Distributed Intelligence in Critical Infrastructures for Sustainable Power

ENK5-CT-2002-00673

Dependable ICT Support of Power Grid Operations

Rune Gustavsson, Per Mellstrand, and Björn Törnqvist

BTH

Hans Akkermans

EnerSearch AB

Identifier:	**BTH_DIS _WP24_002_04**
Date:	**2005-03-25**
Class:	**Deliverable**
Responsible Partner:	**BTH**
Annexes:	**None**
Distribution:	**EC RESTRICTED**
Overview:	**Security and dependability models supporting CRISP experiments of WP III. Background material can be found in D1.6 Information Security Models and their Economics**

The CRISP consortium consist of:

ECN	Principal Contractor & Coordinator	The Netherlands
ENECO	Principal Contractor	The Netherlands
INPGrenoble	Principal Contractor	France
Schneider Electric	Principal Contractor	France
EnerSearch AB	Principal Contractor	Sweden
Sydkraft AB	Principal Contractor	Sweden
Blekinge University of Technology	Principal Contractor	Sweden
ABB AB	Principal Contractor	Sweden

Control Versions:

Version	Date	Author	Description of Changes
Version 1	2004-07-10	Rune Gustavsson	Initial draft
Version 2	2004-07-20	Rune Gustavsson, Per Mellstrand, and Björn Törnqvist	Extensions of Scenarios
Version 3	2004-12-29	Rune Gustavsson, Per Mellstrand, Björn Törnqvist, and Hans Akkermans	Final draft
Version 4	2005-03-25	Rune Gustavsson, Per Mellstrand, Björn Törnqvist, and Hans Akkermans	Final version

Document Description

This deliverable D2.4 *Dependable ICT support 0f Power Grid Operations* is a link between deliverable D1.6 *Information security models and their economics* and planned activities in WP III, that is Experiments A, B, and, C. Background CRISP material includes deliverables of D2.2 *Design document and multi-agent simulation tool for distributed demand-supply matching* and D2.3 *Design document and simulation tool for diagnostics of high-DG power networks*. Furthermore there are links between this document and the deliverables D1.7 *Report on distributed network architectures* and D1.8 *Reports on case study simulations and results.*

In short, this document specifies and extends the general background on security models and dependability models of deliverable D1.6 with CRISP specific material of D2.2 and D2.4 towards the experiments and tests of WP III.

Table of content

Executive summary

This deliverable D2.4 *Dependable ICT support 0f Power Grid Operations* is a link between deliverable D1.6 *Information security models and their economics* and planned activities in WP III, that is, Implementation, Experiments and Tests. Background CRISP material includes deliverables of D2.2 *Design document and multi-agent simulation tool for distributed demand-supply matching* and D2.3 *Design document and simulation tool for diagnostics of high-DG power networks*. Furthermore there are links between this document and the deliverable D1.7 *Report on distributed network architectures* and D1.8 *Reports on case study simulations and results.* In short, this document specifies and extends the general background on security models and dependability models of deliverable D1.6 with CRISP specific material of D2.2 and D2.4 towards the experiments and tests of WP III.

The focus of this deliverable is on dependable ICT support of power grid operation. By recasting the three CRISP experiments into three Scenarios in Chapter 2 we claim that we have a good description of benefits and challenges related to future virtual utilities. Among the challenges are securing trustworthy operation from a technical operation side (avoid disturbances such as blackouts) as wee as from a user-centric business point of view (value added power related services).

Our investigation on proper means to safeguard operations of future virtual utilities begins with an assessment of lessons learned from recent (2003) big blackouts worldwide in Chapter 3. We propose an accident diagnosis and repair model (STAMP++ in Section 3.2) suitable for the complex socio-technical system we envisage for future cell-based virtual utilities (Figure 1-6).

From this analysis and the background material from deliverable D1.6 Information security models and their economics, we then reassess the dependability concerns related to the CRISP related scenarios of Chapter 2.

The deliverable provides some novel ideas and models that we claim are useful beyond the CRISP project. Having said that, there is much more work to be done along those the lines stated in the deliverable. A good start is the planned CRISP experiments.

1 Background and overview of the deliverable

This deliverable D2.4 *Dependable ICT support of Power Grid Operations* is a link between deliverable D1.6 *Information security models and their economics* and planned activities in WP III, that is, Implementation, Experiments and Tests. Background CRISP material includes deliverables of D2.2 *Design document and multi-agent simulation tool for distributed demand-supply matching* and D2.3 *Design document and simulation tool for diagnostics of high-DG power networks*. Furthermore there are links between this document and the deliverable D1.7 *Report on distributed network architectures* and D1.8 *Reports on case study simulations and results.* In short, this document specifies and extends the general background on security models and dependability models of deliverable D1.6 with CRISP specific material of D2.2 and D2.3 tailored towards the experiments and tests of WP III.

As we have discussed in Section 4 Information security and network security of D1.6 and in *Section 3 Assessments of dependability in critical infrastructures* of this deliverable, we need *operational definitions* of security and dependability in order to be able to improve those qualities. To that end we propose the following operational definitions:

- Security and dependability are *constraints* on the behaviour of the system components at each level.
- Accessibility is *control* of interactions with components at each system level.
- Sustainability is maintaining of *invariants* of the running system.
- System improvements are enabled by *failure models* and the derived feedback from analysis of accidents.

In this Chapter 1, we specify in Section 1.2 *Goals of the CRISP experiments.* The bottom line is to define safety and security concerns relevant to the CRISP experiments as such but also in the more general context of dependable future cell-based energy systems. Our approach to that end is to take into account the general architecture models and safety and security models of D1.6 *Information security models and their economics*, as well as background material from D2.2 *Design documents and multi-agent simulation tool for distributed demand-supply matching* and D2.3 *Design document and simulation tool for diagnostics of high-DG power networks* transformed into the contents of D3.1 *Experiments and test set-up specification and preparation*.

In *Chapter 2 Setting the scene – Architectures* we start by generalising the three CRISP experiments into three scenarios. Following that, we assess the given architectures of the experiments and define in *Figure 2-12 a reference architecture* for embedded ICT supporting critical infrastructures such as future virtual utilities. The following *Chapter 3 Assessments of dependability in critical infrastructures*, begins with an assessment of recent (2003) big power blackouts (Section 3.1). In *Section 3.2 Analysis of failure in socio-technical systems* we assess different failure models and their purposes. Our investigation leads to a proposal of a new comprehensive accident model to enable development and maintenance of sustainable dependable socio-technical systems such as future virtual utilities (*Section 3.2 The STAMP++ accident model*). Chapter 3 ends with a discussion of *critical dependencies*

and the relation to our scenarios (*Section 3.3 Critical interdependencies* and *Section 4.3 Dependability issues related to Scenario 3*).

Chapter 4 Dependability issues related to scenarios, identifies and summarises dependability concerns related to our three scenarios. Section 4.4 introduces a *layered security model in Figure 4-1*. This layered model emphasises a layered approach to security and dependability mechanisms. To that end we introduce in *Chapter 6 Tasks, protocols and dependable coordination* high-level *patterns* (coordination, tasks, dialogue models) for analysis and abstraction of algorithms related to the CRISP experiments. As a result we propose a *unified coordination model to increase dependability of critical infrastructures* (Section 6.4) and some pointers to dependable protocols (Sections 6.5 and 6.6). Chapter 5 describes different coordination models of the future virtual utility based on Chapter 2.

Critical issues related to implementations are brought forward in *Chapter 7 Dependable implementation*. We specifically address issues of safe execution environments of unreliable software. The deliverable ends with some *conclusions* (Chapter 8) and a list of *references*.

1.2 Goals of the CRISP experiments

The CRISP experiments of WP III are:

- Experiment A: Real-time monitoring and control of electricity supply and demand in a commercial setting to avoid short term market imbalance due to intermittent renewables.
- Experiment B: Fault detection and restoration in power system management.
- Experiment C: Intelligent load shedding as a mean to circumvent outages.

The following five figures (Figure 1-1, Figure 1-2, Figure 1-3, Figure 1-4, and Figure 1-5) from the experiments descriptions of Deliverable D3.1 *Experiments and test set-up specification and preparation*, capture the main characteristics of the experiments.

1.2.1 Experiment A

Goals of the experiment are:

- To demonstrate the ease of computerisation of a power network aggregation in a commercial setting spread over a large geographical region with common, off-the-shelf ICT-components.
- To demonstrate the ability to characterise and model the behaviour of the majority of installations in the experimental setting from an energetic perspective from a set of initially collected monitoring data. The operational effects on the Eneco installations will be assessed by simulation. In this way the way will be paved to a staged testing methodology of new control strategies in critical infrastructures without impairing or harming vital utility assets.

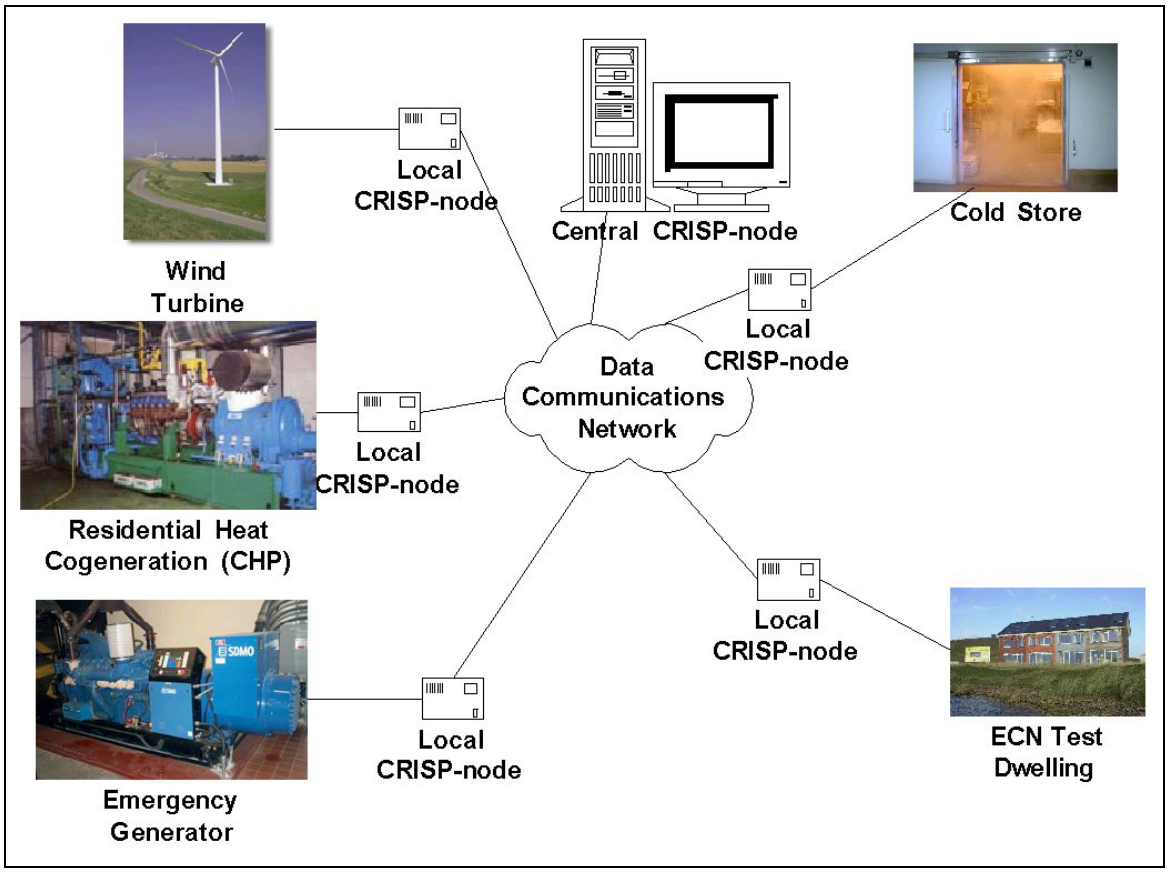


Figure 1-1 Loose coupling in the IRS-network in the field test configuration

- To demonstrate the added value of novel bottom-up control strategies, that use information, derived from monitoring real-time data, for diminishing the imbalance cost caused by the intermittence of DG-RES resources in a liberalised market situation.

