ReSIST: Resilience for Survivability in IST

A European Network of Excellence

Contract Number: 026764

Deliverable D33: Resilience-Explicit Computing: final

Report Preparation Date: December 2008
Classification: Confidential

Contract Start Date: 1st January 2006
Contract Duration: 39 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 Ciencas 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

Contents

1 Introduction

This report forms part of Deliverable D33, from ReSIST Work Package 1 (Integration Technologies). In accordance with the Programme of Work (D22), the deliverable is:

> *a second edition on support for resilience explicit computing, covering the augmentation and enhancement of the RKB, and reporting on the challenge problem workshops*.

The report is an update to Deliverable D11 - Support for resilience-explicit computing (first edition), which demonstrated "how resilience mechanisms can be represented in terms of resilience metadata" and described the "extended resilience ontology, with reference to the content and organisation of the validated knowledge base."

In common with the first edition Deliverable D11, this deliverable has two components. First, the augmented ReSIST Resilience Knowledge Base (RKB) contains "second edition" descriptions of resilience mechanisms in terms of resilience metadata, based on the ReSIST resilience ontology. These descriptions include extensive coverage (at an overview level) of techniques described in IEC 61508 (Part 7). The second component is a summary report on the three challenge problem workshops held during 2008, reviewing the aims of each workshop, the participating organisations, issues covered and next steps.

1.1 Resilience-Explicit Computing

A full description of Resilience-Explicit Computing is provided in Deliverable D11. To summarise, the *resilience-explicit* (Res-Ex) approach aims to support the achievement and prediction of system resilience by making explicit the resilience-related properties of components and infrastructure. These properties are described in terms of *metadata*, which can be used at design time to inform decisions about the choice of design patterns and development tools, or potentially at run-time to tune or reconfigure, maintaining resilience.

We use the term *resilience-explicit computing* to encompass both the design-time and run-time use of resilience-related metadata. Res-Ex computing requires effective descriptions of metadata and of the mechanisms that may be deployed or configured in order to meet a resilience target. We use the term *resilience mechanism* to refer to any design pattern, technique or tool intended to improve system resilience. Examples include fault-tolerant architectural patterns (e.g. n-version programming) and development tools (e.g. robustness testing tools).

We use the term *metadata* to refer to resilience information, about components or mechanisms, on which human or machine decision-makers act. Potential metadata range from a person's workload in a socio-technical system to known failure modes declared in the functional specification of a system. Such metadata could be perceived by an observer, and could be predicted or historical. The metadata could also be declared at different levels in the system, such as components, the whole system or even for the user-interface of the system.

We seek to inform the decisions to select a particular resilience mechanism from among alternatives and to instantiate or configure the mechanism for a specific application. Such decisions may be made statically, at design time, or implemented dynamically within a running system. In either case, in order to plan to reach a

resilience target, the decision-maker requires metadata about the characteristics (e.g., failure rates) of components, infrastructure and environment, and descriptions of the resilience mechanisms in terms of their effects on metadata, for example failure rate of a fault tolerant assembly in terms of failure rates of its components, or metadata generated by a robustness testing tool. The resilience mechanism descriptions may be combined with system metadata to obtain a prediction of the consequences of a particular selection or configuration.

The goal of our work in ReSIST has been to encourage the community to give descriptions of mechanisms and metadata that support these decision-making processes. In particular, we wish to promote the contribution of mechanism descriptions in a form that might enable automated analysis. There is currently very little support for gathering such descriptions or for making use of them. The descriptions of resilience mechanisms available to practitioners at present are deeply embedded in the scientific literature and are in many cases hard to extract. We wish to encourage researchers developing new mechanisms to give descriptions that help answer the question "What exactly does this mechanism achieve in terms of resilience?" We hope thereby to encourage research to evaluate existing and new mechanisms, and scholarship in codifying that information and making it readily available to practitioners.

The work of ReSIST Task IT-T2 has been to develop a means of recording descriptions of resilience mechanisms that are based on metadata and which integrate with the Resilience Knowledge Base (RKB). This allows mechanism descriptions to be linked to other resilience knowledge through the emerging ontologies and through the research and training/education data embedded in the RKB.

1.2 Approach

In order to make progress towards our goal of providing resilience-explicit guidance for the developer community, our aim was to provide metadata-based descriptions of a large number and wide range of resilience mechanisms. To this end we asked specialists across the ReSIST network to provide a preliminary and broad-ranging set of "first edition" descriptions. The mechanism descriptions provided were described in Deliverable D11.

A "second edition" set of additional mechanisms have been contributed by Network partners, augmented by outline descriptions of techniques listed in IEC61508 Part 7, a list of techniques specifically related to safety. The complete set of mechanism descriptions now available is described in overview in Section 2, and these have all been included in the on-line Resilience Knowledge Base (RKB). This is accessible to readers at http://resist.ecs.soton.ac.uk/resex/.

The resulting set of mechanisms provides an extension of the content of the RKB, but it is also necessary to extend our understanding of how the mechanism descriptions can contribute to more effective design and development for resilient systems.

To help us to begin to develop our understanding, the second strand of work in Task IT-T2 has been to hold a series of workshops centred on a set of "challenge problems". Preparation for each workshop delineated the topic and a core of ReSIST participants; the workshops were then opened (via the ResEx SIG) to all of the

ReSIST partners and to any identified external participants with a close interest in the designated topic.

Three workshops were held during 2008-09: "Challenge Problems for Resilience-Explicit Computing in Grids", held at Pisa, July 2008; "Challenge Problems for Resilience-Explicit Computing in Critical National Infrastructures", held at Malvern, November 2008; and "Challenge Problems for Resilience-Explicit Computing with Assistive Technologies", held at Newcastle, December 2008.

The generic objectives for the workshops were to establish working groups that could explore:

- how resilience meta-data could be exploited at design-time and/or run-time
- how resilience mechanism descriptions in the RKB could be utilised
- how meta-data and descriptions might be enhanced for greater effectiveness, and
- how to focus future research towards techniques and tools for modelling, analysis, design and implementation of resilience-explicit systems.

1.3 Report Structure

This report is a guide to the second edition mechanism descriptions and a summary of the efforts of the ResEx Challenge Workshops. Section 2 indicates the mechanisms that have been incorporated at this stage (these are enumerated in Annex 1). Full information on the Resilience-Explicit mechanism descriptions is stored in the on-line RKB (http://resist.ecs.soton.ac.uk/resex/). Section 3 provides a summary of the aims and operation of each of the three Challenge Workshops. Finally, in Section 4 we summarise, and look forward to future work beyond the ReSIST project, aimed at maintaining the legacy of the RKB mechanism descriptions and promoting the working groups to develop the Challenge Problem definitions.

2 Second Edition Resilience Mechanisms

In this section, we review the collection of example mechanisms included in the second edition of the Res-Ex support embedded in the RKB. The first edition mechanisms were initially offered by members of the Res-Ex SIG and, later, by other ReSIST partners. The collection included classical architectural mechanisms such as n-version programming, dynamic mechanisms such as dynamic function allocation, and design-time tools such as ModelWorks. They represent contributions from each of the initial ReSIST Working Group areas in resilience building (Architectures, Algorithms, Socio-technical systems, Verification and Evaluation).

Our second edition content aimed to address gaps in the coverage provided by the first edition sample. A list of candidate second edition mechanisms was developed with the aim of increasing the involvement of a broader group of ReSIST partners and improving the coverage of mechanisms across working groups in the ReSIST network. A number of these mechanisms have been described either in full or in outline. In addition, relevant safety-related techniques listed in IEC 61508 Part 7 (Overview of techniques and measures) have been added at an "outline" level of detail. IEC 61508 is the international standard for electrical, electronic and programmable electronic safety related systems. Part 7 lists techniques and measures to support the process of ensuring that systems are designed, implemented, operated and maintained to provide the required safety integrity level (SIL).

As a result of this effort, the total number of mechanisms included in the second edition is close to 160, with 24 of these having a complete description. The range of contributors to the mechanisms has increased, with seven of the ReSIST partners providing mechanism descriptions. Annex 1 to this report provides a full list of the second edition mechanisms.

3 Challenge Workshops

A conclusion of the initial deliverable on Resilience-Explicit Computing was that, alongside ongoing research in the realisation of support for a Res-Ex approach, there is a need for greater understanding and experience of the approaches along with their successful integration. To support this, the final stage of ReSIST work on Res-Ex computing aimed at initiating development of a series of challenge problems, including applications that required the run-time use of metadata to support dynamic reconfiguration for resilience. The development of the challenge problems was expected to deliver insight into the need for and role of metadata, and to provide a driver by which we hope to advance and integrate research.

The challenge problems are intended to serve as benchmarks for researchers and developers of tools supporting the design process or providing dynamic reconfiguration capabilities. Each is based on a specific application area. The three workshops to develop the challenge problems were all held in 2008:

- "Challenge Problems for Resilience-Explicit Computing in Grids", held at the University of Pisa, Italy, 14 July 2008
- "Challenge Problems for Resilience-Explicit Computing in Critical National Infrastructures", held at Qinetiq, Malvern, UK, 20-21 November 2008
- "Challenge Problems for Resilience-Explicit Computing with Assistive Technologies", held at Newcastle University, UK, 5 December 2008

Each workshop was attended by co-ordinators of Res-Ex work (Tom Anderson and Steve Riddle), and organised by a local host. The approach proposed for defining a challenge problem was common to all workshops:

- to select a problem (or small set of problems) which would fit the aims of ResEx;
- to create a problem statement that encompasses functional and dependability properties;
- to consider the range of conventional resilience/dependability design approaches that have been employed;
- to select one or more mechanisms to incorporate/achieve dependability, and convert these to a ResEx approach by developing a ResEx mechanism descriptor;
- to deploy the ResEx approach, by identifying the strengths and weaknesses of the ResEx description, determining the effectiveness of role of metadata and recording the experience of utilising ResEx approach, noting strengths and limitations;
- to establish membership of an ongoing, active workgroup.

The final bullet in this list is perhaps the most essential since a single workshop meeting could only prepare for and instigate the process outlined above.

3.1 Resilience-Explicit Computing in Grids

The Pisa workshop was organised by Cinzia Bernardeschi and Andrea Domenici.

The motivation for study of grids originated from discussions at a Res-Ex workshop in late 2007. In a production Grid, resilience is a property that is required both for the infrastructure itself and of the Grid services or mechanisms that applications may use to satisfy their own resilience requirements. The nature of the Grid demands availability of a large number of resources, resource virtualisation, brokering services and data replication. Tools such as GridView and SAM provide monitoring services and reliability metrics which can provide a valuable source of metadata for long-term and dynamic analysis of resilience properties. Thus it was expected that resilience-related properties and mechanisms of Grid infrastructures could be feasibly characterised in terms of the ReSIST ontology, and related to ResEx mechanisms.

Attendees at the workshop included representatives of ReSIST partners (Pisa, Malvern, Southampton, Newcastle) and domain experts (CERN, INFN). The domain experts, including members of the team responsible for the Large Hadron Collider, were able to give valuable insights into the stringent reliability and availability requirements for Grid infrastructure. A working group was established to continue the collaboration among the participants. One of its commitments was to produce a document summarizing known Resilience-explicit mechanisms that could be applied in Grid computing. The workshop report presented in Annex 2 to this report is an output from that commitment.

3.2 Resilience-Explicit Computing in Critical National Infrastructures

The Malvern workshop was organised by Nick Moffat and Colin O'Halloran (QinetiQ). The aim was to bring together people from the communities of Resilience and Security. Critical National Infrastructures (CNIs) include transport networks, utilities, emergency services and telecommunications. The workshop focussed on possible threats and security challenges to CNIs and, as with the other workshops, sought to identify a challenge problem and establish a working group.

A particular focus of the workshop was a discussion of possible electronic attack vectors against Critical National Infrastructure. Specific sectors addressed were Energy, Food, Water, Transport, Communications, Government, Health, Emergency Services and Finance. These sectors correspond with the classification by the UK Centre for Protection of National Infrastructure. Discussion concluded that the most effective attacks can be those that adversely impact public perceptions, and that modelling of security may be difficult due to the wide variety of complex factors relevant to typical national systems. Other topics included security modelling tools and their representation as a resilience mechanism, and the role of semantic web-based tools for information discovery in the context of terrorist incident reports.

Attendees at the workshop included representatives from Newcastle, Southampton and QinetiQ, St Andrew's University, and practitioners from the Centre for the Protection of National Infrastructure. Planned follow-up activities included the setting up of an email forum to include a wider range of participants throughout the UK and mainland Europe who had not been able to attend the workshop, and the arrangement of a follow-up workshop to further expose resilience and modelling techniques to

practitioners, and to make use of the Resilience Ontology work in collaboration with related projects.

The full workshop report is presented in Annex 2.

3.3 Resilience-Explicit Computing with Assistive Technologies

The Newcastle workshop was organised by Steve Riddle and Patrick Olivier (Newcastle University). Its aim was to bring together researchers and practitioners from the communities of Resilience and Assistive Technologies.

The field of Assistive Technology covers a wide range of technologies designed to support "independent living": that is, the ability to take part in normal, day-to-day activities in spite of infirmity brought on by age, disease or disability. Examples include support for medication, diet and nutrition; personal mobility; and virtual communities. Each of these examples has significant potential risks to individual safety and security: these include danger of overdose, falls or other accidents, and phishing attacks.

Attendees at the workshop came from ReSIST partners (Newcastle, Southampton, Birkbeck) and domain experts (Centre of Excellence for Life Sciences, and University of Dundee). The workshop was structured with introductory presentations followed by a specially filmed video, featuring actors in a scenario highlighting many of the problems in Assistive Technology. The workshop concluded with breakout groups discussing one of four case studies, as potential sources of material for a Challenge Problem. The full workshop report is presented in Annex 2.

4 Report summary

The overall objective for the ResEx work in ReSIST is very much for the long term: to build up our capability for designing and developing resilient systems by more effectively deploying architectures, components, on-line processes and supportive methodologies that exploit the resilience-explicit approach. In the third and final year of ReSIST we set the dual objective of expanding the scope of our coverage of "resilience mechanisms" in the RKB, and of initiating a programme of challenge workshops that would investigate, explore and refine the ways in which the ResEx approach can contribute.

4.1 Resilience-Explicit Mechanisms

Thanks to a much improved input facility and template, supported by appropriate guidance information, we were able to double the number of mechanisms that are described in detail to 24. It will always be possible to improve the description of a mechanism, especially if subsequent investigation and study provides better, and ideally quantified, information on the resilience parameters of the mechanism. The opportunity for future upgrading of mechanisms is an obvious follow-on capability that continued availability of the RKB will naturally support.

In addition to the fully described mechanisms, there are now a number of partially completed mechanisms within the RKB. Basically, these are descriptions for which readily available information has been inserted, but for which completion of all remaining fields would entail a more significant investigation. With limited resources

available we concentrated our effort on broadening and extending coverage rather than ensuring that all descriptions were completed.

It was suggested that the international safety standard IEC 61508 enumerated a breadth of safety related "mechanisms". Since this defined a valuable and wide-ranging corpus of mechanisms in which system developers would have a natural interest, we thought these mechanisms would be a valuable addition to the RKB. Our aim here was to include all suitable/relevant entries from part 7 of the standard, and that has been done. Many of the entries follow an outline format, which basically gives the name of the mechanism, and a short description of what it is and what it does but will normally include a link to some other, external, source of more detailed information. In some cases we have been able to extend these outlines to partially completed descriptions by adding information in many of the supplementary fields of the template.

Once again, the continued availability of the RKB affords the opportunity to elaborate more fully all of the outline and partial descriptions – and, indeed, to further extend the coverage by adding new mechanisms. Two obvious possibilities for strengthening the RKB's capability with respect to resilience mechanisms, in terms of review and improvement, in completion of partial descriptions, and in adding new mechanisms is via

(i) student exercises (advanced undergraduate/postgraduate/doctoral students can all have a role here), and

(ii) within the future programmes of the challenge workshops.

The quantified status of the RKB with respect to resilience mechanisms is subject to change, since we are trying to maintain an ongoing, but now small-scale, programme of development. However, we can record that in early March 2009 the total number of mechanisms included was just below 160.

4.2 Challenge Workshops

The resilience-explicit approach needs considerable further investigation before anything that could be considered a methodical development regime can be recommended. Nevertheless the approach has obvious merit, in that it proposes that decisions at design time and runtime should be rationally based on an analysis of resilience capabilities and characteristics. Converting this potential into a realizable engineering benefit will require insight and experience – which should be derived from investigation and (ideally) empirical studies. The proposed challenge workshops were intended to initiate this process.

We envisaged a modest number of workshops, and succeeded in holding three. We wanted these to be in diverse application domains, one in the mainstream of dependability (grid computing and services was selected), one in security (we chose critical national infrastructures) and one that related to the ambient model of future IST (for this we chose assistive technologies). Each workshop had the same set of objectives:

1. to bring together resilience specialists and domain experts;
2. to select one (or more) candidate problems;
3. to compare current resilience practice with a resilience-explicit approach;
4. to consider how resilience metadata about mechanisms might help;

5. to establish a legacy working group.

It should be fairly clear that meeting all of these objectives in full at a single first meeting was a little unrealistic, and this proved to be the case at all three workshops. But in each case there was solid success on the key objectives, namely numbers 1 and 5. Success on objective 1 was realized in terms of extremely positive and constructive interaction between the two camps (resilience and domain) at each of the three workshops. [This was particularly gratifying for the overall WP1 coordinators and for the individual workshop organizers.] Success on objective 5 was realized in terms of setting up champions for a future thread of activity and designating individuals present at the workshops as champions to take these matters forward.

The remaining objectives (2-4) were discussed at varying levels of detail at the separate workshops and, although no firm conclusions were determined, the discussions will inform the future deliberations of the working groups – which we hope will have a lasting presence, will take forward the objectives, build up understanding, report back via publications and the RKB, and support the long term aims of the ResEx activity.

Annex 1: Second Edition Mechanism Descriptions

The mechanism descriptions in this Annex are not presented in a structured way. Indeed, many classification schemes can be considered. Table 1 provides classification of those mechanisms which have been fully described in the RKB according to a variety of key characteristics:

- The partner responsible for contributing the mechanism description.
- The mechanism objectives.
- Whether the mechanism is an architecture, a process or a tool.
- The development/operational phase during which the mechanism can be applied (design, development more generally or run-time).
- Whether the mechanism provides fault detection, fault forecasting, fault removal, and/or fault tolerance.
- The resilience-building technology (RBT) with which the mechanism is most closely associated (corresponding to ReSIST Working Groups):
  - ***Arch****itecture* – resilience architecting and implementation paradigms.
  - ***Algo****rithms* – resilience algorithms and mechanisms.
  - ***Socio****-Technical* – resilient socio-technical systems.
  - ***Verif****ication* – methods and tools for verifying resilience.
  - ***Eval****uation* – methods and tools for evaluating resilience.
- The resilience-scaling technologies with which the mechanism is most closely associated (corresponding to ReSIST Working Groups):
  - ***Evolv****ability* – Resilience evolvability, maintaining resilience during activities such as upgrading, recovery and fault handling, adaptation and reconfiguration.
  - ***Assess****ability* – Resilience assessability, the ability of a system to assess its correct functioning and quality of service delivered under both nominal and stressful conditions.
  - ***Usab****ility* – Resilience usability, achieving or assessing usability of systems, particularly ubiquitous ones. Helping users interacting with ubiquitous systems to understand the potential effects of their actions as well as preventing them from taking actions with unwanted and difficult to anticipate system-level effects.
  - ***Divers****ity* – Resilience diversity, the use of components that can perform similar functions in the system context but differ in some essential aspect that affects their vulnerability.
- The types of system to/within which this mechanism can be applied.
- The main direct benefits of the mechanism, in terms of improved system resilience (in the case of run-time deployment mechanisms) or assurance of key system properties (in the case of pre-deployment

analysis tools); the latter type of benefit could lead to improved resilience of whatever systems are deployed, through bug-finding and potential freeing up of resources for other development activities, by achieving assurance at reduced cost.

