# 5.0 – Minutes of the NATO RTO Workshop on “Building Robust Systems from Fallible Construction”

Prague, 9 and 10 November 2006

**Prepared by Yves van de Vijver**

## 5.1 AGENDA

**Thursday, 9 November**

| | |
|---|---|
| 9:00 | Welcome, logistics, introductions of participants<br>- Morven Gentleman and Milan Snajder |
| 9:15 | Introduction to NATO Research and Technology Organization, and the Information Systems Technology Panel<br>- Lt. Col. Patrick Prodhome, IST Panel Executive |
| 9:30 | Introduction to topic and process<br>- Morven Gentleman |
| 9:45 | First working session: framing the problem – what are the challenges today and why is the research done in the past not enough?<br>• Systems of systems<br>• Pre-existing third-party code, including COTS and Open Source<br>• Installation and configuration errors<br>• Misunderstood interfaces and protocols<br>• Invalid operator input<br>• Malicious attacks<br>• Rate of upgrades<br>• … |
| 10:30 | Break |
| 10:45 | First working session resumes |
| 12:30 | Lunch |
| 14:00 | First working session resumes |
| 15:30 | Break |
| 15:45 | First working session resumes |
| 16:30 | Summary of day’s discussions |
| 17:00 | End of first day |

**Friday, 10 November**

9:00 Second working session: what new technology or new approaches might reduce exposure risk or facilitate damage recovery
- Education to ingrain fault sensitivity awareness
- Architecture, especially Service Oriented Architecture
- Internet and Web technologies, especially web services
- Virtual machines
- Genetic programming
- Trouble sensors
- Situational awareness and autonomic frameworks
- Forward rather than rollback error recovery
- Dependability of adaptive systems
- …

10:30 Break

10:45 Second working session resumes

12:30 Lunch

14:00 Second working session resumes

15:30 Break

15:45 Way forward: Summary of conclusions and next steps

17:00 End of workshop

## 5.2 DAY 1, 9 NOVEMBER 2006

### 5.2.1 Welcome and Objective

The chairman of the task group and the workshop, Dr. Morven Gentleman (Canada), opens the meeting and welcomes the attendees to the workshop. The objective of the task group is to produce a report for the NATO community on the subject of robust systems/software development. The topic of robustness/ reliability is not new, but goes back all the way to the early days of computers and software. Therefore, the types of questions to be addressed in this workshop are the following:

- What has changed in the world of fault-tolerance and building systems in the last 30 years?
- What is different now compared to then?
- What has not been addressed before?
- What new techniques for building systems exist, and how do they deal with reliability?
- ….

After this welcome and introduction to the objective of the workshop, the NATO/RTO National Coordinator of the Czech Republic, <Name>, welcomes the attendees to the beautiful city of Prague and the magnificent workshop location at the Czech Ministry of Defence.

Next, Lt. Col. Patrick Prodhome, the IST Panel Executive, introduces NATO/RTO and the IST Panel. He shows the central position of the RTO in the NATO Structure and explains how the RTO works.

### 5.2.2 First Working Session

The first working session consisted of a number of short introductions describing the problem area and the invited experts' views on this area, followed by longer, in-depth presentations of the work of some of these experts.

#### 5.2.2.1 Short Introductions and Views

**Dr. Morven Gentleman**

The first working session starts with a problem definition by the chairman of the task group, Dr. Morven Gentleman. He demonstrates the problem by means of a Geographical Information System (GIS), for instance an internet site with a map of the city of Prague, with public buildings, tourist attractions, public transport, etc., which you may ask for routes from A to B and the time it takes to get there. These systems usually contain static data, such as maps, roads and rivers, which change infrequently, and dynamic data, such as public transport schedules, and road works. More sophisticated systems may also take time-based data and weather into account (e.g. to calculate time to move from A to B). Often, this information is not in the system itself, but retrieved from external sources. And these external sources are systems of their own, developed and maintained by third-parties not under control of the GIS operations. Also, the usage of these external sources may not be as originally foreseen by the developers of that system. Finally, a main source of errors is wrong operator input, and this can happen anywhere in the system itself, or any of the external sources used.

