Measuring Software Dependability by Robustness Benchmarking", IEEE Transactions of Software Engineering, vol.23 no.6, pp.366-376, 1997.

[Ruiz et al. 2004] J.-C. Ruiz, P. Yuste, P. Gi land L. Lemus, "On Benchmarking the Dependability of Automotive Engine Control Applications", Proc. IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 857-866, 2004.

[SIGDeB] http://www.laas.fr/~kanoun/ifip_wg_10_4_sigdeb/

[Tsai et al. 1996] T. K. Tsai, R. K. Iyer and D. Jewitt, "An Approach Towards Benchmarking of Fault-Tolerant Commercial Systems", Proc. 26th Int. Symp. on Fault-Tolerant Computing (FTCS-26), (Sendai, Japan), pp.314-323, IEEE CS Press, 1996.

[Vieira and Madeira 2003] M. Vieira and H. Madeira, "A Dependability Benchmark for OLTP Application Environments", Proc. 29th International Conference on Very Large Data Bases (VLDB 2003), pp. 742-753, Berlin, Germany, 2003.

[Wilson et al. 2002] D. Wilson, B. Murphy and L. Spainhower. "Progress on Defining Standardized Classes of Computing the Dependability of Computer Systems," Proc. DSN 2002 Workshop on Dependability Benchmarking, pp. F1-5, Washington, D.C., USA, 2002.

[Zhu et al. 2003] J. Zhu, J. Mauro and I. Pramanick. "R3 - A Framwork for Availability Benchmarking," Proc. Int. Conf. on Dependable Systems and Networks (DSN 2003), pp. B-86-87, San Francisco, CA, USA, 2003.

GA6 - Model complexity

1- Definition

Applying model-based resilience evaluation to increasingly complex systems leads unavoidably to complex and large models. Model complexity is a general term indicating that the model construction process and/or the model solution process are "difficult" to perform. These difficulties come from some system's characteristics, like heterogeneity and largeness, which finally result in well known problems like state-space explosion, other problems such as stiffness for the analytical model solution, the rare event problem for simulation and intrusiveness of the monitoring system for the experimental evaluation methods.

2- Examples/Application domains

The model complexity problem is not domain-specific; it can be encountered in many application domains. Systems particularly affected are those showing **heterogeneity** in terms of available network connections (GPRS/UMTS/WLAN), types of hardware/software components (ad-hoc/off-the-shelf), and types of faults (accidental/malicious) [HIDENETS 2006]. Another notable example is that of critical infrastructures where modelling the interdependencies between the payload infrastructure (power or gas or water generation and distribution) and the computer and communication control infrastructure leads to very high complexity [CRUTIAL 2006].

3- Current approaches

Several techniques have been developed to cope with the model complexity issue; they can be grouped in two complementary sets [Nicol et al., 2004]: **largeness avoidance** and the **largeness tolerance** techniques. The techniques belonging to the first set try to circumvent the generation of large models using many different approaches like state truncation methods (e.g., [Muppala et al. 1992, Obal 1998]), lumping techniques (state-level, model level or compositional) (e.g., [Buchholz 1994, Derisavi et al. 2003]), and model decomposition and approximate solution techniques (e.g., [Courtois 1977, Lanus et al. 2003, Kaâniche et al. 2003]). However, largeness avoidance techniques alone may be insufficient, and, thus, largeness tolerance techniques are needed supporting practical generation of large state-space models by structured composition and hierarchical modeling approaches (e.g., [Sanders and Meyer 1991, Betous-Almeida and Kanoun 2004]). The basic idea is to build the system model from the composition of submodels describing system components and their interactions.

Both techniques are generally used in combination when detailed and large dependability models need to be generated and processed to evaluate dependability measures characterizing real-life systems.

4- Research challenges

Continued research in dealing with model complexity is needed, both in model construction and model solution. In addition, the properties to be evaluated for current ubiquitous computerized systems pose some new research challenges.

The complexity of the scenarios to be considered for the assessment of large, dynamic and heterogeneous systems calls for the development of **holistic evaluation approaches** including complementary evaluation techniques, covering simulation, analytical modeling and experimental measurements. Each of these methodologies could be successfully applied to analyze well confined parts of the system, but probably in isolation none of them can manage the huge model complexity. Mechanisms are needed to ensure the cooperation and the unified integration of these techniques, in order to provide realistic assessments of architectural solutions and of systems in their operational environments.

Evaluation methods must also consider **metrics at an increasingly high level of abstraction**, to express the impact of the computing infrastructure on an enterprise business. In this context it is crucial to determine the adequate level of detail of the model and abstract the remaining parts of the system. This may also require the cooperation among different evaluation techniques capturing the system’s behavior at different levels of abstraction.

Of increased significance is also the use of quantitative evaluation methods **to support the effective use of adaptation mechanisms** prevalent in modern-day systems. Mechanisms must be developed to monitor the conditions of the system’s environment and to dynamically adapt to their changes, and well-founded quantitative techniques are necessary to run such systems in resilient fashion.

Besides assessing the impact accidental threats, extensions are also needed **to quantify the impact of malicious threats**. In the last decade, malicious faults have received an increasing attention from the developers and the users of computer-based systems because of the exponential increase of reported vulnerabilities and malicious threats, and the openness of systems and their interconnection to the Internet. From the evaluation point of view, quantitative evaluation techniques have been mainly used to evaluate the impact of accidental faults on systems dependability. On the other hand, the evaluation of security has been mainly based on qualitative evaluation criteria. Clearly, there is a need for a **comprehensive modeling framework** that can be used to assess the impact of accidental faults as well as malicious threats in an integrated way.

5- References

[Betous-Almeida and Kanoun 2004] C. Betous-Almeida and K. Kanoun, "Construction and Stepwise Refinement of Dependability Models", Performance Evaluation, 56 (2004), Elsevier, pp.277-306, 2004.

[Buchholz 1994] P. Buchholz, "Exact and Ordinary Lumpability in finite Markov Chains", Journal of Applied Probabilities, 31, pp.59-74, 1994.

[Courtois 1977] P.J. Courtois, "Decomposability: Queuing and Computer System Applications", Academic Press, New York, 1977.

[CRUTIAL 2006] CRUTIAL Consortium (2006): IST FP6 STREP 027513 (CRUTIAL, CRitical UTility InfrastructurAL Resilience) http://crutial.cesiricerca.it/

[Derisavi et al. 2003] S. Derisavi, H. Hermanns, and W. H. Sanders, "Optimal State space lumping in Markov Chains", Information Processing Letters, 87 (6), pp.309-15, 2003.

[HIDENETS 2006] HIDENETS Consortium (2006): IST-FP6-STREP-26979 (HIDENETS: Highly DEpendable IP-based NET-works and Services). http://www.hidenets.aau.dk/

[Kaâniche et al. 2003] M. Kaâniche, K. Kanoun and M. Rabah, "Multi-level modeling approach for the Availability assessment of e-business applications", Software: Practice and Experience, John Wiley & sons, pp. 1323-41, 2003.

[Lanus et al. 2003] M. Lanus, L. Yin, and K. S. Trivedi, "Hierarchical Composition and Aggregation of State-based availability and Performability Models", IEEE Trans. on Reliability, 52 (1), pp.44-52, 2003.

[Muppala et al. 1992] J. K. Muppala, A. Sathaye, R. Howe and K. S. Trivedi, "Dependability Modeling of a Heterogeneous VAX-cluster System Using Stochastic Reward Nets", in Hardware and Software Fault Tolerance in Parallel Computing Systems (D. R. Avresky, Ed.), pp.33-59, 1992.

[Nicol et al. 2004] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based Evaluation: From Dependability to Security", in IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, pp 48-65, 2004.

[Obal 1998] W. D. Obal, "Measure-Adaptive State-Space Construction Methods", PhD Dissertation, Univ. of Arizona, 1998.

[Sanders and Meyer 1991] W. H. Sanders, J. F. Meyer, "Reduced Base Model Construction Methods for Stochastic Activity Networks," IEEE Journal on Selected Areas in Communications 9(1): 25-36, 1991.

GA7 - Metrics/models For Evolution Processes

1- Definition

Threats and defences together undergo a continuous process of competitive co-evolution. As new threats develop, new defences are created and deployed. New threats also follow the development of new information systems, networks, services and usages.

Adopting an evolutionary and dynamic view, the distinction between preventive and responsive measures becomes meaningless. Rather the essential security issue becomes the relative speeds with which the attacker and defender are able to adapt.

Unfortunately in this competition, defenders are generally constrained by the economic or contractual environment in which the protected system operates. An example of such constraints is seamless service availability and continuity. Attackers are generally not so constrained. It is thus essential that systems be designed and built according to principles that negate, or at least minimize, this inherent attacker advantage.

2- Examples/Application Domains

Formulation of metrics of system evolvability provides an essential input into the formulation of security strategies. The most difficult high-level problem faced by defenders is that of where to invest limited time and resources. Creation of security metrics, in particular differential security metrics, allows defenders to measure the return on security investment.

3- Current Approaches

Current approaches to security metrics tend to focus on system specification and are decided at design time, thus are static and absolute rather than dynamic and differential. We tend to measure compliance rather than the speed and cost of attaining and maintaining compliance. Numerous examples exist: percentage of machines at the most recent patch level, percentage of machines running anti-virus software, percentage of machines with no remotely detectable vulnerability.

While these figures are important, it is also important to know about the lag curve between the release of a patch and its installation, the lag between new viruses and matching detection capabilities, the lag between new virus detection capabilities and the installation of these, etcetera, as well as the factors that affect this lag (volume of signatures, mutation capability, communication availability, etc.).

Ultimately, this calls for characterizing, describing and evaluating the continuous co-evolution of all participants and contextual information. Processes aimed at ensuring stability and accountability often come with the cost of increased complexity and hence potentially reduced security. As such these processes need to be adapted to promote stability, simplicity and visibility while still allowing dynamism.

4- Research Challenges

The research challenges fall into two general categories. The first is formulation of metrics to evaluate the evolvability of a system. Many of the most basic such metrics are merely differential adaptations of existing metrics. The second category is creation of technologies that permit evolvability while ensuring service stability. Many examples already exist within this category, and are generally referred to as good engineering practices, but they are ad hoc and incomplete.

GA8 - Evaluation of dynamic systems

1- Definition

Specific applications and environments that intrinsically feature evolution dimensions such as mobile, ad hoc based, autonomous, self-organized and ubiquitous systems require that specific modelling and evaluation techniques be developed. Such techniques are needed to support the assessment of resilience architectural solutions and algorithms covering different layers (communication, middleware, application, etc.).

2- Examples/Application Domains

The target application domains concerned with the problems raised for this gap include mobile, ad hoc based, applications and services whose topology and use environment feature dynamic evolution that could be related to the mobility of the users or the target services. Examples include the application domains listed for the gap GE2 "Resilient Ambient Systems", e.g., domestic and infotainment applications (home entertainment, smart homes, assisted living support systems for elderly or handicapped persons), intelligent transportation and cooperative driving systems [Radimirsch et al. 06], cooperative robotics and nano-robotics for exploration, manufacturing, medecine and surgery, battlefield and rescue related applications based on sensor and mobile ad-hoc networks, and more generally cooperative services with both Internet-connected and disconnected periods [Killijian et al. 04], etc.

3- Current Approaches

To the best of our knowledge, little research has been done on the evaluation of ubiquitous systems. Currently, the common evaluation approach is simulation as several parameters and complex phenomena that need to be taken into account are difficult to describe analytically (geographic localisation of the nodes, communication range and accessibility of the nodes, mobility of the nodes, failure characteristics in wireless environments, etc.) [Chlamtac *et al.* 2003]. In most cases, the proposed algorithms are evaluated and validated using wireless network simulators such as NS-2, Glomosim or JiST/SWANS. Since simulators use a model of physical components, such as network cards and location systems, this raises concerns as to the representativity of the assumptions that underlie the simulation [Cavin et al. 02]. Real world experimental studies are rather limited due to the difficulty of setting up realistic test beds at a small scale [Jadhav *et al.* 2005, Tschudin *et al.* 2005]. Also, significant effort has been devoted to the development of mobility models that can be used to support the analysis and validation of mobile based dynamic systems [Camp *et al.* 2002, Bai & Helmy 2004]. As regards the objectives and the targets of the analyses carried out, attention has been focussed so far mainly on the assessment of communication and routing algorithms and architectural solutions considering performance measures. The development of assessment techniques for the quantitative evaluation of dependability and resilience properties, including in particular middleware and application layers is still at an early stage. However, we can mention the work presented in [Burnett & Rainsford 01] looking at a general approach to evaluating ubiquitous systems. In the paper, the authors argue that quantitative measurements should be complemented with qualitative evaluation. The argument is that there is a number of problems for which evaluation cannot be easily quantified. Thus an evaluation should be conducted using an hybrid quantitative/qualitative strategy. Also, the work presented in [Basu et al. 01] has investigated the definition of appropriate metrics for the evaluation of distributed applications running on ubiquitous systems. Finally, we can mention that a few studies dealing with the evaluation of users experience and human-computer interfaces in the context of ubiquitous systems also exist (see e.g., [Spasojevic & Kindberg 01]).

4- Research Challenges

The considered systems are dynamic in terms of topology, connectivity, and communication channels characteristics. Mobility contributes to most of the dynamics. Thus the development of quantitative assessment strategies with realistic mobility patterns and failure models is needed to provide a faithful validation of the quality of service provided by the target applications, services and communication protocols.

Simulation techniques are widely used for the quantitative assessment of dynamic and mobile-based systems and applications. However, several problems can be mentioned: 1) the validity of the results depends highly on the accuracy of the simulations, as discrepancies have been observed between the results obtained from different simulators commonly used in research studies on wireless communications and mobile environments [Cavin *et al.* 2002, Andel & Yasinsac 2006]; 2) analytical models are needed that are able to faithfully capture some of the relevant parameters and phenomena instead of estimating them through simulation. Another complementary approach that needs to be investigated, to improve the accuracy of resilience evaluation techniques in mobile and pervasive environments, concerns the development of experimental techniques that are able to emulate at a small scale the phenomena and behaviours occurring at a larger scale in real operational environments.

To this aim, a laboratory-scale realistic platform is necessary to evaluate and validate fault-tolerance algorithms (e.g., group membership and replication protocols, backup mechanisms, etc.) targeting systems comprising a large number of communicating mobile devices equipped with various sensors and actuators. The objective is to be able to build an experimentation platform allowing for reproducible experiments (including mobility aspects) that will complement validation through simulation. To build such a platform, one major challenge is related to changes of scale so as to emulate as many various systems as possible.

More generally, the complexity of the scenarios to be considered for the assessment of dynamic systems calls for the development of holistic evaluation approaches including complementary evaluation techniques, covering simulation, analytical modelling and experimental measurements. Mechanisms should be developed to ensure the cooperation and the unified integration of these techniques, in order to provide realistic assessments of architectural solutions and of systems in their operational environments.

5- References

[Andel & Yasinsac 2006] T. Andel and A. Yasinsac, "On the Credibility of MANET Simulations", Computer (July), pp.48-54, 2006.

[Bai & Helmy 2004] F. Bai and A. Helmy, Wireless Ad Hoc and Sensor Networks, Kluwer Academic Publishers, 2004.

[Basu et al. 01] Prithwish Basu, Wang Ke, and Thomas D.~C. Little. "Metrics for performance evaluation of distributed application execution in ubiquitous computing environments", Workshop on Evaluation Methodologies for Ubiquitous Computing at Ubicomp'01, 2001.

[Burnett & Rainsford 01] Mark Burnett and Chris P. Rainsford, "A hybrid evaluation approach for ubiquitous computing environments", Workshop on Evaluation Methodologies for Ubiquitous Computing at Ubicomp'01, 2001.

[Camp et al. 2002] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", Wireless Communication and Mobile Computing (WCMC) - Special Issue on mobile ad-hoc networking: research, trends and applications, 2, pp.483-502, 2002.

[Cavin et al. 2002] D. Cavin, Y. Sasson and A. Schiper, "On the Accuracy of MANET Simulators", in The Second ACM International Workshop on Principles of Mobile Computing (POM'C02), (Toulouse, France), pp.38-43, ACM Press, 2002.

[Chlamtac et al. 2003] I. Chlamtac, M. Conti and J. Liu, "Mobile ad hoc Networking: Imperatives and Challenges", Ad Hoc Networks, 1 (1), pp.13-64, 2003.

[Jadhav et al. 2005] S. Jadhav, T. X. Brown, S. Doshi, D. Henkel and R. G. Thekkekunnel, "Lessons Learned Constructing a Wireless Ad Hoc network Test Bed", in 1st Workshop on Wireless Network Measurements (WiNMee-2005), (Berlin, Germany), 2005.

[Killijian et al. 04] Marc-Olivier Killijian, David Powell, Michel Banâtre, Paul Couderc, and Yves Roudier. "Collaborative backup for dependable mobile applications.", In Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware 2004), pages 146--149, Toronto, Ontario, Canada, October 2004. ACM.

[Radimirsch et al. 06] M. Radimirsch, I. De Bruin, A. Casimiro, A. Fosslie Hansen, G. Huszerl, T. Ingvalsden, M. Kaaniche, M.-O. Killijian, M. Löbbers, E.V. Matthiesen, M. Reitenspiess, N. Rivière, I. E. Svinnset, H. Waeselynck. "Use-case scenarios and preliminary reference model.", Hidenets project, IST 26979, deliverable d1.1, Technical Report 06235, LAAS, Toulouse, 2006.

[Spasojevic & Kindberg 01] Mirjana Spasojevic and Tim Kindberg. Evaluating the cooltown user experience. Workshop on Evaluation Methodologies for Ubiquitous Computing at Ubicomp'01, 2001.

[Tschudin et al. 2005] C. Tschudin, P. Gunninberg, D. Lundberg and E. Nordstrom, "Lessons from Experimental MANET research ", Ad Hoc Networks, 3, pp.221-225, 2005.