- Providing a practical sample scenario to validate simulations in WP-2 of the CRISP-project. Within the CRISP Project ECN and Enersearch have developed novel mechanisms for matching demand and supply of electricity. These supply and demand matching (SDM) mechanisms are based on electronic markets, which are a form of distributed optimisation; in this case especially targeted at imbalance reduction.

The local process computers will be loosely coupled to the IRS-network according a topology as sketched in Figure 1-1. Apart from the ECN-test dwelling, only measured data will be transmitted from field process computers. The impact on operational characteristics of the other installations will be derived from installation models that are constructed and validated from the data measuring campaign in the initial phase of the project. The IRS-network of Crisp test-systems will be linked in a VPN.

1.2.2 Experiment B

The problem addressed is fault localization with DR or without DR as has been described in work package 1.4 of the CRISP project (see [crisp D1.4]). The application targeted is not yet solved conveniently in the existing network without DR: it takes time in general (minutes or hours) and entails numerous break of power for the feeder involved. While the main role of the distribution EPS is to supply loads, the break of power during short time period is not so expensive or problematic. But with the DG and the DG-RES distributed in these networks, and the constraints associated with brutal disconnection or slow and delayed reconnection ask for questions. Two axes are followed in the experiment proposed: study and make the fault diagnostic tool adapted to a high penetration of DG and DG-RES, and also boost the localization limiting the unsupplied area. This experiment is based on the possible application of a dedicated fault diagnostic tool including an important role to ICT and is defined in the work package 2.3 of the CRISP project (see [crisp D2.3]).

Goals of the experiment are:

The equipment associated with ICT used for the HTFD application will have to achieve conversion, calculation, and communication with remote equipments. The goal of the experiment is to check the ability of dedicated ICT to comply with the application in terms of calculation amount and time and in terms of information security aspects.

- The EPS timing process is evaluated for an expected realistic future situation based on the technical knowledge of existing electrical devices. The amount of calculation will be taken in coherency with existing capability of fault diagnosis real time devices (see in [crisp D1.3]).

The questions that the experiment is intended to answer are:

- *Estimates*: What is the ICT equipment that may comply with the application? What is the expected possible cost involved by the solution? What is the associated software requirement?

- *Challenges*: What are the difficulties to face when developing the solution proposed? What are the recommendations for ICT, HMI, EPS intermediate equipment (crossed constraints between existing EPS automatic processes and programming a flexible tool adapted to the main existing MV networks and easy-to-use by a network operator)?

- *Constraints*: What are the timing constraints of the different subtasks defined in the application due to existing limitation of the equipment and of the protocols? What are the possible future improvements?

- *Integration:* What is the possible real time response during such localization in order to integrate this sequence into the main protection sequences? (total time expected for localization, time remaining for decision making)

- *Dependability*: What are the different aspects of the information security in this application? How to characterize the interoperability, the reliability and non intrusion capacity of the information network? (problem caused by shared data between applications as topology data) (see [crisp D2.4])

The main interest of ICT is to use its powerful calculation and communication ability: the need is a large communication network/system between numerous and various equipments, and also a certain level of local analysis. The cost constraints are high in the MV (compared with the transmission system) and the ICT is expected to bring an adapted low cost solution in the long run. A first approach consists in simulating the EPS system and test the expected calculations. Then knowing the constraints of existing devices, and the ability of them to boost the information, a timing process and data preparation is defined to test the ICT devices and associated software.

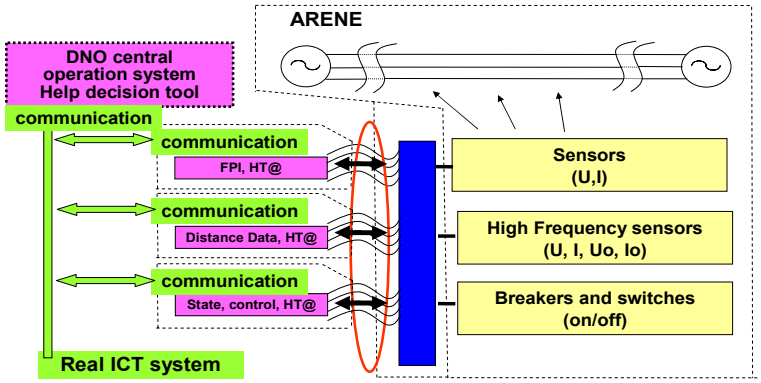


Figure 1-2 The architecture of the ARENE real time experiments

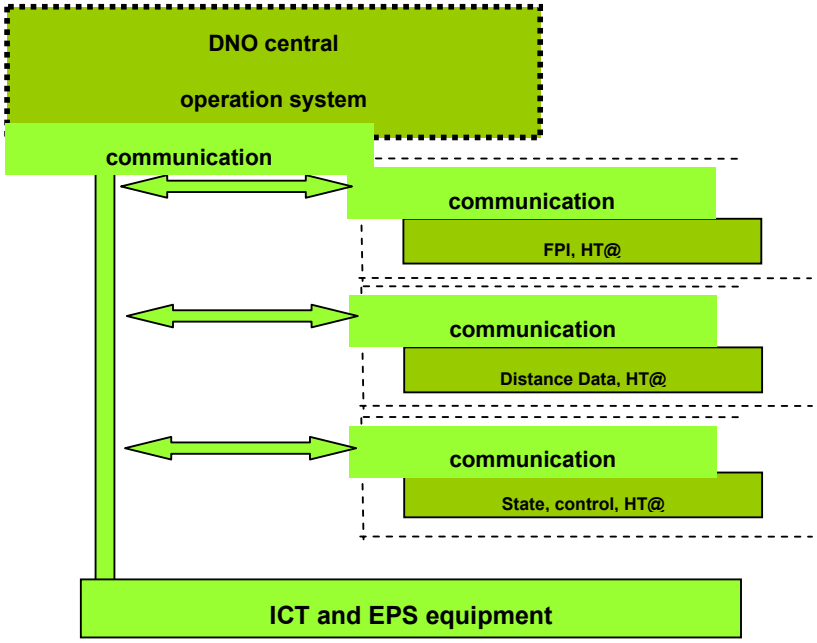


Figure 1-3 External ICT components and EPS to be tested

The experiment in open loop for the ICT is then followed to check their real time response and compare with the expected values (theory, simulation).

From the results reached, the closed loop will be initiated to study the possible interactions between the EPS real time signals, the ICT information resulting, and the feedback control to the EPS switches.

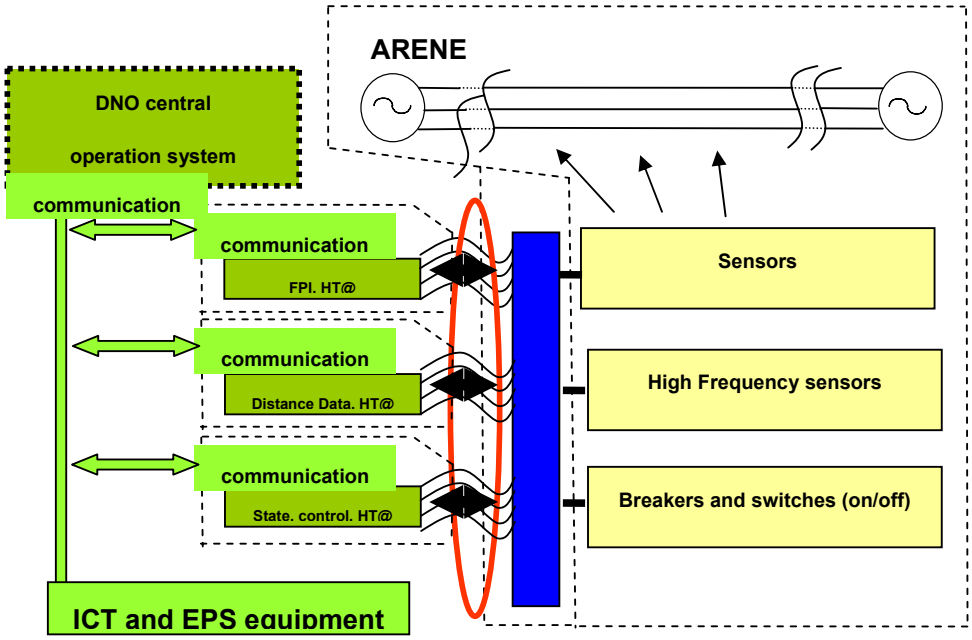


Figure 1-4 Expected closed loop experiments with Arene Real time

Experiments results will be based on three-phase and two-phase faults. The codes associated with the ICT components, physical link representation, TCP/IP description and setting of ports, management of the inputs/outputs of communication (in asynchronous mode) will be described in the experiments results.

Using Arene real time is not necessary for the main results of the experiment. The inputs for the experiment have been prepared with Arene (generation of comtrade files, results of simulation). Depending on the obtained results, the closed loop will be used in order to show the interest in a fast protection new application.

From exchanges with BTH and ECN in Amsterdam the 4th-5th of March 2004, the technical solution the most adapted to our application is using TCP/IP communication protocols. The ICT components are clearly identified in a level 1 cell to allow the operator a simple re-configuration process. In effect, we are developing a tool that will automatically detect a fault in the (cell-based) grid after triggering of the automatic protection system. Today the most of the fault detection and restoration is done manually *in situ*. The aim is to automate large parts of those processes. Crucial components to that end involve proper decision support systems and processes as well as adequate communication and control structures (Section 3.2). The first part of the experiment is to set up and evaluate performance parameters of a dedicate ICT structure based on adaptations of standard protocols.

1.2.3 Experiment C

The problem domain to be studied is related to intelligent load shedding, in systems with a large amount of dispersed generation. According to

How do the wind power mills react to different power system conditions, such as:

- “normal faults” in the mainland power system
- “normal faults” in the Öland system;
- large disturbances and power oscillations – if they occur during the recording period.

What is interaction between load and power supply system during stressed or faulty situations, such as:

- what is the load response to a supply voltage decrease;
- what is the system voltage response to a load change.

The problem is to identify load and system interaction to be able to design efficient and “intelligent” load shedding schemes, in systems with a high degree of distributed generation.

Goals of the experiment are:

1. For faults occuring on Öland, resulting in loss of load for about half a minute, the system response to a load relief can be investigated.
2. For faults on the main land, resulting in weakening of the supply source, the load response to a voltage decrease can be studied.
3. For all kind of disturbances, the response from the distributed wind power can be studied.