The above example shows the many things that can go wrong when building an application on the basis of a collection of (external) components of which the reliability is unknown, or questionable. Building such an application from scratch, however, is an expensive solution, and hardly ever done (especially not in commercial applications).

The classical approaches towards reliability mostly duplicated components (different implementations) and used a voting system to determine which "answer" was the correct one, or used an "oracle" to decide whether an answer was ok, or not. A basic assumption in many of the classical approaches was the presence of an ability to roll-back to a previous (correct) state and try again from there. In current applications, such as the one above, this is not always possible. So what to do then?

This last question is also very relevant to the NATO context, in which coalitions have to be formed in a very short time frame, and in which the composition of coalitions change regularly. Each of the partners brings their own systems and these have to be put together quickly.

**Prof. Alexander Romanovsky**

Prof. Romanovsky (University of Newcastle) is an expert in (software) fault tolerance and explains that a shift has taken place in this field. Traditionally, software fault tolerance was built into systems to deal with hardware failures. But nowadays, faults are introduced because of architectural mismatches between different components or subsystems within the system, or because of faults in the underlying infrastructure in which the components rely. These different types of faults ask for different types of techniques to deal with them. And to add further to the problem, also in implementing fault tolerance, faults are introduced. Therefore, he would like to direct research towards "software engineering for fault tolerance", i.e. a method for software engineering in which fault tolerance is an inherit part of the method, not an add-on afterwards.

**Prof. Mary Shaw**

Prof. Shaw (Carnegie-Mellon University) is an expert in software architecture models. Recently, she has been researching high-level system design methods in which costs (e.g. for high assurance) are taken into account as an integral part of the design. The needs of the user play an important role in this. If, for example, the GIS application described by Dr. Gentleman earlier will be used by an ambulance service, reliability and accuracy must be much better than if the application will be used by tourists exploring the city. Same application, but a need for different designs (resulting in different costs).

Prof. Shaw also mentions an important difference between system design thirty years ago, and system design now: lack of central control. Where large systems in the past were usually designed by one chief designer, or a small team of designers, who had control over the whole system design, large systems today are more and more composed out of distributed components under distributed control. This introduces the problem that any of the components of the system may change, without notice, at any time.

**Tomas Feglar MSc PhD**

Tomas Feglar is an international consultant in Information Systems Research and Architecture and an expert in process integration and systems engineering. He has a large experience in working with the Czech Ministry of Defence. He notices that the eastern European countries and their defence organisations are modernizing and professionalizing fast. This challenge is accompanied by a rapid rate of change in Information Technology. He has been involved in meeting these challenges for the Czech Army and separated the two types of challenges by introducing business process models. First, a technology independent set of business models were composed for Army Force Management and Force Development. Then, these models are populated with technology and risks and threats to this technology are identified. In the systems engineering activity, these risks and threats are explicitly taken into account in the architectural design process, risk management process, information management process, and security management process.

As a final remark, Mr. Feglar would like to see robustness become a major topic in Service Oriented Architectures.

**Maarten Boasson**

Maarten Boasson is an expert in software architectures, especially for distributed applications. He worked for a major Dutch defence company and work on distributed, real-time architectures for, among others, frigates of The Royal Netherlands Navy. He always believed architecture was the answer to all problems, but over the years he has become more sceptical. When comparing software engineering with other engineering disciplines, he notices that there is no software engineering "discipline": no discipline, no theory, no guidelines. "We are just trying to build systems".

Most efforts within software engineering are addressed to produce systems with "very few" faults, but one fault may be too much. Most of the efforts in fault-tolerance are attempts to build a way around faults, but who are we fooling, but ourselves.

Robust systems must contain no faults. And in order to ever get there, software engineering should become a "real" engineering discipline.

**Christophe Dony**

Christophe Dony is an expert on exception handling. His current research consists of analysing languages for the construction of systems (object oriented, component based, service oriented architectures), and defining a new one, which will include exception handling mechanisms. Such mechanisms already exist to some degree in software languages (e.g. Java), but not for distributed systems, component based systems, etc.