Table 1 shows the coverage achieved in the first and second phases of mechanism description. Beside the 24 mechanisms listed in this table, a further 8 are listed as "stub" entries requiring further effort to provide a full description. These are, typically, mechanisms which have been referred to as related to full mechanism definitions: they are: *Accord; Deductive Reasoning; Dependability Benchmarking; Model-checking; Self-managed cells; Static Analysis; Verification; Decryption Mixes.*

In addition to this list, approximately 120 further mechanisms have been included from the International Standard IEC 61508 (Functional safety of electrical/electronic/programmable electronic safety-related systems), Appendix 7 (Overview of techniques and measures) at an "outline" level of detail. This means that we have described their objective and given a description of the technique, and links to relevant papers describing the technique. IEC 61508 is the international standard for electrical, electronic and programmable electronic safety related systems. Part 7 lists techniques and measures to support the process of ensuring that systems are designed, implemented, operated and maintained to provide the required safety integrity level (SIL).

The techniques listed are drawn from relevant annexes of Appendix 7 of the standard. From Annex B ("Techniques and Measures for Avoidance of Systematic Failure"), the techniques include the following:

General measures and techniques*: Separation of safety-related systems from non-safety-related systems; Diverse hardware; Structured specification; Operation and maintenance instructions; Limited operation possibilities; Protection against operator mistakes; Modification protection; Input acknowledgement*

Formal methods*: Semi-formal methods: Finite state machines/state transition diagrams; Time Petri nets*

Computer-aided specification tools*: Entity models; Checklists; Structured design; Modularisation; Computer-aided design tools*

Verification and Validation techniques*: Simulation; Inspection (reviews and analysis); Walk-through; Functional testing; Black-box testing; Statistical testing; Functional testing under environmental conditions; Interference surge immunity testing; Static analysis; Dynamic analysis; Expanded functional testing; Worst-case testing; Fault insertion testing*

Failure analysis techniques*: Failure modes and effects analysis; Cause consequence diagrams; Event tree analysis; Failure modes, effects and criticality analysis; Fault tree analysis; Worst-case analysis*

From Annex C ("Techniques and Measures for Achieving Software Safety Integrity"), the techniques and measures include:

Requirements and detailed design techniques*: Structured methods; CORE (Controlled Requirements Expression); JSD (Jackson System Development); MASCOT (Modular Approach to Software Construction, Operation and Test); Real-time Yourdon; SADT (Structured Analysis and Design Technique); Data flow diagrams; Structure diagrams*

Formal notations*: CCS (Calculus of Communicating Systems); CSP (Communicating Sequential Processes); HOL (Higher Order Logic); LOTOS; OBJ; Temporal logic; VDM (Vienna Development Method); Z*

Design and coding standards*: Defensive programming; No dynamic variables or dynamic objects; On-line checking during creation of dynamic variables; Limited use of interrupts, pointers and recursion; Structured programming; Information hiding/encapsulation; Modular approach; Use of trusted/verified software modules and components*

Architecture design*: Fault detection and diagnosis; Error detecting and correcting codes; Failure assertion programming; Safety bag; Software diversity (diverse programming); Backward recovery; Forward recovery; Re-try fault recovery mechanisms; Memorising executed cases; Graceful degradation; Artificial intelligence fault correction; Dynamic reconfiguration*

Development tools and programming languages*: Strongly typed programming languages; Language subsets; Certified tools and certified translators; Library of trusted/verified software modules and components; Suitable programming languages*

Verification and modification*: Probabilistic testing; Data recording and analysis; Interface testing; Boundary value analysis; Error guessing; Error seeding; Equivalence classes and input partition testing; Structure-based testing; Control flow analysis; Data flow analysis; Sneak circuit analysis; Symbolic execution; Formal proof; Complexity metrics; Fagan inspections; Walk-throughs/design reviews; Prototyping/animation; Process simulation; Performance requirements; Performance modelling; Avalanche/stress testing; Response timing and memory constraints; Impact analysis; Software configuration management*

Functional safety assessment*: Decision tables (truth tables); Hazard and Operability Study (HAZOP); Common cause failure analysis; Markov models; Reliability block diagrams; Monte-Carlo simulation*

	Partner	Objectives	Category	Phase	Fault Actions	Res-Building Tech.	Res-Scaling Tech.	Systems	Direct Benefits
Consensus Mechanism	Newcastle	Group agreement in the presence of faults.	Architecture	Run-time	Forecasting	Algo	Divers	Distributed computer systems	Agreement among correct components
Dynamic Function Allocation	Newcastle	To support the human operator effectively and resiliently in carrying out their control tasks.	Architecture	Run-time	Tolerance	Socio	Assess	Human-machine control	Effectiveness (incl. efficiency) and resilience
N-Self-Checking Programming/1/1	Newcastle	To tolerate faults through the use of components with the ability to check their own dynamic behaviour.	Architecture	Run-time	Tolerance	Arch	Assess, Evolv, Divers	Systems with self-checking components	Fault tolerance
N-Version Programming/1/1	Newcastle	To utilise design diversity and voting in order to tolerate software faults	Architecture	Run-time	Tolerance	Arch	Assess, Evolv, Divers	Systems with diverse components	Software fault tolerance
Recovery Blocks/1/1	Newcastle	To provide backward recovery to isolated sequential programs	Architecture	Run-time	Tolerance	Arch	Assess, Evolv, Divers	Sequential programs	Error recovery
Robust re-encryption mixes	Newcastle	To provide ballot secrecy by anonymising ballot receipts.	Process, Architecture	Run-time	Tolerance	Algo	Assess	Ballots	Security, anonymity, auditability
Model based stochastic dep. evaluation tool	BUTE	Model-based evaluation of architectural alternatives from the point of view of availability and reliability.	Tool	Design	Forecasting (& Removal)	Eval	Assess	Distributed	Assurance of availability and reliability
Robustness testing	BUTE	To generate and execute test cases to assess the robustness of a computer system.	Tool/ Process	Development	Forecasting (& removal)	Verif	Divers/ Assess	ICT + stressful environments	Assurance of robustness
Supervisory Systems	BUTE	To support real-time monitoring and visualisation of state and non-functional properties of hardware, software and service components in general	Architecture	Run-time	Detection	Arch	Assess	General purpose IT	Real-time monitoring and visualisation

		IT infrastructures.							
Cooperative Backup	LAAS	Long-term availability of data produced by mobile devices.	Architecture	Run-time	Tolerance	Arch/ Algo	Divers	Systems containing mobile devices	Long-term-availability, reduced local storage need
Modelworks	QinetiQ	To provide scalable dependability assessment of systems	Tool	Design	Forecasting (& Removal)	Eval	Assess	Distributed	Assurance of safety, liveness and security
Autonomic Computing Architecture	ReSIST affiliate researcher	To provide an architectural approach to autonomic computing: self-configuration, self-healing, self-protection, self-optimisation.	Architecture	Run-time	Tolerance, Removal	Arch	Evolv	Autonomic	Efficient and resilient sharing of resources
Ad-hoc routing in resilient ambient systems	Darmstadt	A basic network primitive, determines the resilience of distributed systems/applications.	Process	Run-time	Tolerance	Algo	Evolv	Distributed computer systems	Resilience of applications
Byzantine quorum systems	Lisboa	Tools for ensuring the consistency and availability of replicated data despite the benign failure of data repositories.	Architecture	Run-time	Tolerance	Algo	Evolv	Distributed computer systems	Consistency of replicated data
CLawZ	QinetiQ	To verify that Ada code meets a MathWorks MATLAB Simulink (control law) specification	Tool	Design	Removal, Prevention	Verif	Assess	Industrial control-law systems	Verification
CRIA - Critical Interaction Analysis Method	Deep Blue	Validation technique for safety assessment in complex systems	Architecture	Design	Forecasting, Removal	Arch	Assess	Traffic management	Safety assessment
Dynamic Function Allocation (Adaptive Automation)	Newcastle	To support the human operator effectively and resiliently in carrying out their control tasks	Architecture	Design	Removal, Prevention	Arch	Assess	Human interaction	Control automation
Heuristic Evaluation	Deep Blue	Usability engineering method for quick, cheap, and easy	Process	Design	Removal, Prevention	Eval	Assess	Humain interaction	Quick UI evaluation

		evaluation of a user interface design.							
Malporte	QinetiQ	To find exceptional behaviours in source code.	Tool	Design	Removal, Prevention	Verif	Assess	Command and Control, Aerospace	Static analysis
Patterns of cooperative interaction	Newcastle	Patterns of cooperative interaction that have been observed to improve resilience in a variety of systems	Architecture	Design	Tolerance, Removal, Prevention	Arch	Divers	Socio-technical systems	Resilience patterns
Self-healing for Wireless Sensor Networks	Darmstadt	Self-organized autonomic systems such as Wireless Sensor Network (WSN) require self-healing techniques in order to maintain the required availability .	Architecture	Run-time	Tolerance	Arch/Algo	Evolv	Autonomic systems	Resilient evolvable systems
State machine replication	Lisboa	General method for implementing fault-tolerant services in distributed systems by replicating servers and coordinating client interactions with server replicas	Architecture	Run-time	Tolerance	Arch/Algo	Divers	Distributed computer systems	Server replication
Trust and Cooperation Oracle	LAAS	To evaluate locally the level of trust of neighbouring entities and to manage cooperation incentives.	Architecture	Run-time	Prevention	Algo	Assess	Consumer products	Trust evaluation
WS-Mediator	Newcastle	An architectural solution to improving the dependability of Web Services	Architecture	Run-time	Tolerance	Arch/Algo	Evolv/Divers	Web-based systems	Dependable composition

Table 1: Summary of Resilience Mechanisms

Annex 2: Reports from Challenge Workshops

Contents:

Workshop on Challenge Problems for Resilience-Explicit Computing in Grids

Pisa, 14 July, 2008

Report on the application of Res-Ex mechanisms to Grid computing[1]

Editors: Cinzia Bernardeschi, Andrea Domenici
DIIEIT, University of Pisa, v. Diotisalvi 2, I-56122 Pisa, Italy

December 3, 2008

[1]This work is supported under the ReSIST Network of Excellence, which is sponsored by the Information Society Technology (IST) priority in the EU Sixth Framework Programme (FP6) under contract number IST 4 026764 NOE.

Contents

Introduction

At the workshop on *Challenge Problems for Resilience-Explicit Computing in Grids*[1] supported by the ReSIST European Network of Excellence[2], held in Pisa on 14 July, 2008, a working group was established to continue the collaboration among the participants. One of its commitments was to produce a document summarizing known Resilience-explicit mechanisms that could be applied in Grid computing. This report is the result of that commitment, and its contents are a selection of published materials from the ReSIST NoE deliverables.

In particular, Sections 2 (Resilience-Explicit Computing), 3 (The Res-Ex Approach), and 4 (Related Work) are extracted from Deliverable D11 (*Support for Resilience-Explicit Computing* [39]), while Sections 5 (Service Oriented Architectures), 6 (Building Resilient Architectures with Off-the-Shelf Components), and 7 (Dependability Benchmarking) are from Deliverable D12 (*Resilience-Building Technologies: State of Knowledge*) [40].

Topics from the deliverables have been chosen with the intent to provide Grid developers and maintainers both with a catalogue of results that may address relevant issues in their activities, and with an introduction to the approach that the Resilience-Explicit community (or, more generally, the dependability community) may take to the same issues. In fact, Grid practitioners are familiar with many resilience-enhancing techniques, but they may be interested in seeing how these techniques fit in the conceptual framework emerging from research, and contributing to it.

1 Grids and Resilience

Dependability and resilience are crucial requirements for Grid systems. In a Grid system, resilience comes into play as *resilience of the infrastructure* itself and *resilience for applications*, i.e., available Grid services or mechanisms that applications may use to satisfy their resilience requirements.

Some forms of resilience mechanisms are implicit in the nature of the Grid, given the availability of a large number of resources, resource virtualization, the availability of resource brokering services, and the use of data replication. In particular, information and monitoring services are a valuable source of metadata both for long-term and run-time analysis of resilience-related properties. Indeed, several tools are already in place that provide current and historical availability and reliability metrics for various services.

2 Resilience-Explicit Computing

A long-term goal of current research is the provision of methods and tools that support the development and operation of ICT systems that exhibit predictable levels of resilience. Current system development methods rarely treat resilience-related information explicitly, making it difficult to predict system resilience and identify weaknesses. By contrast, in a *resilience-explicit* (Res-Ex) approach,

[1] `http://resist.isti.cnr.it/wiki/doku.php`
[2] `http://www.resist-noe.org/`

information about the resilience-related properties of components and infrastructure are stated explicitly in the form of *metadata* published by components themselves, or by observers. Such metadata can be used at design-time to inform the choice of design patterns and development tools, or at run-time to tune or reconfigure, maintaining resilience. We use the term *resilience-explicit computing* to encompass both the design-time and runtime use of resilience-related metadata.

We use the term *metadata* to refer to information on which human or machine decision-makers act in order to maintain or enhance a system's resilience. We use the "meta-" prefix in order to differentiate this from the data over which a system is performing its functionality. Examples of metadata include: a person's workload in a socio-technical system, descriptions of known failure modes declared in the functional specification of a component, or historical availability statistics. Metadata could also be declared at different levels, such as components, the whole system or even for the user-interface of the system. We may even conceive of a market in trustworthy metadata, whereby metadata on service resilience might be provided by third parties and used to govern run-time selection of components and services.

In order to support machine-assisted decision-making, especially at run-time, it is necessary to develop languages for representing resilience metadata. Examples of representations include simple enumerations (e.g., component integrity levels), numeric representations (e.g., probabilities), or possibly formal logical descriptions (e.g., functional preconditions). Semantics are required for metadata so that analyses can be conducted consistently and with machine support. In particular, common semantics are required to ensure compatibility of metadata from heterogeneous sources (e.g., to ensure that metadata labelled "failure rate" from two different component providers are either interchangeable or convertible). The analyses that we envisage going on at run-time or design-time as part of the decision to adapt or reconfigure may involve calculation over numeric metadata, logical deduction or, most likely, a mixture of the two.

As well as precise descriptions of metadata, Res-Ex computing requires that we have descriptions of the mechanisms that may be deployed or configured in order to meet a resilience target. We therefore use the term *resilience mechanism* to refer to a design pattern, technique or tool intended to improve system resilience. Examples include fault-tolerant architectural patterns (e.g., n-version programming) and development tools (e.g., robustness testing tools). In order to exploit resilience metadata in machine-supported decision-making, we require theories that describe the characteristics of the resilience mechanisms that may be deployed or configured within a system in terms of the relevant metadata.

We focus on the decisions to select a particular resilience mechanism from among alternatives and to instantiate or configure the mechanism for a specific application. Such decisions may be made statically, at design-time, or dynamically within a running system. In either case, in order to reach a resilience target, the decision-maker requires metadata about the characteristics (e.g., failure rates) of components, infrastructure and environment, and descriptions of the resilience mechanisms in terms of their effects on metadata, for example failure rate of a fault tolerant assembly in terms of failure rates of its components, or metadata generated by a robustness testing tool. The resilience mechanism descriptions may be combined with metadata to obtain a prediction of the consequences of a particular selection or configuration.

The goal of the Res-Ex in ReSIST is to encourage the community to give descriptions of mechanisms and metadata that support this decision-making process. In particular, the group wishes to promote the contribution of mechanism descriptions in a form that enables automated analysis. There is currently very little support for gathering such descriptions or for making use of them. The descriptions of resilience mechanisms available to practitioners at present are deeply embedded in the scientific literature and are in many cases hard to extract. We wish to encourage researchers developing new mechanisms to give descriptions that help answer the question "What exactly does this mechanism achieve in terms of resilience?". We hope thereby to encourage research to evaluate existing and new mechanisms, and scholarship in codifying that information and making it available to practitioners. The work of ReSIST Task IT-T2 is to develop a means of recording descriptions of resilience mechanisms that are based on metadata and which integrate with the emerging Resilience Knowledge Base (RKB). This allows mechanism descriptions to be linked to other resilience knowledge through the emerging ontologies and through the research and training/education data embedded in the RKB.

2.1 A Scenario

In order to further clarify the resilience-explicit computing concept, consider a simple scenario. A designer requires a system that tolerates one (sequential) hardware fault and/or one software fault. The designer has limited resources available and wishes to provide a cost effective solution. However, the system must also be as reliable as possible. The designer knows about three fault-tolerant architectures that would provide the necessary level of tolerance. These are, in our terms, *resilience mechanisms*:

- Recovery Blocks (RB/1/1);
- N-Version Programming (NVP/1/1);
- N-Self Checking Programming (NSCP/1/1).

Which of these mechanisms provides suitable cost and reliability levels? *Metadata* can be obtained for the three alternatives, including number of components, structural overheads, and operational time overheads in normal operation and when errors occur. For example, for RB/1/1, metadata includes[3]:

- Total number of variants required (= 2);
- Total number of hardware components required (= 2);
- Ratio of Development and Maintenance Cost of fault Tolerant versus Cost of non-FT software (Min 1.33; Avg 2.17; Max 1.75);
- Probabilities of detected and undetected failures on demand (as functions of probabilities of independent faults in components and decider).

[3]These metadata are derived from the comparative study in [89].

3 The Res-Ex Approach

In order to make progress towards our goal of providing resilience-explicit guidance for the developer community, we aim to provide metadata-based descriptions of a large number and wide range of resilience mechanisms. We have begun this task by asking specialists across the ReSIST network to provide a preliminary and broad-ranging set of "first edition" descriptions.

The full first edition mechanism descriptions have been included in the on-line Resilience Knowledge Base (RKB) and are accessible to readers at http://resist.ecs.soton.ac.uk/resex/. The RKB is a key integrative technology contributed by ReSIST, gathering information on projects, publications, people, resilience mechanisms, educational materials and course descriptions. The addition of Res-Ex mechanism descriptions is part of the ongoing expansion of the value-added content of the RKB. Incorporating the mechanism descriptions requires utilisation of the existing ontological capability of the RKB, but expanding it to cover mechanisms and metadata via a Res-Ex ontology. More information on this aspect is included in Section 4 of [39].

Giving the RKB the capability of holding Res-Ex mechanism descriptions is not sufficient to support expansion of the collection. There must also be an interface whereby mechanism descriptions can be fed into the RKB and maintained once entered. A prototype of such an interface has been developed and used for recording the first edition mechanisms. This interface guides the creator of a mechanism description to answer the right questions in the right context so that they deliver the required information in the appropriate format. The interface and some of its rationale are considered in Section 3 of [39].

4 Related Work on Application Areas

The Res-Ex work on support for resilience-explicit computing has concentrated on providing a means of recording metadata-based descriptions of resilience mechanisms with the intention that the descriptions can be used to assist in the selection and configuration of mechanisms at design-time and run-time. Looking forward to run-time exploitation of metadata, there are several technologies supporting dynamic selection of components and services. In this section, we identify relevant existing work and place Res-Ex computing in this context, particularly noting those technologies in which metadata already plays a role. We see these as technologies that may be able to utilise metadata-oriented descriptions of resilience mechanisms.

For the purposes of this report, only the work more closely related to Grid computing has been selected from Section 5 of [39]. Namely, Section 4.1 (of the present report) relates on work explicitly referring to Grid systems, while Sections 4.2 (Dynamic reconfiguration) and 4.3 (Component-Based software) relate on issues that are expected to be relevant for Grids.

4.1 Grid computing

Computational grids are infrastructures that provide access to shared computing resources for a great number of users involved in large-scale collaborations. In the LHC Computing Grid (LCG) and Enabling Grid for E-sciencE (EGEE)

projects, the Grid Laboratory Uniform Environment (GLUE) schema defines a common conceptual data model for Grid resource discovery and monitoring [116].

There are neither protocols nor standards in the Grid community for dealing with ontologies [66]. However, ontologies can be used in Grids for several purposes: for describing policies and sharing information, services and computing resources in virtual organization, and for describing formal and informal properties of Grid resources and services. The OntoGrid project[4] aims at developing a reference Semantic Open Grid Service Architecture (S-OGSA) for the development of distributed applications that need to use explicit and distributed metadata. The Web Services Data Access and Integration Ontology realisation (WS-DAIOnt) defines the data access services that are needed for dealing with ontologies in Grid environments [66]. Reference [37] describes the approach for metadata management proposed in the context of the S-OGSA. Reference [81] analyses the problem of resource discovery in the Semantic Grid, showing how to solve this by utilizing Atlas, a P2P system for the distributed storage and retrieval of RDF(S) data. Atlas is being used to realise the metadata service of S-OGSA in a fully distributed and scalable way.