GA9 - On-line Assessment for Resilience

1- Definition

On-line (run-time) assessment is concerned with system operating in highly dynamic and evolving environment (e.g. network congestion, new/changing service customers, effects of exploits of faults/vulnerabilities). Its goal is assessment of the implications of the changing environment on system resilience ranging from simply signalling a sub-standard operation to triggering a system's adaptation.

2- Current Approaches, and Examples/Application Domains

The traditional approaches to assessment, which dominate the current assessment practices – are: i) pre-deployment assessment, i.e. collecting data in a simulated environment (e.g. "statistical testing", "dependability benchmarking", etc.), and/or ii) processing the measurement data accumulated in real operation at a later stage, e.g. periodic reviews widely used in some safety-critical industries such as the nuclear sector.

These approaches have limitations. Pre-deployment assessment is limited by its nature – the impact on system dependability/performance cannot be known for unforeseen environments. Pre-deployment assessment, however, can provide *a priori* knowledge about how the system is expected to operate, especially if the simulated environment is "close" to the operational environment post-deployment.

Processing the measurements collected in real operation at a later stage, e.g. in periodic reviews, may be inadequate too: by the time the observations are processed the operational environment may have changed to yet something not seen before.

An example of on-line assessment combined with adaptation is the Model-based programming [Williams, B. C., *et al.*, 1996], [Williams, B. C., *et al.*, 2003], an approach proposed by Brian Williams and successfully applied to create "long-lived autonomous systems such as deep space explorers, distributed satellites, unmanned air vehicles, intelligent offices and automobiles".

Another example where on-line assessment, using Bayesian assessment, is known to be useful is management of a legacy system upgrade by replacing "old" components with newer releases [Gorbenko, A., *et al.*, 2005]. The actual replacement only takes place when the new components are shown (by the on-line assessment) to be sufficiently good to replace the old ones.

3- Research Challenges

The main research challenge is that the assessment should be *fast* and *accurate* at the same time, which is difficult to achieve. In more detail:

- Dealing with very large scale systems would make observability of the entire system problematic for any observer. Simplified models of the system (e.g. in the form "my part of the system is modelled in detail, the rest of the system as a monolithic whole") are likely to be used. When several simplified models are applicable, selecting the best one is not trivial. From other context (e.g. software reliability growth modeling [Brocklehurst, S., *et al.*, 1990]) it is known that the best predictive model is impossible to know in advance: which of the available alternatives is best depends on the model itself and on the data to which the prediction is applied. Objectively assessing the model's predictive quality in the context of large systems operating in a changing environment is a particularly important and hard problem to solve.

- All methods based on learning from observing the system will have problems providing speedy results. Attempts to make the models using methods based on crude approximation will be problematic in terms of accuracy and hence may trigger oscillating reconfigurations.
- Model-based programming's primary aim is to improve observability in an embedded system as a result of failures. It is unclear, whether the approach would work well in the context of common mode/cause failures (which may exploit widespread vulnerabilities likely to be present in large complex systems).
- Accurate measurements (data collection) is problematic, especially failure detection in the case of non-crash failures. Here, as our recent work shows [Gashi, I., *et al.*, 2007], diversity has great potential in terms of both failure detection and diagnosis of the failed components.

4- References

[Brocklehurst, S., *et al.*, 1990] Brocklehurst, S., Chan, P. Y., Littlewood, B. and Snell, J. (1990). Recalibrating software reliability models. IEEE Transactions on Software Engineering. **SE-16:** 458-470.

[Gashi, I., *et al.*, 2007] Gashi, I., Popov, P. and Strigini, L. (2007 (to appear)). Fault Tolerance via Diversity for Off-The-Shelf Products: a Study with SQL Database Servers. IEEE Transactions on Dependable and Secure Computing (TDSC).

[Gorbenko, A., *et al.*, 2005] Gorbenko, A., Kharchenko, V., Popov, P. and Romanovsky, A. (2005). Dependable Composite Web Services with Components Upgraded Online. Architecting Dependable Systems III. De Lemos, R., Gacek, C. and Romanovsky, A., Springer Verlag. **3549:** 92-121.

[Williams, B. C., *et al.*, 2003] Williams, B. C., Ingham, M. D., Chung, S. H. and Elliott, P. H. (2003). Model-based Programming of Intelligent Embedded Systems and Robotic Space Explorers. IEEE Special Issue on Modeling and Design of Embedded Software. **9:** 212-237.

[Williams, B. C., *et al.*, 1996] Williams, B. C. and Nayak, P. (1996). A Model-Based Approach to Reactive Self-configuring Systems. AAAI-96, Menlo Park, CA, USA, AAAI Press 971-978.

GA10 - Trust and Cooperation

1- Definition

The development of ubiquitous systems is generating increasingly complex applications in which self-organization and/or interaction of multiple components of the system is required in order to achieve both performance and fault-tolerance. In such systems, trust should rely on the observation of the behaviour of entities with respect to the rest of the system rather than identification through semantic-free names.

2- Examples/Application Domains

Cooperation based trust assessment is being used in several famous large scale applications, for instance peer-to-peer file sharing [Lee et al. 2003], secure routing in mobile ad-hoc networks [Marti et al. 2000], and Internet auctions and e-commerce services such as eBay and Amazon. Assessment in such applications is relatively simple however. Cooperative backup [Killijian et al. 2004] is another good example of application domain in which cooperation assessment is rendered more difficult by the non-atomicity of the backup operation.

3- Current Approaches

Assessing cooperation represents an essential task for achieving resilience that makes it possible to ensure the availability of the monitored resources. Cooperation assessment can be achieved based on either reputation mechanisms or remuneration schemes. Such assessment makes it possible to detect and segregate malicious entities. Incentives based on assessment are also introduced to foster participant cooperation. The efficiency of the assessment is generally validated through game theoretic models. Because of the large scale of ubiquitous systems, cooperation assessment is generally distributed, as described in more detail in [Oualha & Roudier].

4- Research Challenges

Cooperation assessment mechanisms adapted to various application dynamics should be investigated as well as their performance. In particular, many interactions develop over several phases, while the methods developed so far only address the instantaneous observation of cooperation. Important issues in this area are the use of non-atomic observations, and the introduction of more dynamic game theoretical tools and of cooperation logics.

Distributing cooperation assessment opens new alleys for coordinated attacks or even plain denial of service, as illustrated by Sybil attacks [Douceur 2002]: cooperation assessment mechanisms therefore have to be designed carefully so as to resist such attacks. Developing collusion avoidance techniques in reputation systems represents a central issue in this respect. Probabilistic fault-tolerance techniques might provide new solutions, but the performance of these remains to be demonstrated. Multi-layer models for reputation might also prove worthwhile investigating. Concerning remuneration schemes, research is also required in designing distributed algorithms that protect from double spending and ensure fair exchange and at the same time can be integrated easily in various applications.

5- References

[Lee et al. 2003] S. Lee, R. Sherwood, B. Bhattacharjee. "Cooperative peer groups in NICE". In INFOCOM'03, April 2003

[Marti et al. 2000] S. Marti, T.J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks", *Mobile Computing and Networking 255–265*, 2000

[Killijian et al. 2004] M.-O. Killijian, D. Powell, M. Banatre, P. Couderc, Y. Roudier, "Collaborative backup for dependable mobile applications," Proc. 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (a Workshop of IEEE/IFIP Middleware 2004), Toronto, Ontario, Canada, October 18-22, 2004.

[Oualha & Roudier] N. Oualha and Y. Roudier. "Cooperation Incentive Schemes". Section Algo5, ReSIST Deliverable D12 "Resilience-Building Technologies: State Of Knowledge".

[Douceur 2002] J. R. Douceur. "The sybil attack". In *Proc. of the IPTPS02 Workshop, Cambridge*, MA (USA), March 2002

GA11 - Verification of Mobile Computing Systems

1- Definition

Mobile computing calls for the use of mobile devices (handset, PDA, laptop, etc.) that move within some physical areas, while being connected to networks by means of wireless links (Bluetooth, IEEE 802.11, GPRS, etc.). Mobile-based applications and services may involve device-to-device or device-to-infrastructure communication. They have to be aware of, and adapt to, contextual changes.

2- Examples/Application Domains

Mobile computing technology enables a wide variety of applications, ranging from location-based infotainment services to critical services in disaster-response situations.

3- Current Approaches

There is currently no established framework for specifying, designing and verifying such applications.

Some low-level formalisms based on process algebras, like "Mobile Ambients" [Cardelli and Gordon 2000a], offer a core set of primitives to represent mobility. Basically, the mobility primitives consist of entering and exiting administrative domains that are hierarchically organized. This view is mostly adequate to express logical mobility (mobile computation), but physical mobility requires further investigation, e.g., to account for dynamic ad-hoc networking. Also, low-level process algebras are more designed for theoretical studies than for specifying systems. Some authors have proposed UML extensions loosely inspired from the ambients view of mobility [Baumeister *et al*. 2003] [Grassi *et al. 2004*] but none of these extensions has been given a formal semantics.

As regards testing, contributions came from the protocol testing community using the SDL modelling language [Cavalli *et al.* 2004, Ngani Noudem and Viho 2005]. Since SDL does not offer mobility-related concepts, the authors had to use modeling tricks. Model-based test generation can then be used but requires a baseline scenario to be selected a priori (e.g., a predetermined number of nodes and a few predetermined topology changes). The determination of the baseline scenario remains an open issue. From a more technological viewpoint, other work is focused on the implementation of testing using platforms based on emulation/simulation facilities [Satoh 2003, Morla and Davies 2004].

4- Research Challenges

Adequate specification and design formalisms still need to be proposed for mobile computing systems. Besides a core framework (e.g., process algebras) there is a need for high level formalisms usable by engineers. The formalisms should allow the specification of mobile systems according to multiple views: behavioural models of fixed and mobile nodes, declarative properties at system level (with possibly both temporal and spatial modalities [Cardelli and Gordon 2000b]), and scenarios/use cases. They should offer convenient abstraction for the relationships of nodes (e.g., who is connected with whom) as well as for the dynamic evolution of the system structure (e.g., a mobile node can become alive at any time and any location, future location changes are constrained by some mobility model). Assuming such formalisms become available, with their semantics defined based on some theoretical framework, an open issue is how to tackle formal verification (see e.g., undecidability and complexity results from [Charatonik et al. 2003]).

Testing also faces the unavailability of a modelling framework to account for mobile settings. As a result, model-based test generation techniques and associated coverage criteria still need to be defined. A mix of deterministic and probabilistic test approaches could be used to explore a sample of potential behaviours. Finally, technological problems associated with implementing the test experiments should not be underestimated. Complex test platforms may be required to possibly account for a variety of equipments (mobile devices, fixed infrastructure), communication between them, mobility models, 3D-models of geographical areas, etc. The cost of field testing, controllability constraints and observability constraints, imply that part of the testing activities will rely on emulation/simulation facilities. This raises the issue of the representativeness with respect to real operational conditions. For example, it is well known from the evaluation community (see also GA8 *Evaluation of dynamic systems*) that two different network simulators may yield experimental assessment results that strongly diverge [Cavin *et al.* 2002]. The impact on verification results is still to be investigated.

5- References

[Baumeister *et al.* 2003] H. Baumeister *et al.* "UML for Global Computing". *Global Computing: Programming Environments, Languages, Security, and Analyisis of Systems, GC 2003*, LNCS 2874, Springer-Verlag Berlin Heidelberg, 2003, pp. 1-24

[Cardelli and Gordon 2000a] L. Cardelli and A. Gordon, "Mobile ambients", *Theoretical Computer Science*, 240(1):177–213, 2000.

[Cardelli and Gordon 2000b] L. Cardelli and A. Gordon, "Anytime, anywhere: Modal logics for mobile ambients", in *Proc. 27th ACM Symp. on Principles of Programming Languages (POPL 00)*, Boston, USA, pp. 365-377, ACM Press, 2000.

[Cavalli *et al.* 2004] A. Cavalli, C. Grepet and S. Maag, "A validation Model for the DSR protocol, " in *Proc. of the 24th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'04)*, Tokyo, Japan, pp.768-773, IEEE CS Press, 2004

[Cavin *et al.* 2002] D. Cavin, Y. Sasson and A. Schiper, "On the accuracy of MANET simulators", in *Proc. 2nd ACM Workshop On Principles Of Mobile Computing (POMC'02)*, Toulouse, France, ACM Press, pp. 38-43, 2002.

[Charatonik *et al.* 2003] W. Charatonik , S. Dal-Zilio, A. Gordon, S. Mukhopadhyay, and J-M. Talbot, "Model checking mobile ambients", *Theoretical Computer Science*, 308(1-3): 277-331, 2003.

[Grassi *et al. 2004*] V. Grassi, R. Mirandola and A. Sabetta."A UML Profile to Model Mobile Sytem", *UML 2004*, LNCS 3273, Springer-Verlag Berlin Heidelberg, 2004, pp.128-142

[Morla and Davies 2004] R. Morla and N. Davies, "Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment", *IEEE Pervasive Computing*, 3(2):48-56, July-September 2004

[Ngani Noudem and Viho 2005] F. Ngani Noudem and C. Viho. "Modeling, Verifying and Testing the Mobility Management in the Mobile Ipv6 Protocol", in *Proc. 8th International Conference on Telecommunications (ConTEL 2005)*, Vol.2, pp. 619-626, IEEE CS Press, 2005.

[Satoh 2003] I. Satoh, "A Testing Framework for Mobile Computing Software", *IEEE Transactions on Software Engineering*, 29(12):1112-1121, 2003.

GA12 - Abstraction

1- Definition

In computer science, *abstraction* usually defines a transformation between the original and the abstracted model of the system. The relation of the two models is that the abstracted model contains less detail of the system's specification, thus resulting in a compact representation of the system.

The main attributes of an abstraction are *soundness* and *completeness*, which are related to a *property* of the system (e.g. fault-tolerance properties). The abstraction is sound and complete if the property holds in the abstracted system if and only if it holds in the original one.

Abstract interpretation theory [Cousot & Cousot 1977] gives a formal meaning to what is meant by abstraction: A concrete semantics (incomputable, in general) is connected, through a Galois connection, to a simpler abstract semantics. As for abstraction in general, the abstract semantics can be computed to obtain sound approximations of (generally a set of) concrete executions, the converse (completeness) is not true or necessary in general though.

2- Examples/Application Domains

The application of *formal methods* has become inevitable in fault-tolerant systems. The main limitation of applying formal methods in the V&V of real systems is *branch explosion,* i.e., the vast number of possible system executions making formal analysis infeasible. An established solution to this problem is use of sound and/or complete abstractions in order to downscale the complexity of the original problem.

The analysis of the abstracted model can be performed using deductive reasoning, exhaustive simulation, etc. *Model checking* [Clarke et al. 1986] is an attractive method, since the analysis can be performed in a fully automated manner. Related approaches present general frameworks for abstraction-based model checking of general systems (e.g., [Clarke et al. 1994]). In particular, the SLAM project [Ball & Rajamani 2002] applies the technique presented in [Clarke et al. 1994] to support the model checking of device drivers of Windows operating systems.

We remark that many approaches (e.g., [Steiner et al. 2004]) use "ad-hoc" abstractions, i.e., the soundness and/or completeness of the abstraction is assumed (by intuition) rather than proved. In case of formal verification, such approaches can undermine the overall soundness of the verification. Therefore, our focus is on formally proven abstractions.

3- Current Approaches

Approaches like *predicate abstraction* [Graf & Saidi 1997], *parameterized networks* [Kurshan & McMillan 1995], etc. propose abstractions to enhance the verification of general systems. These frameworks propose solutions, which yield only limited reduction of the state space, and also define complex conditions for their applicability. The extent of complexity reduction is limited because general solutions usually apply to a general class of systems; therefore, substantial detail must be added to the abstracted model to guarantee soundness and\or completeness of the analysis method. On the other hand, the evaluation of conditions that express whether the abstraction is applicable for a particular system often requires considerable mathematical expertise. In fact, the main research challenges stem from the latter issue, i.e., proposing solutions to make abstraction-based formal verification more applicable.

4- Research Challenges

A research gap can be identified in defining *"intuitive" abstractions* to guide the user in the abstraction process. As previously discussed, abstractions are intended to be sound and/or complete; the proofs however remain hidden from the user. Such abstractions are usually specific to an application domain (e.g., synchronous distributed applications). The approach also provides for an increased reduction of the state space, since the semantic information of the specific application domain can be exploited in the abstract model.

In particular, abstract interpretation is clearly appealing in the context of the assessability of properties of complex, ubiquitous systems. The main drawback arises when already developed abstract interpretations are not applicable, abstract interpretation theory can be considered mostly as a set of prescriptions of what should be provided, and what should be proved, to obtain a sound (or complete) abstraction, requiring great specialized skills. Research in abstract interpretations suitable for concurrent and mobile systems is still young, and few examples can be found in the literature.

5- References

[Ball & Rajamani 2002] T. Ball and S. K. Rajamani, "The SLAM project: Debugging system software via static analysis.", in *Symp. Principles of Programming Languages*, pp. 1–3, 2002.

[Cousot & Cousot 1977] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints", in *Proc. of POPL '77,* pp. 238–252, 1977.

[Clarke et al. 1986] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, 1986.

[Clarke et al. 1994] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 5, pp. 1512–1542, 1994.

[Graf & Saidi 1997] S. Graf and H. Saidi, "Construction of abstract state graphs with PVS", in *CAV '97*, pp. 72–83, 1997.

[Kurshan & McMillan 1995] R. P. Kurshan and K. L. McMillan, "A structural induction theorem for processes", *Information and Computation*, vol. 117, no. 1, pp. 1–11, 1995.

[Steiner et al. 2004] W. Steiner, J. Rushby, M. Sorea, and H. Pfeifer, "Model Checking a Fault-Tolerant Startup Algorithm: From Design Exploration To Exhaustive Fault Simulation", in Proc. of *DSN '04,* pp. 189–198, 2004.