Approach

For the field test recordings phasor measurement units (PMUs) will be used. Recording of positive sequence voltage and current phasors will be made once every power system cycle, to a local PC with extended memory, for the whole recording period. The PMUs will be synchronized via the global positioning system (GPS). Since no communication is available the recordings will be put together after the finalization of the recording period.

Scenarios and strategies involved

A utility top level manager introduced the term “Intelligent load shedding” after a severe disturbance that led to a system blackout. To his knowledge a load shedding system was installed, to take care of severe, rare disturbances, in such a way that a system wide blackout should be avoided. The load shedding system had obviously failed and the reaction from the management was immediate and powerful – stating that “we need a more intelligent load shedding system”.

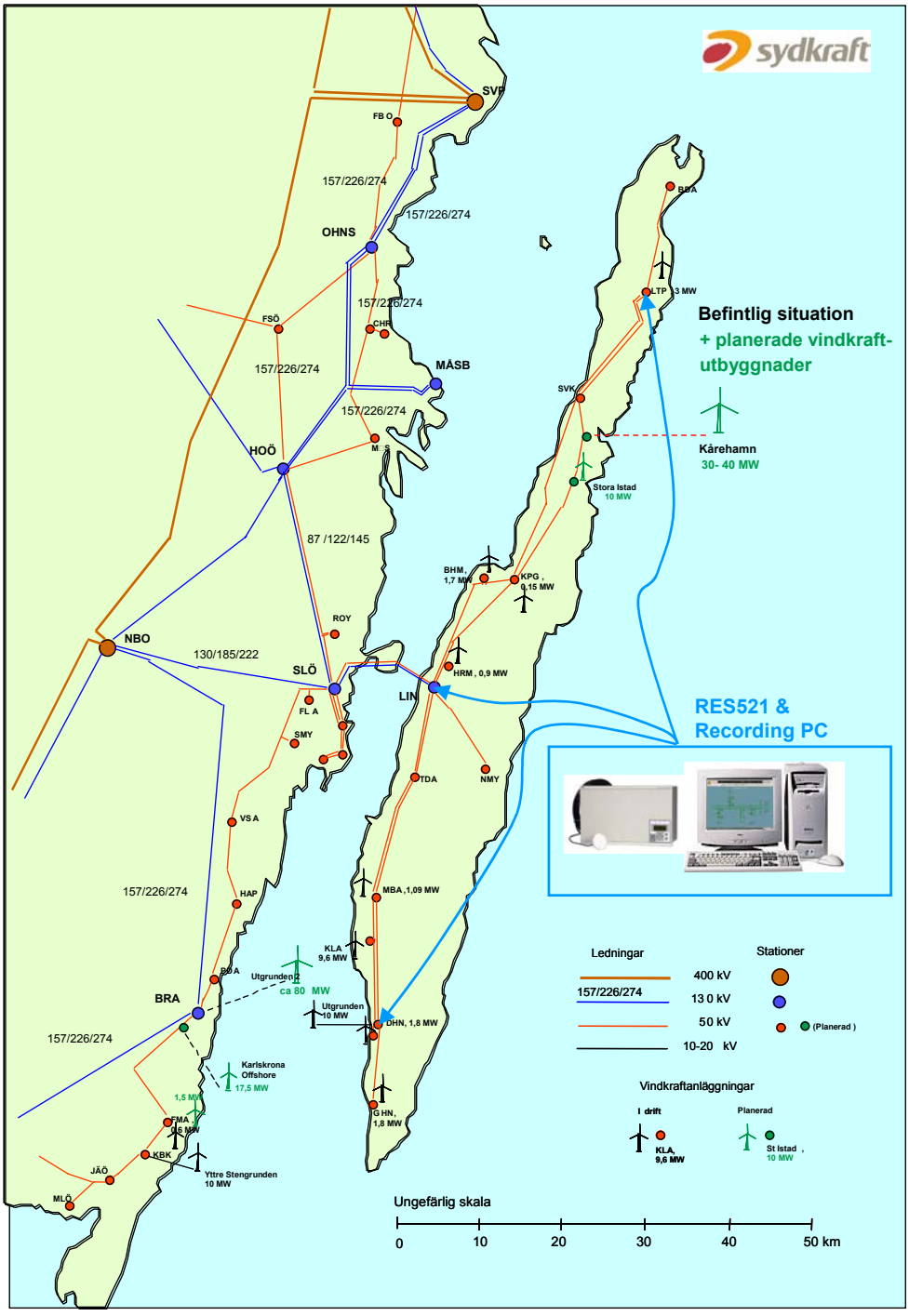


Figure 1-5 The test area of Öland, with wind power farms and recording nodes marked

Today we interpret “Intelligent load shedding” as:

1. a means to improve *power system stability*,
2. by providing *smooth load relief*,
3. in situations where the power system *otherwise would go unstable*.

The aim is clearly to improve power system stability, i.e. keep the bulk power or transmission system energised together with as much of the load as possible. The way to improve power system stability is by *smooth load relief*, which can be achieved in a number of ways. The situations to activate smooth load relief, is when the power system *otherwise would go unstable*. “Intelligent load shedding” thus deals with:

1. Detecting situations that will go unstable if no remedial actions are taken, and
2. to take proper action in such a way that stability is restored by minimum cost load shedding.

Algorithms

Load shedding can be made either on a contractual base, with respect to the statements in the contract, between the power supplier (or grid operator) and the customer, or as an emergency non-discriminative action to save the system. Load shedding can be caused by long term problems, such as energy shortage or generation capacity shortage (peak-shaving), or by short term problems, such as imbalance between generation and load (frequency problems), power oscillations or voltage instability (network problems).

The problem of identifying the load to shed for a certain disturbance is related to a number of parameters, such as:

1. the power system geographical problem area,
2. the severity of the problem,
3. the time available to take proper actions,
4. the load shedding infrastructure and preparation, and
5. the cost for the load shedding.

1.2 The CRISP experiments from a security and dependability point of view

From the overview of the CRISP experiments given in the previous section it follows that we are addressing potential external and internal threats against the (future) socio-technical virtual utility system of systems. The following Figure 1-6 captures a systems perspective of the situation.

In Figure 1-6 we emphasise the main components of a socio-technical system (e.g. future virtual utilities) as well as the dynamics of the system related to coordination (e.g., interactions of components and constraints of their behaviour) to meet the requirements of the business processes (e.g., the systemic goals of the expressed in terms of value chains and input – output relations). Furthermore, we indicate that the threats to the systems can be expressed as internal or external given the boundaries of the system as such. We will return to this figure in *Chapter 3 Assessment of dependability in critical infrastructures*.

Classically dependability issues have been related to the Quality of Service (QoS) of the technical system in Figure 1-6. Whereas security has two intended classical meanings. One meaning is related to the operations of the technical system (should not harm people in maintenance or operations), the other meaning is related to ICT security issues (c.f. D1.6).

The most held present position by people working with security and dependability of systems of systems is that *dependability* could be a common denominator for security and dependability concerns of socio-technical systems as expressed in Figure 1-6. We support that strand in general, but sometimes it is more convenient to express concerns as security concerns or dependability concerns where the context gives the intended reading.

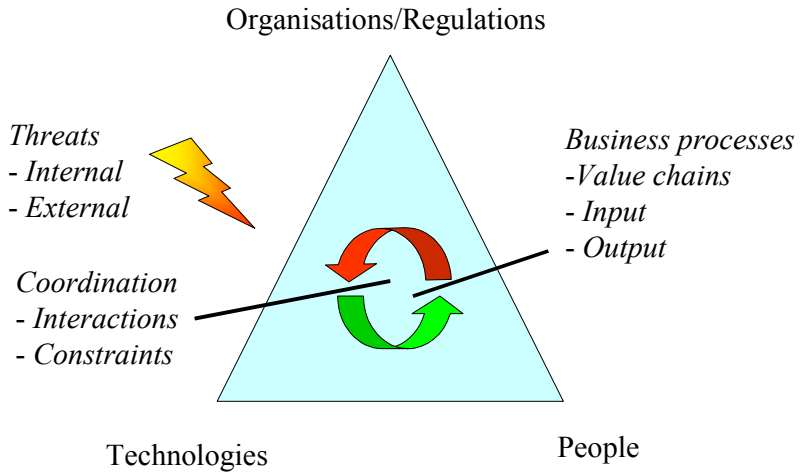


Figure 1-6 The main components of dependability and security assessments of socio-technical systems

In Figure 1-6 we emphasise the main components of a socio-technical system (e.g. future virtual utilities) as well as the dynamics of the system related to coordination (e.g., interactions of components and constraints of their behaviour) to meet the requirements of the business processes (e.g., the systemic goals of the expressed in terms of value chains and input – output relations). Furthermore, we indicate that the threats to the systems can be expressed as internal or external given the boundaries of the system as such. We will return to this figure in *Chapter 3 Assessment of dependability in critical infrastructures*.

Classically dependability issues have been related to the Quality of Service (QoS) of the technical system in Figure 1-6. Whereas security has two intended classical meanings. One meaning is related to the operations of the technical system (should not harm people in maintenance or operations), the other meaning is related to ICT security issues (c.f. D1.6).

The most held present position by people working with security and dependability of systems of systems is that *dependability* could be a common denominator for security and dependability concerns of socio-technical systems as expressed in Figure 1-6. We support that strand in general, but sometimes it is more convenient to express concerns as security concerns or dependability concerns where the context gives the intended reading.

Assessments of the three experiments of CRISP gives that experiments B and C have clear *technical system dependability* focus (e.g., ICT supported fault detection and intelligent load shedding). Experiment A has a more clear focus on supporting *business processes* in future virtual utilities (e.g., supply – demand matching). However, the ELECTRA game by ECN designed to assess the business processes related to Experiment A has clearly demonstrated the involving actors concerns related to *information security* (e.g., the non-confidentiality, integrity, availability and denial of service described in D1.6). These information security concerns are related to the regulations related to the organisations involved as well as with the rules of engagement and behaviour of the people involved (Figure 1-6).

In short, the CRISP experiment A-C clearly demonstrated that the dependability and security model of Figure 1–6 is viable both for the CRISP setting as such but more importantly for describing and assessing the trustworthiness of future virtual utilities. In *Section 3.2 The STAMP++ accident model* we will describes a *dependability assessment model* based on Figure 1-6.

As we have discussed in Section 4 Information security and network security of D1.6 and in *Section 3 Assessments of dependability in critical infrastructures* of this deliverable, we need *operational definitions* of security and dependability in order to be able to improve those qualities. To that end we propose the following operational definitions based on Figure 1-6:

- Security and dependability are *constraints* on the behaviour of the system components at each level.
- Accessibility is *control* of interactions with components at each system level.
- Sustainability is maintaining of *invariants* of the running system.
- System improvements are enabled by *failure models* and the derived feedback from analysis of accidents.

2 Setting the scene - Architectures

The CRIS Experiments A, B, and C described in previous section can be architecturally summarised by the following Figure 2-1.