**Frédéric Painchaud M. Sc.**

Frédéric Painchaud works for the Defence Research and Development Canada and is relative new to the area. He started researching this field because a client needs a framework for building fault-tolerant distributed systems. The short term goal of the research is to produce an overview of the available products for such a framework. The long term goal is to develop a new framework. Frédéric has kindly offered the results of his research so far for a state-of-the-art section in the task group's final report, for which the task group is very grateful.

**Dr. Morven Gentleman – View of Experts not Present**

The chairman of the task group, Dr. Morven Gentleman, finishes this first working session by shortly introducing some views of experts who could unfortunately not attend the meeting in person.

The first view considers autonomic computing in the form of self adapting or self healing software a solution to the problem of fault-tolerance. The software monitors itself and, in case of bad performance, adjusts the built-in control loops. Each individual component does this for itself, and the system for the system as a whole. This solution is considered most applicable in systems where "faults" do not occur instantaneously, but do occur as a result of degradation of performance over time.

The second view introduces recovery oriented design as a solution. Given the fact that errors will always be there (e.g. human operator input errors), design the system in such a way that it can always go back to a previous, safe state.

The third view abandons the concept of predictive engineering (think what can go wrong and build something to deal with that) and promotes to run a system in many different situations to find the faults that (may) occur, for instance by applying genetic algorithms. This view also promotes no to go back to previous safe states, but go to a stable state from where to continue. Of course, the problem is: "which are the stable states?"

#### 5.2.2.2 Presentations

After the short introductions, the first working session was resumed with longer presentations, some of which were based on the position papers submitted.

**Tomas Feglar MSc PhD – "SOA Robustness Roadmap"**

Tomas identifies four important models to be developed for a system with a Service Oriented Architecture (SOA):

- SOA Robustness Decomposition;
- Enterprise Integration Model;
- Enterprise Application Model; and
- Decision Support Model.

In this presentation, he focuses on the first of these models. The system is decomposed in business processes, which provide one or more services to other business processes. System characteristics such as agility and ability are covered within the business process tier of the (layered) architecture. Reliability is covered within the SOA tier of the architecture.

For each of the business processes, risk profiles are defined in which the various threats and impacts are identified. These threats may be technical (e.g. hardware failure), but also physical (e.g. destruction of parts of the infrastructure because of war-time activity). For each risk identified, risk measures are determined. These measures may be measures for fault-tolerance (e.g. in case of hardware failure), but also management or logistic measures (e.g. spare parts).

The idea behind taking risk management into account from the beginning is that future changes to the system will be more economical, resulting in lower Total Ownership Costs.

During the work performed for the Czech Army, Tomas has discovered recurring patterns in these risk profiles and measures, and identified a number of "SOA Enterprise Robustness Patterns" which are used in numerous places throughout the overall system description.

The modelling activities are supported by a software tool, in which both the process view and the design view are maintained and synchronized. Tomas demonstrates the use of the tool taking an example from the position papers of F. Michaud and F. Painchaud.

**Prof. Mary Shaw – "Strategies for Achieving Dependability in Coalitions of Systems"**

Mary starts her presentation with trends in systems development and use: from local to distributed, from independent to interdependent, from insular to vulnerable, from installations to communities, from central administered to user managed, from software to resource, from single systems to coalitions. As a result of these trends, failures will ripple through these "systems of systems".

The dependability issues with these systems of systems are numerous:

- Different types of users require different levels of dependability.
- Costs matter (money, delay, disruption), but few can afford high dependability.
- Uncertainty is inevitable because specifications will never be complete, and actual operational environments are unknown.
- Integration is a bigger challenge than components.

In order to structure the problem domain and the discussions in the rest of the meeting, Mary shows a classification of types of measures for reliability. The first distinction is the time at which a problem will be considered: before using the system (preventive), or during use of the system (reactive). Preventive measures "validate" the system against a standard. This may be a global standard, a relative standard, or a policy standard. Validation against a global standard is the "traditional" approach, based on analysis and careful development. Other measures in this category are formal methods, and testing. Validation against relative standards take the user needs into account, and is therefore categorized as "User-centred". Validation against policy standards is an approach in the case of large systems of systems, in which negotiations among stakeholders drive the standard. This category is therefore called "Ultra-large scale". Reactive measures, or "remediation", can be further decomposed in technological reactive measures ("fault-tolerance"), technological adaptive measures ("self-adaptive, self-healing"), and economical reactive measures ("compensation").