Grid computing encompasses resource discovery and resource allocations at run-time; it covers issues related to semantics and performance. It is an ideal application area for resilience-explicit computing. Techniques such as S-OGSA explicitly use metadata at run-time and propose a middleware architecture supporting metadata to be displayed, and shared among the different Grid entities. Focus is given on service provisioning and access control. Research on building reliable and high available Grid services could benefit from Res-Ex work: for example, metadata based descriptions of resilience mechanisms could allow strategies for enhancing reliability of services in S-OGSA.

4.2 Dynamic reconfiguration

Run-time reconfiguration of services has been reported in Wapee [85], a specific middleware that supports dynamic reconfiguration in case of detected faults. A dedicated Fault-Manager detects faults and a Run-time Service Manager triggers reconfiguration once faults have been detected. A Monitoring Service provides real-time monitoring and feedback status of jobs submitted to services. The detection of faults and the choice of the replacement component are based on formal description of faults, of functionality of services and of context information (resource requirements). Three ontologies are defined: fault ontology for types of faults and their causes; service ontology for functionality and resource requirements of services; and recovery strategy ontology for fault resolution.

In the field of autonomic computing, a uniform representation and composition of autonomic elements has been proposed [143], encompassing the use of a service-oriented architecture supporting the interactions of these elements, preliminary design patterns and policies. Accord [95] is a programming framework for autonomic applications, supporting the use of rules to control the behaviour and interaction of autonomic components. Dynamic addition, deletion and replacement of components are supported, as well as changes to interactions. Self-Managed Cells (SMC) [48] consist of (heterogeneous) hardware and

[4] `www.ontogrid.net`

software elements and management services integrated through a common publish/subscribe event bus. Managed components are monitored and decisions and actions are taken on the basis of provided policies. SMC elements have well-defined expected interfaces, limiting the possibility for new elements to join the system, especially if they have not been designed by the same team. The SMC scheme does not specifically address the use of metadata, even though elements are monitored, which nevertheless implies that metadata is collected about their behaviour.

Dynamic reconfiguration of services tends to achieve goals similar to those of resilience-explicit computing at run-time. The use of metadata is not always "explicit" in the different approaches, but the use of metadata is present, since monitoring of a component implies checking some specific behaviour / performance / quality of service, etc. The above techniques focus on run-time architectures and middleware. Design-time issues are not yet primarily considered. The Res-Ex approach clearly encompasses dynamic reconfiguration of services and autonomic computing aspects in general. Res-Ex computing intends to have a larger scope: it covers not only run-time activities related to resilience in the large (not only limited to reconfigurations), but also design-time activities by supporting the choice of resilience techniques at design-time.

4.3 Component-Based Software: selecting components

A design-time automated process for selecting, evaluating and testing third party components is presented in [104], based on both metadata and formal specification of the required component (interface and behaviour) and of its context of use. Metadata capture context information and specific criteria of the desired component. The specification is used first to select the possible components on the basis of their expected functionality, and second to derive tests for evaluating the selected components in the targeted environment. This process leads to a ranking of short-listed components according to criteria such as performance, security, or ease of integration. The Z language is used for the specifications. The specification and the metadata are captured with XML. The above selection process has further been extended with AI techniques for classifying components in order to take into account interdependent criteria [103].

This technique allows selection at design-time of the appropriate component. The main criterion is functionality; the use of the tests allows further identification of the components on the basis of additional non-functional criteria (e.g., performance). This type of work is completely in line with the resilience-explicit approach which allows both design-time and run-time use of metadata in order to ensure adequate choice of the right component. Even though the resilience-explicit computing approach is suited for configuration at design-time, and re-configuration at run-time by selecting components or services, it also goes further by supporting the use of metadata in order to define resilience strategies (different from re-configuration strategies).

5 Service Oriented Architectures

As Grid infrastructures are increasingly incorporating concepts from the "service oriented" paradigm, research in the field of service oriented architectures

provides insight on Grid-specific issues [40].

5.1 Introduction

The term Service Oriented Architecture (SOA) refers to a style of information systems architecture in which distributed applications are constructed by combining loosely coupled and interoperable Services that interoperate according to a Service Contract that is independent of the underlying platform and programming language used to implement the service. Because the service contract hides the implementation of the service, SOA-compliant systems can be independent of any particular development technology or execution platform.

In principle, SOAs can be realised using a variety of different middleware protocols (for example, CORBA or Jini), but in practice, the term SOA is often used to refer to an SOA implemented using the web services protocol stack. A Web Service is essentially just a software component with a well-defined interface that can be accessed programmatically over a network using standard protocols. In this sense, web services are no different from conventional client-server applications built using middleware technologies such as CORBA. However, the distinguishing characteristic of web services is the use of XML-based protocols and languages to describe the interface to the web service and the messages that it understands and generates.

Although the benefits of Service Oriented Architectures are certainly of high interest for applications with important dependability requirements, the lack of mature advances regarding the resilience of such architecture is a major impairment to their use in large critical applications. Thus, the big challenge is how to build reliable/secure distributed applications out of unreliable/insecure Web Services and communication infrastructures.

Regarding resilience and dependability of SOAs, we propose to classify the various works and contributions in six research domains. This separation is not always clear as problems and solutions often overlap several domains. It is however a way to summarize our state of knowledge on SOA and related resilience issues:

- Resilience of the executive support for Web Services,
- Resilience assessments and tools,
- Security and authentication issues,
- Reliability issues,
- Transaction, composition and orchestration,
- Quality of service requirements.

The first dependability issue for SOA applications is to improve the reliability of the runtime support of the Web Services, i.e., the platform on which the web service is executed. Conventional dependability techniques can be used to address this aim, from both an architectural and evaluation viewpoint. For example, relevant techniques include replication at various hardware and software levels (OS, middleware, etc.), failure mode analysis using fault injection at various levels, and failure mode analysis proving inputs to the design of fault

tolerance mechanisms. This is why a large portion of current work in this area tackles the problem in this way. In the same way, conventional techniques can be applied to the communication infrastructure and transport protocols.

The second dependability issue is to tackle the problem at the level of actual SOA concepts, i.e., all protocols and software components used to interact with Web Services. The works we are aware of currently address security and reliability issues, transactional problems, flexibility of dependability solutions with respect to the application needs, and orchestration of large-scale applications based on Web Services. Clearly, there are still many open subjects and difficult issues to address in this second dimension. It is worth noting however that although the backbone of an SOA may introduce multiple fault sources [71], the architecture also allows for design diversity in the form of alternative services and communication channels that may be available over the Internet.

5.2 Recent research work in ReSIST

In this section we summarize the work targeting Service Oriented Architectures done by the partners of the network in the relevant period of ReSIST, included as contributions in this deliverable. These contributions relate to the topics proposed in the previous sections [40].

5.2.1 A Fault Tolerance Support Infrastructure for Web Services based Applications

In this paper, researchers from LAAS propose a support infrastructure that enables both clients and providers to add dependability mechanisms to web services used in large-scale applications [124]. To this aim, it is introduced the notion of so-called Specific Fault Tolerance Connectors. The connectors are software components able to capture web service interactions between clients and providers. They implement filtering and error detection techniques (e.g. runtime assertions) together with recovery mechanisms to improve the robustness of web services. The same web service can be used in several service-oriented applications with different dependability constraints and thus taking advantage of different connectors. To implement recovery strategies, connectors can use the natural redundancy of web services. Similar services can also be found to provide an acceptable service instead of the original one, a sort of degraded service. As this approach provides separation of concerns, such dependability mechanisms can easily be adapted to the needs. A central contribution of this work is a dedicated language (a DSL, Domain Specific Language) that has been developed to build reliable connectors. A platform has been developed (services and tools), used to implement dependable web services based applications and tested with various web services available on the Internet.

5.2.2 Secure and Provable Service Support for Human-Intensive Real-Estate Processes

In this paper, researchers from Newcastle introduce SOAR, a service-oriented architecture for the real-estate industry that embeds trust and security, allows for formal correctness proofs of service interactions, and systematically addresses human interaction capabilities through web-based user access to services [41].

The paper demonstrates the features of SOAR through a Deal-Maker service that helps buyers and sellers semi-automate the various steps in a real-estate transaction. This service is a composed service, with message-based interactions specified in SSDL, the SOAP service description language. The implemented embedded trust and security solution deals with the usual privacy and authorization issues, but also establishes trust in ownership and other claims of participants. We also demonstrate how formal techniques can prove correctness of the service interaction protocol specified in SSDL. From an implementation perspective, a main new contribution is a protocol engine for SSDL. A proof-of-concept demonstration is accessible for try-out.

5.2.3 Service-oriented Assurance – Comprehensive Security by Explicit Assurances

Flexibility to adapt to changing business needs is a core requirement of today's enterprises. This is addressed by decomposing business processes into services that can be provided by scalable service-oriented architectures. Service-oriented architectures enable requesters to dynamically discover and use sub-services. Today, service selection does not consider security. In this paper, researchers from IBM introduce the concept of Service-Oriented Assurance (SOAS), in which services articulate their offered security assurances as well as assess the security of their sub-services [82]. Products and services with well specified and verifiable assurances provide guarantees about their security properties. Consequently, SOAS enables discovery of sub-services with the "right" level of security. Applied to business installations, it enables enterprises to perform a well-founded security/price trade-off for the services used in their business processes.

5.2.4 Modelling of Reliable Messaging in Service-Oriented Architectures

Due to the increasing need of highly dependable services in Service-Oriented Architectures, service-level agreements include more and more frequently such traditional aspects as security, safety, availability, reliability, etc. Whenever a service can no longer be provided with the required QoS, the service requester needs to switch dynamically to a new service having adequate service parameters. In the current paper, researchers from the University of Budapest propose a meta-model to capture such parameters required for reliable messaging in services in a semi-formal way [62]. Furthermore, they incorporate fault-tolerant algorithms into appropriate reconfiguration mechanisms for modelling reliable message delivery by graph transformation rules.

Currently these researchers are working on the formal verification of the correctness of the proposed reconfiguration mechanisms using existing verification tools for graph transformation systems. As the next step in the future, they plan to implement the automatic generation of runtime implementation in existing middleware and to create test cases for reliable messaging.

5.3 Other Research on SOA

In the next sections, we report on some significant examples, research projects and current standardization efforts targeting various aspects of resilience for

Service Oriented Architectures, following the research and development topics mentioned above.

5.3.1 Resilient executive support for Web Services

A framework for improving dependability of web services was proposed in DeW (A Dependable Web Service Framework) [3], which can be understood as a register containing the address of different copies of a web service. This register guarantees the physical-location independence and thus a web service-based application is able to continue running as long as a reference to an available copy is reachable through the register. This framework enables non-stop operation in the presence of crash faults affecting the web services or service migration. Active-UDDI [73] is based on a similar research approach.

FT-SOAP [91] enables a service provider to replicate Web Services using a passive replication strategy. This approach is based on interceptors at the SOAP layer enabling the client application to redirect request messages towards replicas in case of failure of the primary. At the server location, interceptors are used to log messages, to detect faulty replicas and to manage available replicas. The important problem of state transfer is controlled internally, i.e., it is part of the implementation of the service. A similar passive replication approach has been developed in the JULIET project, using .NET [110, 46]. In both cases, the approach relies on a specific software layer that must be installed at the client and at the provider platform, which can raise interoperability problems due to inconsistency with the notion of SOA.

FTWeb [125] is another example of an infrastructure providing active replication strategies for web service-based applications. This project is based on the FT-CORBA standard, WS-Reliability [131] (see Section 5.3.4 on reliability of SOA protocols) and a global ordering service [44] of client requests to ensure replica consistency. This project also introduces a specific software layer that may impair interoperability. Beyond that, it is clear that this architecture strongly depends on a particular middleware support, namely CORBA, and thus Web Services must be implemented as CORBA objects.

Thema [107] is a another middleware-based implementation of fault tolerance mechanisms for web services, more precisely aiming at tolerating Byzantine Faults. The communication service relies on former work [121] providing reliable multicast and consensus policies. WS-FTM (A Fault Tolerance Mechanism for Web Services) [98] is a similar research effort on consensus issues for web services.

5.3.2 Resilience assessments and tools

Most of web services must be considered as black-box components, which means that their development process, their design features, and their robustness in the presence of faults is not known. This kind of situation is not new, and also applies to the use of COTS components in dependable systems for which a lot of work has been carried out, targeting operating systems [86], real-time microkernels [9] and middleware such as CORBA [102, 101]. However, to our knowledge, the assessment of the resilience of SOA is rather limited today and currently relies on conventional benchmarking approaches.

Although failure mode analysis has not been performed yet for web services, we can however mention the assessment by fault injection (SWIFI) carried out in

the OGSA platform based on web services [99, 96, 97]. One of the results of this work is the development of WSFIT (Web Services Fault Injection Tool) [96] for the assessment by fault injection of the SOAP protocol. In practice, the SOAP parser Axis 1.1 is instrumented to inject faults on input/output request messages. Recent results presented in [128] also reveal that version 1.3 of the Axis software should not be used in business-critical applications because of several problems like memory-leakage, performance degradation and hang-up situations of the server that require manual restart intervention. This type of work relates to the discussion of dependability benchmarking in Section 7.

One benefit of such work is to propose a fault model for web service based applications, i.e., the kind of faults affecting a web service in operation [38, 63]. Beyond physical faults affecting the runtime support, communication faults are one of the most important sources of faults for large-scale applications on the Internet. However, because of the complexity of the web service infrastructure at runtime (protocol analysers, dynamic code generation, application servers, virtual machines / operating systems), software faults must be considered as a first class of problems [71]. For example, the SOAP parser can be subject to development faults, which could lead to an incorrect analysis of messages and/or a wrong mapping of data types. In short, the development of large-scale critical application in this context must take into account both physical and software faults affecting the executive and communication infrastructure but also evolution faults, e.g., inconsistency between WSDL documents and stubs produced from older versions.

5.3.3 Security and authentication issues

A lot of work has been carried out regarding security in Services Oriented Architectures built out of web services, and this part of the web services protocol stack is relatively mature. There are several security related web service standards, in particular WS-Security [112], which aims at providing end-to-end security including authentication (using security tokens), confidentiality and integrity of messages. WSSecurity, which is based on XML-Signature and XML-Encryption, is implemented at the SOAP request level by using non-functional elements in the SOAP header and by encrypting the message body. XML-Signature [18] is a specification targeting the creation and the processing of electronic signatures of XML documents or parts of them. XML-Encryption [70] is a specification of object encryption and formatting of the encrypted result in XML.

WS-Security defines a basic framework for transmitting web service messages securely. Above this layer there is less agreement, but IBM and Microsoft have proposed the set of layers and software abstractions shown in Figure 1.

WS-Policy [16], WS-Trust [6] and WS-Privacy build directly on WSSecurity to add higher level security characteristics. Thus, WS-Policy specifies the security contract, describing how to express the security requirements of the provider and the capabilities of the client, WSTrust specifies the model of the mechanisms to establish trusted relations, either directly or indirectly via trusted third party services, whilst WS-Privacy (in progress) is a language for the specification of privacy features (on both machine- and human-readable format) that can be interpreted by user agents.

Higher-level protocols build on top of these protocols to solve interoperability issues between heterogeneous security approaches. Thus, WS-SecureConversation

Figure 1: Abstractions and Software Layers for Web Services Security (from [68]).

[5], WS-Federation [15] and WS-Authorization (in progress) provide a general framework for authorization mechanisms in web services architectures.

5.3.4 Reliability of SOA protocols

There is clearly a need for web service messages to be delivered reliably as well as securely, and there are many examples of reliable messaging systems that can provide such guarantees. However, web services cannot depend on the semantics of the underlying transport protocols, and thus, reliable messaging must be implemented at the SOAP level to ensure end-to-end reliability guarantees and interoperability across a range of different transport protocols. A reliable messaging protocol for web services essentially defines an abstraction of a reliable message processing layer that hides the details of the underlying mechanism used to ensure reliable delivery, which is typically some form of message-oriented middleware. The basic reliability guarantee provided is "best effort" delivery, perhaps within a specified time limit, or else a failure indication.

Unfortunately, for historical and perhaps political reasons, there are two rival standards for reliable messaging for web services, namely WS-Reliability [131] and WS-ReliableMessaging [108]. Although the two specifications offer very similar capabilities with respect to reliable messaging, they seem to have a very different approach to specifying the policy associated with a reliable messaging channel. WS-ReliableMessaging appears to take a rather static view in which the policy is negotiated as part of setting up the channel, and then the agreed parameters apply to the whole sequence of messages. This means that it is not necessary to specify the parameters as part of each message, but it also means that it is not possible for the policy to be adapted dynamically except by shutting down the channel and starting again. In contrast, WS-Reliability seems to allow much more flexibility, but at the cost of some overhead associated with each message. Ultimately, the performance of a reliable messaging system depends on the underlying implementation rather than the protocol, but it would seem that the WS-Reliability proposal allows the implementer more flexibility.

5.3.5 Transaction, composition and orchestration

In addition to BTP (Business Transport Protocol) [30], an older technology, WS-AtomicTransaction [25], WS-BusinessActivity [26] and WSCoordination [27] are three complementary specifications supported by IBM, Microsoft and BEA that should support the implementation of synchronous short duration and ACID

transactions (WS-AtomicTransaction) as well as asynchronous long-running business transactions (WS-BusinessActivity).

Transaction management provides a basic infrastructure for coordinating the execution of web services. However, a service-oriented application can be composed of several web services, which requires some sort of language for "composition", "orchestration" or "choreography" of services. The web services included in the composition are coordinated by a workflow that constitutes the business process of the composition.

Several initiatives specify such business processes using XML documents. Three similar specifications have been proposed for the workflow oriented applications: XLANG [137] by Microsoft, WSFL (Web Services Flow Language) [90] by IBM, and WSCL (Web Services Language Conversation) [17] by Hewlett-Packard. BPEL4WS (Business Process Execution Language for Web Services) [7] which is a fusion of WSFL and XLang, is the most mature version and an implementation is already provided by IBM and Microsoft. This integrates the notion of web services transactional specifications to manage the composition and to perform compensatory transactions (a kind of undo operation) when necessary. In web services technology, a transactional service must propose compensatory transactions, which are the only practical way to maintain consistency in case of blocking or failure situations.

Recovery aspects at service level have been investigated in WSCAL (Web Service Composition Action Language) [135, 136], a language for service composition based on XML developed to address fault tolerance of web services, in particular by using forward error recovery strategies. The principle of forward error recovery is clearly more suitable for web services because it does not impose the handling of state management issue to the web service provider. The WSCAL language creates a coordinator seen as a proxy between the client and all web services in the composite application. The main role of the coordinator is to handle exceptions returned by web services, based on an exception tree for all web services belonging to a composite application. When an exception is raised by a service, the proxy is able to check whether this exception has an impact on other services or not, thanks to the exception tree. When necessary, the proxy triggers the compensating transaction or any other recovery action on target services. The implementation of the language is still in progress.

Finally, BPEL4WS has been used to manage the upgrade of individual component web services into composite web services based applications [63]. The idea consists of switching the composite web service from using the old release of the component web service to using its newer release, only when confidence in the new version is high enough, so that the composite service dependability will not deteriorate as a result of the switch.

5.3.6 Quality of service requirements

Quality of service issues in the broader sense have been addressed in [106]. In this respect, except for some security and transaction aspects, there is no formalism today to express properly the expected QoS of a web service, i.e., the core parameters (e.g., delays, resources requirements, etc.) that could be exploited by developers and application designers. In a recent work, IBM proposed a language WSEL (Web Services EndPoint Language) whose aim is to precisely define certain QoS characteristics of a Web Service Endpoint. Its development

is still in progress. In addition, huge efforts are spent to define Service Level Agreements (SLAs) corresponding to some QoS agreement between client and provider, including expected QoS objective criteria (e.g., delay of service restart in case of failure) and penalties when the provider does not fulfil them.

WS-Agreement [8] is an XML language that describes a service-level agreement for Grid Computing and that is supported by the GRAAP working group (Grid Resource Allocation and Agreement Protocol). Service-level agreements distinguish between negotiation of QoS requirements and monitoring of the provided QoS in operation. Hence, quality of service and other guarantees that depend on actual resource usage cannot simply be advertised as an invariant property of a service and then bound to by a service consumer. Instead, the service consumer must request state-dependent guarantees to the provider, resulting in an agreement on the service and the associated guarantees. Additionally, the guarantees on service quality must be monitored and failure to meet these guarantees must be notified to consumers.