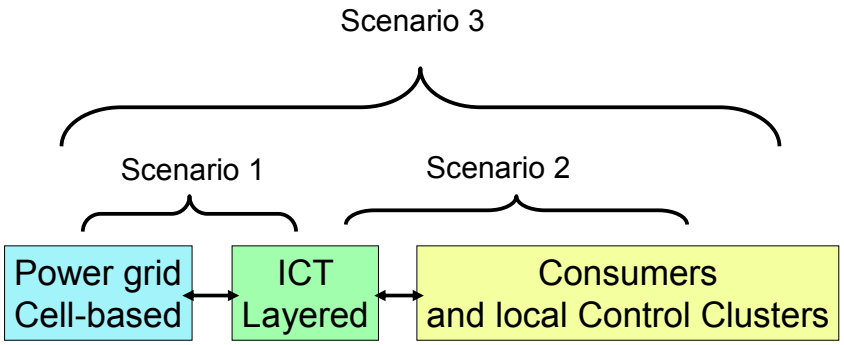


Figure 2-1. The main components of WP 2.4 and interactions with WP 2.2, WP 2.3, and WP3.

Figure 1-3 depicts the three scenarios that are in focus of this deliverable D2.4 *Dependable ICT support 0f Power Grid Operations*. The three scenarios of Figure 2.1 are related to the CRISP experiments as follows:

- Scenario 1. Dependable operations of cell-based power grids incorporate and extend the experiments B and C.
- Scenario 2. Dependable customer- centric business operations based on local control clusters incorporate and extends experiment A.
- Scenario 3. Trustworthy and dependant operations of future virtual grids combines and extends the efforts of all CRISP experiments.

The main issues addressed by the three scenarios are as follow (C.f., discussion related to the experiments in Section 1.3).

2.1 Scenario 1 - ICT support of cell based power nets

This first scenario denotes the ICT support of cell based power nets and is derived from the Experiments B and C where the focus is on dependable support of power grid operations including *Fault finding* and *Restoration* and *Intelligent load shedding* of cell based grids.

Specific dependability issues:

- Performance of ICT support in cell-based power nets.
- Dependability related to restoration. Fault finding and restoration in cell based power nets.
- Performance and choice of algorithms supporting intelligent load shedding.

2.2 Scenario 2 – ICT support of business models based on demand-supply matching

This scenario focus on issues brought forward in Experiment A that is, business models based on computational markets of distributed Demand-Supply matching.

Specific dependability issues:

- Trusted market mechanisms.
- Trust issues related to markets, e.g., information management.

2.3 Scenario 3 – Dependencies between power related networks

In this scenario the focus is on dependencies of the two networks studied separately in Scenario 1 and Scenario 2. There are obviously several conflict situations that might arise and to be resolved by implementation of high-level policies. Scenario 3 is the main focus of this deliverable (Section 3.3 and Section 4.3).

Specific dependability issues:

- Monitoring of system states
- Coordination between the market oriented models and the power net stability models. (Section 4.4)

2.4 Architectures

2.4.1 Architectures related to the experiments

Experiment A:

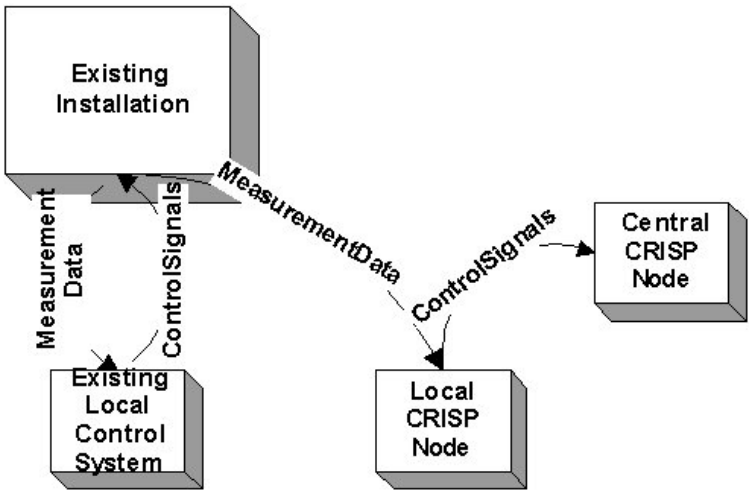


Figure 2-2 Site computer configuration

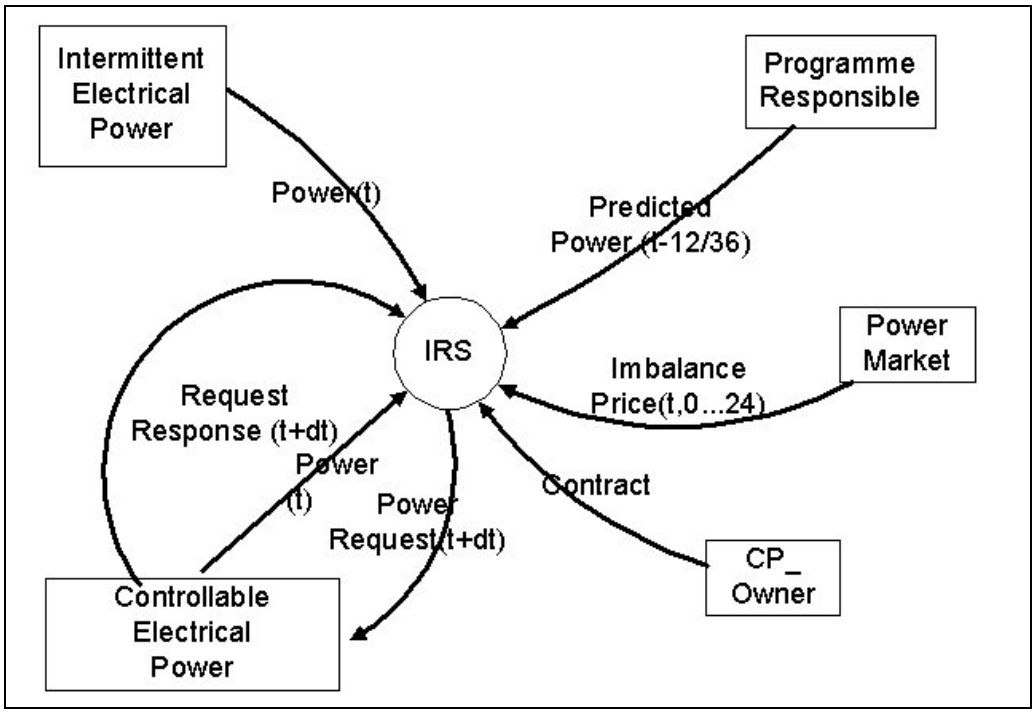


Figure 2-3 Context diagram IRS

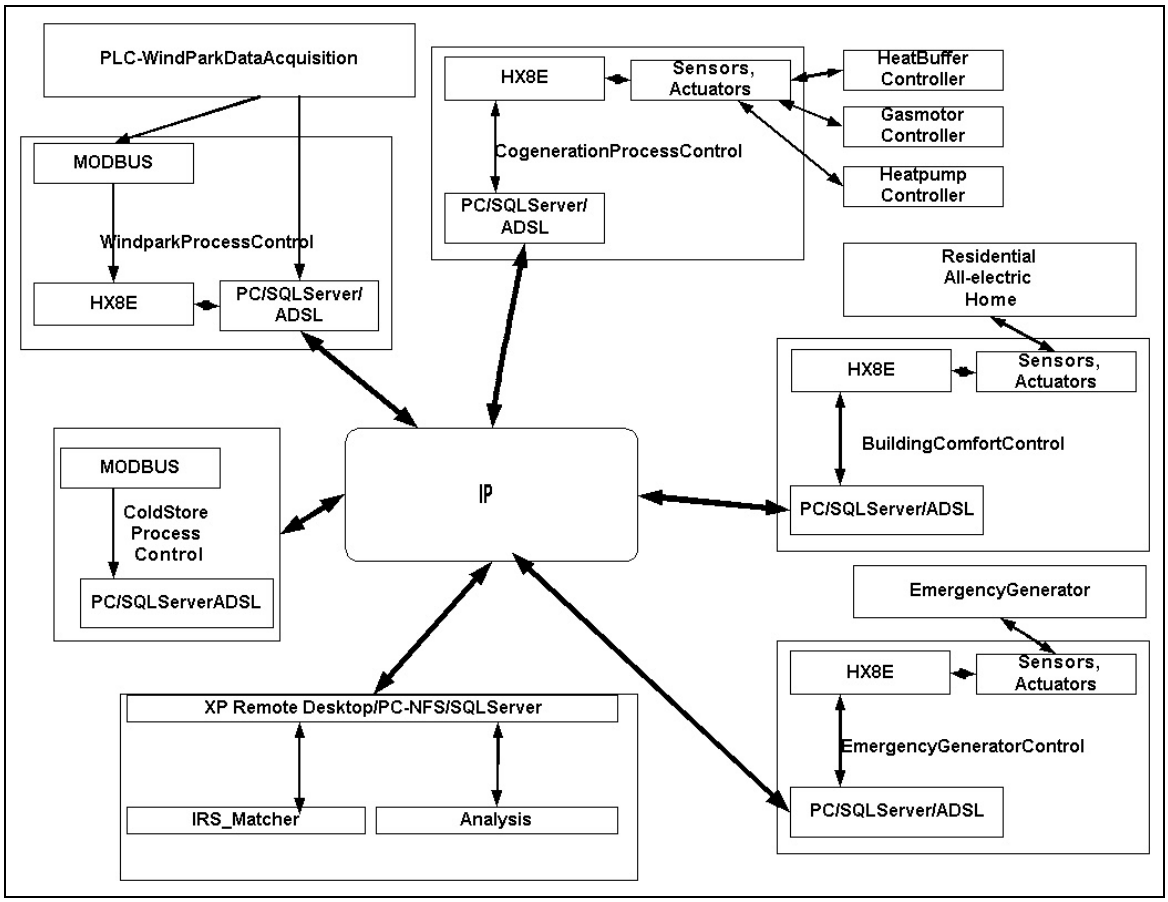


Figure 2-6 Hardware architecture IRS

Experiment B and Experiment C:

The main architectural components of Experiment B are the architecture for fault finding and restoration (Figure 2-5) and the cell-based architecture for the future utility (Figure 2-5) Figure 2.7 is supplemented with Figure 2-6 introducing a phasor measurement set-up that is related to experiment C.