GA13 - Test methods for aspect-oriented systems

1- Definition

Adaptive systems require observation and control capabilities to modify system structure and behaviour at run-time according to changes in the environment. Reflective technologies are seen as a privileged means to introduce these adaptation capabilities into computing systems (see gap GE7 "Design for Adaptation: Framework and Programming Paradigms"). Among these technologies, aspect-oriented programming (AOP) [Kiczales *et al.* 1997] has recently gained considerable attention. An aspect is a module implementing a cross-cutting concern (e.g., for us, an adaptation feature), and defining locations where the cross-cutting code should be injected into the core application. The composition process is called weaving. Modularity, re-usability, separation of concerns and applicability in case of legacy code are clear advantages of AOP.

2- Examples/Application Domains

The AOP paradigm is potentially relevant to any system that needs to adapt its resilience mechanisms dynamically. A typical example of application could be reflective architectures for ubiquitous and mobile computing systems.

3- Current Approaches

However appealing the concept of reflective architecture may be, the use of the related technologies in critical systems will remain questionable as long as assessment issues cannot be properly addressed.

A first contribution focused on testing reflective systems based on MetaObject Protocols (MOPs) [Ruiz-Garcia *et al.* 2001]. It defined an incremental strategy, enabling reflective mechanisms that have already been tested to be reused for verifying the remaining ones. The strategy was applied to validate the MOP of a fault-tolerant architecture for CORBA systems [Killijian and Fabre 2000].

Pioneering work addressing the specifics of aspect-oriented software has emerged recently. Testing based on structural coverage criteria is investigated in [Zhao 2003, Xie and Zhao 2006], while [Xu and Xu 2007] proposes testing from state models with aspect-oriented extensions. The work of [Bækken and Alexander 2006] aims to define a fault model that targets the constructs of AOP languages, and that could be used as a basis for designing test strategies.

4- Research Challenges

An aspect has no independent existence, and must be analyzed for a given deployment configuration. Likewise, the behaviour of the core application code cannot easily be abstracted in terms of module interfaces, since aspects may implicitly change these interfaces. Side effects may make the emergent behaviour depend on a particular weaving order of multiple aspects. From a testing perspective, this means that the notion of testing level has to be revisited. There is a need to make it clear what the equivalent of traditional unit testing would be, and how we could define an integration strategy that incrementally tests the target software from unit to system level. Specifically, this strategy would have to accommodate the multi-layer framework proposed by gap GE7.

For a given testing level, the test selection issue may be tackled by adopting structural or functional criteria. As shown by pioneering work on testing aspect-oriented software [Zhao 2003, Xie and Zhao 2006, Xu and Xu 2007, Bækken and Alexander 2006], there is a need to revisit the notion of control- and data-flow coverage, and to consider the utilization of behavioural models that make the aspects explicit.

5- References

[Bækken and Alexander 2006] J. Bækken and R. Alexander, "A Candidate Fault Model for AspectJ Pointcuts", in *Proc. 17th International Symposium on Software Reliability Engineering (ISSRE 2006),* IEEE CS, pp. 169-178, 2006.

[Kiczales *et al.* 1997] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J-M. Loingtier and J. Irwin, "Aspect-Oriented Programming", in *Proc. 11th European Conference on Object-Oriented Programming (ECOOP'97)*, LNCS 1241, Springer Verlag, pp. 220-242, 1997.

[Ruiz-Garcia *et al.* 2001] J-C. Ruiz-Garcia, P. Thévenod-Fosse and J-C. Fabre, "A Strategy for Testing MetaObject Protocols in Reflective Architectures", in *Proc. 2nd International Conference on Dependable Systems and Networks (DSN'01)*, IEEE CS, pp. 327 – 336, 2001.

[Killijian and Fabre 2000] M-O. Killijian and J-C. Fabre, "Implementing a Reflective Fault-Tolerant CORBA System", in Proc. 19th Symposium on Reliable Distributed Systems (SRDS'2000), IEEE CS, pp. 154-163, 2000.

[Xie and Zhao 2006] T. Xie and J. Zhao, "A framework and tool supports for generating test inputs of AspectJ programs", in *Proc. of the 5th International Conference on Aspect-Oriented Software Development (AOSD' 2006)*, ACM Press, pp. 190-201, 2006.

[Xu and Xu 2007] D. Xu and W. Xu, "State-Based Incremental Testing of Aspect-Oriented Programs", in *Proc. of the 5th International Conference on Aspect-Oriented Software Development (AOSD' 2006)*, ACM Press, pp. 180-189, 2006.

[Zhao 2003] J. Zhao, "Data-Flow-Based Unit Testing of Aspect-Oriented Programs", in *Proc. 27th Annual IEEE International Computer Software and Applications Conference (COMPSAC'2003)*, IEEE CS, pp. 188-197, 2003.

GA14 - Compositional Reasoning

1- Definition

Compositional reasoning can generally be understood as the deduction of system properties from a collection of sub-properties. Commonly, these sub-properties are specifications of the constituent components of a system, such that the correctness of a system can be verified solely on the basis of specifications of the sub-systems. Thus, the decomposition is organized according to the structure of the system, and one can distinguish horizontal decomposition, where properties of components at the same level of abstraction are considered, from vertical composition, where properties are established along a hierarchy of abstract views of the system. Conversely, one can keep the overall system unchanged and take the lead from the structure of the property for decomposition and then check each of these possibly weaker sub-properties for the whole system.

2- Examples / Application Domains

Compositional reasoning is commonly applied to concurrent systems in order to reduce the complexity of reasoning. Applications can be found in diverse areas such as real-time systems [Furia et al., 2007, Parkinson & Gasperoni, 2002], pipelined processors [Manolios & Srinivasan, 2005], security analysis of network protocols [Backes et al., 2006], or the compositional verification of middleware-based software architectures [Caporiuscio et al., 2004].

3- Current approaches

One approach to reasoning compositionally is by using the assume-guarantee paradigm, referring to both the rely-guarantee formalism [Jones, 1981] for shared-variable concurrency, and the assumption-commitment formalism [Misra & Chandy, 1981] for synchronous communication. Extensions have been proposed to include circular assume-guarantee reasoning [McMillan, 1999, Namjoshi & Trefler, 2000], to deal with faults in safety-critical systems [Elmqvist et al., 2005], and to enable abstraction and refinement for timed systems [de Alfaro et al., 2002] and for hierarchical hybrid systems [Henzinger et al., 2001]. Another compositional reasoning approach is stepwise refinement, which allows one to deduce refinement properties of a composite system from such properties of its components. In the abstract interpretation framework [Cousot & Cousot, 1979], composition and decomposition of abstract interpretations can be applied to decompose the analysis into a set of simpler sub-analyses that can be carried out independently. Compositional reasoning is employed both in deductive verification using theorem proving and in automated verification using model checking [Moffat & Goldsmith, 2006].

4- Research challenges

Although plenty of theoretical approaches exist, there is still a gap to the effective use of compositional verification in practice, in particular regarding the automation of reasoning. Challenges concern deciding how best to decompose the system, or the property, and also how to find suitable assumptions that can complete an assume-guarantee proof. There has been work on automatically computing such assumptions, but further effort is necessary to make these techniques scale.

Compositional reasoning is currently used for verification, i.e. to establish the correctness of a system with respect to its specification. By contrast, certification authorities will not certify components and then certify their composition. The reason for this is that a dependability property like safety is only meaningful at a system level, such as an aircraft. However, future reconfigurable and evolving systems will require a compositional certification framework for them to be affordable. To extend compositional verification to certification, a number of issues need to be addressed through the use of a dependability case (see GA3 "Dependability cases"). A key research challenge for dependability cases is that of decomposing dependability claims and is strongly related to these research challenges for compositional reasoning. [Rushby, 2007] explores some of these issues where one approach is advocated by employing components with integrity guaranteed through partitioning mechanisms that also preserve properties of subsystems. Certification of physical systems will involve dealing with hybrid (discrete and continuous) models of the controlled units and their physical environment, hence compositional verification techniques for hybrid systems need to be developed further. Compositional reasoning will also have to be extended to deal with degraded behaviours of components.

5- References

[Backes et al., 2006] Backes, M., Datta, A., Derek, A., Mitchell, J.C., Turuani, M. (2006) Compositional Analysis of Contract-Signing Protocols. Theoretical Computer Science, Volume 367, Issue 1, November 2006, Pages 33-56.

[Caporiuscio et al., 2004] Caporuscio, M., Inverardi, P., Pelliccione P. (2004) Compositional verification of middleware-based software architecture descriptions. In Proceedings of the 26th International Conference on Software Engineering (ICSE'04). May 2004, IEEE, Pages 221- 230.

[Cousot & Cousot, 1979] Cousot, P., Cousot, R. (1979) Systematic design of program analysis frameworks. In Proceedings of the 6th ACM Symposium on Principles of Programming Languages, San Antonio, Texas, 1979. ACM Press, New York, Pages 269-282.

[de Alfaro et al., 2002] de Alfaro, L., Henzinger, T. A., Stoelinga, M. (2002) Timed Interfaces. In Proceedings of the Second International Workshop on Embedded Software (EMSOFT'02), Lecture Notes in Computer Science, Volume 2491, Springer, 2002, Pages 108-122.

[Elmqvist et al., 2005] Elmqvist, J., Nadjm-Tehrani, S., Minea, M. (2005) Safety Interfaces for Component-Based Systems. In Proceedings of the 24th International Conference on Computer Safety, Reliability and Security (SAFECOMP'05), Lecture Notes in Computer Science, Volume 3688, Fredrikstad, Norway, September, 2005, Pages 246-260.

[Furia et al., 2007] Furia, C. A., Rossi, M., Mandrioli, D., Morzenti, A. (2007) Automated compositional proofs for real-time systems. Theoretical Computer Science, Volume 376, Issue 3, 2007, Pages 164-184.

[Henzinger et al., 2001] Henzinger, T. A., Minea, M., Prabhu, V. (2001) Assume-guarantee reasoning for hierarchical hybrid systems. In Proceedings of the Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01), Lecture Notes in Computer Science, Volume 2034, Springer, 2001, Pages 275-290.

[Jones, 1981] Jones, C.B. (1981) Development Methods for Computer Programs including a Notion of Interference. PhD thesis, Oxford University, 1981.

[Manolios & Srinivasan, 2005] Manolios, P., Srinivasan, S. K. (2005) A complete compositional reasoning framework for the efficient verification of pipelined machines. In Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 06 - 10, 2005. IEEE Computer Society, Washington, DC, Pages 863-870.

[McMillan, 1999] McMillan, K.L. (1999) Circular Compositional Reasoning about Liveness. In Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, Lecture Notes in Computer Science, Volume 1703, Springer, 1999, Pages 342-345.

[Misra & Chandy, 1981] Misra, J., Chandy, K.M. (1981) Proofs of Networks of Processes. IEEE Transactions on Software Engineering, Volume 7, Issue 4, 1981, Pages 417-426.

[Moffat & Goldsmith, 2006] Moffat, N., Goldsmith, M. (2006) Assumption-Commitment Support for CSP Model Checking. In Proceedings of the Sixth International Workshop on Automated Verification of Critical Systems, AVoCS 2006, Nancy, September 2006. Revised version available in Electronic Notes in Theoretical Computer Science, 185 (SPEC. ISS.) pp. 121-137. 2007.

[Namjoshi & Trefler, 2000] Namjoshi, K.S., Trefler, R.J. (2000) On the Completeness of Compositional Reasoning. Lecture Notes in Computer Science, Volume 1855, Springer, 2000, Pages 139-153.

[Parkinson & Gasperoni, 2002] Parkinson, P., Gasperoni, F. (2002) High-Integrity Systems Development for Integrated Modular Avionics using VxWorks and GNAT. Reliable Software Technologies - Ada-Europe 2002 : Reliable Software Technologies - Ada-Europe 2002, Lecture Notes in Computer Science, Volume 2361, Springer Berlin / Heidelberg, Pages 77-102.

[Rushby, 2007] J. Rushby, J. (2007) Just-in-Time Certification. In Proceedings of the 12th IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS'07), Auckland, New Zealand, July 2007 (to appear).

GA15 - Emergent behaviours in large-scale socio-technical systems

1- Definition

Emergent behaviour is a common issue in the systems addressed by ReSIST. Increasing system scale (in the various directions considered in ReSIST) may produce system properties and behaviours that cannot apparently be predicted by the laws that describe small-scale versions of the system [Johnson, 2006]. Work in networks and distributed computing systems has long dealt with issues of scalability and how properties change with simple emasures of size. The ubiquitous systems of interest to ReSIST pose the problem of increased internal heterogeneity, most significantly in that they include people in tightl integration with technical systems. Thus, this discussion uses examples of "emergent behavious" arising from people's interaction with or through ICT. Although other causes are certainly present, the current limiits of practices for modelling human and machine behaviour together (cf GA17, "Modelling Human Behaviour") makes people-related emergent behaviours a challenging category. In large scale socio-technical systems, even those usability issues that are well understood for one person-one computer (or, more generally, small numbers of both) scenarios might have more complex effects.

Scale has several dimensions, but to take the simplest one, size, even simply increasing the number of nodes in a network may cause emergent phenomena. E.g., some interfaces for security administrators are notoriously poorly designed [Reeder, 2005]; errors by a security administrator may have cascading consequences, if e.g. they allow attackers to usurp trusted roles. The size of the damage caused may thus grow non-linearly with the number of nodes involved, and may change the relative importance of different aspects of usability dramatically, depending on the propagation potential of the failures they may cause. Likewise, simple increases in propagation delays caused by certain increases in size might cause various qualitative changes:

- A change or service degradation may manifest itself not as a large scale replication of the local change occurring at one node of a large network, but be amplified non-linearly, and not as a simple step change throughout the system, but as propagating (or standing) waves or more chaotic patterns. Such patterns may make the negative effects (e.g. overloads) worse or give space for better tolerating them;
- Vice-versa, social phenomena that were reasonably stable thanks to high propagation delays might become unstable when delays are reduced by ICT (cf. the plausible common opinion that blames the 1987 stock market crash on positive feedback among automated trading strategies).

2- Examples / Application Domains

Three useful examples are:

- Some service (perhaps society-critical) provided on the web to human users. There are many users and multiple redundant, diverse web service providers. In case of failure or overload of some server[s] ("natural" or perhaps caused by malicious activity), individual users will seek the service from another server or service providers. Thus they will transfer load and affect the perception other users have of the quality of service of their current servers, and thus the behaviour of these other users. The overall effect could be anywhere between smooth load redistribution and a chain reaction in which stampeding users crash all the service providers in succession. This is determined by a combination of the design of the services and the knowledge, perceptions, reaction times of the users. Knowing which effect to expect would be of great interest for safeguarding (or attacking) the service;
- Integrated healthcare through electronic patient records. There are multiple conflicting requirements (efficiency; privacy; accuracy) and, for instance, perception by healthcare staff of poor efficiency may lead to privacy-violating behaviours that in turn may lead to patients providing inaccurate information. In addition to the scenario above, here we have multiple user roles (patients, doctors, medical administrators, insurance companies) with different priorities and cultures dictating attitudes to trust and decision making;
- E-voting, especially the kinds based on sophisticated cryptographic methods. These depend on a distributed network based service, on trust by the voters (which may change, even suddenly and massively, following perceptions of unreliable or suspicious behaviour of the automated systems), on the availability of a distributed web-based service for monitoring ballot counting, and on distributed monitoring by many users.

3- Current approaches

The "complex systems" research area is a flourishing "new" interdisciplinary field with its basis in mathematical techniques for modelling large scale systems [Boccaletti et al 2006; Reka, 2002]. Mathematical methods applied include simple massive simulation, physics-derived methods based on characterising system state via statistical parameters, and random graphs. There is also increasing application, and perception of the need for more extensive application, of these techniques in the ICT area [Bullock and Cliff, 2004; Jones and Randell, 2004]. Human factors inspired studies of emergent properties exist, although they may not focus on large scale systems [Wears, Cook, and Perry, 2006].

4- Research challenges

A first need for research is simply to document "emergent behaviours" (actual – empirically observed – or potential – identified through mixes of mathematical and empirical work) and so identify those large-scale socio-technical systems in which they are a concern, if there are any. The simple discovery of the potential for such behaviours may itself be a rewarding result.

Secondly, the creators and users (and the enemies) of ReSIST socio-technical systems would wish to know under which conditions to expect specific categories of "emergent" behaviours; and to what extent they can predict these behaviours as a function of system characteristics. For instance, designers need to know to what extent they can treat the joint behaviour of many people as many non-interacting sessions, each between one human and some automated counterpart. I.e., a desired outcome is to move some behaviours from the set of "emergent behaviours" in the sense of "possible surprises" to a list of potential behaviours whose likelihood in a system may be assessed –with more or less precision and confidence– before deployment.

Methodological questions arise, concerning for instance:

- To what extent the behaviour of these large systems is predictable. There will clearly be differences between different systems, but should we generally expect rather precise predictions to be possible for a specific system (e.g., "the rate of loss from event A will be roughly proportional to parameter lambda")? Or will only basic predictions of the qualitative patterns of behaviour be possible? This "degree of predicatbility" will determine whether tools to be used for studying emergent behaviours will be closer to those common for assessing computer system dependability or those used in complexity research

- The relationship to social sciences and their results. In some cases, the presence of ICT may make no difference compared to studying the collective behaviour of people, i.e. the questions would reduce to those usually addressed by sociologists, economists or psychologists, with various degrees of success: practitioners involved with large socio-technical systems would "only" have to become familiar with established knowledge from these fields. On the other hand, ICT may make a large enough difference, possibly making it easier to design systems for collective behaviour to be predictably resilient (e.g. by allowing better co-ordination), or in other cases making collectives of people less predictable (e.g. by reducing propagation delays on positive feedback loops).

5- References

[Boccaletti et al 2006] Boccaletti S., Latora V., Moreno Y., Chavez M., Hwang D.-U (2006). Complex networks: Structure and dynamics, Physics Reports, 424 (4-5), pp. 175-308.