The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime. Currently, WS-Agreement is just a draft, but this is the most promising and advanced work with respect to other research work like WSOL [138], WS-QDL [146, 147, 67], etc.

6 Building Resilient Architectures with Off-the-shelf Components

Grid infrastructures are mostly built out of off-the-shelf components, ranging from operating systems to middleware and applications. This section deals with the relationship between the overall dependability of complex infrastructures and the dependability of its components [40].

6.1 Introduction

The societal impact of the [un]dependability of off-the-shelf (OTS) information and telecommunication components can hardly be overstated. As the "information society" takes shape, people increasingly depend on the proper functioning of a loose, open computing/communication infrastructure whose building blocks (e.g., proprietary or open-source operating systems and web servers) have either established records of poor dependability, or little evidence of good development practices and acceptable dependability. There has now been for several years a trend towards increasing reliance on OTS components: both from developing custom-built components for each new system towards using existing components, and from using components developed for niche markets with high dependability requirements to buying alternatives that offer lower costs thanks to a larger market. These trends have been accompanied by increasing concerns, perhaps mostly about complex OTS software, with design faults leading to more frequent failures and security problems, but also about OTS hardware for the mass market, containing design faults and also becoming increasingly susceptible to transient faults. In embedded computing, increased reliance on OTS components has already created serious challenges for designers. Apart from

headline-making events like the disabling of a U.S. Navy warship by a Windows NT crash [129], industries dealing with hazardous processes face the inevitability of replacing, for instance, safety-qualified hardware sensors, now unavailable, with ubiquitous "smart" sensors containing software, that offer many advantages except comparable evidence of dependability.

Using OTS components is commonly believed to reduce at least the initial cost of deploying complex IT systems. But their actual advantages in terms of Total Cost of Ownership (TCO) are uncertain, and inadequate dependability and security contribute heavily to this cost. For instance, a recent analysis [115] suggests that out of TCO values for OTS systems that vary between 3.6 and 18.5 times the purchase cost of systems, "a third to half of TCO is recovering from or preparing against failures". To such visible costs, one must add the cost of failures that are never detected (e.g., data corruption that causes suboptimal business decisions and other waste), and of catastrophic failures that are too rare to be part of such surveys.

In many of the applications that depend on commercial OTS (COTS) components, the risk from design faults has not yet been addressed adequately. While awareness of these costs and risks slowly grows among users, large vendors of OTS components are slow in improving, due to contrasting market pressures and the sheer difficulty of improving the huge base of installed systems. The supply of many OTS components will be driven by the dependability requirements that satisfy the majority of their mass markets, but are insufficient for specific business and government sectors of the Information Society; this may well remain true in the long run. This view is supported by the recent history of the industries of safety-critical computer applications. As special product lines of high-dependability components became extinct due to competitive pressures these industries have increasingly had to adapt to using general-purpose tools and OTS components, often with insufficient or unknown dependability. The necessary solution is to use fault tolerance against the failures of these components. Side by side with industrial applications of known schemes, the last decade has thus seen a steady growth of the research on the application of fault tolerance specifically to systems built with OTS components.

The fault-tolerant architectures that can be used to preserve system dependability in the presence of (demonstrated or suspected) insufficient dependability of components vary along several dimensions, which it is useful to recall in order to characterise the different lines of currently active research.

First, regarding the form of error detection, confinement and recovery, both architectures using additional components dedicated to monitoring and recovery from failures of OTS components, and ones based on modular (diverse) redundancy appear promising. The former have always been preferred for applications where the cost of failure did not justify the high cost of developing multiple versions of a component. For OTS components, it often takes the form of wrapping, in which a custom-made component filters communications between the OTS component and the rest of the system. But modular redundancy with diversity (i.e., fault-tolerant architectures using diverse, functionally equivalent components) becomes an affordable competitor, since for many functions of OTS components (from chips to complete software packages and hardware-plus-software systems), the market offers alternative, diverse OTS products. Diversification at the level of whole software packages or servers has also been widely advocated for protection of large-scale infrastructures, given the current situation of

widespread vulnerabilities, whereby an attacker, having once discovered a single software bug that opens a security vulnerability, can exploit this knowledge at minimal cost to attack myriads of hosts.

Fault-tolerant architectural solutions also differ in the level (in the decomposition or functional hierarchies in a system) at which fault tolerance is applied. Some research focuses on application-level, end-to-end fault tolerance, to deal with the well-known problems of mass-marketed operating systems and applications. Generic, application-level fault tolerance (e.g. multiple-version software) will to some extent also protect against failures and vulnerabilities in the lower level (e.g., operating systems, processors). However, other research also considers means that are specialised at lower level of granularity (e.g., wrapping is applied at all levels from complete applications to individual units in libraries) or in the software-hardware hierarchy (e.g., specialised to tolerate processor failures).

A related area of interest concerns developing essential building blocks that make fault-tolerant architectures easier to build out of undependable, mass-market OTS components, by guaranteeing properties of low-level mechanisms (e.g., communication or voting). In an attractive scenario, these building blocks would lead to economically viable OTS product lines of standardised, comparatively simple, highly dependable products, possibly shared among various high-dependability applications. Notable example include the Time-Triggered Architecture (TTA) [87], increasingly adopted in the automotive industry, safetycritical railway applications and avionics, and the proposal by Avižienis [11] described further down.

The rest of this section outlines important current lines of research on achieving system resilience with OTS components, and then, in more detail, some more recent contributions [12, 21, 59].

6.2 Lines of research on resilience with OTS components

6.2.1 Identifying vulnerabilities of OTS software, and wrapping against them

Groups at LAAS, led by J.-C. Fabre, J. Arlat and K. Kanoun and at CMU, led by P. Koopman and centred on the BALLISTA project, have worked on two related areas evaluating via fault injection the vulnerabilities of COTS items (POSIX-compliant operating systems, the Chorus microkernel and the CORBA notification service), and specifying wrappers to "cover" such vulnerabilities [1, 9, 45, 88, 102, 114].

The related HEALERS project at AT&T (C. Fetzer's group) aimed to protect library components by automatically generated wrappers (C macros) that intercept calls and check the validity of call parameters and results [56].

M. Swift's group at Univ. of Washington developed Nooks, a subsystem that wraps the Linux kernel and detects improper system calls and predefined exceptions [133].

In these early, influential studies the emphasis was on ad hoc development of wrappers rather than on defining explicit general goals and assessment criteria. For instance, some of this literature does not acknowledge the potential for the wrappers themselves failing, and thus the need to assess at which point increasing the amount of scrutiny performed by wrappers on communications

between components becomes pointless or even counterproductive.

With respect to this last deficiency, the recently completed U.K. project DOTS (a collaboration between the ReSIST members Newcastle and City), produced advances in providing a framework for dealing rigorously with the deficiencies of COTS software. A methodical approach was developed in which protective wrapping was seen as a way of structuring fault tolerance with OTS components [141], to address explicitly the mismatch between what is required from OTS items in a specific system context and what is known about the available candidate OTS items. The approach was demonstrated on a set of case studies[5].

While the early studies mentioned above are mainly concerned with wrapping at the level of the operating systems, wrapping at other levels is also used, e.g. at the level of application software, as discussed in the next sub-section.

Other approaches dealing with vulnerabilities detected at various levels have also been used. The SWIFT technique [120] is a recent extension to the long standing line of research about programmer transparent software solutions for dealing with transient hardware faults. It is a compiler-based software transformation, effective in detecting transient hardware faults, and recently extended [32] to recovery from the detected failures. Software solutions like this offer the users of the OTS hardware, e.g. CPU and memory, a means of reducing the negative effects of transient faults beyond the levels provided by the OTS hardware.

The "Immune System Paradigm" proposed recently by Avižienis [11, 12] is an example of compensating in the system architecture for the lack in modern microprocessors of adequate support for fault tolerance.

6.2.2 Recent work on diversity in replication-based fault-tolerant systems

Diverse redundancy has played a major part in the effort to meet high resilience requirements using existing (including legacy) OTS components, especially when custom-built solutions are either impracticable (due to interconnecting previously existing legacy systems, i.e., systems of systems) or not economically viable due to the limited market needs.

Accepting that fault tolerance with OTS components requires diversity, several groups looked at various aspects of using diverse redundancy:

- B. Liskov's group at MIT developed the BASE approach [29] extending the "state machine" approach to fault tolerance to allow diverse replicas of a component. A "conformance wrapper" guarantees that the states of the diverse replicas remain consistent with an abstract common state, translates between representations of these states, and enables states to be saved and restored. This approach must cope with both faults and permitted behaviour variations between the diverse components; it aims at tolerating Byzantine faults. A prototype demonstrator was developed at MIT for an NFS file system.

- In the above mentioned DOTS Project, the City team undertook an empirical study with complex OTS products, such as several popular database

[5] `http://www.csr.ncl.ac.uk/dots/bibliog.html`

servers, to assess the viability of design diversity with these products and to evaluate, via measurement, the potential benefits in terms of both dependability [58] and performance [130] gains from deploying diversity.

- The "Immune System Paradigm" by Avižienis, mentioned above, provides support for both identical and diverse multichannel computation.

Further insight into design decisions about which fault-tolerant architectures are appropriate came from studies targeted at measuring the actual prevalence of various failure modes in OTS software with the following important contributions:

- At the Univ. of Michigan, Chen and colleagues used several open-source products to study the appropriateness of general purpose recovery schemes [31]. They recorded empirical evidence of serious limitation of these schemes and evidence that a significant proportion of reported faults for the products studied lead to non-crash failures, which reinforces the need for diversity;
- At the University of Illinois at Urbana-Champaign, Ravi Iyer and colleagues recorded empirical evidence of a strong indication of error dependency or error propagation across a network of NT servers [145];
- The City team [58, 59] provided direct empirical evidence of fault diversity with OTS database servers.

6.2.3 Diversity for security

This topic is covered in more detail in the chapter on Intrusion Tolerance of [40]. The text here is limited to aspects related to the use of diversity with OTS components to improve security.

The security research community directly embraces the notion of protective wrapping, and has also developed a considerable interest in fault tolerance via diversity to compensate for the (security) deficiencies of OTS components. Three important research strands have been:

- the DARPA sponsored OASIS project in the USA (which developed prototype architectures, e.g. [57] for web sites made attack-resistant via multiple diverse copies of a web server),
- the European MAFTIA project, coordinated by Newcastle and including other ReSIST members (University of Lisboa, IBM Zurich, LAAS), which delivered a reference architecture, a rationalised terminology framework, and supporting mechanisms,
- the DIT project, in which LAAS-CNRS was involved (associated to SRI International). The project developed a prototype architecture and implementation of an adaptive intrusion tolerant web server using diversity[6].

While significant advances have been made in design and verification of fault tolerance for improved security, progress in the area of quantitative evaluation remains limited [94].

[6] `http://www.csl.sri.com/projects/dit`

6.2.4 Adaptive Fault Tolerance

An important aspect of achieving resilience of computer-based systems, including those developed with OTS components, is managing the evolution of the system configuration or environment during the systems' lifetime. A special case of evolution is the change of the deployed fault tolerance mechanisms. Traditionally, such changes would be implemented statically, i.e., changing the system configuration off-line. Changing the deployed mechanism at runtime, however, has also been explored. The main technical problems with such an approach were outlined and discussed in [83]:

- the difficulty to adapt the mechanisms by means of architectural and/or algorithmic solutions;
- the adaptation efficiency, i.e., the ability to monitor the operational conditions and to react to configuration/environment changes.

Projects such as ROAFTS [84], MEAD [111] and AQuA [123] propose solutions based on middleware, which allows for transparent switching from one mechanism variant to another at the expense of some performance penalty, i.e., by temporarily freezing the service delivered to the user. Chameleon [75] follows the same approach. The adaptation executed by a Fault Tolerance Manager responsible for collecting the user requirements and other pertinent information and then deciding which of the available fault tolerance mechanism will be used. In such systems, adaptation is often driven by thresholds, e.g., as in MEAD [49] and ROAFTS. In AQuA, instead, the adaptation is driven by the QoS criteria defined by the clients of the services (e.g., in terms of crash and/or value failures). For instance, the requested availability can be obtained by increasing the number of replicas. Another interesting study [61] advocates an "Adaptive Fault Resistant System"; it can be seen as a survey of some possible approaches to address the problem of adaptation (e.g., adaptive recovery blocks) and open issues (e.g., reflection as in FRIENDS [54, 134]).

In ReSIST, work carried out at LAAS relies on the notion of multi-level reflective architectures [134] to perform the on-line adaptation. The objective is to limit the impact of the modifications on the service delivered to the user, i.e., without freezing the system but by introducing degraded modes of operation at the non-functional level. This work tackles both the architectural and algorithmic issues to simplify the design and the implementation of on-line adaptive mechanisms.

6.2.5 Infrastructure management

An important aspect of achieving resilience of systems built with OTS components is the so called infrastructure management, a collective label for a multitude of administration tasks and processes and the tools enabling them. Adopting standards of infrastructure management promotes interoperability and best practices and thus reduces the likelihood of misconfiguration of complex systems of OTS components and as a result — poor system performance.

The most widespread infrastructure management standards today are Simple Network Management Protocol (SNMP), Web Based Enterprise management (WBEM) and Java Management Extensions (JMX), which are described briefly below:

- SNMP [69] is widely spread both in terms of usage and industry support. SNMP is best suited for the management of networks and network elements. It is also adapted for computational platform management, but the lack of real object-oriented information representation and security issues makes it inferior compared with the other two solutions.

- WBEM [47] was developed by the Distributed Management Task Force (DMTF), a subsidiary of the Object Management Group (OMG). WBEM is a truly object oriented and model based management standard with a UML-compatible information model — the Common Information Model (CIM). However, although the standards comprising WBEM are available for years now and being supported by the major software vendors its industrial penetration has been limited except for Microsoft Windows NT operating systems. The Windows Management Instrumentation (WMI) is partly a CIM compliant implementation.

- JMX [132] is an extension of the Java runtime platform with management and manageability capabilities. While .NET utilises WMI, for Java a standard defines the Java Management Extensions framework (JMX). The standard is heavily used by the major Java-based application servers, e.g., WebSphere, Apache Tomcat etc.

6.3 Recent Research Work in ReSIST

6.3.1 An Immune System Paradigm for the Assurance of Dependability of Collaborative Self-Organizing Systems

This contribution by the Vytautas Magnus University team addresses an important problem of enhancing the limited support for fault tolerance built in the modern microprocessors and other hardware OTS components.

Most currently available microprocessors and other hardware OTS components have very limited fault tolerance (i.e., error detection and recovery) functions. They also do not possess built-in support for redundant multi-channel (duplex, triplex, etc.) computation either with identical, or with diverse hardware and software in the channels. Recently Avižienis has proposed [11] a network of four types of Application-Specific Integrated Circuits (ASIC) components called the fault tolerance infrastructure (FTI) that can be used to embed OTS hardware residing in one package (board, blade, etc.) and provide it with a means to receive error signals from and to issue recovery commands to the OTS components. Furthermore the FTI provides support for both identical and diverse multichannel computation. The FTI employs no software and is fault-tolerant itself. The design principle of the FTI is called the immune system paradigm because the FTI can be considered to be analogous to the immune system of the human body "hardware".

The most recent work presented here [12] applies the FTI concept to the protection of collaborative self-organizing systems composed of relatively simple autonomous agents that act without central control. Because of the changing structure of such systems the application of consensus algorithms for fault tolerance becomes impractical, while the FTI provides fault tolerance individually for every agent, and consensus algorithms are not needed to protect the entire self-organizing system.

6.3.2 Towards an Engineering Approach to Component Adaptation

This report by the Newcastle team addresses an important problem in building dependable systems by integrating ready-made components: how to deal with various mismatches between components [21]. These mismatches are unavoidable because the components are not built directly for the context in which they are used and because developers of various components typically make a number of assumptions about the context which are not consistent or even conflicting. The well-known solution to these problems is introducing adaptors mediating component interactions. Component adaptation needs to be taken into account when developing trustworthy systems, where the properties of component assemblies have to be reliably obtained from the properties of its constituent components. The adaptor development is still an ad hoc activity, so a more systematic approach to component adaptation is required when building trustworthy systems. In this paper, the authors show how various design and architectural patterns can be used to achieve component adaptation and thus serve as the basis for such an approach. The paper proposes an adaptation model, which is built upon a classification of component mismatches and identifies a number of specific patterns to be used for eliminating them. It concludes by outlining an engineering approach to component adaptation that relies on the use of patterns and provides an additional support for the development of trustworthy component-based systems.

6.3.3 Fault tolerance via diversity for off-the-shelf products: a study with SQL database servers

This report by the City team is a recent update of previous work on fault diversity [58], which presented empirical evidence that design diversity could offer significant dependability gains for OTS relational database management systems (SQL servers) [59]. The report presents results from a second study with more recent faults reported for two open source SQL servers, PostgreSQL and Firebird, the two most advanced and widely used open-source SQL-servers, which have been reproduced on SQL servers from other vendors. The results observed are consistent with the results reported earlier in [58]:

- very few faults cause simultaneous failure in more than one server.
- the failure detection rate in this study is 100% with only 2 diverse server being used.
- The proportion of crash failures is $< 50\%$, consistent with the first study and contrary to the common belief that crash failures are the main concern. Such a high proportion of non-crash failures sheds a serious doubt as to how effective protection is provided by the known database replication solution, developed to tolerate crash failures only.

Ways of diagnosing the failed server were also studied, in the cases this is not evident (e.g., non-crash failures). A variant of data diversity [4] was found to be a promising way of building an efficient rule-based diagnosing system, which only requires a handful of rules to diagnose successfully the failed servers for all faults included in the study [60].

7 Dependability Benchmarking

Operating a Grid requires continuous monitoring and testing, both for performance and functionality. This section covers research in the novel field of dependability benchmarking [40].

7.1 Introduction

Performance benchmarks are widely used to evaluate system performance while dependability benchmarks are hardly emerging. Several popular performance benchmarks exist for many specific domains that constitute invaluable tools to objectively assess and compare computer systems or components. This is a well-established area that is led by organizations such as the TPC[7] (Transaction Processing Performance Council) and SPEC[8] (Standard Performance Evaluation Corporation), and supported by major companies in the computer industry [64]. Benchmarking the dependability of a system consists in evaluating dependability or performance-related measures, experimentally or based on experimentation and modelling, in a well-structured and standardized way. The development and conduct of benchmarks are still at an early stage in spite of major efforts and progress made in recent years both by the IFIP WG 10.4 SIG on Dependability Benchmarking[9] and within the IST project DBench[10] to which several ReSIST partners are contributing or have contributed.

To be meaningful, a benchmark should satisfy a set of properties (representativeness, repeatability, portability, cost-effectiveness, etc.).

7.2 Dependability benchmarking approaches

Dependability benchmarking involves a new dimension to the Workload and (performance) Measure dimensions that typically characterize a performance benchmark: the Faultload. The Faultload defines the types of faults that are combined with the Workload to exercise the target system being benchmarked to obtain relevant Measures. In addition to performance measurements, more specifically performance degradation in presence of faults, the relevant measures attached to dependability benchmarking encompass a wide variety of facets including the assessment of system resilience, the characterization of failure modes and error signalling, the estimation of restart times, etc.

Dependability benchmarks are usually based on modelling and experimentation, with tight interactions between them. In particular, analysis and classification of failure data can support the identification of realistic faultloads for the experimentation. On the other hand, measurements collected during experiments can be processed via analytical models to derive dependability measures. Some possible benchmarking scenarios can be sketched in reference to Figure 2, where layers identify the three steps analysis, modelling and experimentation and arrows A to E represent the corresponding activities and their interrelations [78]. For example a scenario consisting of modelling supported by experimentation would include the three steps and links A, B and E; in that case,

[7] www.tpc.org
[8] www.spec.org
[9] www.dependability.org/wg10.4/SIGDeB
[10] www.laas.fr/DBench

the expected outputs are comprehensive measures obtained from processing the models.