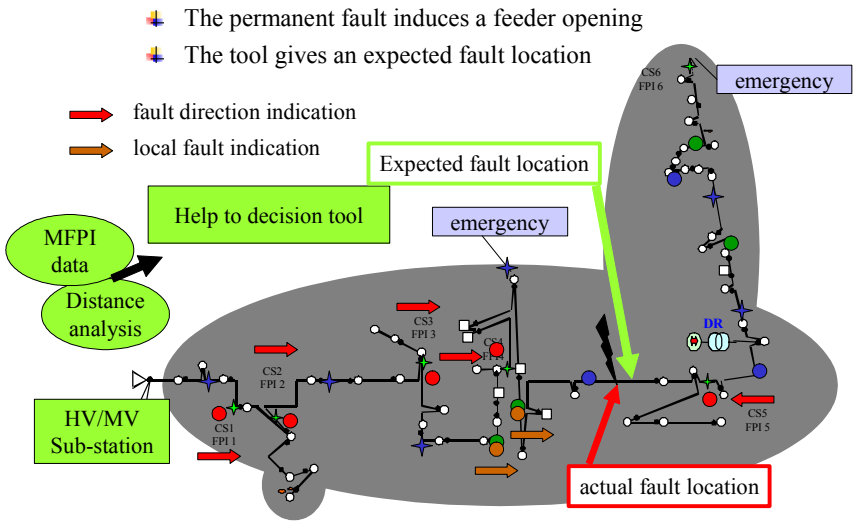
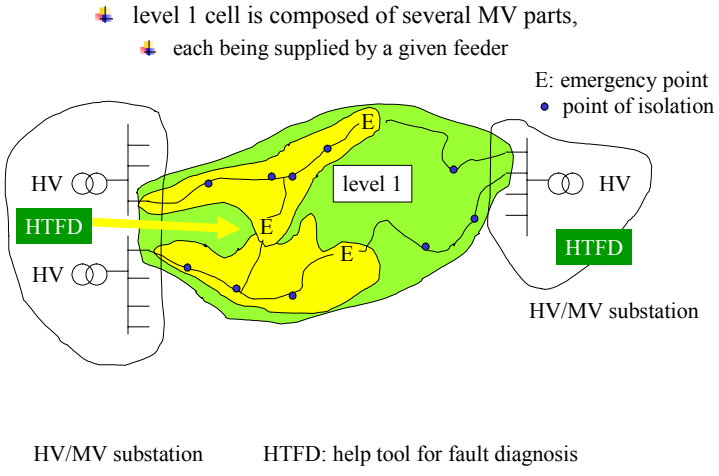


Figure 2-5 Principle for fault detection and restoration in Experiment B



HV/MV substation HTFD: help tool for fault diagnosis

Figure 2-6 Aspects of a MV cell architecture of Experiment B

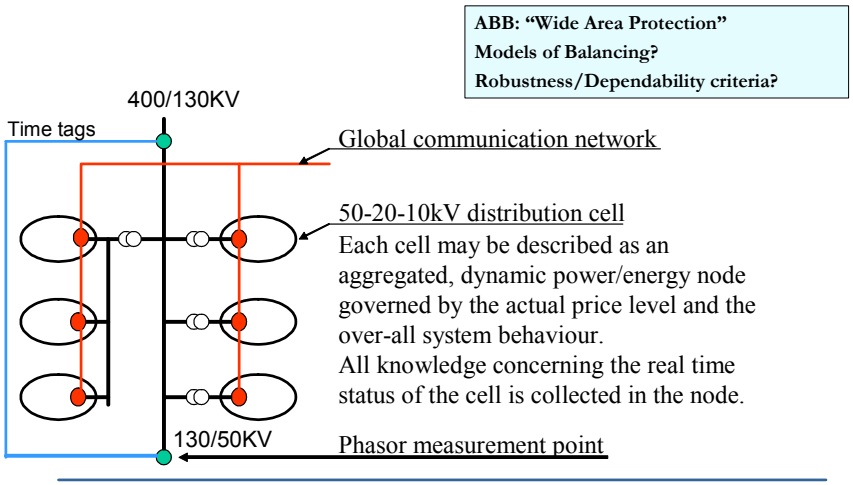


Figure 2-7 Balancing networks of cells in a virtual utility

2.4.2 Generic architectures supporting CRISP experiments

We introduced in D1.6 *Information security models and their economics* some generic architectures supporting security and dependability. From *Section 4 Information security and network security* we include the following figure:

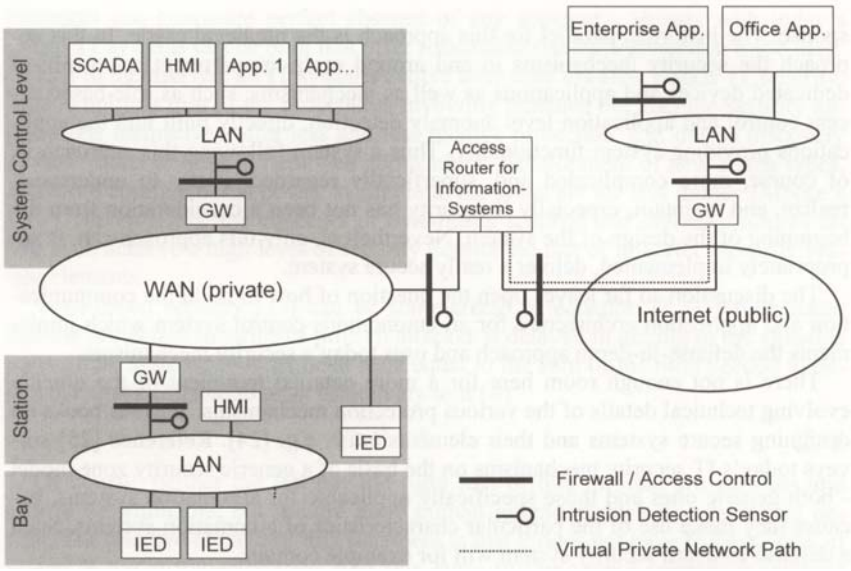


Figure 2-8. Architecture of a security model related to energy based information

From Figure 2-1, however, we observe that virtual utilities typically are characterised by being open, flexible and scalable. A generic architecture supporting the CRISP experiments as well as future virtual utilities is thus middle-ware oriented in the meaning of contemporary approaches in GRID computing and Service-Oriented Computing (c.f., Section 1 Introduction of D1.6).

A traditional IT architecture supporting single client-server applications is given in Figure 2-9.

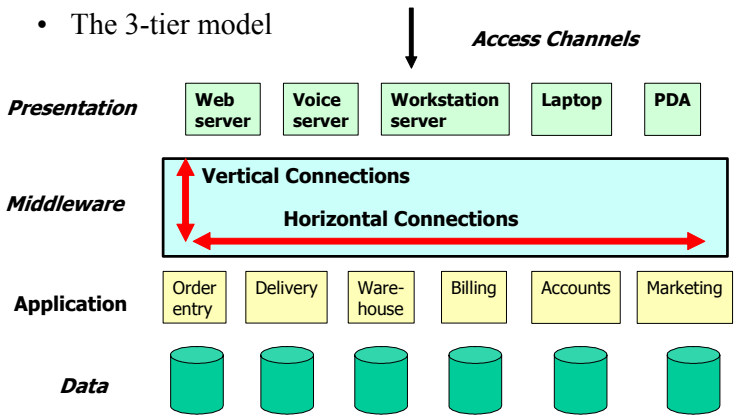


Figure 2-9 A typical client-server ICT architecture

However, this information system is not adequate for our distributed *embedded* ICT support systems of future utilities (c.f., Figure 2-5 to Figure 2-8). The following two figures gives a set (depending of levels of sophistication) of appropriate ICT architectures to our disposal.

The Figure 2-10 illustrates a generic Service Oriented Architecture (SOA) that complements and extends the ICT part of the architecture of Section 1 Introduction of D1.6.

The following Figure 2-11 is a reference architecture of future ICT in advanced applications related to Network-Enabled Capabilities and Mission support (C.f., Section 5 Layered architectures of D1.6)

We conclude our discussion of architecture with Figure 2-12 of embedded ICT architectures in management, maintenance, and support of ICT in critical infrastructures (e.g., future virtual utilities).

- Basic services from the support middleware includes
  - Publication, discovery, selection, and binding of services.
  - *Composition support*; coordination, conformance, monitoring, QoS.
  - *Support for Service Level Agreements (SLA)* of services.

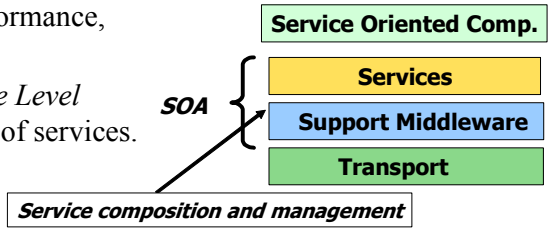


Figure 2-10 Structure and basic middle-ware services of a Service Oriented Architecture

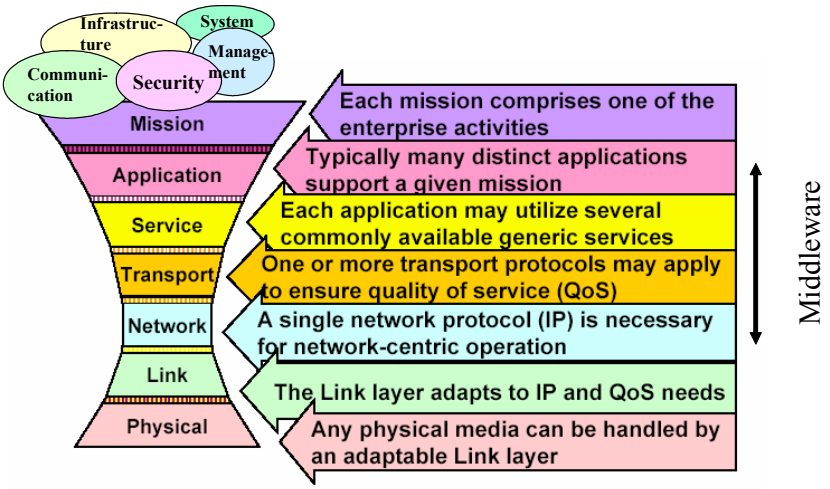


Figure 2-11 Generic ICT reference model for Network-Enabled Capabilities (NEC)

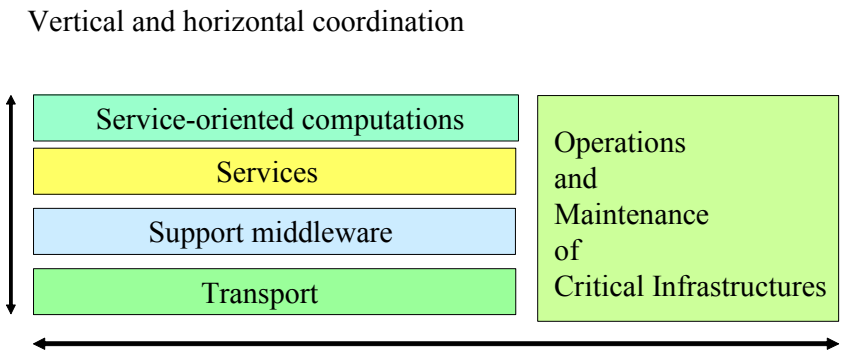


Figure 2-12 A reference architecture for embedded ICT supporting critical infrastructures

3 Assessments of dependability in critical infrastructures

In a very true sense, we can design and implement dependable systems as good as we can *identify*, *assess and remedy the reasons and causes behind system failures*. Recent power blackouts illustrate both the need and focus of assessments of catastrophic failures in critical infrastructures. This section gives a short overview of some of the blackouts of power systems in 2004 and some lessons learned in Section 3.1 *Assessments of recent blackouts*.

In Section 3.2 we give a short background to analysis methods of system failures and their shortcomings in analysing break-downs of complex systems as those we envisage in future critical infrastructures as exemplified by the CRISP experiments.

Our findings of Sections 3.2 and 3.3 leads us to introduce, in Section 3.3, an extension of a System-Theoretic Accident Model and Processes – STAMP developed by NASA to investigate space shuttle failures.

The Chapter ends with a short summary of dependability concerns related to the CRISP experiments in the light of the reasoning in earlier sections.