The preventive measures ("validation") are common in many engineering disciplines. A common factor in these disciplines is the existence of linear models, which may be validated easily. But, in case of systems of systems, such linear models probably do not exist. Therefore, validation will be at least very difficult, if not impossible. As a consequence, for systems of systems, reactive strategies to dependability will probably be more effective.

**Maarten Boasson – "Software Faults"**

Maarten identifies tree types of faults: faults that lead to no results; faults that lead to wrong results; and faults that lead to late results. Faults are the result of errors. In order to prevent faults, first of all, errors must be detected before they occur. Secondly, the impact of errors must be limited, and, thirdly, errors must be repaired. Maarten would like to see that effort and research is focused on the first way of prevention, error detection, or, even better, error prevention ("never stop striving for correct programs"). He therefore is a strong advocate for research into formal data models, formal verification, and minimal dependencies (e.g. by late binding, asynchronous communication, and autonomous components).

**Prof. Joe Sventek – "Closed-Loop Management Patterns"**

Joe presents the idea of using closed-loop management patterns in system management. Closed-loop management is used in many production plants: measuring the output of a process, and adjusting the input and control of the process to steer the process towards producing the desired output. This principle works well for managing processes with slowly degrading output. So why not adopting such an approach to managing systems and software processes?

To make the use of closed-loop management even more powerful, the system to be managed could be designed as a hierarchy of lower-level processes or subsystems. On each of these levels, and in each of the components or subsystems at those levels, the closed-loop management principle can be applied. The advantages are manifold, including the closure of a loop as low as possible in the hierarchy will avoid the propagation of problems, and some well-proven techniques at one level can be reused at other levels.

The open question in applying this principle to achieve robustness, is how well this principle will work for faults, i.e. not a slowly degradation of performance, but a sudden malfunctioning of the system?

**Prof. Cristophe Dony**

Cristophe's interests are agent-based and component-based applications and message oriented programming. The objective of his current research is to specify and implement an Exception Handling System (EHS) for new architectures, for instance, J2EE.

Cristophe defines an exceptional situation as a situation in which the standard execution cannot continue. An Exception Handling System must then raise the exception, associate handlers to entities to deal with the exception, and put back the system to a coherent state.

Current solutions towards an EHS include:

1) Standard signalling methods (e.g. Java), which are stack-oriented and destructive;
2) Separation of treatment; and
3) Contextualization (pass exception to requester who knows the context).

These solutions, however, do not work well when dealing with concurrency, which is inevitable in today's larger and larger, distributed systems. Commercial implementations of exception handling systems for these types of systems are not available yet. But, Prof. Dony et al. have been able to produce a first implementation of an exception handling system for J2EE.

He is currently looking to coordinate his activities with other researchers in the field.

**Frédéric Painchaud**

Frédéric introduces the research he has been performing in the last eighteen months on fault-tolerance. He identifies three sources of faults: the system environment, the hardware, and the software. Most error recovery strategies can be classified as either: retry, fallback (redundancy), or diversity (different implementations of the same algorithm). Most of the times, these strategies fail, because the underlying logical problem is not recognized, and similar mistakes are made independently of each other.

He started researching this field because a client needs a framework for building fault-tolerant distributed systems. The short term goal of the research was to produce an overview of the available products for such a framework. The long term goal is to develop a new framework.

The anticipated solution is a framework for systematic development of software with built-in support for managing diversity, based on the principles of Erlang (Ericsson), and developed for JAVA with support for distributed applications.

## 5.3 DAY 2, 10 NOVEMBER 2006

### 5.3.1 Second Working Session – Presentations

**Maarten Boasson – SPLICE**

The second working session starts with a presentation of the ideas behind SPLICE, a dependable, distributed architecture developed by Maarten Boasson during his activities at Hollandse Signaal Apparaten BV. A commercial implementation (OpenSplice) is available.