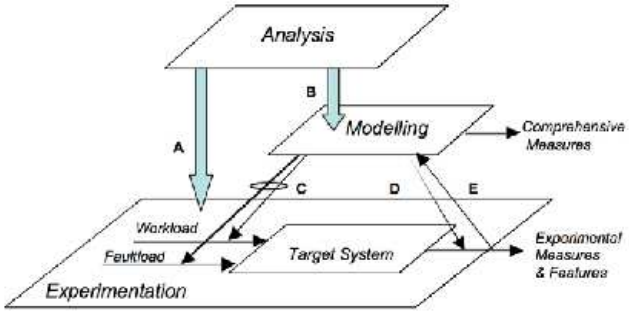


Figure 2: Dependability benchmarking steps and activities.

Favouring the combined effects of the workload and the faultload in order to reduce the cases of nonsignificant experiments (e.g., see [140]) involves complex composability analyses. Nevertheless, it is worth noting that the careful study of the interactions between the workload and the faultload is also central to the work addressing verification via robustness testing. Accordingly, cross-fertilization from these studies is expected to develop more efficient dependability benchmarks.

To be useful and widely accepted by the community as a means to obtain a fair assessment and comparison of systems in a given class, dependability benchmarks must fulfil a set of specific properties. These properties must be taken into consideration from the earliest phase of the benchmark specification. The main relevant properties are briefly identified hereafter:

Representativeness: A benchmark must reflect the typical use of the target system. This requirement is a key issue and encompasses all the dependability benchmarking dimensions:

- It is essential that the workload profile used in the benchmark includes a realistic set of the activities found in real systems for the application area being considered. In practice, this aspect is often handled by considering existing performance benchmarks to implement the Workload.

- Faultload representativeness relies on how well injected faults correspond to real faults, i.e., faults affecting the target system in real use. Field data is the best way to validate whether a set of faults is to be considered in the faultload. The problem is that field data on the underlying causes of computer failures is often not widely available. The emergence of Open Source Software (OSS) paradigms and the wide usage of such OSS components provide new opportunities to address this issue. It is pertinent to note that previous works are available in the dependability literature that has been used to achieve confidence on the representativeness of a given faultload (e.g., see [34, 35, 51, 10, 72, 127]).

- Measure representativeness has to do with the adequacy of i) the features that are reported [144, 19], ii) the measurements that are carried out and

optionally of their post processing to derive comprehensive measures (as illustrated by Figure 1), with the service expected from target system and its usage. In particular, the life cycle and the category of end-user to which the benchmark is focused, have an impact on the relevance of the measures. In practice, multidimension measures are preferred to an averaged figure. Indeed, in the first case, it is still possible to tailor the interpretation to suit end-user primary requirements (e.g., error reporting or continuity of service) [52, 1].

Repeatability: The results obtained for a specific application of the benchmark should be repeatable (within acceptable statistical margins) for the same system set up, even if executed by different end-users. This is clearly a prerequisite for the credibility of the benchmark. The resulting impact of this property differs whether the considered benchmark is provided under the form of a specification (TPC-like approach) or under the form of an executable prototype (SPEC-like approach).

Portability: This property is intrinsic to the concept of benchmark that by definition must be executed on various target systems. Portability depends on the way some key benchmark dimensions (such as the faultload and workload) are specified. For example, the more general the faultload is defined the more easily portable will be the benchmark. In practice, the underlying technology (fault injection mechanisms, measurement tools, etc.) must be portable to different platforms. The availability of standard interfaces greatly facilitates the fulfilment of this property.

Non-intrusiveness: The implementation of the benchmark in the experimentation step (particularly to what concerns the faultload and the measurements) should induce no or really minimal change to the benchmark target (either at the structure level or at the behaviour level). One implication in order to satisfy non-intrusiveness results in a practical rule that is to avoid fault injection within the benchmark target[11]. However, faults can be injected in the parts surrounding the benchmark target. The respect of this property has a restrictive impact on the granularity of the faultload and of the time measurements that can be carried out and reported.

Cost effectiveness: The effort in time and cost necessary to run the benchmark should be reasonably low, otherwise many users could be prevented from using it. It is worth noting that the benchmarking time consists of three parts: i) set-up and preparations, ii) running the actual benchmark (i.e., execution time) and iii) data analysis. Ideally one would acclaim benchmarks that are able to assess the target system thoroughly and accurately, while featuring short benchmarking times. Clearly, in practice, these properties call for a trade-off.

Scalability: The various properties of a benchmark must hold for differing sizes of instantiations of a class of target system. In connection with the viability property, this means in particular that time and cost of running the benchmark must increase at a radically lower pace than the complexity and size of the target system. Usually, scaling is achieved by defining a set of rules in the benchmark specification. Scaling rules mostly affect the workload, but also the size of the faultload as the latter may affect the time needed to execute the benchmark.

[11] It is worth noting that the impact of this restriction can be related to the robustness attribute (i.e., dependability with respect to external faults [13]) and accordingly to the case of robustness testing.

In the subsequent sections, we summarize the current status of the results obtained on dependability benchmarking with respect to two categories of faults: accidental faults and intrusions, both by the ReSIST partners and elsewhere. It is worth pointing out that while quite a substantial body of research has addressed accidental faults, benchmarking with respect to intrusions is almost inexistent.

7.3 Accidental faults

Elaborating on the pioneering work started in the first half of the 1990s (e.g., see [43, 126]) several efforts have been conducted in the past decade to advance the conceptual issues and the development of prototype dependability benchmarks based on specific faultloads encompassing physical faults, software bugs or, to a less extent, operator mistakes.

Physical faults have received first a special attention both from academia and industry. In order to facilitate the conduct of the controlled experiments, these works have significantly built on several variants of the SWIFI technique to implement the faultload, (e.g., see [20, 76, 28, 24, 148]). Indeed, SWIFI exhibits relevant features, such as low intrusiveness and good portability and it can emulate a wide variety of hardware faults and, to some extents, software faults as well [100]. These proposals have covered a wide spectrum of application domains (critical embedded control systems, highly available distributed systems) as well as various components and layers in a computer architecture (processors, middleware, etc.) [126, 139, 102, 144, 23, 149, 122, 127, 36].

Accidental human interaction faults are reportedly a dominant factor in many critical systems and information infrastructures. In particular, the statistics covering the period 1959-2005[12]. show that they steadily contribute as the principal cause. The figures for the period 1996-2005, indicate that they contribute as the major factor to 63% of the accidents in commercial aviation: crew 55%, airport/air traffic control 5%, maintenance 3%. Operator faults constitute also a significant ratio (almost 50%) of the causes of diagnosed failures in large-scale data centres as reported in [113] that has analysed three major Internet sites. This explains why efforts have tried to address this state of affair by including the human-factor aspect in dependability benchmarking proposals (e.g., see [22, 142]). These attempts are still preliminary and much work is still needed to cope with the complexity of the task.

Software faults constitute an increasingly large share of reported service disruptions in many application domains [145, 55, 33]. This trend is easily explained by the penetration of software-intensive computing systems in every day usage (for example a typical cellphone now contains 2 million lines of software code) and also in many critical applications (65 million lines are quoted for the Airbus A380). Increased time to market pressure and the complexity attached to the handling of large software programs impose a heavy burden on the software life cycle (e.g., see [14, 117]). This trend also favours the reuse of previously developed software. Accordingly, several works have addressed the dependability characterization of Off-The-Shelf (OTS) proprietary or open source software components including both the software executive layer (e.g., operating systems

[12] *Statistical Summary of Commercial Jet Airplane Accidents – Worldwide Operations 1959–2005*, Boeing Commercial Aircraft Group, Seattle, USA, 2006 (http://www.boeing.com/news/techissues)

(OS) [86]) and the application layer (e.g., database management systems [142]. automotive application [122] or web services [50]). Due to their central role in a computer architecture, special emphasis has been put on the development of benchmarks to support the assessability and the testing of the robustness of software executives. These especially addressed general-purpose OS kernels, but some relevant works have also considered specific microkernels (e.g., see [9]) or the middleware layer (e.g., see [114]).

As already pointed out, several attempts have been made to propose and implement benchmark prototypes supporting the assessability of OS kernels. Figure 3 depicts the software architecture of a computer system and provides a basis on which this thread of work can be illustrated. As shown on the figure, the kernel features three main interfaces with its environment. The first one (bottom) is basically concerned with hardware interactions, while the other two (top and right) are software related.

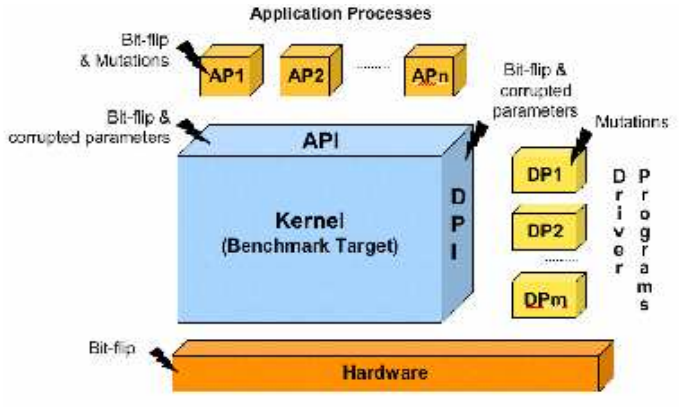


Figure 3: Interactions between an OS kernel and its environment and possible fault injection locations.

The "lightning" symbols in Figure 3 identify possible locations where faults can be applied. These locations and related faults are briefly described as follows:

1) The main interactions concerning the bottom interface are made by raising hardware exceptions. Several studies (e.g., see [9, 65]) have been reported in which faults were injected by means of bit-flips into the memory of the system under benchmark or via special debugging interfaces [122].

2) The top interface corresponds to the Application Programming Interface (API). The main interactions are made via kernel calls. A significant number of studies were reported that target the API to assess the robustness of OS (e.g., under the form of code mutations [51], by means of bit-flips [72], or by altering the system calls [86, 74, 79]).

3) The third type of interactions are made via the interface between the drivers and the kernel. The proposal in [52] has concentrated on drivers code mutation. The work reported in [1] proposes a complementary alternative that explicitly targets the exchanges made between the device drivers and the kernel via their interface the DPI (Driver Programming Interface) [53].

The work reported in [77] and [80] fits item 2 above, while item 3 is exemplified by the work described in [2]. These contributions illustrate the benchmarking efforts targeting general purpose OS that were carried out at LAAS within the DBench project.

7.4 Intrusions

There are no intrusion detection benchmarks per se. However, a number of papers related to testing intrusion-detection systems have been published in the literature. In most cases testing methodology is proposed by the developers of a given intrusion-detection system method or tool, with the purpose of showing their own development at its best. This type of work is considered biased. We put emphasis on three testing approaches: the Lincoln Lab experiments, university and research testing environments and commercial environments, that are more detailed hereafter.

7.4.1 The Lincoln Lab experiments

One of the best known testing experiments in the intrusion-detection community is the Lincoln Lab experiment [93, 92]. The initial purpose of this experiment is to test the various intrusion-detection technologies developed under DARPA funding, and to compare them in order to choose the most appropriate one. This is achieved by simulating a network of workstations to create normal traffic and by inserting a set of attacks carefully chosen to cover various situations, summed up as remote attacks, local attacks and denial-of-service attacks. This experiment has received scientific criticism from the community, most notably in Reference [105]. Our main criticisms are:

- Focus on background traffic: As the test includes behaviour-based intrusion-detection systems, realistic background data must be generated to ensure that these systems will be properly trained. However, while background traffic generation is required for performance testing, the Lincoln Lab testbed does not provide a way of calibrating the characteristics of this background traffic and verifying their compliance with specifications.
- Focus on a broad set of attacks: The Lincoln Lab tests aimed at exercising the largest possible set of attacks for the largest possible set of intrusion-detection systems. However, in some contexts one would like to focus on network traffic close to firewalls. Such kind of analysis cannot be done based on the Lincoln Lab tests that are too wide for such a use.
- Lack of reference point or baseline: The Lincoln Lab tests compare prototypes in a closed circle. There is no notion of a minimal set of requirements that a tested tool has to meet, only relative data comparing them with each other. The lack of a baseline that all tools would have to fulfil was felt as particularly lacking in the Lincoln Lab experiment.

7.4.2 University and research testing environments

The most representative work concerning university tests has been carried at the U. of California at Davis [118, 119] with related and de facto similar work at IBM Zurich [42].

The objective of the UC Davis test is to simulate the activity of normal users and the activity of attackers, using script written in the *expect* language. *Expect* simulates the presence of a user by taking over the input and output streams of a tty-based application, matching expected responses from the application with the displayed output, and feeding appropriate input as if the user typed it on its keyboard.

A similar approach was followed at IBM Zurich. In addition to user-recorded scripts the IBM approach introduced software testing scripts from the DejaGnu platform (also in *expect*) to ensure that all aspects of an application were exercised, regardless of the fact that they were obscure features of an application or not.

7.4.3 Commercial testing environments

Tests classified as commercial include tests published by commercial test laboratories mandated by a particular company, and tests carried out by independent publications.

Several test labs have published test results related to intrusion-detection systems. In particular, Mier Communications has released at least two test reports, a comparative of BlackICE Sentry, RealSecure and NetProwler on one hand, and a test of Intrusion.com SecurenetPro Gigabit appliance. Appropriate queries on mailing list archives show that these test results have been the subject of many controversies. The general feeling is that it is appropriate for the testing laboratory to provide results that highlight the advantages of the product sponsored by the company financing the tests. Even if direct test manipulation is not suspected, at least test configurations for the competing products is likely not to be handled by experts, thus resulting in unfair comparison.

Our feeling is that even this cannot be determined, as the description of the tests is very sparse and does not provide information that would allow an impartial judgement of the tests. Normal traffic generation, for example, is done using commercial products and the test report is considered complete with only the mention of the name of the traffic generator, without any information on the kind of traffic it generates or its configuration.

Journalists have also been involved in the testing of intrusion-detection systems. The most interesting article that we have found is by Mueller and Shipley [109]. It is the only comparative test of commercial intrusion-detection systems that we have found, where a number of intrusion-detection products are compared on equal footing and on live traffic, hence with a higher probability of either missing attacks or triggering false positives. The main drawback of this kind of test is reproducibility: when observing alerts it is quite difficult to reproduce the conditions in which these alerts are generated. Also, we believe that the tuning phase carried out during testing initiation is problematic; the testers describe a process in which alerts that they believe are false are turned off.

The most serious work related to Intrusion-Detection Systems / Intrusion-Prevention Systems testing is regularly published since 2004 by the NSS group (http://www.nss.co.uk). They regularly select a set of Intrusion-Detection Systems / Intrusion-Prevention Systems appliances and provide comparative testing of performance and detection capabilities. Their findings are a widely used source of information for Intrusion- Detection Systems / Intrusion-Prevention

Systems buyers, but the rationale for selecting the tested appliances seems unclear. A complete description of their methodology is available on their web site.

Conclusions

This report collects large excerpts from two deliverables produced by the ReSIST NoE, selected and edited in order to provide a compact introduction to research work on dependability that is expected to be applicable in the field of Grid computing. The report is by no means exhaustive, and much relevant work has been surely left out, but the editors are confident that this document can be a useful tool for people concerned with the dependability of Grid infrastructures.

Readers are encouraged to read the original documents in order to get a wider and more complete view of the field of resilience-explicit computing.

References

[1] A. Albinet et al. Characterization of the Impact of Faulty Drivers on the Robustness of the Linux Kernel. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2004)*, pages 867–876, 2004.

[2] A. Albinet et al. *Robustness of the Device DriverKernel Interface: Application to the Linux Kernel.* IEEE CS Press, 2008.

[3] E. Alwagait and S. Ghandeharizadeh. Dew: A dependable web services framework. In *14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications*, 2004.

[4] P. E. Ammann and J. C. Knight. Data diversity: An approach to software fault tolerance. *IEEE Transactions on Computers*, C-37(4):418–425, 1988.

[5] S. Anderson et al. Web services secure conversation language (ws-secureconversation), version 1.1. `http://specs.xmlsoap.org/ws/2004/04/sc/ws-secureconversation.pdf`, 2004.

[6] S. Anderson et al. Web Services Trust Language (WS-Trust). `http://www6.software.ibm.com/software/developer/library/ws-trust.pdf`, 2005.

[7] T. Andrews et al. Business process execution language for web services, version 1.1, 2003.

[8] A. Andrieux et al. Web services agreement specification (ws-agreement), version 1.1, 2004. Draft 18.

[9] J. Arlat et al. Dependability of cots microkernel based systems. *IEEE Transactions on Computer Systems*, 51(2):138–163, 2002.

[10] J. Arlat et al. Comparison of physical and softwareimplemented fault injection techniques. *IEEE Transactions on Computers*, 52(8):1115–1133, 2003.

[11] A. Avižienis. A fault tolerance infrastructure for dependable computing with high-performance cots components. In *2000 International Conference on Dependable Systems and Networks*, pages 492–500, 2000.

[12] A. Avižienis. An immune system paradigm for the assurance of dependability of collaborative self-organizing systems. In *IFIP 19th World Computer Congress, 1st IFIP International Conference on Biologically Inspired Computing*, pages 1–6, 2006.

[13] A. Avižienis et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE Tansactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[14] M. W. Bailey and J. W. Davidson. Automatic detection and diagnosis of faults in generated code for procedure calls. *IEEE Transactions on Software Engineering*, 29(11):1031–1042, 2003.

[15] S. Bajaj et al. Web services federation language (wsfederation), version 1.0. `http:///www6.software.ibm.com/software/developer/library/ws-fed.pdf`, 2003.

[16] S. Bajaj et al. Web services policy framework (wspolicy), version 1.2. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/wspolicy-2006-03-01.pdf`, 2006.

[17] A. Banerji et al. Web services conversation language (wscl) 1.0. `http://www.w3.org/TR/wscl10/`, 2002. W3C Recommendation.

[18] M. Bartel et al. Xml-signature syntax and processing. `http://www.w3.org/TR/xmldsig-core/`, 2002. W3C Recommendation.

[19] W. Bartlett and L. Spainhower. Commercial fault tolerance: A tale of two systems. *IEEE Tansactions on Dependable and Secure Computing*, 1(1):87–96, 2004.

[20] J. H. Barton et al. Fault injection experiments using FIAT. *IEEE Transactions on Computers*, 39(4):575–582, 1990.

[21] S. Becker et al. Towards an Engineering Approach to Component Adaptation. In *Architecting Systems with Trustworthy Components*, volume 3938 of *LNCS*, pages 193–215. Springer, 2006.

[22] A. B. Brown et al. Including the human factor in dependability benchmarks. In *Dependable Systems and Networks (DSN2002) – Workshop on Dependability Benchmarking*, pages F9–F14, 2002. Supplemental volume.

[23] K. Buchacker et al. Reproducible dependability benchmarking experiments based on unambiguous benchmark setup descriptions. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2003)*, pages 469–478, 2003.

[24] K. Buchacker and V. Sieh. UMLinux — a versatile SWIFI tool. In *4th European Dependable Computing Conference (EDCC4)*, pages 159–171. Springer, 2002.

[25] L. F. Cabrera et al. Web services atomic transaction (wsatomictransaction), version 1.0. `http://www-128.ibm.com/developerworks/library/specification/ws-tx/#atom`, 2005.

[26] L. F. Cabrera et al. Web services business activity framework (ws-businessactivity), version 1.0. `ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf`, 2005.

[27] L. F. Cabrera et al. Web services coordination (wscoordination), version 1.0. `http://www-128.ibm.com/developerworks/library/specification/ws-tx/#atom`, 2005.

[28] J. Carreira et al. Xception: A technique for the experimental evaluation of dependability in modern computers. *IEEE Transactions on Software Engineering*, 24(2):125–136, 1998.

[29] M. Castro et al. BASE: Using Abstraction to Improve Fault Tolerance. *ACM Transactions on Computer Systems*, 21(3):236–269, 2003.

[30] A. Ceponkus et al. Business transaction protocol, version 1.0. `http://www.oasisopen.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf`, 2002. OASIS Committee Specification.

[31] S. Chandra and P. M. Chen. Whither Generic Recovery from Application Faults? A Fault Study using Open-Source Software. In *International Conference on Dependable Systems and Networks*, 2000.

[32] J. Chang et al. Atomic Instruction-Level Software-Only Recovery. In *International Conference on Dependable Systems and Networks*, pages 83–92, 2006.