[Bullock and Cliff, 2004] Bullock, S. and Cliff, D. (2004) Complexity and Emergent Behaviour in ICT Systems. Report prepared for the Foresight Cyber Trust and Crime Prevention Project, London, UK.

[Johnson, 2006] Johnson, C. (2006) What are emergent properties and how do they affect the engineering of complex systems? (Editorial), Reliability Engineering and System Safety 91, 1475–1481, Elsevier.

[Jones and Randell, 2004] Jones, C. and Randell, B. (2004) Dependable pervasive systems. University of Newcastle upon Tyne, report prepared for the Foresight Cyber Trust and Crime Prevention Project, London, UK.

[Reeder, 2005] R.W. Reeder and R.A. Maxion (2005). User Interface Dependability through Goal-Error Prevention, 2005 International Conference on Dependable Systems and Networks (DSN'05), pp. 60-69.

[Reka, 2002] Reka, Albert and Albert-Laszlo Barabasi (2002). Statistical mechanics of complex networks, Reviews of Modern Physics, vol 74,http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0106096.

[Wears, Cook, and Perry, 2006] Wears, R. L., Cook, R. I. and Perry, S. J. (2006) Automation, interaction, complexity, and failure: A case study. Reliability Engineering and System Safety 91, 1494–1501, Elsevier.

GA16 - Modelling Effect of Micro-decisions In The Whole System

1- Definition

In a large system involving many people, the decision making of these people helps determine its dependability. For example, it is conventional wisdom that many COTS systems on the Internet are more vulnerable than they need to be simply because system administrators do not install the necessary patches or do not alter default configurations [Arbaugh 2000]. The resilience of the Internet infrastructure or of any individual service delivered on it thus depends on the individual decisions of many independent agents, who decide on the basis of their local incentives and information. Classical economic theory (effects of incentives and rewards on individuals) can predict some of these behaviours [Cave, 2006]. Others are explained by considering also real decision making of humans with limited knowledge and time, as studied in research on the psychology of judgement and decision making. Applications of the latter to economics is still a relatively young area [Kahneman, 2002]. Predicting, to the extent possible, the behaviour of networks of people connected by ICT requires an interdisciplinary collaboration involving at least economics, psychology and sociology. While there is no guarantee that large scale behaviour in these systems will be predicted more effectively than behaviour of (for instance) markets currently is, insight in the range of possible behaviours and the factors affecting them would be precious for decision makers who determine the composition and design of these systems.

The designation "decision making" covers a broad spectrum of tasks, and in particular decisions that vary greatly in the *immediacy* of their determining stimuli and of their effects. For instance, security-pertinent decisions range from deciding long-term investment in protection against future and hypothetical security breaches, to the decision over a specific, potentially dangerous "click" action on one's computer screen (in between there are decisions like whether to install a security patch, or whether to encrypt a certain set of data). There is no sharp demarcation along this continuum, but the more immediate decisions by people while using computers tend to be studied as part of the domain of HCI, although perhaps with greater emphasis on the cognitive aspects than the motivation aspects of decision. In ubiquitous systems, more study may be needed of their potential emergent effects (cf. GA15, "Emergent behaviours in large-scale socio-technical system"). Here we call attention to issues of motivation and of less immediate decisions.

2- Examples / Application Domains

As widely recognised, advanced economies already depend on the Internet and on the IT used by companies. Hence, the above quoted example of how the aggregation of individual preferences about how much is invested in security administration and assurance affects the resilience of the Internet infrastructure and of systems overlaid on it. Another widely recognised role of individual decisions is in determining choices of IT products, with markets determining the commercial success of comparatively unreliable, insecure off-the-shelf operating systems and applications. Consumer choice of services delivered via the Internet is also acknowledged to have large impacts, and for instance their risk-aversion and risk-proneness necessarily impact the resilience of the services, markets or collaborative enterprises thus created.

3- Current approaches

In some areas of research, there is wide interest in explaining and predicting behaviour of ICT users in terms of incentives and perceptions. For instance, there is a growing body of literature on the economics of security (cf. the literature summaries at http://www.heinz.cmu.edu/~acquisti/economics-privacy.htm, http://www.cl.cam.ac.uk/~rja14/econsec.html) that mostly addresses normative issues (what the appropriate decision is for a rational decision maker) and their consequences for aggregate behaviour (e.g. whether "rational" decisions of self-interested parties would lead to an optimal aggregate behaviour; what criminal behaviours should be expected from "rational" criminals) but which has also started to consider "irrational" behaviours (i.e., behaviours that violate the normative decision-theory view of optimal decisions) due to insufficient information and cognitive biases [Acquisti, 2005]. Researchers are also active in the study of individual preferences of users, both professional and as consumers of Internet-delivered services.

"Irrational" decision making in a broader context has been widely studied. For instance, psychological and economics research of human decision making has provided evidence for what is termed "the default effect". This occurs when there is an increased preference for any alternative which is presented as a default, where this alternative is automatically considered more desirable than it would have been if it were not presented as a default [Johnson et al., 1992]. Effects of default options are well documented in diverse domains (organ donation, choice of pension plans) and can be large and of considerable consequence to the decision maker and others [Benartzi & Thaler, 2001]. While defaults have no status in any economic theory of decision making, they plainly are of considerable significance for understanding choice behaviour. For example, [Thaler and Benartzi, 2004] took advantage of defaults with an attempt to get people to commit in advance to allocate a portion of their future salary increases toward retirement savings. The basic idea was to give workers the option of committing themselves now to increase their savings rate later, each time they get a raise. Once people enroll in the plan, few will ever get around to opting out. The initiative was highly successful, significantly increasing the amounts people save. This issue of the choice between "opt-in" and "opt-out" designs has also been studied in the ICT area [Bellman, 2001].

4- Research challenges

Existing results in economics and psychology have to some extent inspired research about decisions in ICT-rich environments, "porting" existing knowledge and providing empirical verification or calibration. These studies need to be extended to the prediction of behaviours that affect the resilience of large systems and thus to support decision making by their designers; in this context, "design" has a broad meaning, ranging from technical decisions about protocols all the way to law making, as all these decisions affect both a system's structure at a given time and also the possible changes of its components' behaviours as well as its boundaries.

5- References

[Acquisti, 2005] Acquisti, A. and Grossklags, J. (2005) "Privacy and Rationality in Individual Decision Making". IEEE Security and Privacy, 3, 26-33.

[Arbaugh 2000] Arbaugh, W. A., Fithen, W. L. and McHugh, J. (2000) "Windows of Vulnerability: A Case Study Analysis". IEEE Computer, 33, 52-59.

[Bellman, 2001] Bellman, S., Johnson, E. J. and Lohse, G. L. (2001) "On site: to opt-in or opt-out?: it depends on the question". Communications of the ACM, 44, 25-27.

[Benartzi & Thaler, 2001] Benartzi and Thaler, 2001 S. Benartzi and R.H. Thaler, Naive diversification strategies in defined contribution plans, American Economic Review 91 (2001), pp. 79-99.

[Cave, 2004] Cave, J. (2004) The economics of cyber trust between cyber partners. University of Warwick, report prepared for the Foresight Cyber Trust and Crime Prevention Project, London, UK.

[Johnson et al., 1992] Johnson, E.J. J. Hershey, J. Meszaros and H. Kuhnreuther, (1992). Framing, probability distortions, and insurance decisions. Journal of Risk and Uncertainty 7, pp. 35-51.

[Kahneman, 2002] Daniel Kahneman (2002). "Maps of Bounded Rationality", 2002 Nobel Prize Lecture, http://nobelprize.org/nobel_prizes/economics/laureates/2002/kahneman-lecture.html.

[Thaler and Benartzi, 2004] Thaler R. and S. Benartzi. 2004. "Save More Tomorrow: Using Behavioral Economics to Increase Employee Savings", Journal of Political Economy, 112(1), 164-187.

GA17 - Modelling Human Behaviour

1- Definition

Socio-technical systems are dependent on the interaction of people with complex technologies, or among themselves, and rely on the adequate integration of all their elements, including hardware and software artefacts, the people within the system, and the rules and procedures used to support the activities of the humans. Humans are usually considered the critical element of these systems and "human error" the most common cause of incidents and accidents. "Human error" [Reason, 1990] is seen to be an ontological entity, a cause as such, the removal of which, through the automation of functions is expected to increase the dependability of the system. In reality, however incidents and accidents are rarely the result of a single cause such as a human error or a wrong interaction. Most incidents and accidents, in socio-technical systems, are caused by a combination of organisational, managerial, technical, social and political factors. These factors tend to change and evolve during the system operational lifecycle and then they exercise a different influence on the system, involving a continuous change of role and interactions between human, equipment, procedures. Several accidents involving socio-technical systems were caused by a combination of factors and breakdowns in the interaction at organisational, managerial, and technical levels and a lack of consideration of the evolution of these interactions. For example, in transportation [Neumann, 1995], space [Science, 1989], defence [Jaffe, 1989], medical devices [Leveson, 1993] and plant control [van Beek, 1992]. These combinations would probably be recognised as predominant in most of the past accidents, had the investigations paid attention to all of the root, deeper underlying causes, and not just the precipitating event in a particular circumstance [Leveson, 1995].

2- Examples / Application Domains

A typical example of a socio-technical system in which humans play a major role are the modern systems used for the centralised control of railway traffic. The components of such systems can be grouped in three main categories:

- The human component, which includes all the humans involved, even if marginally, in the control activity;
- The equipment component, which concerns all the "non human" physical elements;
- And the procedural component, which regulates the interactions between different components and between components and the external world.

The control functions are directly based on this set of components. The correct execution of the control activity is a result of their interaction and of their adequate integration.

The integration of the components is also important to ensure an adequate level of redundancy. In case of failure of a component, the interaction and correct co-operation of the other components can ensure that the failure is tolerated. Indeed, the most important control functions are usually ensured by more than one component. For example, in process control of industrial plants a common solution adopted by designers is to have an operator monitoring the correct functioning of the automated part of the control system. In the case of failure of the automation, the operator can step in and perform the control function manually.

3- Current approaches

Current methods for dependability assessment are based on the consideration of the socio-technical system elements in isolation, and in a context that is static and aseptic (e.g. not influenced by social aspects and political pressures). Dependability assessment is often based on methods and techniques which have been successful in predicting the behaviour of technical components [Fullwood, 1999]. These methods and techniques do not explicitly include human behaviour in their analysis, and are therefore complemented by techniques for human reliability analysis (HRA), which are generally derived from methodologies for hardware and software reliability assessment, in order to allow smooth integration into existing practices (e.g. probabilistic safety assessment). To suit the needs of probabilistic safety assessment (PSA) methods, HRA usually tries to assess whether humans will perform an action correctly or not, and to assign a probability to the success or failure of the action.

THERP (Technique for Human Error Rate Prediction) was one of the first techniques for HRA. THERP is based on a task analysis of the work to be done by humans in the system; it represents the task model in an HRA event tree. At each node of the event tree there is a binary decision point, which represents the success or failure of an action. The probability of a failure is derived from a data base containing the human error probabilities (HEPs). These probabilities are derived from expert judgement or are based on other data bases. The HEPs are modified by performance shaping factors that are contextual elements influencing the probabilities. The results obtained with this approach can be integrated easily into the PSA study, though possibly with a strong uncertainty inherent in the estimates [Apostolakis, 2004].

There are some important criticisms against this approach, which has frequently shown its limits in the nuclear, industrial and transportation domains in recent years. The first criticism is that the model of human behaviour used in such an approach misses a solid theoretical and empirical basis. Driven by the needs of the engineering approaches, human behaviour is modelled according to the behaviour of hardware and software artefacts, but humans cannot be understood in the binary, context free fashion proposed by this approach. Human activity is complex, and goal based, and it is shaped by the response offered by artefacts supporting the activity, the community within which the activity takes place and the social and political influences of the environment. However, the more recent contributions from cognitive science are still difficult and not practical to be incorporated in PSA studies [Hirschberg, 2004].

Another important criticism is the lack of consideration for the system evolution, which is particularly important for socio-technical systems. Safety assessment tends to provide an instantaneous picture of the situation at the moment the system is assessed, ignoring the potential problems that may arise during the system operational lifecycle because of the continuous change of role and interactions between human, equipment and procedures.

At the moment there are no methods or techniques nor indicators that can support a run time on-line assessment of the system to help understand whether it is still resilient.

4- Research challenges

There is a need for a conceptual framework able to describe the socio-technical system in its entirety, providing a complete representation and description of human behaviour in interaction with hardware and software artefacts and within the context of a given environment and culture [Davoudian, 1994]. There is also the need to understand when and how the interaction, the artefacts and the context can change during the operational life of a system and the influence such changes can have on system resilience.

5- References

[Apostolakis, 2004] Apostolakis, G. (2004), How Useful Is Quantitative Risk Assessment?, Risk Analysis, Blackwell Synergy.

[van Beek, 1992] van Beek, P (1992), FACTS, Database for Industrial Safety Acc.11057, The Netherlands Organisation for Applied Scientific Research, Dept. of Industrial Safety, Apeldoorn, The Netherlands.

[Davoudian, 1994] Davoudian, K, Jya-Syin Wu, George Apostolakis, (1994), Incorporating organizational factors into risk assessment through the analysis of work processes, Reliability Engineering & System Safety, Vol. 45, no. 1-2, pp. 85-105.

[Fullwood, 1999] Fullwood, R, (1999) Probabilistic Safety Assessment in the Chemical and Nuclear Industries Butterworth-Heinemann.

[Hirschberg, 2004] Hirschberg S. and others, (2004), Technical Opinion Paper on Human Reliability Analysis in Probabilistic Safety assessment, CSNI Technical Opinion Papers, OECD.

[Jaffe, 1989] Jaffe, M. (1989), Aegis, Vincennes and the Iranian Airbus, ACM SIGSOFT Software Engineering Notes, pp. 20-21, 14(5).

[Leveson, 1993] Leveson N. G. and C. Turner, (1993), An investigation of the Therac-25 accidents, IEEE Computer, pp. 18-41.

[Leveson, 1995] Leveson N. G. (1995), Safeware - System Safety and Computers, Addison Wesley.

[Neumann, 1995] Neumann P. G. (1995), Illustrative aviation problems, Computer Related Risks, pp. 44-49, Addison Wesley.

[Reason, 1990] Reason J. (1990), Human Error, Cambridge University Press, Cambridge, UK.

[Science, 1989] Science (1989), "Phobos 1 & 2 computer failures", SCIENCE, p. 1045, Vol. 245.

GA18 - Inter-organisation boundary failures

1- Definition

Many system failures, especially in socio-technical systems, are caused by inappropriate responses to communications across the boundaries between organisations (or parts thereof). That is, messages are misunderstood or anyway cause behaviours (of action or inaction) that may be correct in the view of the recipient of the message but different from that which the message was meant to elicit. A large set of these failures could be called "responsibility failures" (a term adopted in the U.K. DIRC project [DIRC, 2007; Dewsbury and Dobson, 2007]). These failures happen when actions that are necessary (during system operation, or at design or configuration time) to ensure acceptable or safe system behaviour are not performed because all the agents invoked assume them to be someone else’s responsibility, or because the agents that are expected to act lack (again, often, because of inter-organisation boundaries) the information or power to do so. Instead of inaction, counter-productively redundant actions may also occur. The inconsistent views of the world that cause the failures may be explicit, accepted views in the organisations concerned, or they may emerge, in the absence of conscious allocations of responsibility, during the response to new or exceptional situations.

2- Examples / Application Domains

These issues affect many application domains. Two instances follow.

- Large-scale critical infrastructures: in the 2003 Italian blackout [UCTE, 2004], miscommunication at the boundary between the Swiss and Italian operators contributed to the failure to shed load in time to avoid the blackout. Some reports also identified lack of regulation assigning responsibility for limiting load as a contributing cause, especially in view of the dynamic markets governing the exchange of energy. The report on the US-Canada blackout [USCPSOTF, 2004] identifies a complex situation of overlapping responsibilities and also states "There is an additional category of conclusions beyond the four principal causes—the failure to act, when it was the result of preceding conditions. For instance, GE did not respond [because it did not have sufficient information or insight to reveal the need for action".

- In E-voting and other uses of ICT in elections [Lock et al, 2007], responsibility for correct operation may be distributed between the vendors of machines (and – often – providers of training and maintenance), public authorities (multiple authorities in the many countries in which elections are a shared responsibility between local and central governments) and political parties or other representatives of legitimate interests in the election process. Knowledge of the law, of the technical problems and of the risk environment are unequally distributed among these actors. Cases of misplaced trust in e.g. vendors taking remedial action are documented.

Similar issues concern health services, finance, and other sectors. In general, the information society recasts well known problems, for example, in organizational response to anomalies and management of subcontracting, in new scenarios in which the socio-technical system controls – in the absence of anomalies – faster and more tightly-coupled processes than assumed before. This is done without necessarily having mechanisms for controlling exceptions on the same time scale, and evolving in faster and less controllable ways (e.g., web services may allow rearrangements of subcontracting relations on a continuous basis and at a very fast pace).

3- Current approaches

Many recommendations regarding critical infrastructure protection (including those triggered by the above-mentioned blackouts) include the need for better definition of responsibilities and for adequate information sharing [GAO, 2004]. Indeed, the need for clearly defined responsibilities (or processes for allocating them in emergency situations) are commonplace statements when dealing with organizational robustness. Research motivated by the needs for resilience in critical infrastructure deals with some of the issues, especially in terms of controlled sharing of information [Gjermundrød et al, 2004].

Responsibility failures in socio-technical systems with large ICT components are seen as a research issue, addressed in the U.K. project INDEED (in which City is a partner). Information sharing is one of the topics addressed in the context of critical infrastructure survivability, like the FP6 IRRIIS project.