3.1 Assessments of recent blackouts

During 2003 at least three major power-system blackouts happened in August-September, that is the blackouts in US – Canada, Italy, and Sweden – Denmark. The causes of these catastrophic events (not the least in economic terms) have been reported in several fora including the reports [a] [b] [c] (c.f., discussion in D1.6).

From the US-Canada report of the 2003-08-14 disaster we find the following listing of causes to the disaster. The listing is coupled to recommendations by the investigators of actions to remedy some of the shortcomings found.

The listing includes the following headings:

Institutional issues related to the disaster

- Insufficient investments
- Lack of training of personnel
- Insufficient maintenance
- Non-functioning procedures of operation

Need of standardisations

- Establishment of *enforceable* standards

Need of technological improvements

- Specifically the role of ICT- Power Grid Management
- Dependable software

The issue of dependable software and operator understanding of operation was the root cause of the US-Canada blackout. In fact, a vulnerability in a General Electric Management System XA/21 (a SCADA system) at FirstEnergy Corporation, Ohio triggered the event. The primary cause was a software bug (software vulnerability) creating a "race condition" in the interactions between modules of a subsystem. This kind of software vulnerabilities are well-known problems in distributed computing. However, the affected software contained more than 3 million lines of code and had run without problems in more than 1 million operation hours before the critical event happened. The triggering of the critical event had a "window" of a few seconds and the event become disastrous when the operator did not know the ramifications of the software failure (updating of system states) but closed down the system and went to lunch.

As a matter of fact the *Blaster IT – worm* have impacted operations of nuclear power plants at FirstEnergy before 2003-08-14. The possible causalities between that event and the later "software –bug" event are not fully understood at this point in time. Anyhow, the Blackout itself clearly shows the critical interdependencies between power grids and ICT.

The reports of the black-outs in Italy and Sweden-Denmark confirms the general picture of the investigation report from the US-Canada case. A complementary information is that the societal cost for the Swedish – Denmark blackout (that lasted just for a few hours) has been estimated to 410 MEUR (4 000 MSEK). Furthermore, that event happened during the CRISP measurement test at Öland by Sydkraft and ABB. The measurements and evaluations assuch have gained international impact and publication.The US-Canada blackout in 2003 is the worst case of that kind so far with over *50 million people* affected and where the power breakdown also affected other critical infrastructures of the society. Among those were *the crounding of over 5000 information networks* due to lack of electric power [7].

The event clearly illustrated the increasing vulnerability to our society due to critical interdependencies of critical infrastructures. As such, this kind of vulnerability has gained international concerns during the last years. There are several ongoing international ongoing efforts and projects on investigations of critical dependencies of infrastructures. The upcoming EC Seventh Frame Program (EC 7FP) will to a large extent address programs in this area. Some references are given at the end of this deliverable.

The following Figure 3-1 illustrates some of the complexities and difficulties associated with the increasing dependabilities [22] [23]. Furthermore, we see that Figure 1-6 adequately captures the situation of the blackouts mentioned above [1]. We will return to this observation in the following sections Section 3.2 *Analysis of failure in socio-technical systems* and *A Systems-Theoretical Accident Model and Processes – STAMP++.*

In summary: The topics addressed by CRISP will be in focus of large international efforts inthe near future. Critical interdependencies between Energy Management System and Business Management System interconnected by ICT are already a manifested fact!

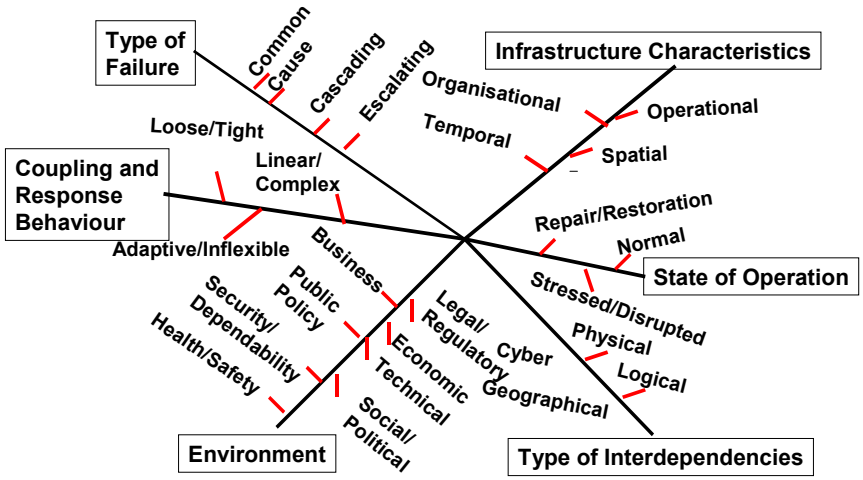


Figure 3-1 Aspects of critical interdependencies

3.2 Analysis of failure in socio-technical systems

In the following section we give a short account of the linear cause-effect analysis of system failures and its limitations. The following section describes a novel analysis model of system failures that is very promising in dealing with failures of complex socio-technical systems that reflects the challenges of the CRISP experiments and their generalisations towards cell-based virtual utilities (Chapter 2).

3.2.1 Linear cause-effect analysis of system failures

It is a well-known fact that the development of robust and sustainable artefacts is closely interlinked with the development of methods and techniques supporting diagnosis and remedies of faults and procedures. Development of tools to design and produce artefacts before the industrial revolution had only to take into account the *physical constraints* of artefact compositions in diagnosis of failures and breakdowns.

The industrial revolution brought forward technologies to create, transform and transport energy to places where it was needed (the steam and combustion engines, electric generation transportation and usage, telecommunication, and the modern transportation systems). Systems of that era were built to respect not only physical constraints but also *electromechanical constraints*.

Typical diagnosis methods for system failures built to conform to physical and/or electro-mechanical constraints have been based on a *linear cause-effect model* (cause-effect *chains* or *trees*).

An underlying assumption of these accident models is that there are common patterns in accidents. By defining those assumed patterns, *accident models may act as a filter and bias* towards considering only certain events and conditions or they may expand considerations with factors often omitted. The completeness and accuracy of the model for the type of system being considered will be critical in how effective the engineering approaches based on it are.

The selection of events to include in an event chain is dependant on the stopping rule used to determine how far back in the sequence of explanatory events goes. It is common to isolate one or more events or conditions (usually at the beginning of the chain) and call them the *cause* or the proximate, direct, or *root* cause of an accident or incident.

Usually, a root cause selected from a chain of events has one or more of the following characteristics:

1. It represents a type of event that is familiar, and, thus easily acceptable as an explanation for the accident.

2. It is a deviation from the standard behaviour.

3. It is the first event in the backward chain for which a “cure” is known.

4. It is politically acceptable as the identified cause of the faiure.

The backward chaining may also stop because the causal path disappears due to lack of information. Typically due to continuing the backtracking “through” a human a root cause is identified as “a human error”. Identifying accident causes in this way can be a hindrance in learning and preventing future accidents.

Event chain models rest on traditional *analytic reduction*: Physical systems are decomposed into separate physical components so that the parts can be examined separately, and behaviour is decomposed into events over time. This decomposition assumes that such separation is feasible, that is, each component or subsystem operates independently and analysis results are not distorted when the components are considered separately. This assumption in turn implies:

1. The components or events are not subjected to *feedback loops and non-linear interactions*.

2. The behaviour of the components is the same when examined alone as when they are playing their part in the whole.

3. The principles governing the assembly of the components as a whole are straightforward, that is, the interactions among the subsystems are simple enough that they can be considered separate from the behaviour of the subsystems themselves.

Clearly, those assumptions are not longer valid in complex systems where software is vital components (the only constraints being logical) and/or when we have to assess accidents in complex socio-technical systems such as those depicted in Figure 1-6 and/or Figure 3-1.

An excellent recent state-of-the-art report on accident analysis based on causal tree models is given in [2]. In short: Causal tree accident models aims to find a cause of a failure (that is,

in terms of events or errors under the assumptions 1 - 3 above). We would like to investigate accidents in more complex systems and *reason* about the ramifications (that is, *why* the events and errors occurred, Figure 1-6). To that end, we will build on recent ideas and models put forward by NASA in investigating accident in their space programmes.

In the following section we will take a closer look at the limitations of causal chain models of accident. Inspirations comes from the recent work by Nancy Levenson [16] and discussions by Siewiorik et. al. [26].

3.2.2 A Systemic approach of analysis of system failures

While there have been attempts to extend traditional safety engineering such as fault tree analysis and probabilistic risk assessment, based on event-chain models of accidents to software intensive systems, the results have not been terribly successful. Perhaps the lack of significant progress in dealing with software in safety-critical systems is the results of inappropriately attempting to extend the techniques that were successful in simpler electromechanical systems based on models of causation that no longer apply (assumptions 1 – 3 of Section 3.2.1).

Accidents can be separated into three types: (1) those caused by failures of individual components, (2) those caused by *dysfunctional interactions* between non-failed components, and (3) a mixture of both. The dysfunctional behaviour in modern high-tech systems orginate in software and/or operators (Section 3.1 Assessments of recent blackouts). In most software-related accidents investigated by NASA, the software operates exactly as specified. However, the software, following its requirements, commands component behaviour that violates system safety constraints or the software design contribute to unsafe behaviour by human operators. As such, the traditional event-chain model, with its emphasis on component failure, is inappropriate for today's software-intensive, complex human machine systems with distributed decision-making across both physical and organisational as well as logical boundaries (Figure 1-6).

A new model, called STAMP (Systems-Theoretical Accident Model and Processes) is proposed by Levenson [16] uses a systems-theoretic approach to understanding accident causation. Systems theory allows more complex relationships between events to be considered (e.g, *feedback and other indirect relationships*) and also provides a way to look more deeply into *why* the event occurred [4] [5] [20].

Accident models based on systems theory consider accidents as arising from the interactions among system components and usually do not specify single causal variables or factors (components). A systems approach to safety takes a broad view by focusing on what was *wrong* with the system's design or operations that allowed the accident to take place.

3.2.3 Safety as an emergent system property

The three fundamental assumptions (Section 3.2.1) behind the event-chain accident model are reasonable for many properties and systems, but they start to fall apart in complex systems.

The systems approach focus on systems taken as a whole, not on the parts examined

separately. It assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facet relating the social to the technical aspects [5]. These system properties derive from the relationships between the parts of systems: how the parts interact and fit together. While components may be constructed in a modular fashion, the original analysis and decomposition must be performed top down.

The foundation of systems theory rests on two pairs of ideas:

1. *Emergency* and *hierarchy*.

2. *Communication* and *control*.

A general model of complex systems can be expressed in terms of *hierarchy levels* of organisation, each more complex than the one below. A level is characterised by having *emergent* properties. Emergent properties do not exist at lower levels; they are meaningless in the language appropriate at those levels.

Safety is an emergent property of systems. Determining whether a plant is acceptable safe is not possible by examining a single valve in the plant. In fact, statements about the “safety of the valve” without information about the context in which that valve is used are meaningless. We can, however, make conclusions about the reliability of the valve, where reliability is defined as “the ability of a system or component to perform its required function under stated conditions for a specific period of time”. This is one of the basic definitions separating safety and reliability. Attempts to assign safety levels to software components in isolation from particular use, as is currently the approach in some international safety standardisation efforts, are thus misguided. For a similar reasoning of trustworthiness of systems, we refer to our own work in [24].