[33] R. N. Charette. Why software fails. *IEEE Spectrum*, September 2005.

[34] J. Christmansson and R. Chillarege. Generation of an Error Set that Emulates Software Faults Based on Field Data. In *26th Int. Symp. on Fault-Tolerant Computing (FTCS26)*, pages 304–313. IEEE CS Press, 1996.

[35] J. Christmansson et al. An Experimental Comparison of Fault and Error Injection. In *9th Int. Symp. on Software Reliability Engineering (ISSRE'98)*, pages 369–378. IEEE CS Press, 1998.

[36] C. Constantinescu. Dependability benchmarking using environmental test tools. In *Annual Reliability and Maintainability Symp. (RAMS'05)*, pages 567–571. IEEE Computer Society Press, 2005.

[37] O. Corcho et al. Metadata management in s-ogsa. In *International Workshop on Collective Intelligence for Semantic and Knowledge Grid (CISKGrid 2007)*, 2007.

[38] D. Cotroneo et al. Improving dependability of service oriented architectures for pervasive computing. In *Eighth IEEE International Workshop on Object-Oriented Real- Time Dependable Systems*, 2003.

[39] Support for Resilience-Explicit Computing — first edition. Technical report, ReSIST, September 2007. Deliverable D11.

[40] Resilient-Building Technologies: State of Knowledge. Technical report, ReSIST, September 2006. Deliverable D12.

[41] E. Ribeiro de Mello et al. Secure and provable service support for human-intensive real-estate processes. In *IEEE International Conference on Services Computing*, pages 495–502, 2006.

[42] H. Debar et al. Reference Audit Information Generation for Intrusion Detection Systems. In *IFIPSEC'98*, pages 405–417, 1998.

[43] L. S. DeBrunner and F. G. Gray. A methodology for comparing fault tolerant computers. In *26th Asilomar Conf. on Signals, Systems and Computers*, pages 999–1003. IEEE Computer Society Press, 1992.

[44] X. Defago et al. Totally Ordered Broadcast and Multicast Algorithms: A Comprehensive Survey. Technical Report DSC/2000/036, Dept. of Communication Systems, EPFL, 2000.

[45] J. DeVale and P. Koopman. Robust Software — No More Excuses. In *International Conference on Dependable Systems and Networks*, pages 145–154, 2004.

[46] V. Dialani et al. Transparent fault tolerance for web services base architectures. In *8th International Europar Conference*, pages 889–898, 2002.

[47] Web-Based Enterprise Management (WBEM). `http://www.dmtf.org/standards/wbem/`, 2004.

[48] N. Dulay et al. Self-managed cells for ubiquitous systems. In *Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 1–6. Springer, 2005.

[49] T.A. Dumitras et al. Architecting and Implementing Versatile Dependability. In *Architecting Dependable Systems III*, volume 3549 of *LNCS*, pages 212–231. Springer, 2003.

[50] J. Durães et al. Dependability benchmarking for Webservers. In *23rd International Conference on Computer Safety, Reliability and Security (SAFECOMP2004)*, 2004.

[51] J. Durães and H. Madeira. Emulation of Software Faults by Selective Mutations at Machinecode Level. In *13th Int. Symp. on Software Reliability Engineering (ISSREE2002)*, pages 329–340. IEEE CS Press, 2002.

[52] J. Durães and H. Madeira. Mutidimensional characterization of the impact of faulty drivers on the operating systems behavior. *IEICE Tansactions on Information and Systems*, E86-D(12):2563–2570, 2003.

[53] D. Edwards and L. Matassa. An approach to injecting faults into hardened software. In *Ottawa Linux Symposium*, pages 146–175, 2002.

[54] J.-C. Fabre and T. Perennou. A metaobject architecture for fault-tolerant distributed systems: The FRIENDS approach.

[55] Norman E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797–814, 2000.

[56] C. Fetzer and Z. Xiao. HEALERS: A Toolkit for Enhancing the Robustness and Security of Existing Applications. In *International Conference on Dependable Systems and Networks*, pages 317–322, 2003.

[57] T. Fraser et al. Hardening COTS Software with Generic Software Wrappers. In *Foundation of Intrusion Tolerant Systems — Organically Assured and Survivable Information Systems (OASIS)*, pages 399–413, 2003.

[58] I. Gashi et al. Fault Diversity Among Off-the-shelf SQL Database Servers. In *International Conference on Dependable Systems and Networks*, pages 389–398, 2004.

[59] I. Gashi et al. Fault tolerance via diversity for off-the-shelf products: A study with sql database servers. Manuscript, 2006.

[60] I. Gashi and P. Popov. Rephrasing Rules for Off-The-Shelf SQL Database Servers. In *6th European Dependable Computing Conference*, 2006.

[61] J. Goldberg et al. Adaptive Fault-Resistant Systems. Technical Report SRI-CSL-95-02, SRI International Computer Science Laboratory, 1995.

[62] L. Gonczy and D. Varro. Modeling of reliable messaging in service oriented architectures. In *International Workshop on Web Services Modeling and Testing*, pages 35–49, 2006.

[63] A. Gorbenko et al. Development of Dependable Web Services out of Undependable Web Components. Technical Report CS-TR-863, School of Computing Science, University of Newcastle upon Tyne, 2004.

[64] J. Gray, editor. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, 1993.

[65] W. Gu et al. Characterization of Linux kernel behavior under errors. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2003)*, pages 459–468, 2003.

[66] M. Gutiérrez et al. Access in grids with ws-daiont and the rdf(s) realization. In *Semantic Grid Workshop, GGF16, Athens*, 2006.

[67] O. Hasan and B. Char. A deployment-ready solution for adding quality-of-service features to web services. In *The 2004 International Research Conference on Innovations in Information Technology*, 2004.

[68] Security in a web services world: A proposed architecture and roadmap. http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/, 2002. Joint White Paper from IBM Corporation and Microsoft Corporation.

[69] Simple Network Management Protocol. Technical Report RFC 3411, IETF, 1991.

[70] T. Imamura et al. Xml encryption syntax and processing. http://www.w3.org/TR/xmlenc-core/, 2002. W3C Recommendation.

[71] D. B. Ingham et al. Constructing dependable web services. *IEEE Internet Computing*, 4(1):25–33, 2000.

[72] T. Jarboui et al. Impact of internal and external software faults on the Linux kernel. *IEICE Transactions on Information and Systems*, E86D(12):2571–2578, 2003.

[73] M. Jeckle and B. Zengler. Active uddi - an extension to uddi for dynamic and fault-tolerant service invocation. In *Web, Web-Services, and Database Systems 2002*, pages 91–99, 2003.

[74] A. Kalakech et al. Benchmarking the dependability of Windows NT, 2000 and XP. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2004)*, pages 681–686, 2004.

[75] Z. T. Kalbarczyk et al. Chameleon: A software infrastructure for adaptive fault tolerance. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):560–579, 1999.

[76] G. A. Kanawati et al. FERRARI: A flexible softwarebased fault and error injection system. *IEEE Transactions on Computers*, 44(2):248–260, 1995.

[77] K. Kanoun and Y. Crouzet. Dependability benchmarks for operating systems. *International Journal of Performability Engineering*, 2(3):275–287, 2006.

[78] K. Kanoun et al. A framework for dependability benchmarking. In *Dependable Systems and Networks (DSN2002) – Workshop on Dependability Benchmarking*, pages F7–F8, 2002. Supplemental volume.

[79] K. Kanoun et al. Benchmarking the Dependability of Windows and Linux using PostMark Workloads. In *16th Int. Symp. on Software Reliability Engineering (ISSREE2005)*, pages 11–20. IEEE CS Press, 2005.

[80] K. Kanoun et al. *Windows and Linux Robustness Benchmarks With Respect to Application Erroneous Behavior*. IEEE CS Press, 2008.

[81] Z. Kaoudi et al. Semantic grid resource discovery in atlas. In *Knowledge and Data Management in GRIDs*, pages 185–199. Springer US, 2007.

[82] G. Karjoth et al. Service-oriented Assurance-Comprehensive Security by Explicit Assurances. In *1st Workshop on Quality of Protection*, LNCS. Springer, 2006.

[83] K. Kim and T. Lawrence. Adaptive Fault Tolerance: Issues and Approaches. In *Second IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 38–46, 1990.

[84] K. Kim and C. Subburaman. ROAFTS: A Middleware Architecture for Real-Time Object-Oriented Adaptive Fault Tolerance Support. In *Third IEEE High Assurance Systems Engineering Symposium*, pages 50–57, 1998.

[85] Y. Kim et al. Ontology based software reconfiguration in a ubiquitous computing environment. In *6th IEEE International Conference on Computer and Information Technology (CIT'06)*, 2006.

[86] P. Koopman and J. DeVale. Comparing the robustness of posix operating systems. In *29th Annual International Symposium on Fault-Tolerant Computing*, 1999.

[87] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[88] N. P. Kropp et al. Automated robustness testing of off-the-shelf software components. In *28th International Symposium on Fault-Tolerant Computing*, pages 230–239, 1998.

[89] J.-C. Laprie et al. Analysis of hardware and software fault-tolerant architectures. *IEEE Computer*, 23(7):39–51, 1990.

[90] F. Leymann. Web services flow language (wsfl 1.0). `http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`, 2001.

[91] D. Liang et al. Ft-soap: A fault-tolerant web service. In *Tenth Asia-Pacific Software Engineering Conference (APSEC'03)*, 2003.

[92] R. Lippmann et al. Analysis and Results of the 1999 DARPA OffLine Intrusion Detection Evaluation. In *RAID 2000*, pages 162–182, 2000.

[93] R. Lippmann et al. Evaluating intrusion detection systems: The 1998 DARPA offline intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition*, 2000.

[94] B. Littlewood and L. Strigini. Redundancy and diversity in security. In *9th European Symposium on Research in Computer Security*, pages 423–438, 2004.

[95] H. Liu et al. A component-based programming model for autonomic applications. In *International Conference on Autonomic Computing (ICAC'04)*, pages 10–17. IEEE Computer Society, 2004.

[96] N. Looker et al. Ws-fit: A tool for dependability analysis of web services. In *The 1st Workshop on Quality Assurance and Testing of Web-Based Applications*, 2004.

[97] N. Looker et al. Pedagogic data as a basis for web service fault models. In *IEEE International Workshop on Service-Oriented System Engineering*, 2005.

[98] N. Looker and M. Munro. WS-FTM: A Fault Tolerance Mechanism for Web Services. Technical Report 02/05, University of Durham, 2005.

[99] N. Looker and J. Xu. Assessing the dependability of ogsa middleware by fault injection. In *22nd International Symposium on Reliable Distributed Systems*, pages 293–302, 2003.

[100] H. Madeira et al. On the emulation of software faults by software fault injection. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2000)*, pages 417–426, 2000.

[101] E. Marsden. Caractérisation de la Sûreté de Fonctionnement de Systemes à base d'intergiciel. Technical report, LAAS-CNRS, 2004.

[102] E. Marsden et al. Dependability of corba systems: Service characterization by fault injection. In *21st IEEE Symposium on Reliable Distributed Systems*, pages 276–285, 2002.

[103] V. Maxville et al. Intelligent component selection. In *28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, 2003.

[104] V. Maxville et al. Selecting components: A process for context-driven evaluation. In *Tenth Asia-Pacific Software Engineering Conference (APSEC'03)*, 2003.

[105] J. McHugh. The 1998 Lincoln Laboratory IDS Evaluation, A Critique. In *RAID 2000*, pages 145–161, 2000.

[106] D. A. Menascé. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[107] M. Merideth et al. Thema: Byzantine-fault-tolerant middleware for web-service application. In *24th IEEE Symposium on Reliable Distributed Systems*, pages 131–140, 2005.

[108] Web services reliable messaging protocol (ws-reliablemessaging). `http://www-106.ibm.com/developerworks/webservices/library/ws-rm/`, 2004.

[109] Patrick Mueller and Greg Shipley. To catch a thief. `http://www.networkcomputing.com/1217/1217f1.html`, 2001.

[110] R. Murty. Juliet: A distributed fault tolerant load balancer for .net web services. In *IEEE International Conference on Web Services*, pages 778–781, 2004.

[111] P. Narasimhan et al. MEAD: Support for Real-Time Fault-Tolerant CORBA. *Concurrency and Computation: Practice and Experience*, 17(12):1527–1545, 2005.

[112] Web services security: Soap message security 1.0 (ws-security 2004). http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, 2004. OASIS Standard 200401.

[113] D. Oppenheimer and D. A. Patterson. Architecture and dependability of largescale internet services. *IEEE Internet Computing*, 6(5):41–49, 2002.

[114] J. Pan et al. Robustness Testing and Hardening of CORBA ORB Implementations. In *International Conference on Dependable Systems and Networks*, 2001.

[115] D. A. Patterson et al. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. Technical Report UCB//CSD-02-1175, UC Berkeley Computer Science, 2002.

[116] A. Delgado Peris et al. LCG-2 User Guide. Technical Report CERN-LCG-GDEIS-454439, LHC Computing Grid, September 2004.

[117] M. Pighin and A. Marzona. An empirical analysis of fault persistence through software releases. In *ACM/IEEE Int. Symp. on Empirical Software Engineering (ISESE 2003)*, pages 206–212. IEEE CS Press, 2003.

[118] N. J. Puketza et al. A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering*, 22(10):719–729, 1996.

[119] N. J. Puketza et al. A software platform for testing intrusion detection systems. *IEEE Software*, 14(5):43–51, 1997.

[120] G. A. Reis et al. SWIFT: Software Implemented Fault Tolerance. In *3rd International Symposium on Code Generation and Optimization*, 2005.

[121] R. Rodrigues et al. Base: Using abstraction to improve fault tolerance. In *18th ACM Symposium on Operating System Principles*, pages 15–28, 2001.

[122] J.C. Ruiz et al. On benchmarking the dependability of automotive engine control applications. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2004)*, pages 857–866, 2004.

[123] C. Sabnis et al. Proteus: A Flexible Infrastructure to Implement Adaptive Fault Tolerance in AQuA. In *7th IFIP International Working Conference on Dependable Computing for Critical Applications*, pages 137–156, 1999.

[124] N. Salatge and J.-C. Fabre. A Fault Tolerance Support Infrastructure for Web Services based Applications. Technical Report 06365, LAAS, May 2006. Research Report.

[125] G. T. Santos et al. FTWeb: A Fault Tolerant Infrastructure for Web Services. In *Ninth IEEE International EDOC Enterprise Computing Conference*, 2005.

[126] D. P. Siewiorek et al. Development of a benchmark to measure system robustness. In *23rd Int. Symp. on FaultTolerant Computing (FTCS23)*, pages 88–97. IEEE CS Press, 1993.

[127] D. P. Siewiorek et al. Reflection on industry trends and experimental research in dependability. *IEEE Tansactions on Dependable and Secure Computing*, 1(2):109–127, 2004.

[128] L. Silva et al. Software aging and rejuvenation in a soap-based server. In *IEEE Network Computing and Applications*, 2006.

[129] G. Slabodkin. Software glitches leave navy smart ship dead in the water. `http://www.gcn.com/print/17_17/33727-1.html`, 1998. GCN Government Computer News.

[130] V. Stankovic and P. Popov. Improving DBMS Performance through Diverse Redundancy. In *25th International Symposium on Reliable Distributed Systems*, 2006.

[131] Web services reliable messaging tc ws-reliability. `http://www.oasisopen.org/committees/download.php/5155/WS-Reliability-2004-01-26.pdf`, 2003.

[132] Java Management Extensions (JMX). `http://java.sun.com/j2se/1.5.0/docs/guide/jmx/`, 2004.

[133] M. M. Swift et al. Recovering Device Drivers. In *6th ACM/USENIX Symposium on Operating Systems Design and Implementation*, 2004.

[134] F. Taiani et al. A Multi-level Meta-object Protocol for Fault-Tolerance in Complex Architectures. In *International Conference on Dependable Systems and Networks*, 2005.

[135] F. Tartanoglu et al. Coordinated forward error recovery for composite web services. In *22th Symposium on Reliable Distributed Systems*, 2003.

[136] F. Tartanoglu et al. Dependability in the Web Services Architecture. In *Architecting Dependable Systems*, volume 2677 of *LNCS*. Springer, 2003.

[137] S. Thatte. Xlang (web services for business process design). `http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm`, 2001.

[138] V. Tosic et al. Management applications of the web service offerings language (wsol). *Information Systems*, 30(7):564–586, 2005.

[139] T. K. Tsai et al. An approach towards benchmarking of faulttolerant commercial systems. In *26rd Int. Symp. on FaultTolerant Computing (FTCS26)*, pages 314–323. IEEE CS Press, 1996.

[140] T. K. Tsai et al. Stress-based and path-based fault injection. *IEEE Transactions on Computers*, 48(11):1183–1201, 1999.

[141] M. van der Meulen et al. Protective Wrapping of Off-the-Shelf Components. In *4th International Conference on COTS-Based Software Systems*, pages 168–177, 2005.

[142] M. Vieira and H. Madeira. Benchmarking the dependability of different OLTP systems. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2003)*, pages 305–310, 2003.

[143] S. White et al. An architectural approach to autonomic computing. In *International Conference on Autonomic Computing (ICAC'04)*, pages 2–9. IEEE Computer Society, 2004.

[144] D. Wilson and H. Madeira. Progress on defining standardized classes for comparing the dependability of computer systems. In *Dependable Systems and Networks (DSN2002) – Workshop on Dependability Benchmarking*, pages F1–F5, 2002. Supplemental volume.

[145] J. Xu et al. Networked Windows NT System Field Failure Data Analysis. In *6th Pacific Rim International Symposium on Dependable Computing*, pages 178–185. IEEE Computer Society Press, 1999.

[146] S. Yoon et al. Ws-qdl containing static, dynamic, and statistical factors of web services quality. In *IEEE International Conference on Web Services*, pages 808–809, 2004.

[147] M. Yu-jie et al. Interactive web service choicemaking based on extended qos model. In *Fifth International Conference on Computer and Information Technology*, pages 1130–1134, 2005.

[148] P. Yuste et al. Nonintrusive softwareimplemented fault injection in embedded systems. In *1st Latin American Symposium on Dependable Computing (LADC2003)*, pages 23–38. Springer, 2003.

[149] J. Zhu et al. Robustness benchmarking for hardware maintenance events. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2003)*, pages 115–122, 2003.

Report of Res-Ex Workshop on Resilience-Explicit Computing for Critical National Infrastructures

20-21 November 2008
QinetiQ Malvern

Nick Moffat (QinetiQ), Steve Riddle (Newcastle University)

Contents

1. Purpose

This document summarises the second of the Resilience-Explicit "Challenge Problems" workshops, which took place on Thursday 20 and Friday 21 November, 2008 at QinetiQ, Malvern, UK.

The workshop brought together people from the communities of Resilience and Security, with the goal of defining one or more "challenge" problems. A challenge problem is difficult issue that is relevant to some application area and that could be used as a benchmark for techniques related to resilience-explicit computing.

2. Programme

Thurs 20th November

- 12:00 - 13:00 *Arrival, buffet lunch*
- 13:00 - 13:15 Talk 1: Vision for the workshop - challenge problems and possible strategy (Steve Riddle, Newcastle + Colin O'Halloran, QinetiQ)
- 13:15 - 13:30 Talk 2: **ReSIST**: Resilience-Explicit Computing (Tom Anderson, Newcastle)
- 13:30 - 14:00 Talk 3: **ReSIST**: Res-Ex Challenge Problems (Steve Riddle, Newcastle)
- 14:00 - 14:30 Talk 4: **ReSIST**: RKB Explorer (Hugh Glaser, Southampton)
- 14:30 - 15:15 Talk 2: **ESII/SERSCIS/CFMS/INDEED**: Overview (Neil Briscombe, QinetiQ)

- 15:15 - 15:45 *Coffee break*
- 15:45 - 16:30 Talk 7: **SERSCIS:** CNI Sectors and Vectors (Alan Hood, QinetiQ)
- 16:30 - 17:15 Talk 8: **ESII/SERSCIS/CFMS:** Semantic modeling tool based Resilience Mechanisms for CNI security (Neil Briscombe, QinetiQ)

Fri 21st November

- 08:15 - 08:30 *Coffee*
- 08:30 - 09:15 Talk 7: **SERSCIS:** System Exploitation 101 (Alan Hood, QinetiQ)
- 09:15 - 10:00 Talk 8: **SEMIOTIKS:** Actionable Intelligence Extracted from Natural Language Sources (Chris Booth and Neil Briscombe, QinetiQ)
- 10:00 - 10:30 *Coffee break*
- 10:30 - 11:30 Talk 9: **ReSIST:** Integrating Disparate Knowledge Bases using Semantic Web Technologies (Hugh Glaser, Southampton)
- 11:30 - 12:00 Discussion and Next steps

3. Outline of Talks

Talk 1: Vision for the workshop

Speakers: Steve Riddle, Newcastle and Colin O'Halloran, QinetiQ

This workshop aims to define one or more "challenge" problems in the area of resilience-explicit computing (see next talk). A challenge problem is a difficult issue, relevant to some application area, that could be used as a benchmark for techniques related to resilience-explicit computing.