4- Research challenges

Responsibility failures and communication failures are frequent despite being obvious possibilities. They may or may not have already been studied in sufficient detail in the areas of management and social sciences for the needs of organisations. However, the creation of complex socio-technical systems of increased openness and dynamicity creates new possibilities for these failures to occur. Relevant research questions include

- How to discover the potential for these failures, where the challenge includes
  - The general need for appropriate modelling of responsibilities and communication exchanges implicit in designs, and the need for matching analysis tools

  - ○ Addressing the opacity introduced by rapid, long range delegation and abstraction of services

- How to estimate (to some extent) their probability so as to identify plausible risks, prioritise them, assess the effects of corrections to the system structure, decide on the acceptability of changes towards greater dynamicity or openness, etc.

5- References

[Dewsbury and Dobson, 2007] Dewsbury G. and Dobson D. (2007). Responsibility and Dependable Systems, Springer.

[DIRC, 2007] DIRC project (2007). Responsibility, http://www.dirc.org.uk/research/themes/responsibility.php.

[GAO, 2004] GAO General Accounting office (2004). CRITICAL INFRASTRUCTURE PROTECTION: Improving Information Sharing with Infrastructure Sectors, July 2004.

[Gjermundrød et al, 2004] Gjermundrød, K. H., Dionysiou, I., Bakken, D., Hauser, C., and Bose, A. (2003) Flexible and Robust Status Dissemination Middleware for the Electric Power Grid", Technical Report EECS-GS-003 School of Electrical Engineering and Computer Science Washington State University.

[Lock et al, 2007] Lock, Russell, Tim Storer, Natalie Harvey, Conrad Hughes, and Ian Sommerville (2007). Observations of the Scottish elections 2007. Project Working Paper 3, InDeED Project, June 2007.

[Sagan, 2004] S.D. Sagan (2004). "The Problem of Redundancy Problem: Why More Nuclear Security Forces May Produce Less Nuclear Security", Risk Analysis, vol. 24, no. 4, pp.935-946.

[USCPSOTF, 2004] USCPSOTF U.S.-Canada Power System Outage Task Force (2004). Final report on the August 14th Blackout: Causes and Recommendations, April 2004

Usability

GU1 - Usability Metrics

1- Definition

Usability is the "extent to which a computer system enables users, in a given context of use, to achieve specified goals effectively and efficiently while promoting feelings of satisfaction" [Ivory & Hearst, 2001] The measurement of usability is a significant area in human computer interaction research. These measures are usually related to controlled laboratory experiments, comparing interface variations by measuring error rates, delays or hesitations, time to complete tasks, recoveries, references to help and so on. Usually these analyses revolve around systems that involve well defined goals and idealized tasks to achieve the goals. The challenge in terms of usability metrics is to develop unobtrusive "in the wild" approaches to measuring usability – there are some existing examples, mainly in the context of web based systems (see for example [Hilbert & Redmiles, 2001] and further measures that relate more effectively to those characteristics of usability that relate to resilience in ubiquitous systems.

2- Examples / Application Domains

It is relevant to a large class of interactive systems. For example, in the context of air traffic control how do we measure its resilience in terms of the human controller's activity within the system, using the various embedded computer based systems. Can this information be used to provide a justification that the system is resilient (safe or secure)? Another example is a decision support system, for example a system to support the analysis of mammograms, can we measure the way the system biases the diagnosis. Can we measure the impact of a ubiquitous computing system designed to improve waiting times in an accident and emergency room and reduce patient frustration.

3- Current approaches

Dix et al. comment that current usability metrics "rely on measurements of very specific user actions in very specific situations. When the designer knows what the actions and situation will be, then she can set goals for specific observations." [Dix et al, 2004]. What they mean is that the metric must be determined specifically for the user interface of the device being evaluated. Furthermore any measures are only relevant to the specific context in which the device is being evaluated. The same mammography system may have different user characteristics because of the different contexts of the two hospitals in which the systems are being used. There are therefore important questions about how to use the measures to compare across contexts. Examples of usability metrics already discussed within the HCI literature (see specifically ISO standard 9241) include:

- Time to complete a task; proportion of task completed; proportion completed within a unit of time;
- Ratio of successes to failures
- Time spent in errors; percentage or number of errors;
- Frequency of help use;

- Number of repetitions; number of available commands not invoked; number of disruptions from a work task;
- Number of times loses control;
- Time to learn;
- Time spent correcting errors.

A more comprehensive list (with references) can be found in pp. 237-24 of [Dix et al, 2004].

4- Research challenges

What do we measure when the action is implicit, inferred from context? How do we measure for example the extent to which a ubiquitous system leads people within the system to be more risk averse and thereby create a more effective safety culture?

When measuring the usability of systems in their intended contexts what variables can be measured that provide a clear understanding of usability? Two issues are important here: (1) how to gather the data in context; (2) how the data relates to usability. How can these measures be used to compare interactive systems between contexts?

For example, if "hesitation" is an important indicator of usability how do we measure it in the context without affecting the behavior of the people in the system? How does this measure of hesitation relate to the usability of the system? In a decision support tool, how do we assess the bias imposed by the tool, influencing the judgments made? What variables do we measure? How do we analyze these values to assess the bias? This relates directly to the City study of mammography systems.

What is the relationship between usability and safety and in particular what is the impact of usability on safety?

5- References

[Dix et al, 2004] Dix, A., Finlay, J., Abowd, G.D. and Beale, R. Human-Computer Interaction. Prentice Hall, 3rd Edition 2004

[Hilbert & Redmiles, 2001] Hilbert, D.M. and Redmiles, D.F. (2001) Large-Scale Collection of Usage Data to Inform Design by In Hirose, M. (ed.) Proceedings of Interact'01 pp 569-576. IOS Press

[Ivory & Hearst, 2001] Ivory, M.Y. and Hearst, M.A. (2001) The state of the art in automating usability evaluation of user interfaces. ACM Computing Surveys 33(4) pp. 470-516.

GU2 - UCD & resilience engineering and development processes

1- Definition

There is an increasing demand for reliable, safe, usable and resilient information technologies that can help organizations to face increasingly complex business applications and ever-changing risk operational environments. Such organizations are becoming more risky (in terms of working environments) because involving a human in the control of such complex safety critical interactive systems (which is practically compulsory), can unfortunately bring in human unreliability issues. This is because of the unpredictable nature of humans, in addition to more classical technical unreliability. However, if we see the human only as the unpredictable, unreliable and weak component of the system we miss a very important point, i.e. we do not acknowledge that the flexibility and the adaptability of the human work is "normally" the reason for its efficiency and capability to protect the organization from threats and perturbations that are faced very often (e.g. changes in the external environment). User Centered approaches aim at involving users in various stages of the development process, namely requirements gathering, user interface design and usability evaluation. This gap focuses on the latter as this is where there is a stronger interest in ReSIST network. Efficient testing policies should be implemented throughout the development process to help developers ensure the quality of applications. The objectives of testing can be created during requirements analysis and applied during different phases of the development process.

Software design covers specification, implementation, testing, maintenance and deployment phases – all of which require ongoing testing. The current practice in software testing includes many different testing techniques such as model-checking of specifications, running test scenarios on applications, black box and white box testing and so on. However, the development of resilient information technologies should not only check possible failures in software but also measure the impact of human factors on software and system interaction and also prevent usability problems by applying ergonomic and usability guidelines for example.

2- Examples / Application Domains

Issues of dealing with users' concerns are not specific to any application domain. Conversely, it is generic to a specific class of systems: interactive systems. The challenge of integrating user centered approaches and "classical" resilience processes and approaches is particularly relevant for any kind of safety critical interactive system such as command and control systems of power plants; control rooms (Air Traffic Control, satellite ground segment, …) and cockpits (aircraft, ground vehicles, …).

3- Current approaches

Lack of usability has been proved to be an important source of errors and mistakes performed by users [Reason, 1990]. Faced with poor user interfaces (designed from a non-user centered point of view [Norman & Drapper, 1986]), users are prone to create alternative ways of using the applications thus causing hazardous situations. In the worst cases, lack of usability may lead users to refuse to use the applications, potentially resulting in financial loss for companies with respect to the need for redesign and possibly additional training. For detecting and preventing usability problems it is important to focus on the design process from the point of view of people who will use the final applications. The technique called User-Centered Design (UCD) [Gilmore et al., 1994] is indeed the most efficient for covering user requirements and for detecting usability problems of user interfaces.

Dealing with both resilience and usability at the same time is still marginally addressed by the research community. Contributions usually deal with incidents and accidents (such as [Johnson 2005]) rather than addressing issues at the process level. Few exceptions can be found in [Basnyat et al. 2006] where the approaches call for a multi-disciplinary perspective to address the challenges of safety and usability. An advantage of addressing usability and reliability within the same development process or framework is that it is likely to have synergistic advantages.

4- Research challenges

The design of a usable, reliable, safe and error-tolerant safety-critical interactive system is a goal that is hard (actually impossible, because we cannot guarantee either of these characteristics) to achieve because of the unpredictability of the humans involved, but can be more closely attainable by taking into account information from previous known situations that relate to both usability and reliability, for example accident reports.

Whilst system-centered and user-centered designs are clearly complementary, there is a gap in combining both strategies in a sound development process. This gap could be covered by revisiting development processes and introducing usability as an operational resilience requirement. The convergence of these system-centered development processes and user-centered design processes is not just a foundation for a better coverage of best practices during the development process but it is a natural outgrowth of resilience in information technologies.

5- References

[Basnyat et al. 2006] Basnyat, S., Chozos, N., Palanque, P. (2006) Multidisciplinary perspective on accident investigation. Reliability Engineering & System Safety Volume 91, Issue 12, December 2006, Pages 1502-1520.

[Gilmore et al., 1994] Gilmore, D.J., Winder, R., Detienne, F (Eds). (1994) User-Centred Requirements for Software Engineering Environments, Springer-Verlag

[Johnson 2005] Johnson, C. W. (2005). Applying the lessons of the attack on the world trade center, 11th September 2001, to the design and use of interactive evacuation simulations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Portland, Oregon, USA, April 02 - 07, 2005). CHI '05. ACM Press, New York, NY, 651-660.

[Norman & Drapper, 1986] Norman, D & Drapper, S (Eds.). (1986)User Centred System Design. L.Erlbaum,U.S., ISBN-10: 0898597811

[Reason, 1990] Reason, J (1990) Human Error, Cambridge University Press

GU3 - Context Confusion

1- Definition

In ubiquitous computing systems action occurs through a combination of human initiated action and "system" inference. The original vision of ubiquitous computing [Weiser, 1991] was to produce comfortable, informed and effortless living, making the computer invisible, focusing on new ways to support and enhance people's lives. Rogers [Rogers, 2006] notes that the specifics of the contexts surrounding people's day to day lives are much more fluid and idiosyncratic than theories of context have led us to believe. As a result systems that make assumptions about user's intentions can overwhelm everyday life rather than assist it. Rogers argues that it would be preferable to think of the user engaging with their environment rather than seeing the user as a complicit agent. In this way context is used to help interpret what the user actively intends to do. Examples of context include: physical location; personal profile (vegetarian, product preferences etc.), activity trail to the current point, intentions, ambient physical conditions (lighting, access to network).

Systems such as these raise a number of resilience issues, in particular for example how to demonstrate that the actions that the user or the system carries out do not have the effect of compromising resilience characteristics (safety, security, privacy). While many of these issues in relation to the humans in the system are conventional notions related to usability, security and privacy other relate to issues associated with how in control the user might be or feel to be. In this sense issues associated with function allocation become more important [Dearden et al., 2000].

2- Examples / Application Domains

This situation will occur in any environment in which an action by the user and an inference by the system may be confusing, hence movement from one location to another (for example) may mean that controls provided by a hand-held device may now be interpreted differently.

A number of examples were described as part of an EU roadmap for ambient and mobile intelligences in ISTAG Scenarios for Ambient Intelligence in 2010. (ftp://ftp.cordis.europa.eu/pub/ist/docs/istagscenarios2010.pdf). In these scenarios, because they are influenced by Weiser's vision of Calm technology, user engagement is not a priority. However in the so-called "Maria road warrior" scenario the user uses her wrist device to change communications access thresholds. The buttons on the wrist device has many different meanings depending on context. In the D-Me scenario the persona (here persona is used to describe the type of person engaged in this scenario) uses the nearest functioning device. In the example described by [Loer and Harrison 2005] the mobile device provides an interface that enables its owner to control the nearest valve or pump (hence actions depend on the device's location).

3- Current approaches

Current approaches relate to three aspects of the problem:

- Models of context [Bellotti et al., 2002., Benford et al., 2005., Salvador et al., 2003];
- Analyses of systems in terms of context [Loer & Harrison, 2005];
- Concepts associated with implicit computing [Schmidt, 2000];
- Methods of function allocation, see for example [Dearden et al, 2000].

4- Research challenges

The research challenges are concerned with the semantics of mode contingent action where mode is a result of a device being embedded in an appropriate context. Models require appropriate notions of mobility in terms of how the user's context is updated through movement and changing circumstances. These models are required to analyze what characteristics of systems lead to user confusion, privacy and security issues. It is clear that user confusions that arise because of context can lead to a variety of types of failure that may be safety, security or business critical. Techniques are required that will enable analysts to assess the implications of a particular design in terms of these notions of resilience

5- References

[Bellotti et al., 2002] Bellotti, V., Back, M., Edwards, K., Grinter, R., Henderson. A., Lopes, C.: Making sense of sensing systems: five questions for designers and researchers. In Proceedings of CHI'2002, ACM Press, (2002) 415-422

[Benford et al., 2005] Benford, S., Schnädelbach, H., Koleva, B., Anastasi, R., Greenhalgh, C., Rodden, T., Green, J., Ghali, A., Pridmore, T., Gaver, B., Boucher, A., Walker, B., Pennington, S., Schmidt, A., Gellersen, H., Steed, A.: Expected, sensed, and desired: A framework for designing sensing-based interaction. ACM Trans. Comput.-Hum. Interact. 12 (2005) 3-30

[Dearden et al., 2000] Dearden, A., Harrison, M.D. and Wright, P.C.(2000) Allocation of function: scenarios, context and the economics of effort., International Journal of Human -Computer Studies, Volume 52, Issue 2, pp 289-318 Elsevier

[Loer & Harrison, 2005] K. Loer and M. Harrison. Analysing user confusion in context aware mobile applications. In M. Constabile and F. Patern`o, editors, INTERACT 2005, number 3585, pages 184–197. Springer Lecture Notes in Computer Science, 2005.

[Rogers, 2006] Y. Rogers. Moving on from Weiser's vision of calm computing: Engaging ubicomp experiences. In P. Dourish and A. Friday, editors, Proceedings of Ubicomp 2006, pages 404–421, Heidelberg, Berlin, New York, 2006. Springer.

[Salvador et al., 2003] Salvador, T., Anderson, K. Practical Considerations of Context for Context Based Systems: An Example from an Ethnographic Case Study of a Man Diagnosed with Early Onset Alzheimer's Disease. In UbiComp'03 Proceedings, A.K. Dey et al. (Eds.), LNCS 2864,Springer-Verlag Berlin Heidelberg, 243-255, 2003

[Schmidt, 2000] Schmidt, A. (2000) Implicit human computer interaction through context. Personal and Ubiquitous Computing Volume 4, Numbers 2-3 pp. 191-199

[Weiser, 1991] Weiser, M.: The computer for the 21st century. Scientific American (1991) 94–104

GU4 - Modeling Aspects Of HCI

1- Definition

As today's information systems become more complex, the development effort increases disproportionally due to the fact that the developers loose their complete overview of the system design at a point. Model-based software development aims to counter this by separating the design and implementation phases so the complete overview is only necessary during design. This phase is further aided by *domain specific modeling* (DSM), which presents the designer with concepts and relations (the *ontology* or *metamodel*) of the problem domain. Metamodeling can be realized in a formally specified way which makes it possible to develop automated processing of the system models via *model transformations*. There are two simultaneous goals to reduce the development effort: one is the automation of the implementation of the system model and the other is platform independence in modeling and automation of mapping the platform independent model to various target platforms.

Model-based approach can be applied to user interface design at multiple levels. Using *platform specific* models to develop applications to a particular target platform can be considered the most basic level. There is ongoing research and a few tools available that enable *platform independent* modeling of graphical user interfaces which represent higher abstraction levels. Generally the wider range of targeted platforms requires higher abstraction levels.

Human to computer interaction (HCI) in ubiquitous information systems typically involves the utilization of user interfaces running on hardware with a great variety of user interaction capabilities (e.g.: regular computers with large displays, keyboard and mouse vs. mobile equipment with small displays and few buttons and no pointing devices or some exotic devices such as motion/acceleration sensors). Due to the high development effort needed to create application spanning such wide scale of platforms, platform independent model based development can have a great impact. This requires a platform independent metamodel of a very high abstraction level. Implementing automated platform mapping and code generation tools from such highly abstract source models is problematic because it requires some creativity, and there are typically many possible solutions.

2- Examples / Application Domains

Nowadays model driven development is applied in some extent to graphical user interface design in a platform specific manner to replace the tedious, non-expressive coding with a visual design.

The ability to deploy to both locally running client applications and web-based applications from the same platform independent model [Calvary et al, 2003] and providing a variety of web-based technologies with different requirements on the client browser are two emerging applications of model based development. Google's web applications are a typical example that use a rich AJAX based web application but provide a simplified fallback user interface for unsupported browsers. Another example to this kind of application is to provide different web-based user interfaces tailored for small hand-held mobile devices and full featured web browsers.

The formal nature of the user interface models can also facilitate usability and resilience analysis [Campi et al., 2004].

3- Current approaches

Platform specific GUI design tools are available for most GUI platforms. However most of them are only capable of code generation and do not expose the actual internal model, so the development process is not considered to be model-based. Model export facility was only recently added to for example Qt and .NET frameworks.

There is a number of attempts to create standard ontologies and model formats for platform independent GUI modeling ([XAML], [MXML], [XUL]). They define an abstract set of widgets and syntax for defining basic behavior, and some basic facility to define custom widgets.