Emergent properties associated with a set of components at one level in a hierarchy are related to *constraints upon the degree of freedom* of those components. In a systems-theoretic view of safety (or other emergent properties) are *controlled or enforced* by a set of safety constraints related to the behaviour of the system component. Safety constraints specify those relationships among system variables or components that constitute the non-hazardous or safe system states we want to maintain.

An example: An aspect of Safety in the system depicted in Figure 1-6 can be expressed as the following constraints of the behaviour of the basic system components in the vertices of the triangle.

- The behaviour of the system should be restricted by the rules of operation and the training by operation personnel and the sensor and actuator information in such a way that the system does not enter a state of blackout.

Accidents result from interactions among system components that violate these constraints – in other words, from a lack of appropriate constraints on system behaviour.

We have now to introduce the means of *enforcement of constraints*, that is, communication and control. Regulatory or *control* actions is the enforcement of constraints upon the activity at one level of a hierarchy, which defines the “level of behaviour” at that level yielding activity meaningful a higher level. Hierarchies are characterised by control processes operating at the interfaces between levels [5].

Control in open systems (those that have inputs and outputs from the environment) implies the need for *communication*.

In system theory, open systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. A system is not treated as a static design, but as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. To be safe, the original design must not only enforce appropriate constraints on the behaviour to ensure safe operations (the enforcement of the safety constraints), but must continue to operate safely as changes and adaptations occur over time. In other words, the set of constrains should be *sustainable invariants* to ensure proper behaviour.

3.1.2 System-theoretic approaches to safety – The STAMP model

In response to the limitations of the event-chain models (Section 3.2.1), system theory has been proposed as an appropriate approach towards models of accident causation. When using a system-theoretic accident model, accidents are viewed as the result of *flawed processes* involving interactions among system components, including people, societal and organisational structures, engineering activities, and the physical system (Figure 1-6 and Section 3.1 Assessments of recent blackouts).

Leveson has identified an accident model called STAMP (System-Theoretic Accident Modelling and Processes) [16]. In STAMP, accidents are conceived as resulting *not* from component failures, *but* from inadequate control or enforcement of safety-related constraints on the design, development, and operation of the system. In the Space Shuttle *Challenger* accident, for example, the O-rings did not adequately control propellant gas release by sealing a tiny gap in the field joint. In the MARS Polar Lander loss, the software did not adequately control the descent speed of the spacecraft – it misinterpreted noise from a Hall effect sensor as an indication that the spacecraft had reached the surface of the planet. The Milstar-3 satellite launch in February 1999 is perhaps the most unmanned losses in the history of Cape Canaveral launch operations ($ 1 233 million). The failed process did not detect and correct a human error in the manual entry of a roll rate filter constant. The value entered should have been –1,992476, but was entered as –0,1993476.

Accidents such as these, involving engineering design errors, or human process errors, may in turn stem from inadequately control of the development process, i.e., risk is not adequately managed (through communication or feedback) in the design, implementation, and manufacturing processes. Control is also imposed by the management functions in an organisation – the Challenger accident involved inadequate controls in the launch-decision process, for example – and by the social and political systems within which the organisation exist (Figure 1-6).

STAMP is constructed from these basic concepts: constraints, hierarchical levels of control, and process models. These concepts, in turn, give rise to a *classification of control flaws* that can lead to accidents.

Constraints

The most basic concept in STAMP is not an event, but a constraint. Safety-related constraints specify those relationships among system variables that constitute the non-hazardous or safe system states. The control processes that enforce these constraints must

limit system behaviour to the safe changes and adaptations implied by the constraints.

Hierarchical levels of control

A second basic concept of STAMP (and in systems theory) is hierarchical levels of control. Figure 3-2 shows a (NASA inspired) simplified generic hierarchical safety control model.

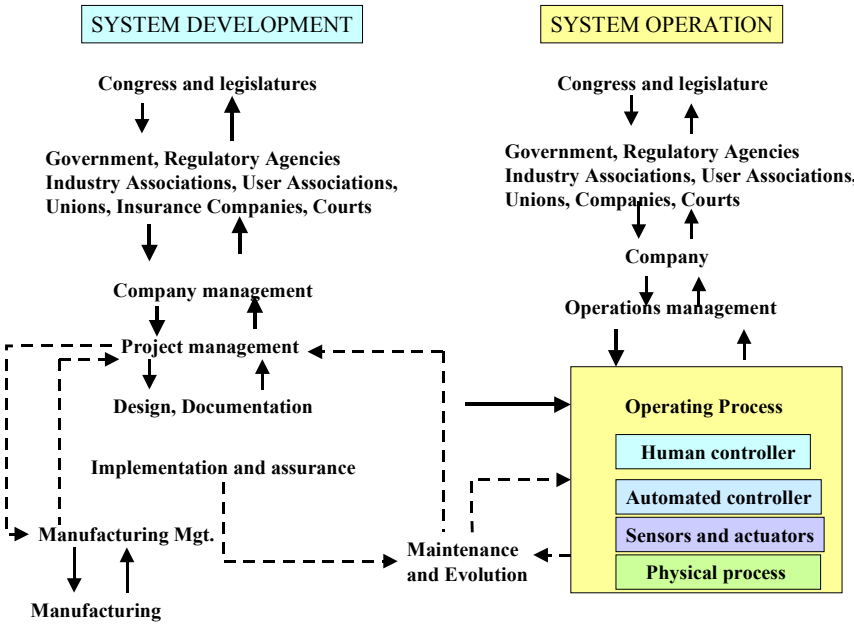


Figure 3-2 General form of a model of socio-technical control

The model in Figure 3-2 has *two basic hierarchical control structures* - one for system development (on the left) and one for system operation (on the right) – with interactions between them. Safety must be designed into a system. The *link* between the two process chains is Maintenance and Evolution. Manufacturers must communicate to their customers the assumptions about the operational environment upon which the safety analysis was based, as well as information about safe operating procedures. The operational environment, in turn, provides feedback to the manufacturer about the performance of the system during operations.

Between the hierarchical levels of each control structure, effective communication channels are needed, both a downward *reference* channel providing the information necessary to impose constraints on the level below and a *measuring* channel to provide feedback about how effectively the constraints were enforced.

Degradation of the safety-control structure over time may be related to *asynchronous evolution*, where one part of a system changes without the related necessary changes in other parts.

For an accident model to handle system adaptation over time, it must take into account the processes involved in accidents and not only simple events and conditions.

Process models

Beside the constraints and hierarchical levels of control, a third basic concept of the STAMP is process models. Figure 3-3 shows a typical process-control loop with an automated controller supervised by a human controller.

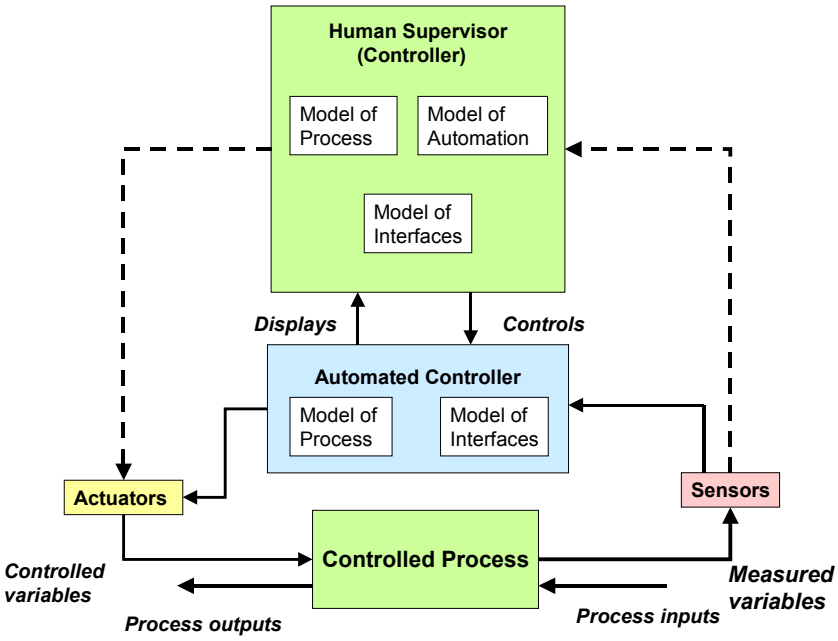


Figure 3-3 A standard hierarchical three-level control loop

A classification of control flaws leading to accidents

In basic systems theory, to effect control over a system requires four conditions:

- Goal Condition: The controller must have a goal or set of goals, e.g., to maintain a setpoint or to maintain the safety constraints.

- Action Condition: The controller must be able to affect the state of the system in order to keep the process operating within predefined limits or safety constraints despite internal or external disturbances. Where there are multiple controllers and decision makers, the actions must be coordinated to achieve the goal condition. Un-coordinated actions are particularly likely to lead to accidents in the boundary areas between controlled processes or when multiple controllers have overlapping control responsibilities.

- Model Condition: The controller must be (or contain) a model of the system, as described above. Accidents in complex systems frequently result from inconsistencies between the model of the process used by the controllers (both

human and software) and the actual process state. For example, the software thinks the plane is climbing, when it is actually descending and, as a result, applies the wrong control law or the pilot thinks a friendly aircraft is hostile and shoots a missile at it.

- Observability Condition: The controller must be able to ascertain the system state from information about the process state provided by *feedback*. Feedback is used to update and maintain the process model used by the controller.

Using systems theory, accidents can be understood in terms of failure to adequately satisfy these four conditions.

1. Hazards and the safety constraints to prevent them are not identified and provided to the controllers (goal condition)

2. The controllers are not able to effectively maintain the safety constraints or they do not make appropriate or effective control actions for some reason, perhaps because of inadequate coordination among multiple controllers (action condition).

3. The process models used by the software or by human controllers (usually called mental models in the case of humans) become inconsistent with the process and with each other (model condition).

4. The controller is unable to ascertain the state of the system and update the process models because feedback is missing or inadequate (observability condition).

We have then the following list of control flaws leading to hazards, Figure 3-4.

Control Flaws Leading to Hazards

• **Inadequate control actions (enforcement of constraints**)
- Unidentified hazards
- Inappropriate, ineffective, or missing control actions for identified hazards
- Design of control algorithms(process) do not enforce constraints
- Process models inconsistent, incomplete, or incorrect (lack of linkup)
- Inadequate coordination among controllers and decision makers

• **Inadequate Execution of Control Action**
- Communication flaw
- Inadequate actuator operation
- Time lag

• **Inadequate or missing feedback**
- Not provided in system design
- Communication flaw
- Time lag
- Inadequate sensor operation (incorrect or no information provided)

Figure 3-4 A classification of control flaws leading to accidents

When using STAMP, the control flaws identified in Figure 3-4 are mapped onto the components or the control loop and are used in understanding and preventing accidents.

In the next section we indicate some extensions of the STAMP accident model to more fully cope with dependability and security and the complexities of the socio-technical system depicted in Figure 1-6.