The basic idea behind the architecture is that a system works when "the right information is at the right place at the right time". The design principles behind SPLICE are: make inter-process data visible, minimize dependencies, and maximize component autonomy. The resulting architecture is deceptively simple, but very powerful.

SPLICE contains a number of mechanisms for fault-tolerance: passive replication ("cold-start"), semi-active replication ("hot-start"), and active replication. For robustness purposes, all messages should pass absolute states, not relative states, in order to allow components to send the same message multiple times without changing the meaning (e.g., turn left 10 degrees cannot be sent more than once, because if two messages are received, the ship would turn 20 degrees; a message "at time T, change course to 70°" may be sent as many times as needed to assure one message is received).

After this presentation, a number of questions were raised by the audience:

Q1: Is there a notion of reflection?

A1: Yes, processes can look at system data and determine which processes are subscribed to what, which processes are running, etc.

Q2: Is deadlock/ live-lock used as a programming tool as it is often in blackboard systems?

A2: Deadlock/Live-lock never occurred, so the situation is not explicitly dealt with. Starvation may happen, but deadlock not.

Q3: How to ensure real-time behaviour?

A3: Hard real-time is not practical, so just assure empirically, or by calculation, that messages are passed quickly enough for proper working of the system. Usually, passing time-stamped information is enough to make a successful hard real-time system. Most hard real-time requirements are not really hard. Usually it is some sort of periodic calculation.

Q4: Some data passed is large ("blobs") and fixed length messages are not very useful, which means you have to send sequences of messages. How is this done in SPLICE?

A4: There are two types of messages in SPLICE. The first type is "fire and forget", for which length is irrelevant. The second type is "order maintained", for which messages are chopped up, sent, and assembled as an atomic action before giving the message to the subscriber.

Q5: How do you establish that a collection of processes together are a semantically correct system? Is there a chief designer?

A5: In practice, there has always been one "group" of designers with control. But the architecture provides some support in the form of "watchdog processes", which monitor the running and dying processes. Furthermore, the publish/subscribe mechanism allows publishers without subscribers, and there may even be multiple publishers, but this is not recommended. Crucial for the working of the system, is to get all required data structures right.

The discussion finally led to the following problem statement by Mary Shaw that would guide the remainder of the working session:

> "If you compose your system by using externally developed components or systems (e.g., a database available through the internet), how do you assure your overall application is correct?"

This is a specifically new challenge since this way of composing systems is relatively new. And in the area of Service Oriented Architectures (SOAs), which aim to meet this challenge, research into robustness of SOAs is not yet covered (very well).

With this problem statement, presentations by attending experts have finished, and the meeting moves to a smaller round-table room in order to facilitate interactive discussions better.

### 5.3.2 Second Working Session – Round Table Discussion

The chairman of the meeting, Morven Gentleman, starts the round table discussions with some more ideas on what is different now compared to thirty years ago, and how these differences may be used to improve robustness. One major difference is computing power and network capacity, which was once scarce but now often in excess. Why not use the excess power for speculative computing (e.g. data mining) on archived data such as event logs and incident logs to discover something has gone wrong or might go wrong? Add "audit routines" to the software that, based on earlier results, suggest that something is not working correctly. In the example of the GIS application at the start of the meeting (beginning of Section 5.2.2.1), structured data could be audited to get an idea of the quality of the data. Also, consistency with unstructured data could be checked (is a road in the structured data visible on the satellite image?). But finding inconsistencies is one side of the problem; the other side is what to do if one is found? How do you notify the running application that something's wrong, and what is the application supposed to do then?

Mary Shaw suggests that such an approach should be coupled with a model of confidence for the application and raise flags, but maybe should not directly impact the running application. Morven Gentleman replies that there should be grades in audit routines, such that serious problems are dealt with immediately, and minor problems are left to deal with later, unless more audit routines raise the same issue, and the "problem level" is increased.

## Patrick Prodhome – Guidelines for Final Report

Because some attendees will have to leave the meeting earlier, the round-table discussion is shortly adjourned to enable Patrick Prodhome, NATO RTO IST Panel Executive, to provide the attendees with some guidelines for contributing to the final report of the task group. An important prerequisite is the submission of a publication clearance through the proper national NATO channels, in order for RTO to be able to publish the final report. Patrick also shows how to log onto the RTO web site and what materials may be found there.