Higher levels of abstraction are achieved by focusing on the desired functionality (for example present a number of alternatives for the user to choose) rather than the widgets (selection list, button). Examples of such approaches can be seen in: [Constantine 2003], [Da Sylva & Patton, 2003].

4- Research challenges

The state of the art of user interface models evolves in two dimensions. Firstly the size of the entire user interface of the system increases (typically this means for example the complexity of the navigation model, not the complexity of the distinct units) and secondly, the number of aspects covered and the detailedness also increases. Such aspects can be, for example, the visual structure, the event handling, the navigation model, the layout etc. but as the system evolves eventually new aspects need to be added.

Generally speaking the size of the model is an issue of model editor technology. However the structure of the metamodel concept is also important, since it determines how much hierarchical simplification can be made to provide overviews that can be handled by the designer.

Adding new aspects to the modeling requires changes to the metamodel. Today's technologies only allow metamodel extensions by adding new elements if extensions are allowed at all. Adding new aspects (potentially for non-functional requirements, like: response time, reliability, security) requires defining and adding new roles to the existing concepts. Furthermore this is also an issue of model editors, that must be able to handle the aspects separately.

5- References

[Calvary et al, 2003] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, Jean Vanderdonckt: A unifying reference framework for multi-target user interfaces, Interacting with Computers, 2003

[Constantine 2003] Larry L. Constantine: Canonical Abstract Prototypes for Abstract Visual and Interaction Design, Proceedings of DSV-IS, 2003 - Springer

[Da Sylva & Patton, 2003] Paulo Pinheiro da Silva, Norman W. Paton: User Interface Modelling with UML, Proc. 10th European-Japanese Conference on Information Modelling and Knowledge Bases, 2003

[Campi et al., 2004] Alessandro Campi, Eliseo Martinez, Pierluigi San Pietro: Experiences with a Formal Method for Design and Automatic Checking of User Interfaces, Proceedings of MBUI2004

[XAML] XAML http://www.xaml.net/

[MXML] MXML http://www.adobe.com/devnet/flex/articles/paradigm.html

[XUL] XUL http://www.mozilla.org/projects/xul/

GU5 - Usable Privacy

1- Definition

The field of ubiquitous computing envisages an era when human owns hundreds or thousands of mobile and embedded computing devices. These devices will perform actions based on the context of their users, and therefore ubiquitous systems will store, process and exchange much more personal information than computers do today. One's vision is that ubiquitous computing has the potential to create an invisible surveillance network, covering an unprecedented share of our public and private life. For instance people location or personal health information is a particularly useful form of context in ubiquitous computing, that can be used beyond the individual's control. Such personal information shall be considered as private, and therefore mechanisms which allow users to control the dissemination of these data are essential. Actually a trade off between functionalities and protection of personal data is necessary to develop efficient ubiquitous systems. One of the main challenges for these systems is to "give end-users security controls they can understand and privacy they can control for the dynamic, pervasive computing environments of the future" (Computing Research Association 2003) Whereas security mainly focuses on integrity and confidentiality, privacy is the claim for users to determine when, how, and to what extent information about them is communicated to others.

2- Examples / Application Domains

All applications that imply an interaction of users with their environment are concerned with usable privacy. These applications include, for example, access control for mobile systems (such as smart phones [http://cups.cs.cmu.edu]), e-commerce, e-health, and any system that makes use of a localization service.

3- Current approaches

Two main issues are studied.

One is to "educate" the users to make them understand how the interactions they have with the environment may have an impact on the protection of their private information (e.g. the General Public Tutorial of the PRIME project [PRIMEa])

A second point is to design a usable system for users to control privacy. For instance, in the PRIME [PRIMEb] project that focuses on identity management, users can choose among different profiles while sending requests on Internet, from anonymous profiles to well identified ones. Refer to SOUPS [SOUPS] and DIMACS [DIMACS] conferences for the most current results on these topics.

4- Research challenges

An important gap related to usability is to develop tools for the users to effectively manage their privacy while requesting services in a ubiquitous system.

In this context, a big challenge is also to adapt those tools to people who are not aware of technology and its threats (dichotomy between gaps in knowledge and excessive fear of the users). Indeed, if end users struggle to comprehend the privacy decisions with which they are presented, they’re more likely to misconfigure—and thus jeopardize—their privacy.

Moreover, users often deliberately disable or ignore privacy configuration to get their work done. For instance, if a privacy procedure is too difficult, users won't deploy it, they may configure it incorrectly, or simply just switch it off.

5- References

[PRIMEa] PRIME general public tutorial https://www.prime-project.eu/tutorials/gpto/index_html/document_view

[PRIMEb] PRIME - Privacy and Identity Management for Europe http://www.prime-project.eu/

[SOUPS] SOUPS : http://cups.cs.cmu.edu/soups

[DIMACS] DIMACS : http://dimacs.rutgers.edu/Workshops/Tools

GU6 - User experience

1- Definition

User experience, often abbreviated UX, is a term used to describe the overall experience a user, customer, or group member has with a product or service. In the Usability field, this experience is usually defined in terms of ease-of-use or the system is referred to as being user-friendly. However, the experience encompasses more than usability, but the understanding of the subject resulting from the information and emotions accumulated through all of the senses.

UX design pertains to the creation of the architecture and interaction models which impact a user's perception of a digital device or system. The scope of the field is directed at affecting "all aspects of the user’s interaction with the product: how it is perceived, learned, and used." [Norman 1999 & Norman 2002] Consequently, UX design fully encompasses traditional human-computer Interaction design and extends it by addressing all aspects of a product or service as perceived by users; that is, it addresses the user's initial awareness, discovery, ordering, fulfillment, installation, service, support, upgrades, and end-of-life activities in addition to the traditional design of the interaction between a human and a device.

The general field has its roots in human factors and ergonomics, a field that since the late 1940's has been focusing on the interaction between human users, machines and the contextual environments to design systems that address the user's experience [HFESa & HFESb] . However, only recently it has received a major attention within user-centered design principles and software development methods being applied mostly to the design of commercial web sites.

2- Examples / Application Domains

The application domains of UX are found whenever the availability of context sensitive information plays a fundamental role to enable newcomers to appropriate the environment for the task at hand. Consequently, User Experience is a "generic" issue for ubiquitous computing.

- Electronic patient journal: This is an area which has received a lot of attention. With the introduction of pervasive computing, new and exciting possibilities for supporting the staff are emerging. However, in most hospitals mobile applications are not used as an integrated part of the day to day care. This is possibly due to the fact that using mobile applications often is cumbersome, and can easily distract the user with irrelevant or inaccessible information. The use of context-aware technology can reduce the intrusiveness and support the staff in their daily activities.
- User in an Airport: Consider a system developed to help passengers experience a sense of place in the unfamiliar setting of an airport. One might imagine a combination of ambient displays, kiosks and mobile services for hand-held devices. They combine together to provide an environment in which passengers can obtain the information they need, in a form that they can use it, to experience the place.

3- Current approaches

User Engineering employs a well structured process for designing competitive offerings and is based on a thorough understanding of the business, market and user domains. The process goes through a well identified sequence of phases distinguished as Business opportunities, User understanding, Initial design, Development and Deployment.

Design is all about business - maximizing value to the supplier, customer, and user. There is an intimate understanding of the different user groups, their goals, what they want to do, and how they envision doing it. Design of the total user experience must be progressed against specific measures with challenging targets. At each phase the design is evaluated against business, market, and user requirements, and their respective targets. There is a physically stimulating and enticing experience that encourages user adoption and regular use.

Users must feel very comfortable with the offering. In fact, they should find it seductive and totally in line with their life style. This affinity should be so strong that users are proud to be associated with the offering, demonstrating great loyalty.

The practical consequence of the approach is a shift from tasks to roles and goals oriented design.

The emphasis is:

- On identifying and describing individual stakeholder and user groups in terms of their skills, physical, and environmental characteristics, and on defining the relationships between them,
- On developing a true understanding of stakeholders' and users' goals, to identify where those goals are complimentary or are in conflict, and on identifying key goals and user tasks as candidates for the measurements.

4- Research challenges

The design of experiences isn't any newer than the recognition of experiences. As such, it is really the combination of many previous disciplines incorporating aspects of psychology, anthropology, computer science, graphic design and industrial design. UX may also involve content design disciplines such as communication design, instructional design, or game design.

The grand research challenge is to define UX design as a discipline. In fact, UX design is still in its infancy and it is so new that its very definition is not well consolidated. Many see it only as a field for digital media, while others view it in broad-brush terms that encompass traditional, established, and other such diverse disciplines as acting, graphic design, storytelling, exhibit design, theme-park design, online design, game design, interior design, architecture, and so forth. The list is long enough that the space it describes is not formally defined.

One issue is that such systems must be very generic, and as a consequence possibly inefficient. The challenge is whether it is possible to produce models in such a way that it becomes feasible to explore experience issues in the design of these systems. An approach that combines scenarios with formal property checking has provided some results in the past and its applicability to the design of ubiquitous systems would merit to be investigated.

5- References

[Norman 1999] Donald Norman (1999) Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex and Information Appliances Are the Solution. MIT Press. 1999, ISBN 978-0262640411

[Norman 2002] Donald Norman (2002) The Design of Everyday Things. Perseus Books Group. 2002, ISBN 978-0-465-06710-7

[HFESa] Human Factors and Ergonomics Society. HFES History. http://www.hfes.org/web/AboutHFES/history.html

[HFESb] Human Factors and Ergonomics Society.

Diversity

GD1 - Diversity for security

1- Definition

This research challenge deals with the controversial issue of the extent of, and limits to, the usefulness of applying the well known techniques of diversity to deal with security issues. Diversity among replicated components has proved to be useful for providing resilience against accidental faults. Recently, its use has also been investigated in the context of security with the goal of improving the intrusion tolerance capabilities of systems, and enhancing the efficiency of intrusion detection systems by reducing false negatives and/or false positives. However, the use of diversity in security raises several concerns and a large amount of work remains to be done in order to get a workable solution.

Diversity is seen as a classical mechanism to improve the resilience of systems to accidental software faults. Regarding security, clearly, having a diverse IT system reduces its vulnerability to specific threats such as worms targeting specific OSs etc. However, maintaining a large number of diverse systems is much more complicated than dealing with monocultural environment. This is particularly relevant when looking at security problems since (as discussed in the Evolvability sections) threats are evolving rapidly.

2- Examples / Application Domains

Web Server farms are classically presented as the kind of target system where diversity could be used as an efficient method to resist intrusion. Of course, as of today they are homogeneous and monocultural by design, for economical reasons. Among the research questions that arise, there is that introducing some level of diversity into them would also imply increasing the amount of problems to be taken care of. Could it be that the added complexity would reduce security instead of improving it? How can we evaluate this apparent paradox where diversity may increase security on one hand and decrease it on the other one?

3- Current approaches

The idea of building intrusion-tolerant systems, taking advantage of replicated servers that would not suffer from common failures modes is not new [Blain, 1990]. Similarly, the use of diversity for security has been discussed on several occasions during the 90's and remains a controversial topic: despite some research work, much debate is still occurring on the general issues involved (see, for example, [Deswarte, 1998; Linger, 1999] and the panel position papers in [NSPW, 2005]). Several projects have built prototypes, showing the usefulness of the approach in various specific application domains (e.g., [Reynolds, 2003; Saydane, 2003; Majorczyk, 2005] among others). However, very few proposals have looked at the issue in a more formal way. In [Littlewood, 2004], the authors discuss the roles and limits of replication and diversity in security. They review to what extent lessons from research on their use for reliability can be applied to security, in areas such as intrusion detection. They discuss the factors affecting the efficacy of redundancy and diversity, the role of "independence" between layers of defense, and some of the trade-offs facing designers. The main positions stated are that decisions about whether, and in which form, to apply replication and diversity must consider the specific system addressed and the desired trade-off between security (or generally dependability) attributes (e.g. privacy and availability); and that the forms of probabilistic modelling of diversity invented for reasoning about reliability can bring clarity about its effect on security as well. The interested reader will find in this publication a large collection of pointers to the related literature.

4- Research challenges

We must have a better understanding of how common mode failures might occur in the context of malicious faults. Indeed, nothing forbids a malicious user from launching different attacks against different components at the same time. Thus one must assume that a malicious user can launch all possible attacks simultaneously, unless this is impossible by design, or if, by design, launching one would make another one impossible to run. The extent to which these forms of impossibility properties can be achieved is an open question, although specific examples are available: e.g., in the design of an intrusion-tolerant web server based on duplicated servers, a possible solution is to send the same input request to a proxy machine that duplicates it to the distinct web servers. Thus, unless both machines are vulnerable to the same attack, it is impossible for the attacker to attack the two systems simultaneously.

Solutions for designing and implementing diversity should take into account the diverse nature of the population of attackers and the fact that attacks generally result from coordinated and synchronized actions between several attackers.

We should think about the several dimensions for enforcing diversity, both at the process level and at the product level, during the life cycle. We should investigate on practical system examples, following the directions stated for instance in [Littlewood, 2004], to what extent traditional ways of thinking about this problem when dealing with accidental faults are applicable to the case of malicious faults. For example, in the context of security, we should take into account the possibility of collusions between people involved and attacks carried out by insiders. This problem does not really exist when dealing with accidental faults.

5- References

[Blain, 1990] Laurent Blain and Yves Deswarte, "An intrusion-tolerant security server for an open distributed system," *Proc. of the European Symposium on Research in Computer Security (ESORICS'90)*, Toulouse (France), 24-26 October 1990, pp.97-104

[Deswarte, 1998] Yves Deswarte , Karama Kanoun and Jean-Claude Laprie, "Diversity against Accidental and Deliberate Faults," P*roceedings of the Conference on Computer Security, Dependability*, 1998, ISBN:0-7695-0337-3

[Linger, 1999] Richard C. Linger, "Systematic Generation of Stochastic Diversity as an Intrusion Barrier in Survivable Systems Software," *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*-Volume 3 - Volume 3 table of contents, 1999, ISBN:0-7695-0001-3, IEEE Computer Society

[Littlewood, 2004] Bev Littlewood and Lorenzo Strigini., "Redundancy and diversity in security," *Proc. ESORICS 2004, 9th European Symposium on Research in Computer Securit*y, Sophia Antipolis, France, September 2004.

[Majorczyk, 2005] Frédéric Majorczyk, Eric Totel, and Ludovic Mé, "COTS Diversity Based Intrusion Detection and Application to Web Servers," *Proceedings of the 8th International Symposium on the Recent Advances in Intrusion Detection (RAID)*. Springer Verlag, LNCS , September 2005.

[NSPW, 2005] Panel: Diversity as a Computer Defense Mechanism. In Christian F. Hempelmann and Victor Raskin (Eds), *Proceedings of the 2005 Workshop on New Security Paradigms*, September 20-23, 2005, Lake Arrowhead, California.

[Reynolds, 2003] James C. Reynolds, James Just, Larry Clough and Ryan Maglich "On-Line Intrusion Detection and Attack Prevention Using Diversity, Generate-and-Test, and Generalization," *Proceedings of the 36th Annual*

Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9, 2003, ISBN:0-7695-1874-5, IEEE Computer Society

[Saydane, 2003] Ayda Saidane , Yves Deswarte and Vincent Nicomette, "An intrusion tolerant architecture for dynamic content internet servers, " *Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*: in association with 10th ACM Conference on Computer and Communications Security, p.110-114, October 31-31, 2003, Fairfax, VA

GD2 - Large-scale diversity for intrusion tolerance

1- Definition

Intrusion tolerance, also dubbed trustworthiness or sometimes Byzantine fault tolerance (with a somewhat different meaning than the classical), is a research trend which has been explored for some time [Fraga and Powell 1985], but that recently gained great momentum, based on the idea of applying the tolerance paradigm to the problem of building resilient network services. In the context of distributed systems, the idea is often to scatter the system's components redundantly over several physical machines in such a way that if an attacker manages to get control over some of these machines, the system still behaves (functionally) correctly. However, this notion is useful only if there is low correlation between the way machines are corrupted, because it rests on the assumption that an attacker cannot corrupt more than a certain threshold number of machines in a certain window of time. This requires that the machines do not share the same vulnerabilities, since an intrusion is always the result of an attack that exploits an existing vulnerability. The commonly accepted way to enforce this property is by having diversity in the set of machines, be it in terms of hardware, installed software combination, operators etc How to enforce this diversity is a research problem, especially if the number of machines is large, e.g., hundreds or thousands.

2- Examples / Application Domains

The typical application area is that of critical Internet services (DNS, Web, certification authorities, etc.).

3- Current approaches

There are currently some well-accepted ways in which diversity can be obtained in systems with a small number of nodes [Deswarte et al. 1998, Littlewood and Strigini 2004, Obelheiro et al. 2006]. Machines should use different hardware, different operating systems, different implementations of COTS software (libraries, virtual machines, databases, etc.), and different implementations of software specific for the system, which has to be obtained using N-version programming. Moreover, it is also very important to have administrative diversity and diversity of physical location.

There are two problems with this approach, though. The first is that it does not scale. Using N-version programming or different operating systems is only possible for a small set of different versions of a machine. It is already impossible if one wishes to create unique diverse replicas among tens of nodes, not to mention hundreds or thousands. Therefore, there is a need for automatic generation of diversity.The second is that protocols for coordination among the replicas are complex and difficult to design, not to speak of implementing and deploying.

In order to decrease the window of time during which corrupted replicas may endanger the system's trustworthiness, for instance by reducing the available level of redundancy, a technique called proactive recovery has been proposed [Ostrovsky and Yung 1991]. The idea is that machines are recovered periodically from a clean state, so that a successful attack would have to be performed in a short time frame. Proactive recovery can also be used to simultaneously implement automatic diversity between the machines, so that they no longer share the same vulnerabilities after recovery.