It was proposed that we consider the following vision: that the knowledge base under development within ReSIST could be a useful generic resilience resource when used with domain-specific knowledge bases.

Talk 2: ReSIST and Resilience-Explicit Computing

Speaker: Tom Anderson, Newcastle

This talk gave an overview of the ReSIST project's activities focused on resilience-explicit computing (ReSIST work package 1). Resilience is a form of dependability, focused on "maintaining delivery of dependable systems over time". WP1 aims to lay foundations for facilities to assist engineers in selecting and deploying resilience "mechanisms" at design time; dynamically; during system operation and evolution.

The word "mechanism" is deliberately vague, intended to be very inclusive: basically any way of helping improve resilience. For example it could be a tool, a manual procedure, a technique, a methodology, etc.

The Resilience-Explicit Computing (Res-Ex) activity is complemented in WP1 by

- development of a Resilience Thesaurus and Ontology (Res-On)

- development of a Resilience Knowledge Base (RKB) which uses the Res-On

Talk 3: Res-Ex Challenge Problems

Speaker: Steve Riddle, Newcastle

This talk described the basic objective for the Res-Ex challenge workshops, including this one, as being to identify something like a 'grand challenge problem' suitable for benchmarking current technology, benchmarking the resilience-explicit approach. It was hoped candidate problems would be identified during the workshop, and that workshop participants might form an ongoing working group to settle on a candidate and use it for benchmarking.

Desired characteristics of such challenge problems were that they would: be manageable, be practitioner-oriented, and exhibit some degree of resilience. To serve as a test-bed for the exploration, evaluation and adoption of Res-Ex approaches at design and/or run time, they would ideally have potential for metadata exploitation, provide guidance and support for design rationale, include runtime reasoning policies and reconfiguration services, and offer opportunities for verification.

The following approach was proposed:

1. Select a problem
2. Draft the problem statement
3. Record a summary of resilience/dependability approaches that have been applied
4. Record approaches in the form of Res-Ex mechanism descriptors
5. Establish a working group
6. Deploy Res-Ex approach

Talk 4: RKB Explorer

Speaker: Hugh Glaser, Southampton

This talk gave an overview of the ReSIST Resilience Knowledge Base (RKB) Explorer - contents, interface and underlying technology. Some highlights follow.

The RKB Explorer (http://RKBExplorer.com/) uses semantic web technologies to provide flexible user interaction. For example, results of a search for entries related to a particular person can be queried further to reveal the relationships.

Saarbrucken University have used classical machine learning to assess the relatedness of many published resilience-related papers – statistical measures of the 'distance between papers' (not based on citation).

The RKB includes information about resilience-related courses developed and described by ReSIST partners. One feature is the ability to show locations of courses on a map.

The RKB supports queries about resilience mechanisms that have been characterised (in terms of metadata) and recorded via the project's wiki interface. An example shown was to find mechanisms that are appropriate for Hardware and Aerospace. One can then inspect the metadata to learn more about the mechanisms, for example to compare the operational overheads associated with detection of faults. SPARQL queries can be submitted to the RKB, for example returning metadata associated with mechanisms in the ontology.

Talk 5: Project Overviews

Speaker: Neil Briscombe, QinetiQ

This talk gave an overview of some QinetiQ research projects related to emergency planning and critical infrastructure protection.

Before describing the projects, it was noted that the UK MOD Defence Technical Strategy states an ambition to use semantic web technology to better exploit the World-Wide Web.

Security & Trust Semantic Modelling: This recurring theme is developing a method for capturing security/trust requirements associated with a system. In general the full set of such requirements cannot be met, so the method aims to support an operator's search for an acceptable 'best case'. On a case by case basis, an operator asks about the cost of particular mechanisms failing; the cost is revealed in terms of implications for requirements not being met. The approach relies on ontologies to enable discovery of dependencies between mechanisms. There is associated tooling.

ESII (Enabling Security Information Infrastructure):

This project generated a set of security/trust models. It demonstrated a number of benefits of the approach: increased understanding of security and business motivation, an ability to address system safety while enabling their flexible management at run time, and automatic generation of analysis documentation. Furthermore, with extra tooling the same models can be used as a basis for decision support and provision of audit trails.

SERSCIS (Semantic Enhanced and Resilient and Secure Critical InfraStructure): Web services are increasingly coming together to form critical infrastructures. SERSCIS is looking at the whole life cycle of such infrastructures, including syntactic, semantic and pragmatic aspects. The idea is to use Memex Links to capture all relevant design decisions (the ‘whats’) in addition to the security decisions captured in the tool of the project above. The whole service lifecycle is covered and, crucially, to also include the reasons for design decisions (the ‘whys’). The ‘whys’ are modelled along each dependency path so that if something goes wrong there is a better chance of explaining why the relevant design decisions were made.

CFMS (Centre for Fluid Mechanics Simulation): This project is using the same tooling that underpins the previous two projects. The main difference is that it is applied to a specific use case and integrates with specific middleware and user application products. The use case covers the management of IPR issues in multi-organisation design and fluid flow analysis/simulation of aircraft, cars and boats using SOAs. The system allows execution of collaborative analysis and design service orchestrations with policy enforcement as well as situational awareness of security in workflows as they execute with decision support features to assist in circumstances such as new requirements or system faults.

INDEED (INterdisciplinary DEsign and Evaluation of Dependability): The aim of this project is to develop knowledge, methods and tools that firstly contribute to the understanding of the dependability of socio-technical systems and secondly support developers of dependable systems. Some of the projects above are related to INDEED.

On behalf of St. Andrews University the speaker presented developments from areas such as HAZOP analysis for communication links and modular dependability cases.

Akogrimo: This is a completed EU co-funded project that had a Disaster Handling and Crisis Management scenario. The scenario involved a dirty bomb terror attack in Bristol with 'domino' status. Grid Services for Knowledge Management, Decision Support, Risk Management, and a Common Operational Picture for disaster response were discussed. This included contribution from the Bristol City Council emergency management team.

Talk 6: CNI Sectors and Vectors

Speaker: Alan Hood (QinetiQ)

This talk gave some opinions regarding a range of possible electronic attack vectors against critical national infrastructure. It provided a very high level view based on the speaker's knowledge of CNI, combined with long experience monitoring and analysing a wide variety of electronic threats and countermeasures.

The CNI sectors addressed were chosen to match the classification by the Centre for Protection of National Infrastructure (CPNI – see www.cpni.gov.uk): Energy, Food, Water, Transport, Communications, Government, Health, Emergency Services, Financial.

We mention some major points made during the talk, first for particular sectors and then generally.

Energy: This consists primarily of the gas and electricity utilities. Gas is more dangerous due to risk of explosion, but the gas infrastructure is very much designed to mitigate this, ironically making it more vulnerable to denial of supply (as it fails safe to avoid the risk of explosions). Both utilities have vulnerabilities due to a reliance on SCADA (Supervisory Control and Data Acquisition), which can be securely deployed but is increasingly exposed in an ever more interconnected world. Monitoring of energy supply is typically ICT-based, but this is not always the case with delivery – opening the potential for an attacker to blind the monitoring through ICT mechanisms while performing a more traditional/manual attack on supply.

Food: The most likely risk of attack against food supplies would seem to be manipulation of *perception* - e.g. declare that certain food could be contaminated with mercury.

Water: Similar to energy with respect to monitoring. Again, the real threat is perception. There is a certain degree of reliance on ICT in water supply, similar to energy supply.

Transport: *Air* is well protected, though there seems to be potential for physical attack against controlling mechanisms, such as air traffic control, and the potential for attack on the ICT at individual airports potentially leading to the higher control systems. *Sea traffic* seems to be more locally focussed, with defences considered on a per port basis. One hopes that if individual port authorities are penetrated this would not cause widespread damage. *Road traffic* would, it seems, be less effectively attacked through ICT than sea and air – note that we are considering malicious intent here (however common or frustrating unintended jams may be). Regarding modelling, the large number of factors suggest it would be infeasible to consider them all in a single 'monolithic model'. *Rail* appears more vulnerable than roads and sea, as the 'bad guys' will know where a train is

going to be (and roughly when) and the control and monitoring of rail networks is often ICT-based.

Communications: Wired and mobile telephone networks are highly computerised and could be attacked using ICT. Mobile telephones rely on cell sites with microwave antennae - these can be "mimicked" by an attacker with a more powerful broadcast mechanism. Assuming total control over phones, it is not clear there would be a 'major detrimental impact' comparable to the potential impact in other sectors; on the other hand, it could have a large impact in terms of revealing information.

Government: Government is very diverse, especially given the European context. Governments will be networked to greater or lesser degrees. The biggest impact of an attack may be loss of social security (e.g. unemployment) benefits.

Emergency Services: These are relatively well served in terms of tactical operation - for example, their mobile comms are generally regarded as robust (e.g. TETRA).

Health: Here the need for access to data outweighs data protection. As a result, there is a distinct potential for attack (not least due to the number of users on health-related networks), but the degree to which this could lead to major detrimental impact is uncertain.

Financial: Financial systems are highly networked, but spread across numerous distinct commercial organisations. Financial institutions are already attacked routinely, but it is difficult to obtain details of such attacks for analysis, particularly in the current financial climate where institutions are keen to avoid any perception of error.

General Vectors: A number of attack techniques are likely to be effective in all sectors: DOS, Local Access, Network Penetrations, WLAN Infrastructure access, Web client attacks, Web application attacks, E-mail, Portable data services.

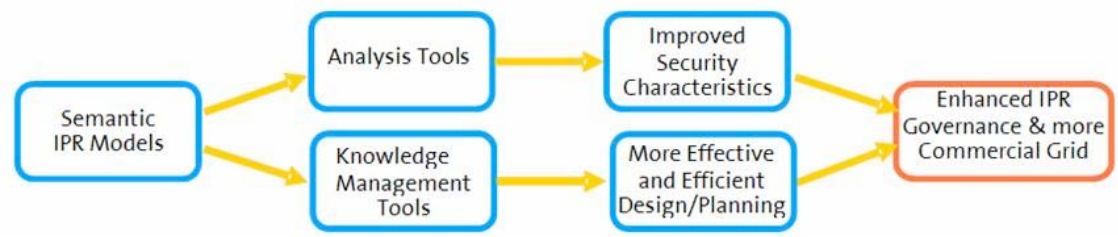
The conclusion of this high level assessment of the potential for ICT attacks on critical infrastructure sectors highlighted two points: 1) adverse impacts on perception can be the most serious effect of an attack, 2) modelling is likely to be difficult in some cases because of the large number of factors relevant to the effectiveness of typical national systems.

Talk 7: Resilience Mechanisms for CNI security

Speaker: Neil Briscombe, QinetiQ

The focus of this talk was to describe the security modelling described in Talk 5. The presentation was intended to promote discussion with ReSIST researchers as to how the tooling could be best represented as resilience mechanisms in the RKB.

There are two main reoccurring themes in the use of the tooling: knowledge management and automated security analysis. The following diagram is an example from the CFMS project:



Traditional diagramming and design tools are overly complicated, provide too much unrelated functionality and cannot guarantee compliance of security diagrams to their methodology.

In contrast, tooling implemented specifically for the purposes of security modelling with a particular methodology can be very much simpler, having fewer but more relevant and domain-specific functions that allow quick and simple construction and at the same time guarantee methodology compliance 'by construction'.

Features of particular note highlighted and discussed were:

- Dynamically 'Collapsible' grouping, which reduces complexity for both diagrams and the automatic analysis features;
- Automatic Compromise Path Analysis, which shows all of the exploit paths available to an attacker;
- Automatic Service Path Analysis, which shows all of the routes available for legitimate service subscribes (or system users);
- A hierarchical 'systems of systems' editor, which has specific views of model subsets to provide different perspectives on the same infrastructure, user groups or their connections (to model system ownership in interconnected CNI for example);
- 'Memex Links', which provide a machine-interpretable modelling framework designed to capture design decisions and enable flexible integration with other tools.

Some other related projects were presented that examined the benefit of tooling that has been implemented specifically for particular methodologies and models that are encoded with formalised semantics, as follows.

Information Assurance Awareness (IAA):

In this project an enhanced version of the above tool records sharing of data so that it can be determined what a recipient is already known to know and what risks may be inherent in sharing additional information. Additional risk may be from repeatedly telling the same type of information (e.g. repeated locations give a route that may be projected to a destination) or different parts of information that is more important when combined (e.g. location and orders can give away specific target).

Data mining across compartments:

Incorporated aspects of IAA but also allowed for the planning of sharing to say that certain tasks would occur between certain groups, with a list of what would be shared and

what should be hidden. Aimed to support situations where it may be possible to share data-mined results but where the whole data set is impractical/impossible to retrieve and where the information requester does not want the information holder to know what is being queried.

'Springboard' from ESII:

Various visualizations produced by this tool were presented and compared to the RKB visualisation method.

ReSIST researchers discussed how the Springboard visualisation approach compares with RKB tooling. A difference noted was that 'Springboard' can pull in information from any ontological model and find relationships apparent through the model whereas the RKB tooling has several visualisation enhancements that are specifically coded against the structure of Res-On.

Audience comment: Could some of these mechanisms be captured in ReSIST terms and used interestingly in one of the 'challenges'? The speaker responded that, yes, it was his intention to characterise some of these mechanisms during the lifetime of ESII and SERSCIS, and he would like to be involved in exploring a challenge problem.

Talk 8: System Exploitation 101

Speaker: Alan Hood, QinetiQ

This talk gave an overview of pen(etration) testing techniques used by 'white hats' (the good guys), with observations regarding further techniques available to 'black hats'. The speaker has several years' experience as a leading member of QinetiQ's Penetration Testing Team. Note that the (UK) Computer Misuse Act is not limited by national boundaries, so all techniques described are only ever to be performed with full awareness and written permission of the system owner.. The four main areas of Penetration Testing are: Local access, network-level (infrastructure), web application testing, wireless networks.

Local access: The typical attack route is social engineering and then privilege exploitation. But local access can mean limited environments, so techniques to bypass such limitations are sometimes needed. Local access can obviously be achieved by obtaining password credentials. Password guessing is basic but effective – 3 strikes and out is the usual rule, so two guesses can often be made without the user finding out. Password locations are well known, and they are stored on Microsoft systems using weak hashing (on pre-Vista systems, there is a LANMAN hash present in the SAM file for each user which can be readily broken and show the related password). Another option is to gather passwords from a logged in PC using U3 techniques, which involve removable USB data keys with 2 partitions, one of which mimics a CD ROM so can have 'autoexec' code executed automatically. Based on this, a tool called *USB Switchblade* silently recovers information from Windows 2000 or higher computers. Switchblade can be used to get information off the system, specifically passwords or website credentials.

Network Penetration: This can provide credentials, connectivity and admin privilege. It consists of reconnaissance of the given network, typically following domains on Microsoft-based networks, and can lead to remote exploitation of unpatched systems, and

where privileged password credentials are obtained can spread through techniques such as psexec.

Reconnaissance ("Recce") involves getting network access (local access to some connected machine), 'recce'ing the network to obtain connectivity data (e.g. using tools such as the Backtrack live Linux distribution), and reviewing the recce data to plan an attack (for example, through bruteforcing of services which allow credentials to be tried, or through exploitation of unpatched or poorly configured services).

One technique which can be used on a given Microsoft-based platform is checking for cached credentials - if an administrator has used the system, for example, their password details can be gathered using the correct technique, and once obtained, these credentials can be used on other systems on the network where the account is trusted.

The psexec utility can be used to execute processes on other systems without having to install client software. This requires administrative credentials, so is a perfect combination with the caching technique described above.

Remote exploitation relies on finding vulnerabilities in remote systems, but unpatched networked services are increasingly rare. One can sometimes exploit using the public ‘metaploit’ framework, but public exploits are now much rarer than historically.

Web application testing:

Forceful browsing is the accessing files that are publicly visible, but should not be. It can be useful to infer likely website structure. An example is Intentia vs. Reuters, where a reporter guessed the URL of a company's third quarter results from previous URLs.

Manipulation of HTTP is a real danger as an attacker can manipulate anything in the HTTP – it is completely under local control.

Hidden fields are often used in forms to send extra data along with user input; this data is not displayed by browsers but can be found by inspection of the webpage source and arbitrarily modified. To guard against malicious exploitation of fields, websites should always use end server validation. Exploitation of hidden fields can involve manipulation of data sent to the server (e.g. SQL injection exploits weak parsing of user-supplied data with the purpose of adding commands which will be executed at the server) or cross-site scripting (when a web application allows users to modify the content of the site, and where another user observing that content can trigger an undesired effect). An amusing example was given where a QinetiQ pen tester exploited weak server parsing to make a customer's gambling site accept negative bets.... this resulted in him winning large amounts of money when he ‘lost’ bets. (Being a white hat, he returned all this money to the customer!)

Wireless networks:

WiFi networks are increasingly present, but not always well protected - the Mumbai attackers used unprotected WiFi networks to send emails from innocent third parties who had failed to secure their home systems.

Wardriving is the process of locating WiFi networks, which report their security mechanisms - these are usually WPA/WPA2 (which is generally good, but needs appropriately-long passwords), WEP (which is trivially broken), or unprotected (which

can be directly connected to by a third party). The data received in Wardriving is being publicly broadcast, so the process is generally regarded as legal provided no inappropriate access is made to the network. This raises the difficulty of auto-connection - if a laptop automatically connects to any "open" WiFi network nearby, and a user has set their network up without any protection (due to lack of understanding), it would be difficult to prove intent to penetrate the network. Typical tools for wardriving and the subsequent penetration of located networks include the Kismet, Netstumbler, and Aircrack-ng tools.

Another technique is the impersonation of a WiFi access point – where a user who has a system set to access, say, HOME_NETWORK goes out of range of that network, their system continues to request it; an astute attacker can simulate that network, and the user could auto-connect to the malevolent network (and could be attacked using whatever techniques the attacker wishes).

Talk 9: Actionable Intelligence Extracted from Natural Language Sources

Speaker: Chris Booth, QinetiQ

Improvised explosive devices (IEDs) are used by people in a highly unstructured way: new targets are chosen because they are not as well defended as previous targets; attack methods are constantly changing. Explosive ordnance disposal (EOD) is therefore a constantly moving target, but there is much of value that can be captured from the reports written by EOD operators. The best place to document the constantly changing conditions is in the last field of the EOD report, which allows a freeform narrative of the incident to be captured. The intelligence to be obtained from these freeform fields would be an extremely valuable resource, if it could be extracted at a reasonable price. Furthermore, open source texts may also be a valuable resource for intelligence analysts, but currently only people can reliably use open sources for intelligence.

The Triton reports, produced by Hazard Management Solutions Ltd in Shrivenham, show just how much value there is in open source material. They summarize terrorist incidents as reported in open media. The armed forces and the Intelligence community both use these reports. But a considerable amount of human skill and experience is required to interpret the source material.

There are some tools that allow information discovery in natural language texts, but they are relatively crude and don't have the sort of understanding that a person does. This work aims to leverage the ontologies and natural language processing algorithms in order to automate the extraction of structured information from natural language texts.

The Semantic Web underpins new tools for rapidly mining semantic gems. Tools based on parsing natural language to see its structure, coupled with algorithms for recognizing instances of ontology concepts, can mine semantic information at an affordable price. Semantic information provides analysts with a sound basis for delivering actionable intelligence to their commanders.