## Round Table Discussion Continued

The discussion continues on the use of the spare capacity in the military environment. Where commercially "just-in-time" seems to be the prevailing attitude (which usually turns out to be "just-too-late"), military systems are specifically designed to handle peak loads (war-time). When not needed (peace-time), spare capacity can be used to do training exercises, or simulations, to ensure that the system will work when it is really needed. This way of using resources is part of the military culture, where personnel are used in a similar way. Therefore, military systems are good candidates to apply the suggestions of Morven Gentleman to.

Cristophe Dony remarks that adding more constructs to a system for robustness usually tends to obscure the structure of the system and make the system more difficult to understand. Alexander Romanovsky follows up on that remark by stating that the contrary might be the case if constructs are used that allow separating normal computation from exceptions. In this case, the understanding of the system may even be increased. This view is supported by Tomas Feglar, who uses risk management strategies to build robustness into systems top down. He claims the resulting system is better, needs less repairing during operational lifetime, and therefore the initial structure is better maintained.

After these comments, the discussion is again focussed on the question: What is different now, compared to thirty years ago? Mary Shaw has been looking into "Ultra Large Scale Systems" recently, and found the following list of differences:

- Decentralised operation and control.
- Conflicting, unknowable, diverse requirements.
- Continuous evolution and deployment.
- Heterogeneous, inconsistent, and changing elements.
- Indistinct people/system boundary.
- Normal failures.
- New forms of acquisition and policy.

After lunch, Morven Gentleman, re-starts the discussion from a slightly different angle. In the seventies, techniques of redundant data structures, which allowed to check damage in data, and sometimes even allowed automated repair, were popular. Have these techniques been forgotten, or are they still around? Alexander Romanovsky knows that diversity in data structures, either by a mapping function, or by double implementation, is still used to recover data structures. Mary Shaw reminds that in user interfaces, checkable redundancy (e.g. address and postal code) is sometimes built in to check consistency of manual input. But, in general, it may be worthwhile to look into some of these "old" techniques, and see what has become of them, to include in the final report. There may even be techniques that once where abandoned because of, for instance, lack of computing power, and that could be revived because of the changed conditions in computing.

Alexander Romanovsky follows up this discussion with the observation that there seem to be many diverse implementations of the same functionality on the Internet. Maybe this is a new form of redundancy that could be exploited for fault tolerance? Morven Gentleman, however, remarks that, although the interface (web page) is different, the underlying implementation could still be the same, and that there is less diversity than you expect. Mary Shaw supports that view by adding that most weather sites are using the same underlying weather data from a single source. Alexander counters that, even in this case, the location of the site may be important for speed, and depending on your application, this could be a very important requirement.

The discussion is wrapped-up with some final thoughts on how all of the techniques and issues discussed in the previous two days will in the end better support the commander in asymmetric warfare; because that is the kind of question the NATO and its member states would like to see answered in the final report. The answer to this question, borrowed from Maarten Boasson in his presentation of SPLICE, is that these techniques will eventually help to get:

> "The right information, at the right place, at the right time."

## 5.4 MEETING CLOSURE

The meeting is closed with some ideas on how to produce the Task Group's Final Report:

- Morven Gentleman will, with the help of Yves van de Vijver, produce and circulate an outline to the meeting attendees and task group members, who are requested to give feedback and submit missing items.
- The possibility of a next meeting, for instance in June 2007, to actually work on the report will be investigated.
- Presentations of this workshop will be included in the Final Report as an appendix. Authors are requested to get publication approval through their national channels in order to enable the inclusion.

Finally, Morven Gentleman, as chairman of the task group and the meeting, thanks the attendees for their enthusiasm and valuable contributions to the meeting and the task group's activities, and the local organizing committee for hosting the meeting in such an inspiring environment.

And, finally, Milan Snajder, task group member and part of the local organizing committee, surprises the attendees by some nice gifts to remember the workshop and the visit to the beautiful city of Prague.