There are already a few methods to automatically obtain software with some kind of diversified, or randomized, properties [Xu et al. 2003, Barrantes et al. 2003, Wang et al. 2001]. However, these methods have not been used with the purpose of obtaining large-scale diversity, and the level of diversity they provide has not been assessed by exposing them to malicious attackers on the Internet. One technique randomizes the memory layout of the processes and/or the operating system in memory, with the purpose of making attacks that insert data or code in the memory more difficult [Xu et al. 2003]. Such address-space randomization is already available for several operating systems, including Microsoft's Windows Vista. A similar technique randomizes the instruction set of the processor in order to prevent attacks attempting to inject binary code in the memory [Barrantes et al. 2003]. The randomized instructions are unscrambled at runtime, so code injected can only be executed if scrambled using the same random function as the code of the process. Higher-level code obfuscation methods have also been studied; they include more modifications with the explicit goal of making code reverse engineering hard for an attacker [Wang et al. 2001]. Code obfuscation can be automated, resulting in different implementations of the same program.

4- Research challenges

Some of the open issues with diversity techniques for intrusion tolerance are the following:

- Which automatic diversification techniques can be used to obtain diversity of vulnerabilities?
- Can automatic diversification be used at all necessary levels: operating system, middleware, and applications?
- Are there other techniques that are more effective?
- Diversity has a cost, so what kind of diversity (COTS, application software,…) is more cost effective, i.e., avoids more common vulnerabilities for the lowest cost?
- Assuming enough diversity can be created, how does the cost of running a system implemented by a distributed protocol, affect the system in its environment? How much higher would such systems place the bar for typical attackers?
- How can large intrusion-tolerant systems be realized in practice?

5- References

[Barrantes et al. 2003] Barrantes, E., Ackley, D, Palmer, T, Stefanovic, D. and Zovi, D. (2003). Randomized instruction set emulation to disrupt binary code injection attacks. In Proceedings of the 10th ACM Conference on Computer and Communications Security, pages 281–289.

[Deswarte et al. 1998] Deswarte, Y, Kanoun, K and Laprie, J-C (1998) Diversity against Accidental and Deliberate Faults. Computer Security, Dependability, & Assurance: From Needs to Solutions, IEEE Press.

[Fraga and Powell 1985] Fraga, J. S. and Powell, D. (1985). A fault- and intrusion tolerant file system. In Proceedings of the 3rd International Conference on Computer Security, pages 203–218.

[Littlewood and Strigini 2004] Littlewood, B and Strigini, L (2004) Redundancy and diversity in security. Proc. ESORICS 2004, 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, September, pp. 423-438, Springer-Verlag, Lecture Notes in Computer Science 3193.

[Obelheiro et al. 2006] Obelheiro, R, Bessani, A N, Lung, L C, Correia, M (2006) How Practical Are Intrusion-Tolerant Systems? Technical Report DI/FCUL TR-06-15.

[Ostrovsky and Yung 1991] Ostrovsky, R. and Yung, M. (1991). How to withstand mobile virus attacks. Proceedings 10th ACM symposium on Principles of distributed computing (PODC), pages 51-59, ACM, 1991.

[Wang et al. 2001] Wang, C., Davidson, J., Hill, J., & Knight, J. (2001). Protection of Software based Survivability Mechanisms. In Proceedings of the International Conference of Dependable Systems and Networks, pages 193–202.

[Xu et al. 2003] Xu, J., Kalbarczyk, Z., and Iyer, R. (2003). Transparent runtime randomization for security. In Proceedings of the 22nd IEEE Symposium on Reliable Distributed Systems, pages 260–269.

GD3 - Interoperability for diversity

1- Definition

Using functionally-equivalent but diversely implemented off-the-shelf (OTS) software components in replication-based, fault-tolerant configurations is an attractive approach to achieving system resilience at low cost. This approach may be used for different purposes, such as:

- developing a *replacement* for an *existing* off-the shelf product of insufficient quality. For instance, replacing a particular database server such as Oracle with a databases replication solution, in which several off-the-shelf database servers from different vendors are used;
- developing a *new product*: a fault-tolerant solution of high resilience to meet a particular demand, not intended to replace an existing product in use by existing systems.

In the current market situation, sets of similar OTS software products are often available (from different vendors, or even from a single vendor, usually offering a range of similar products targeted at different market segments) as building blocks for diverse replicated systems. Applying this approach in practice, however, is often problematic, because the available diverse OTS items are not functionally identical: they may present different interfaces for the same functions, or semantic differences between the meanings of these functions, or the sets of functions they offer may not be identical, although they overlap. Such functional differences may be relatively easy to alleviate for the *new product* option above: one may limit the functionality of the fault-tolerant solution to only the subset of the functionality of the diverse OTS items used that is syntactically and semantically identical, or alleviate some of the differences by wrapping (see *infra*). The problem is harder for the replacement option: it may be difficult to achieve for the newly developed fault-tolerant solution full compatibility with an existing product (for instance, to create a fault-tolerant configuration of diverse OTS database server products that can act as plug-in replacement for an Oracle databases server). Even with diverse OTS products that are functionally identical, difficulties in designing fault-tolerant, diverse OTS-based solutions may arise from non-functional properties. For instance, with regard to performance, combining a very fast product and a slow product, even if fully compatible with each other, may result in a fault-tolerant solution that is too slow; with regard to dependability, combining diverse OTS components that turn out not to be reliable enough (individually) or not diverse enough in their failure behaviour, may not produce a substantial dependability gain. Until these various practical problems are solved, the practical utilization of diversity with OTS products will remain limited. We believe that further research can provide the necessary ideas for full exploitation of the benefits of software diversity in a wide range of application contexts.

2- Examples / Application Domains

The potential for using diversity with OTS components has been studied, for instance, for database servers (see [Gashi et al, in print] and references therein), documenting the multiple compatibility issues that arise even among SQL standard-compliant servers, and need to be solved via additional system components (for instance to translate between SQL "dialects") and/or application programming rules (e.g., avoiding the use of certain features of each product).

A wide range of application domains can be envisaged where the approach of diversity with OTS components could be a viable means to address various aspects of resilience (e.g. fault tolerance against accidental faults and/or intrusion tolerance), and thus research on this topic would bring benefits. These domains are characterised by the presence of a significant level of standardisation, be it at the level of a product type, e.g. databases, web-services, etc., or of an application domain, e.g. healthcare.

3- Current Approaches

The current approach is simply to *statically* select the most suitable products to be used in the intended software fault-tolerant architecture.

Such an approach is easier in the case of products built to comply with well defined *standards*. Standards are created by different bodies: e.g. ANSI/ISO, OASIS, individual companies creating *de facto* standards, etc. They address for instance:

- Standards for categories of general-purpose products, e.g.:
    - SQL standard [ANSI/ISO, 2003] for relational DBMS products. Products usually comply with at least the "Entry Level" of the SQL language. In some cases, "de facto" standards may be adopted, including their proprietary extensions/deviations from the international standards. A very recent trend is to adapt existing products so that they become API compliant with the leading product in a particular problem domain. Notable examples of this trend are EnterpriseDB [EnterpriseDB, 2006], and Fyracle [Janus-Software, 2006], based on open source products PostgreSQL and Firebird, respectively, which offer a degree of compliance with Oracle database server.
    - Sun's J2EE Platform for application servers, which has led to a set compatible products.
    - Web-services related standards, e.g., UDDI [Clement, 2004], WSDL [Chinnici et al., 2007], Web Service Interoperability (WS-I) [WS-I, 2007].
- Application domain specific standards, which allow for easy data exchange between organisations in the same domain (e.g., hospitals, banks, etc.) despite the different software solutions they may use:
    - High Level Seven (HL7) [HL7, 2007] in the healthcare domain, which uses XML technology. The purpose of this standard is to define a common data format for patient records,
    - Society for Worldwide Interbank Financial Telecommunication (SWIFT) Network [SWIFT, 2007], which operates a worldwide financial messaging network allowing banks and other financial institutions to reliably and securely exchange messages.

There are also efforts towards formalising the process of selecting OTS software items for interoperability. A recent summary of this body of research (with pointers to other pertinent sources) [Bhuta et al., 2007] defines a total of 38 attributes of interoperability. Although such research does not explicitly address the problem of selecting products for building diverse solution, one might expect that products with higher interoperability scores will be more suitable for deploying diverse redundancy than those with lower scores. However, the set of attributes selected in the cited study (general attributes such as name, role, type, version, some interface attributes such as "binding", control, data, error handling, etc.) appears inadequate for capturing the potential compatibility issues specific to a type of application.

Selection of OTS components according to non-functional properties only, dependability, performance etc. was recently addressed in [Gashi et al., 2007]. This work ranks the six pairs of SQL servers that can be constructed with four different SQL servers, using the logs of known faults as indication of future reliability of the diverse pairs. A major problem in the study was addressing the *functional incompatibility* between the products, due to their implementing different SQL "dialects".

Some functional incompatibilities between the products can be dealt with using wrappers to compensate for the differences between them (see e.g. [Popov et al., 2004; Gashi et al., 2006] in the context of DBMS).

4- Research Challenges

Solutions are needed to support the use of diverse off-the-shelf software for resilience. Among these:

- *Wrapping* has demonstrated potential for solving the problem of incompatibility in the case of statically deployed and executed systems. However, developing wrappers as bespoke software may be failure-prone and defeat the purpose of achieving high system dependability at low cost. A possible research direction is automating the process of building wrappers, e.g. by first defining the differences between OTS products that the wrapper must compensate for, and then using these definitions as inputs to tools to generate wrappers, possibly aiming to achieve formal proof that the resulting wrappers guarantee the required compatibility properties given the definition of the differences.
- Adequate solutions are needed to bridge the incompatibilities (just in the syntax or also in semantics and sets of functionalities) between off-the-shelf items for cases when they are to be selected at *run-time* (e.g., in the use of web-services). It is a largely unexplored area, and it is even unclear as yet whether the problem is solvable.
- *Selecting* off-the-shelf items for diversity is an important step irrespective of whether the replicated system is built statically (prior to deployment) or at run time. The state of knowledge in the area of interoperability is immature for this purpose and further study is needed. Methods based on measurement [Gashi et al., 2007] are also problematic, especially to deal with multiple attributes. Further research is therefore needed to overcome these difficulties.

5- References

[ANSI/ISO, 2003] ANSI/ISO. Information technology - Database languages - SQL.

[Bhuta, J., et al., 2007] Bhuta, J. and Boehm, B. (2007). Attribute-Based COTS Product Interoperability Assessment. Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), Banff, Alberta, Canada, pp. 163-171.

[Chinnici et al, 2007] Chinnici, R., Moreau, J.-J., Ryman, A. and Weerawarana, S. (2007). "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language." Retrieved 22 August 2007, 2007, from http://www.w3.org/TR/wsdl20/ W3C.

[EnterpriseDB, 2006] EnterpriseDB.. "EnterpriseDB." from http://www.enterprisedb.com/.

[Gashi 2006] Gashi, I. and Popov, P. (2006). Rephrasing Rules for Off-The-Shelf SQL Database Servers. The 6th European Dependable Computing Conference (EDCC-6), Coimbra, Portugal, pp. 139-148.

[Gashi 2007] Gashi, I. and Popov, P. (2007). Uncertainty Explicit Assessment of Off-the-Shelf Software: Selection of an Optimal Diverse Pair. Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), Banff, Canada, pp 93-102.

[Gashi et al., in print] Gashi, I., Popov, P. and Strigini, L. Fault Tolerance via Diversity for Off-The-Shelf Products: a Study with SQL Database Servers. IEEE Transaction on Dependable and Secure Computing, in print. http://doi.ieeecomputersociety.org/10.1109/TDSC.2007.70208

[HL7, 2007] HL7. (2007). "Health Level Seven." from http://www.hl7.org/.

[Janus-Software, 2006] Janus-Software. (2006). "Fyracle." from http://www.janus-software.com/fb_fyracle.html.

[Jordan 2007] Jordan, D. and Evdemon, J.. "Web Services Business Process Execution Language Version 2.0." Retrieved 22 August 2007, 2007, from http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html OASIS.

[Popov, P., et al., 2004] Popov, P., Strigini, L., Kostov, A., Mollov, V. and Selensky, D. (2004). Software Fault-Tolerance with Off-the-Shelf SQL Servers. 3rd International Conference on COTS-based Software Systems, ICCBSS'04, Redondo Beach, CA USA, Springer, pp. 117-126.

[SWIFT, 2007] SWIFT. "Society for Worldwide Interbank Financial Telecommunication." from http://www.swift.com/.

[WS-I, 2007] WS-I. "Web Service Interoperability Organization." from http://www.ws-i.org/.

GD4 - Human diversity and human-machine diversity

1- Definition

Large socio-technical systems (e.g., health care systems, air traffic control) have increasingly complex architectures, where multiple diverse users may be using the same piece of technology (e.g. a computerised decision support) or diverse versions of technology. Users and pieces of automation act as redundant parts in these complex systems, providing error detection and correction for one another's tasks; hence diversity among their failure behaviours (diversity among different users' failures as well as between the users' and the machines' failures) must be ensured and evaluated, as a vital factor of resilience.

Different people will bring to their tasks different working practices, background experience, levels of expertise, strategies to obtain and use information, etc. This heterogeneity of people may play a positive role – as effective diversity, improving resilience – by reducing the probability of their failing together when cast in redundant roles in the same system. But it also complicates the achievement and assessment of effective diversity, because different people may "interchangeably" fulfill the same role in a system (or in many replicas of a system), so that the resulting resilience must be assessed as an average across individual variations.

Additionally, users will adapt to their perception of the reliability (or not) of other components (whether human or computer) and this also affects diversity: for instance, acquired trust in another component may reduce a user's tendency to detect its failures.

Understanding (measuring, predicting and modeling the effects of) human-human diversity (as well as human-machine diversity) is crucial to determine the effect of the existing levels of diversity on actual resilience, and therefore needs to be considered in the design, implementation and evaluation of complex systems.

2- Examples / Application Domains

Examples of these issues, in complex human-machine subsystems embedded in large socio-technical systems, exist for instance in health care delivery, especially in the forms in which it is expected to evolve. For example, decision support tools for intensive care monitoring are expected to be used by clinicians with different roles and goals (e.g. nurses and doctors) and different degrees of expertise (e.g. consultants and junior doctors). In a single instance of use, different members of staff provide a degree of protective redundancy while relying on a piece of technology for advice as well as for communication among them.

Similar situations can be observed in other application domains: in air traffic control, where pilots and operators with different or redundant roles monitor the air space using the same set of computerised tools; or in some E-voting schemes [Ryan, 2005; Ryan et al, 2006], where combinations of people (in multiple roles) and machines are entrusted with detecting failures and attacks, manifested in either human or computer behaviour.

3- Current approaches

A line of research on fault tolerance by City University has focused on analyzing and modeling diversity and common-mode failure in redundant and diverse computer (typically software) components [Littlewood, Popov and Strigini, 2001]. More recently, this approach has also been applied to studying diversity and redundancy between a computer tool and its human user, leading to useful and novel findings [Strigini, Povyakalo and Alberdi, 2003; Alberdi et al. 2005]. This work only considered a single user and a single machine (e.g. a clinician using a computerized decision support tool), and can be seen as adding mathematical and psychological results to an existing literature strand on "automation bias". In summary, these new results demonstrate via both empirical and modelling evidence how (human-human & human-machine) diversity in complex socio-technical systems will be affected by: a) the differences among users who may "interchangeably" assume a given role in the system; and b) the users' evolving perceptions of the machine's (and other users') dependability.

Human-human diversity and redundancy have been addressed by psychological studies dealing with group decision making [Hastie, 1983; Zarnoth, Sniezek, and Janet, 1997; Sorkin, Hays and West, 2001] and the integration or pooling of judgements from multiple experts [Lock, 1987; Budescu and Rantilla, 2000]. Although the prescriptive (normative) and descriptive (behavioral) outcomes of these lines of research can provide useful insights for addressing some aspects of this gap, they are normally far removed from the practical constraints of the human-computer interactions in complex socio-technical systems.

Some of the issues of adaptation and its effect on diversity in socio-technical systems are addressed in the UK project INDEED.

4- Research challenges

The challenges include:

- Extending previous models of human-machine diversity to more complex socio-technical architectures including more humans and machines;
- Using these models (complemented with empirical work) to determine what combinations of users and users and technology are more effective, e.g., by analyzing the correlations of failure probabilities among the users and between the different users and the computerised tools;

- To this end, studying the psychological mechanisms involved in human adaptation and its effects on diversity.

5- References

[Alberdi et al. 2005] Alberdi, E., Povyakalo, A., Strigini, L., Ayton, P., Procter, R., Hartswood, M., and Slack, R. (2005) The use of Computer Aided Detection tools in screening mammography: A multidisciplinary investigation. British Journal of Radiology, 78, 31-40.

[Budescu and Rantilla, 2000] Budescu, D. V. and Rantilla, A. K. (2000) Confidence in aggregation of expert opinions, Acta Psychologica, 104(3), 371-398.

[Hastie, 1983] Hastie, R. (1983) Review Essay: Experimental Evidence on Group Accuracy. In B. Grofman & G. Owen (Eds) Information Pooling and Group Decision Making: Proceedings of the Second University of California, Irvine, Conference on Political Economy. Greenwich, CT: JAI Press.

[Littlewood, Popov and Strigini, 2001] Littlewood, B., Popov, P. and Strigini, L. (2001). Modelling software design diversity - a review. ACM Computing Surveys, 33, 2, 177-208.

[Lock, 1987] Lock, A. (1987) Integrating group judgment in subjective forecasts. Judgmental forecasting; edited by George Wright and Peter Ayton. pp. 109-127.

[Ryan, 2005] Ryan, P. Y. (2005) A variant of the Chaum voter-verifiable scheme. In Proceedings of the 2005 Workshop on Issues in the theory of Security, WITS '05, pp 81-88.