We use existing information extraction techniques, such as named entity recognition, and synonym recognition, to identify some useful information from texts. In addition, we have developed tools that combine the power of a natural language parsing algorithm

with a specially developed ontology – in this case an IED ontology – to extract structured data from texts.

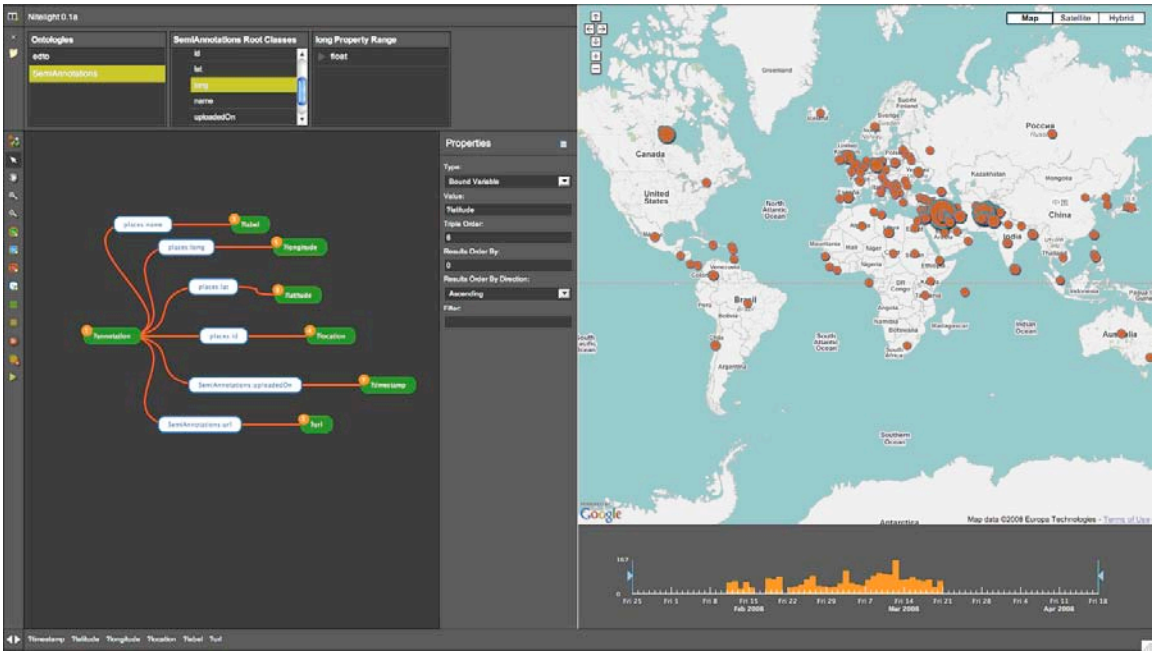
The structured data that we extract is represented as RDF triples, where each triple is compatible with the IED ontology mentioned above, and can be understood as a simple subject-predicate-object statement, which can visualized as a pair of labelled nodes linked together by a labelled arc. If two nodes share the same label, the triples that contain them can be merged to form a graph.

The extraction proceeds by parsing each sentence to identify its linguistic structure. Some sentences match patterns that we have previously identified. If the words or phrases in those sentences correspond to concepts and relations from the ontology, then we extract a triple that approximates the meaning of the sentence. Each document thus gives rise to a collection of triples, or a graph.

This graph can be shown to a user in our Manual Annotation tool, where it can be checked and edited so that the information in the graph is accurate. Then that information is stored in a central Knowledge Repository. If any of the nodes from the document share a label with nodes already in the Knowledge Repository, they are also merged. We can thus begin to aggregate information from multiple sources about a single entity, which is a form of information fusion.

We have shown how traditional information extraction, combined with natural language processing and ontologies can be used together to extract semantic information from natural language texts, i.e. to deliver actionable intelligence from natural language sources.

Related capabilities – designed to combine with the above capability – were developed by Paul Smart's team in Southampton University as part of the SEMIOTIKS project. Of particular note was NITELIGHT, which generates SPARQL queries graphically in a fully reflective tool, as illustrated in the following diagram.



There followed some discussion about how one could apply lessons from SEMIOTIKS to the RKB.

Tom Anderson asked whether it was possible to set out an agenda for investigating application of semiotics.

Neil Briscombe observed that benefits of using semiotics could include tracking of emerging threats, security accreditation, extraction of pragmatics from text and automatic generation of summaries.

Nick Moffat asked whether one could track emerging threats, look at documents and make inferences (using models), and generate a useful representation for analysis. If so, would this be at design time or runtime?

Neil replied that it is difficult to fuse documents, so these tools are better for arming the human analyst.

Neil also commented that NITELITE's ability to produce SPARQL queries graphically may broaden the appeal of the RKB.

Talk 10: Integrating Disparate Knowledge Bases using Semantic Web Technologies

Speaker: Hugh Glaser, Southampton

This talk explained in more detail the architecture of the RKB and associated software.

The different sorts of Knowledge Bases were described, and then the more than 30 different ones were named explicitly. Issues of Semantic Web/Linked Data were presented, and then many components described.

It is highly desirable to avoid incorrect conflation of resources – identifying multiple URIs for one resource. This is the major problem of co-reference, which was discussed in some detail. The solution used in the RKB was explained.

In the conclusion it was stated that major data fusion is possible using Semantic Web technologies and that many types of system can be cast in a Semantic Web framework. In particular, linked data and RDF are effective techniques, and "a little ontology goes a long way". Co-referencing is the biggest problem that arises so an effective co-reference architecture is crucial.

Audience comment: Might a mapping between ontologies be definable between, say, a resilience ontology and a sector-specific CNI ontology that could be exploited by tools? The suggestion was that CNI tools might make use of the CNI ontology, which might reference the resilience ontology. The speaker responded that he would see the mapping as a component to access different knowledge databases, enabling distributed queries across different databases. He emphasised that one would need a way to translate between the two (perhaps using RDF).

Discussion

Joe Bradbury (from the CPNI) had to leave early, but he first remarked that he would be keen to be engaged in follow-up discussions/workshops.

Those present were asked to describe what reservations they may have about the Res-Ex approach. In particular, why not instead go to Wikipedia for mechanism descriptions? It was suggested that a major benefit of the Res-Ex approach is that a wide variety of mechanisms can be described in a uniform manner, and that in contrast Wikipedia descriptions are not (formally) structured. This gives an opportunity to develop tools to exploit the information in a uniform manner. It was further noted that one could use Google to find mechanism descriptions, but that again these would not be structured.

The consensus was that the Res-Ex approach seems a good idea. But the question becomes how to get people to accept/use it. This is a difficult problem, which may become easier when the RKB is fully fledged.

It was remarked that we can refer to the Res-Ex approach in proposals, but there was a cautionary note that the approach must be accessible to the practising engineer.

On the subject of how to obtain a challenge problem, it was suggested that projects such as those described by QinetiQ might be good places to look. Neil Briscombe repeated that he would like to describe some of his tools as resilience mechanisms within the lifetime of existing projects. This was welcomed, but with the reminder that the RKB is in the public domain.

The question of how to engage interest from a wider community arose. One response was that existing stakeholders could be approached, such as customers for existing or recent projects (e.g. Bristol City Council). Also, Joe Bradbury may know of stakeholders in the finance sector. Transport contacts could be found through the Highways Agency.

It was agreed that a sensible way forward would be to build up contacts by e-mail until a sufficiently large and interested group has been achieved, including practitioners. For example, it was noted that Lorenzo Strigini of City University had been keen to attend the workshop but was unable to do so. Lorenzo has valuable expertise in human factors issues and it was agreed to invite him to be involved in the e-mail forum. Other obvious potential contacts were identified at York, Edinburgh and Bristol. There was also a strong interest in expanding representation to include contacts from mainland Europe. Once the forum is established, it was agreed it would be worthwhile organising a follow-up workshop to expose to practitioners a summary of the techniques described above.

There was some discussion regarding a suitable co-ordinator, with full agreement that Neil Briscombe would be welcomed as the initial co-ordinator (assisted by Nick Moffat). Neil's co-ordination role may be a temporary until a challenge problem is chosen, after which the problem owner may be asked to co-ordinate.

To illustrate the potential a poster (produced by Bristol City Council) of the issues relating to Bristol (a core city with domino status) was presented and discussed at length. Debate included scenario issues that indicated how the themes of the workshop would be recognised by CNI protection practitioners. Also examined was a technical focus in the poster of exploring the geospatial issues of risk-focused crisis management and how using the RKB might be beneficial.

Tom Anderson summarised the two days from Newcastle's perspective: The workshop had not completed the specific objective of identifying a challenge problem but the talks

had stimulated plenty of discussion. There were clearly some promising opportunities for collaboration and for extending the group to include more practitioners.

Hugh Glaser gave Southampton's perspective: Critical Infrastructure Protection is a good domain within which to investigate ontologies. Hugh was uneasy about the absence of example CNI features/problems. For example, it would be helpful to know specific information about the infrastructures QinetiQ is looking at. He said he would be interested in further collaboration on topics discussed.

Next Steps

Closer links between those present at the workshop could of course be forged through collaboration on future research projects. For example, Hugh Glaser was invited to propose extensions to existing projects in which Southampton is involved with Neil Briscombe's QinetiQ team. Such collaboration could focus on themes explored during this workshop.

A forum is planned that will

- be lightweight to encourage wide interest
- use a mailing list to give exposure to problems of interest and methods/tools
- involve two communities: 'critical infrastructures' and 'emergency planning'
- act as an informal introduction service to help focus research on realistic problems
- be chaired by Neil Briscombe initially

A major objective of the forum will be to identify a specific challenge problem definition, then define it and invite the community to propose Res-Ex solutions, perhaps through a further workshop meeting. The likely topic is "Emergency Planning".

Neil Briscombe is keen to make use of Res-On within the SERSCIS project and to characterise as resilience mechanisms within the RKB a variety of methods and tools developed within his projects. In addition, he intends to exploit the RKB in some of these projects to support reasoning about resilience. This will use internal QinetiQ tasking.

List of Attendees

* denotes a member of ReSIST.

Austin Anderson	–	Southampton University
Tom Anderson*	–	Newcastle
Stuart Bertram	–	QinetiQ
Chris Booth	–	QinetiQ
Joe Bradbury	–	Centre for the Protection of National Infrastructure, CPNI
Neil Briscombe	–	QinetiQ
Hugh William Glaser*	–	Southampton University
Alan Hood	–	QinetiQ
Russell Lock	–	St Andrew's University
Ian Millard*	–	Southampton University
Nick Moffat*	–	QinetiQ
Colin O'Halloran*	–	QinetiQ

Stephen Riddle*	–	Newcastle
Tim Sheppard	–	QinetiQ
William Simmonds*	–	QinetiQ

Apologies were sent by Andrew Hartley of Bristol City Council and Lorenzo Strigini of City University.

ReSIST workshop report

Challenge Problems for Resilience-Explicit Computing with Assistive Technologies

Culture Lab, Newcastle University
5 December 2008

1. Introduction

The ReSIST workshop on ***Challenge Problems for Resilience-Explicit Computing with Assistive Technologies***[1] aimed to bring together researchers and practitioners from the different communities of *Resilience* and *Assistive Technologies.* The goal, as with all three workshops, was to define one or more Resilience-Explicit Computing "Challenge" problems. Such a problem would highlight a relevant issue to be used as a benchmark for resilience-explicit computing techniques, and would have the potential to lead to the development of technologies and tools to support developers of Assistive Technologies.

The ReSIST workpackage on *Resilience Explicit Computing* aims to improve the extent to which system resilience can be predictably maintained, by stating the resilience-related properties of components in the form of *metadata* published either by the components themselves or by observers. This metadata may be used at design-time to guide the choice of design patterns and development tools, or at run-time to support reconfiguration. Examples of metadata include descriptions of known failure modes declared in a component's functional specification, a person's workload in a socio-technical system, and historical availability statistics. The phrase "resilience mechanism" is used to refer to any design pattern, technique or tool intended to improve system resilience. Examples include fault-tolerant architectural patterns (e.g., n-version programming) and development tools (e.g., robustness testing tools).

Challenge problems are a mechanism for demonstrating progress and galvanising the research community. A well-known example is the Grand Challenges in Computing Exercise, which has identified a number of long-term research initiatives on an ambitious scale. The Resilience-Explicit Challenge Problems are envisaged on a smaller scale, but are intended to help us to acquire and expand knowledge of the state of the art by encouraging multiple approaches to a common problem, and facilitating comparison between existing technologies.

The field of Assistive Technology covers a wide range of technologies designed to support "independent living": that is, the ability to take part in normal, day-to-day activities in spite of infirmity brought on by age, disease or disability. Examples include support for medication, diet and nutrition; personal mobility; and virtual communities.

[1] The workshop was run in collaboration with the EPSRC Inclusive Digital Economy Network and the EU FP7 OASIS project http://www.oasis-project.eu/

Each of these examples has significant potential risks to individual safety and security: these include danger of overdose, falls or other accidents, and phishing attacks.

The aim of the workshop was to facilitate discussion of these technologies and the potential threats to resilience, and the scope for applying the Resilience-Explicit approach, with the goal of defining one or more challenge problems and forming a working group to work on the further definition and application of these problems.

The workshop attendees were: Tom Anderson, Brian Randell, John Fitzgerald, Steve Riddle, David Greathead, Patrick Olivier, Stephen Lindsay, James Thomas, John Shearer, Jon Hook, Phil Heslop (Newcastle University); Giovanna Di Marzo Serugendo (Birkbeck, University of London); Stuart Colmer (Centre of Excellence for Life Sciences); Alan Newall (University of Dundee); Ian Millard (University of Southampton)

The workshop was structured with introductory presentations followed by a specially filmed video, featuring actors in a scenario highlighting many of the problems in Assistive Technology. The workshop concluded with breakout groups discussing one of four case studies, as potential sources of material for a Challenge Problem. Each of these elements of the workshop will be summarised below.

2. Presentations

The first presentation from *Steve Riddle,* Newcastle University, set the scene for the workshop and attempted to summarise key concepts of dependability and resilience for the benefit of the attendees who were not familiar with the terminology.

This was followed by guest speaker *Stuart Colmer,* from the Centre of Excellence for Life Sciences (North East). Speaking on the commercial context for Assistive Technologies, Stuart highlighted the problem of a lack of professional careworkers coming into the industry, and the resulting need for technological support to promote independent living. Opportunities for Assistive Technology in daily life included medication compliance, home safety and security, nutrition and cognitive support, support for mobility and virtual communities. He made the point that we do not strive for perfection in these technologies, and should not underestimate the abilities of those that are in need of assistance. It is more important to provide a useful function, ideally making use of standard equipment, if the technology is to have a chance of being used. It is crucial to be aware of user interface design and to think about unintended consequences of use of the technology. Ideally, such technology should have a natural, familiar interface where factors of *security* and *Quality of Service* are a given.

Research Associate *Stephen Lindsay* from Newcastle University then presented a research agenda for Assistive Technologies. His main focus is assistance with Dementia, one of the fastest growing issues in public health today. It is estimated that one in three people in Britain over the age of sixty-five will die with dementia, with a cost estimated at £4.6 billion per annum. Challenges faced by patients with dementia include behaviour (aggressiveness, depression), wandering, memory loss and a decrease in self-control. The

major resulting issues are loss of personhood and loss of independence. Current research projects to relieve these problems include use of GPS location systems for 'elopement detection' and recovery, and RFID-tagged Smart Environments; and Activity recognition techniques to help with food preparation, medication reminders and shopping lists. Ethical issues have been identified, including acquisition of personal medical data which could be used against the owner, and the ability of the owner to consent to its use. Dependability issues include the transmission of personal data over insecure wireless networks and the reliability of GPS location systems, but also less obvious factors such as the potential trauma caused by poorly functioning devices and by the need for maintenance of a device. Consistency of interface design is a further common factor.

3. Video: "Relative confusion"

The video was prepared for the workshop by *Alan Newell* of the University of Dundee. The video featured three actors involved in a scenario based around the installation of a new digital television, but also included episodes based around the use of 'everyday' technology such as telephones, answering machines and GPS devices. One actor remained 'in character' for a very instructive discussion after the video. It was clear that much more could be done to keep interfaces simple and familiar, and that successful assistive technology needed to be accessible first and foremost, then have a useful function, and then provide a dependable service.

4. Breakout groups

The participants divided into groups. Each group discussed a scenario provided by the OASIS project: the scenarios all related to a proposed device or technology to assist vulnerable people. These included a route guidance system, nutritional adviser and a driving health assistant. In discussing the scenarios, participants focussed on the following questions relating to resilience and trust:

1. What does "Resilience" mean in this scenario?
2. What faults or problems do these kinds of system need to be resilient to – what could go wrong, and what might be the causes?
3. What features might we require from the 'silicon component', such as high reliability, or availability?
4. What would be the ideal, reliable, trustworthy solution?

Discussions are summarised below.

Scenario 1: *Route guidance and transport information system*. This scenario involves assistance to a pedestrian or driver, in a similar manner to a GPS navigation system. Additional features would be choice of safe or scenic routes to a destination, re-planning if the user takes a different route, and information about points of interest or current events. Some of this information might be provided by approved service providers. The transport information part of the system would provide alternative routes from a given

origin to a destination, making use of a given set of possible forms of transport (car, foot, bus, train, etc).

Discussion identified failures in positioning, communications and currency of map information as the most likely problems, with incorrect information being the common effect. There was some concern about the possibility of malicious attack to block signals, or send the user through an unsafe area. Some of these problems could be clearly mitigated, for example through the use of the most recent data if up-to-date data became unavailable. The use of user profiles to record favourite routes, typical walking speed and preferred forms of transport brings further concerns about privacy. It was proposed that an ideal solution would be "as good as someone leading the user by the hand", and the required level of trust in the information provided could be achieved by a large user community providing routing information and giving a level of trust, either implicitly or explicitly.

Scenario 2: *Driver health assistant.* This scenario proposes a device worn by a driver which would monitor their health status, measuring for example blood sugar levels and pulse, and take appropriate action when a dangerous situation was observed. Such action might include an audible alarm to the driver or calling for medical assistance. Medical records might be available in the car or be referred to based on a lookup from a unique code in the device.

A number of problems were identified, such as communication failures (as in the previous scenario), the reliability of the device and the danger of false positives with consequential emergency callout, the need for the device to be worn by the user (who may forget), and the danger to the device itself through mishandling. The accuracy of the decision parameters and policies for medical diagnosis are a further issue, which would result in a high level of criticality for the device. As before, the key issue which would affect whether the system would be used is the level of trust the user would have in the system.

Scenario 3: *Vehicle accident detection.* This scenario concerned a proposed system which would determine when a vehicle had experienced an accident or collision, when the driver was not able to make an emergency call themselves. The scenario has some similarities with the previous one, and would have similar potential failures such as communication failures and false positives. A further feature (also relevant to the previous scenario) is the need for feedback to the user to reassure them that emergency callout has been effected.

Since the vehicle itself could have had a major collision, the callout equipment would need to exhibit robust fault tolerant behaviour such as redundant sensors, an externally monitored heartbeat, and a rugged design. Other common issues identified include GPS failures, malicious interference, inadequate maintenance and false manual activation or cancellation.

5. Summary and Follow-on work

A wrap-up provided by Tom Anderson (Newcastle) drew out the following common threads:

1. Assistive Technology, in order to be practical and widely used, must be accessible and functional, must bring the user some benefit, and must be *adequately* dependable. While we as proponents of dependability might demand that dependability is the priority feature, it must be accepted that technology will not be used unless there is some benefit – and users will put up with some minor faults if the benefit outweighs the inconvenience.
2. The stress has to be on the interface to the user: this must be simple, consistent, familiar, providing clear guidance which is comprehensible to the novice user. Returning to the tenet proposed by Stuart Colmer, Assistive Technology should provide *natural interfaces* where *security* and *Quality of Service* are *given*.

It was clear that there was genuine interest in conducting follow-up work to define a challenge problem drawing on the scenarios discussed during the workshop. To this end a local interest group was identified which would be tasked with coordinating followup activities. Co-ordinated by local representatives of the Resilience and Assistive Technology Communities, the initial activities of this group will be to recruit participants from both communities and begin the selection and definition of a challenge problem drawing on the scenarios discussed previously. Participants will be invited to develop solutions to these problems, with an emphasis on the selection and use of appropriate resilience mechanisms.