[Ryan et al, 2006] Ryan, P. Y. at al, "High assurance voting systems" in *Resilience-Building Technologies: State of Knowledge* , ReSIST NoE Deliverable D12, 2006, http://www.resist-noe.org/Publications/Deliverables/D12-StateKnowledge.pdf

[Skitka, Mosier and Burdick 1999] Skitka, L. J., Mosier, K. L. and Burdick, M. (1999). Does automation bias decision-making? International Journal of Human-Computer Studies, 51, 5, 991-1006.

[Sorkin, Hays and West, 2001] Sorkin, R. D., Hays, C. J. and West, R. (2001) Signal detection analysis of group decision making. Psychological Review, 108, 1, pp.183-203

[Strigini, Povyakalo and Alberdi, 2003] Strigini, L., Povyakalo, A. and Alberdi, E. (2003). Human-machine diversity in the use of computerised advisory systems: a case study. In: Proceedings of DSN 2003, International Conference on Dependable Systems and Networks, San Francisco, pp. 249-258.

[Zarnoth, Sniezek, and Janet, 1997] Zarnoth, P., Sniezek, J. and Janet, A. The social influence of confidence in group decision making. Journal of Experimental Social Psychology. Vol 33(4), 345-366.

GD5 - Spontaneous Redundancy in Large Systems

1- Definition

Many of the systems studied in ReSIST provide, as a side effect of how they are built, large amounts of redundant resources (technical and human) and redundant executions of tasks, with varying degrees of diversity. It is important to assess and harness the effect of this "undesigned" redundancy on system resilience:

- If we assess a system without considering this aspect, we may end up with over-pessimistic expectations and thus sub-optimal decisions;
- We need to be able to assess whether changes (designed or spontaneous) in a system would reduce this "unseen" redundancy and diversity, and thus degrade the system's resilience
- It may be useful to alter the designed parts of the system to better harness this "free" redundancy.

With spontaneous redundancy, the first problem may be *discovering* it, followed by assessing its effectiveness, i.e. the level of failure diversity between the redundant elements. The latter, in turn, will normally vary between threats (causes of failure), and more so in the case of "spontaneous" diversity: two communication links might never fail together due to software bugs in the software controlling them, but be vulnerable to injudicious digging because they run through the same trench; many people may fail essentially independently through attention slips, but be vulnerable to the same cognitive biases when presented information in a certain format. The problem that this presents is a dual problem of the standard problem considered in the literature on software fault tolerance, namely, the problem of how to select aspects of the redundant component to make diverse so as to *create by design* sufficient failure diversity to be effective against the threats of interest.

2- Examples / Application Domains

Examples abound in multiple application domains:

- Systems with many users sharing common resources provide extra opportunities for error detection and correction. A few instances follow:
    - Open-source software development is often claimed to produce more effective, fast, and/or cost-effective dependability improvements than allowed in more controlled co-operative environments: "given enough eyeballs, all bugs are shallow". The development community is a comparatively low-technology, loosely coupled system, but as an example it highlights some important research questions that may arise in the presence of largely undesigned redundancy: how many eyeballs are really watching [Dinh-Trong, 2005]? How effectively diverse are their contributions? What degree of diversity would we need, for the dependability growth to beat that of a comparable non-open development community [Bosio et al, 2002]?
    - A similar, but dynamic "community effect" exists to some extent in the monitoring of new threats on the Internet
    - In E-voting, many proposed schemes [Randell, B. and Ryan, P.Y.A., 2006] expect that "enough" voters will effectively act as monitors of the orderly progress of voting and vote-counting
    - In health care, there is a trend towards patient record databases allowing multiple "health providers" to share the complete clinical history of a patient in a far more complete and efficient way than before. Thus, multiple physicians may be able to repeat diagnoses and query previous clinical decisions, noticing slips and other mistakes. This phenomenon could be exploited for monitoring clinical errors (an important cause of mortality) or even for correcting them (counterbalancing the loss, though automation, of some traditionally available forms of redundancy – e.g., checks of prescriptions by pharmacists). Both goals could well bring great benefit, or backfire through unintended social consequences.
- It is now common to be able to contact a person through multiple electronic means, e.g. mobile phones, land line phones, Email through various alternative servers. This combination is commonly used in everyday business and has been exploited to preserve connectivity despite communication cuts caused by natural disasters or hostile action. However, the actual degree of diversity among these various media changes dynamically, both with short-term changes (an ISP changing supplier of physical cable bandwidth) and longer-term trends (diversification or consolidation in an industrial sector).

3- Current approaches

There are many studies about the potential effects of redundancy in large systems modelled as networks of interconnected nodes; fewer about how to discover this redundancy; and fewer yet about the degree of diversity that such redundancy affords and the factors determining diversity. A number of studies about large systems address the effect of topology on failure propagation, one of the possible sources of common-mode failures, for instance comparing the resilience to attack of networks with different topologies, characterised by macroscopic properties (e.g., scale-free networks). A review can be found in [Boccaletti, 2006]. In related studies the consequences of network topology for maintenance strategies have also been studied [Buzna et al., 2007]. Here the interest is in cascade failures and epidemic-like spreading (i.e., failure of a node in the network affects its neighbours in a way similar to the propagation of infectious diseases; a similarity exploited in some studies of attacks on the Internet [Pastor-Satorras and Vespignani, 2001]). There have also been studies in sociology and engineering about how a system of *weakly-coupled* components with oscillating dynamics can achieve synchronous dynamical behaviour [Strogatz, 2001]. Current approaches use a mixture of empirical work, high-fidelity simulation and approximate physics-based network models (e.g. using mean field theories) [Reka et al., 2002; Boccaletti, 2006].

In contrast with this degree of interest in redundancy and failure *propagation,* there is relatively little research on the genesis of diversity with respect to *common cause* failures, due to a cause that simultaneously affects multiple redundant components. Research on common cause failures exists for engineered systems [Littlewood et al, 2001] but only with respect to small numbers of variants. With spontaneous diversity, we may need to assess the probability of common failures among a potentially large number of variants (for instance multiple people, whose skills and attitudes may vary in a wide multi-dimensional spectrum), affected by common environmental conditions, types of attacks, etc. The FP6 IRRIIS project is addressing initial steps in the required modelling, as part of modelling interdependencies between critical infrastructures.

4- Research challenges

There are various empirical and theoretical questions:

- Mathematical models of the effects of diversity have mostly been studied for systems with small numbers of redundant components belonging to a few discrete types [Littlewood, 2001]. We need to model many components with diversity of various types (technology, geography, ...) some of which may vary along a continuum (e.g. people's skills). Another difference from previous work is that the redundancy in large systems may result in more complex architectures than those studied for small embedded computer systems. The results, apart from general insight, should include definitions of useful metrics of effective diversity, to be used in empirical studies;
- Empirical studies need to characterise the actual effects of unplanned redundancy in its various forms, some of these studies requiring the application of social science methods, others of engineering measurement; apart from descriptive results, methods may be needed for helping to organise the discovery of redundancy and diversity, in evolving systems with ill-defined boundaries;
- Algorithms and protocols are needed to organise the exploitation of unintended diversity. To some extent these already exist in the form of various distributed and peer-to-peer schemes, but there are open questions about analysing such algorithms and protocols with respect to common cause failures and exploiting redundancy between people and across different technologies.

5- References

[Buzna et al., 2007] L. Buzna, K. Peters, H. Ammoser, C. Kühnert and D. Helbing (2007) Efficient response to cascading disaster spreading. Physical Review E 75, 056107, http://link.aps.org/abstract/PRE/v75/e056107

[Boccaletti, 2006] Boccaletti, S., V. Latora, Y. Moreno, M. Chavez and D.-U. Hwang (2006). Complex networks: Structure and dynamics, Physics Reports, Volume 424, Issues 4-5, February, 2006, pp 175-308. http://www.sciencedirect.com/science/article/B6TVP-4J0WTM2-1/2/739f254d99ff14ca96565d3d34a6d77a.

[Littlewood et al, 2001] Littlewood, B., Popov, P. and Strigini, L. (2001) "Modelling software design diversity - a review". ACM Computing Surveys, vol 33, No 2, pp 177-208.

[Pastor-Satorras and Vespignani, 2001] Romualdo Pastor-Satorras and Alessandro Vespignani (2001). Epidemic spreading in scale-free networks, Physical Review Letters, vol 86, pp 3200-3, http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0010317

[Randell, B. and Ryan, P.Y.A., 2006] Randell, B. and Ryan, P.Y.A. (2006). Voting Technologies and Trust, IEEE Security & Privacy, Volume 4, Issue 5, pp 50-56

[Reka, 2002] Reka, Albert and Albert-Laszlo Barabasi (2002), Statistical mechanics of complex networks, Reviews of Modern Physics, vol 74, http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0106096

Rosato V., L.Issacharo, F.Tiriticco, S.Meloni, S. De Porcellinis, R.Setola (2006) Modelling interdependent infrastructures using interacting dynamical models, Int. J. Critical Infrastructures, Accepted for publication.

[Strogatz, 2001] Strogatz SH.,Exploring complex networks (2001), Nature, Mar 8;410(6825):268-76.

[Dinh-Trong, 2005] Trung T. Dinh-Trong, James M. Bieman (2005). The FreeBSD Project: A Replication Case Study of Open Source Development, IEEE Transactions on Software Engineering, Vol. 31, No. 06, pp. 481-494, Jun., 2005.

GD6 - Reconfiguration and Contextual/environmental issues

1- Definition

Command and control systems have to handle a large amount of increasingly complex information. Current research work in the field of Human-Computer Interaction promotes the development of new interaction and visualization techniques in order to increase the bandwidth between the users and the systems they are interacting with. These techniques employ multimodal combinations of input/output devices, and increase the bandwidth of the interaction by involving more perceptive and motor element from the user, so that for instance when one of the user's senses – e.g., sight – is overloaded, another – e.g., hearing – is used.

Such an increase of bandwidth can have a significant impact on efficiency (for instance number of commands triggered by the users within a given amount of time) and also on error rate (the number of slips or mistakes [Reason, 1990] made by the users). However, in the case of adverse events the user interface part of the command and control system may have to be **reconfigured** (possibly at run time i.e. while the system is still in use) in order to provide an optimal level of usability according to the new capabilities of the system/operator couple. In this discussion we emphasize the usability-related aspects of reconfiguration (as described in the examples provided in the next sections) but the concepts are generic enough to be applicable to other aspects such as packets re-routing in computing networks or, more generally, to resource management.

Multi-modal interfaces offer, as far as input is concerned, the possibility to propose variations in the interaction techniques and, as far as output is concerned, the possibility to exploit different output devices according to different criteria. Such **diversity** in the combination of interaction techniques supports **usability** (by adapting the interaction to the users' capabilities or tasks) or **availability** and **reliability** (in case of a malfunction of some equipment for instance). The level of usability must remain similar, and should not go below a certain level, for any combination of input/output devices.

2- Examples / Application Domains

The first example is extracted from the cockpit of the Airbus A380 where 8 display units are available. 4 of them offer interaction via a mouse and a keyboard by means of an input device called KCCU (Keyboard Cursor Control Unit). Interactive applications are allocated to the various display units. The ND (Navigation Display) and the PFD (Primary Flight Display) are not interactive and are of primary importance. These two application are allocated to two display units located on the external side of the cockpit while the other applications like, radio, Eccam … are distributed on the other four central display units. If one of the display units fails, then the applications are migrated to available display units according to predefined criteria (like the importance of the application with respect to the current flight phase, the current screen space availability ...).

A second example is a Ground Segment multimodal application. Interaction can take place through 4 input devices: 2 graphical input devices (like a mouse and a trackball available at the same time), a touch screen and a microphone. Failure of an input device requires the **reconfiguration** of the interaction techniques. Synergistic use of multiple input devices (like voice + graphical selection used for opening a modification menu by uttering "modify" and clicking on a procedure) requires **reconfiguration** of interaction techniques taking into account only the available input devices and exploiting the "best" interaction technique available.

3- Current approaches

Management of input configurations has been addressed in work such as ICon [Dragicevic & Fekete, 2002] but this work only addresses the static aspects of input configurations. Related work such as that on plasticity of user interfaces (see [Thevenin & Coutaz, 1999] for a research agenda) addresses both output and input but places far greater emphasis on the output [Calvary et al., 2004]. In such work, the set of input devices is predefined and static but the interaction technique may evolve according to the context of use (i.e. whether the application is used on a PDA or on a workstation).

As for modelling techniques, research work address user interface descriptions at different levels of abstraction with the main goal of "generating" the user interface according to the characteristics of the input and output devices available on the target platform such as work described in [Bertie et al., 2004a] or [Molina, Massó et al., 2005]. Instead of using high level modelling languages (as for instance formal description techniques) for supporting assessability, testing and validation such work focuses on implementation aspects making the production of user interfaces for multiple platforms cheaper and faster, by using programming level languages like XML-based dialects [Bertie et al., 2004b]. Work has already been done on multimodal interfaces and multimodal interaction high-level modelling such as [Willans & Harrison, 2001], [Navarre et al., 2005] (or see [Navarre et al., 2006] for a more precise state of the art in this field) but this work only addresses the modelling of a user interface and not how to specify multiple user interfaces and interaction techniques according to the diversity of input and output devices.

The same concerns apply to the other phases of the lifecycle, like validation or testing. These phases are addressed even less in the literature in general. Exceptions can be shown, with work such as [Madani et al., 2005] which, however, addresses only superficially the issues raised by usability. New interfaces such as user interfaces for games, or, more generally speaking, multimodal user interfaces are bringing additional requirements and constraints that are summarized in [Palanque et al., 2006].

No research contribution has been identified related to the reconfiguration of interaction techniques in a resilient context. This is due to the fact that designers in the field of Human-Computer Interaction mostly consider the design of interfaces for a given application and not of the interaction in general. This aspect has been emphasized in [Beaudouin-Lafon, 2004]. However, this paper's "related work" section provides useful information for addressing that gap.

4- Research challenges

Assessment of the usability and resilience of interfaces and interactions configurations is critical for the deployment of solutions to this gap in the fields of context aware, ubiquitous or safety-critical applications. While usability assessment has been addressed in the past (even for multimodal interfaces) the diversity aspect of the problem creates new and difficult challenges.

Hardware technology is already available and issues related to methods and tools have already (partly) addressed the static aspect, meaning that this gap could be addressed within a short time frame. The main difficulties can be foreseen in handling the dynamic aspects while providing ways of guaranteeing usability, safety and the assessability of the various configurations and reconfigurations.

5- References

[Reason, 1990] Reason, J. Human Error, Cambridge University Press (1990)

[Dragicevic & Fekete, 2002] Dragicevic, P & Fekete, J.D. (2002). ICON: Input Device Selection and Interaction Configuration. Companion proceedings of UIST'02, 15th Annual Symposium on User Interface Software and Technology, Paris, October 2002.

[Thevenin & Coutaz, 1999] Thevenin, D & Coutaz, J. Plasticity of User Interfaces: Framework and Research Agenda In Proc. Interact99, Edinburgh, , A. Sasse & C. Johnson Eds, IFIP IOS Press Publ. pp.110-117

[Calvary et al., 2004] Calvary, G., Coutaz, J., Dâassi, O., Balme, L., Demeure,(2004) A. Towards a new generation of widgets for supporting software plasticity: the "comet", Joint EHCI-DS-VIS (Engineering HCI, Design, Specification and Verification of Interactive Systems), Hambourg, July 2004.

[Bertie et al., 2004a] Berti, S., Correani. F, Mori, G., Paternò, F., Santoro, C. (2004) "TERESA: A Transformation-Based Environment for Designing Multi-Device Interactive Applications, ACM CHI 2004, Extended Abstract, pp.793-794, ACM Press, April 2004, Vienna, Austria.

[Bertie et al., 2004b] Berti, S., Correani, F., Paternò, F., Santoro, C. (2004) The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels, Proceedings Workshop on Developing User Interfaces with XML UIXML: Advances on User Interface Description Languages, May 2004, pp.103-110.

[Molina et al., 2005] Molina Massó, J.P, Vanderdonckt, J., Montero, Simarro, F., González López P. (2005) Towards virtualization of user interfaces based on UsiXML. Proceedings of the tenth international conference on 3D Web technology table of contents, Pages: 169 – 178, 2005.

[Willans & Harrison, 2001] Willans, J.S., Harrison, M. D. (2001). Prototyping pre-implementation designs of virtual environment behaviour. 8th IFIP Working conference on engineering for human-computer interaction (EHCI'01) 2001. LNCS, Springer Verlag

[Navarre et al., 2006] Navarre, D, Palanque, P., Dragicevic, P., Bastide, R.(2006) An Approach Integrating two Complementary Model-based Environments for the Construction of Multimodal Interactive Applications. Interacting with Computers, vol. 18, n°5, 2006, pp. 910-941.

[Navarre et al., 2005] Navarre, D., Palanque, P., Bastide, R., Schyn, A. Winckler, M., Nedel, L.P., Freitas, C. (2005) M.D.S. A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications. Proceedings of INTERACT 2005, Roma, Italy, September 2005, Lecture Notes in Computer Science, Springer Verlag.

[Madani et al., 2005] Madani,L., Oriat, C., Parissis, I., Bouchet, J., Nigay, L.(2005) Synchronous Testing of Multimodal Systems: an Operational Profile-Based Approach. Proceedings of ISSRE 2005, 16th IEEE International Symposium on Software Reliability Engineering, Chicago, Illinois, USA, November 8-11 2005, IEEE Computer Society, pp. 325-334.

[Palanque et al., 2006] Palanque, P., Bernhaupt, R., Boring, R., Johnson, C. (2006) Testing Interactive Software: a Challenge for Usability and Reliability. Special Interest Group, ACM CHI 2006, conference, Montréal, Canada, 2006.

[Beaudouin-Lafon, 2004] Beaudouin-Lafon, M. (2004). Designing interaction, not interfaces. In Proceedings of the Working Conference on Advanced Visual interfaces (Gallipoli, Italy, May 25 - 28, 2004). AVI '04. ACM Press, New York, NY, 15-22.

[Fitts, 1954] Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47, 381-391.