3.2 The STAMP++ accident model

The STAMP accident model of assessing system failures has been very successful in the breakdown analysis and diagnosis for the kind of socio-technical systems (Figure 1-6) that can be described by the Figures 3-2 and Figure 3-3. Given the discussions in Section 3.1 *Assessments of recent blackouts*, we claim that suitable adaptations (mainly of Figure 3-2 and 3-4) of the STAMP methodology can qualify for a *backbone in accidents models of future virtual utilities*. We denote such extensions with STAMP++. A modern overview of systems theory related to ecosystems and information ecosystems are [6] and [14].

We recall that the basic assumptions behind the STAMP approach was (Section 3.2):

1 Security is expressed as constraints of the behaviour of components.

2 The behaviour of components is modelled as interactions between reliable components. Hazards arise as a consequence of flawed processes.

3 A focus is on internal threats (Figure 1-6).

4 Control conditions related to: Goals, Actions, Models, and, Observability.

Obviously the assumption 2 is not valid in software-intensive embedded systems. Design, implementation and maintenance of (large) *software-intensive* systems is still a challenge (D1.6 Section 5.2 *Secure execution of unreliable software*) Again, the assumption 3 is not valid if we take terrorism into account or if the system is exposed *to information threats* (D1.6 Chapter 4 *Information security and network security*).

The control conditions of assumption 4, especially adequate modelling and adequate means of observation and action, become more challenging as we extend the relatively simple models of Figure 3-2 and Figure 3-3 into more complex socio-technical systems. We have to that end developed systems in the area of Network Enabled Capabilities (NEC) [9]. Specific challenges are related to the modelling aspects (Figure 3-3). In fact we advocate a system architecture allowing different views simultaneously and a system model based on missions composed of explicit goals, effects to be pursued (Effect Based Operations), connection and coordination of necessary capabilities and support for assessing different views and levels of the systems state (Figure 3.5).

A particular challenge is to define and control the behaviour (and related constraints) of the system. To that end we propose a linkage between the formal semantics (offline semantics) of software designs and the actual behaviour of the system as embedded process algebras (online engineering) [13].

The following Figure 3-5 illustrates some aspects of online engineering as a mean to support control conditions. The interaction engine supports assessing the state of the system (monitoring) as well as performing actions to maintain security and dependability constraints.

- System architecture: Three main roles – different concurrent perspectives

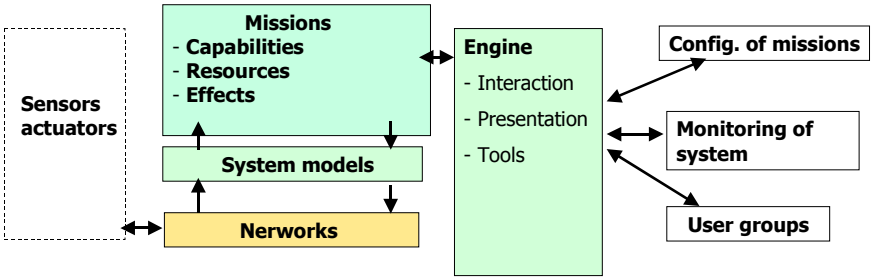


Figure 3-5 Architecture of systems supporting Network Enabled Capabilities

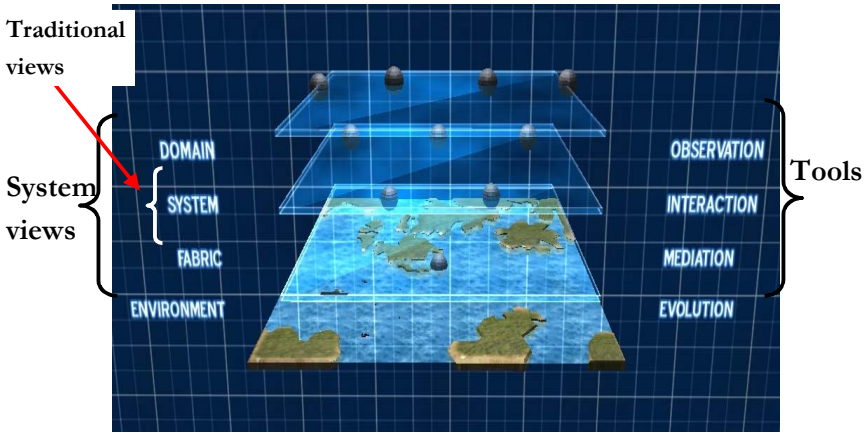


Figure 3-6 Layered online system views and tools

Figure 3-6, illustrates the different hierarchical levels (Domain, System, Fabric) we have found useful in understanding the system behaviour supported by ICT systems. Especially, the environment level provides an excellent possibility to understand and react to the embedding of the system in the physical world. We have also developed tools supporting system observation and interaction to control mediation and evolution while preserving mission constraints (Figure 3-3).

In figure 3-4 we indicate the main components of our methodology [6] [9] [10] [11] [12] [13] [15].

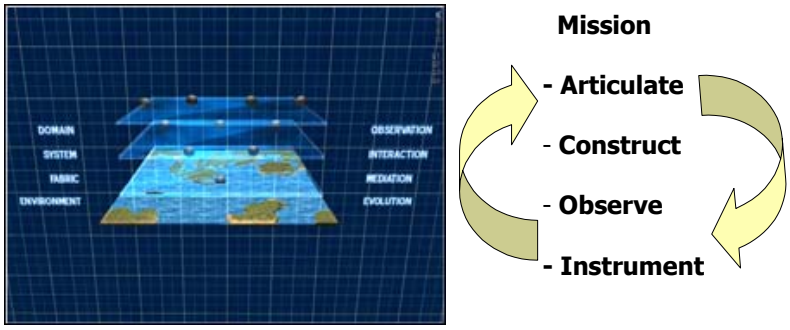


Figure 3-7 Main components of our methodology

Of crucial importance is the abilities to instrument and observe the system in an appropiate way (Figure 3-3, Figure 3-5, and Figure 3-7). Recent advancements in networking, hardware, and middleware technologies (Figure 2-10) have been a major catalyst for the recent approaches in grid-based computation [3] [8]. Grid computing is characterised by their very high computing and resource requirements. Thus, it needs powerful and active instrumentation (the process of putting probes into software to record systems' operation state data).

Originally, instrumentation was used to debug and test applications that run on single processor machines and for analysing the performance of real-time systems. The parallel computing community later adopted instrumentation to debug, evaluate and visualise parallel applications. More recently distributed application developers have recognised the potentials of instrumentation, used in a dynamic regime, to monitor and manage today's distributed applications [21].

It is now generally accepted that systems' introspection and general runtime monitoring for instance; inconsistency and fault detection requires software instrumentation/sensor services [3] [8]. However, the quality of service (QoS) of sensor/actuation service has received little or

no attention. There is a lack of focus on management and control issues related to sensors and actuation (effectors) for grid and web services environments.

In short, a more comprehensive systems-theoretical accident model (STAMP++) has to be complemented by such sensor/actuation services but also methods and environments related to secure execution of unreliable software [17] [18] [19] and appropriate information security measures. Furthermore we have to extend the models and tools for inspection and control as outlined in Figures 3-5, 3-6, and, 3-7. The STAMP++ methodology has then to be configured to the specific purpose at hand.

3.3 Critical interdependencies

From the analysis of *Section 3.1 Assessments of recent blackouts*, we learn that critical infrastructures become evermore interdependent, *systems of systems* (Figure3-1). A system has always a boundary and the STAMP++ Accident model of the previous section is applicable within such a boundary. In order to cope with systems of systems we must have a model of *critical interaction points between* those systems to have a handle to an inter-system control policy. We will return to this issue in the next Section 3.4 *Scope of dependability assessments of CRISP experiments*.

A starting point in modelling critical interdependencies in system of systems is to go back to the ideas of causal chains: The following Figure 3-8 emphasises that from a failure analysis point of view we do not discriminate between internal or external threats (Figure 1-6).

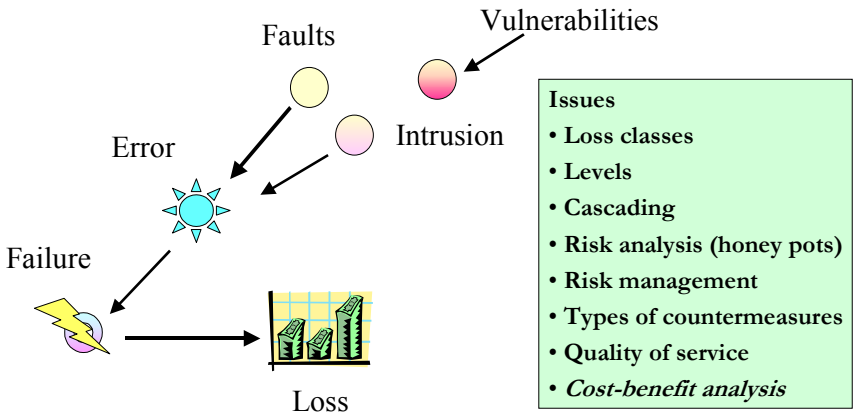


Figure 3-8 Dependability model in systems of systems

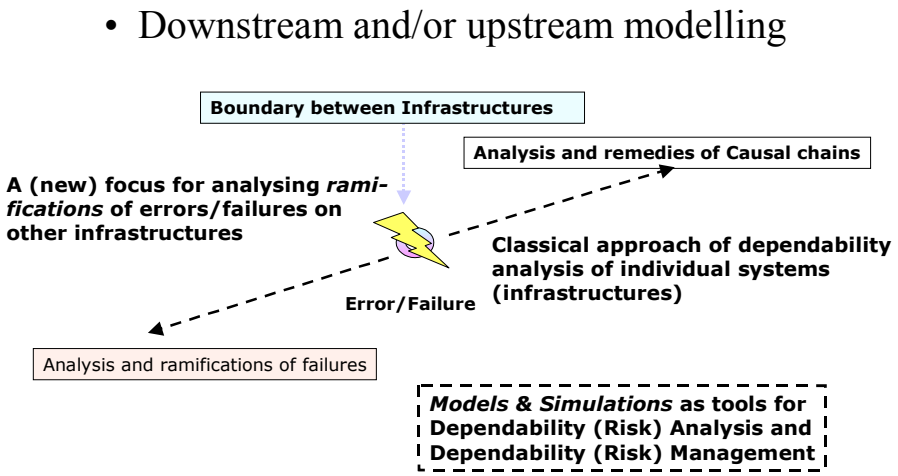


Figure 3-9 Export – import of disturbances in systems of systems

Figure 3-8 captures some important relations between the concepts Vulnerabilities, Faults, Errors, and Failures.

We chose to model failures depending on accidental, as well as intentional or malicious faults in a common framework of dependable systems. A system failure occurs when the delivered service deviates from fulfilling the system function as specified. An error is the specific part of the system state that is liable to lead to subsequent failure. The adjudged cause of an error is a fault. A vulnerability is a weakness in a system that can be exploited with malicious intent creating a malicious error and possible failure. We use the term intrusion to describe the type of fault caused by the exploitation of a vulnerability.

Figure 3-9 states that from an interaction point between critical infrastructures we can "export – import" the disturbance between the systems (Section 4.3 *Dependability issues related to Scenario 3*).

3.4 Scope of dependability assessments of CRISP experiments

In order to solve technical dependability issues related to the three scenarios in *Chapter 2 Setting the scene* based on the content of *Section 1.2 Goals of the Crisp experiments*, and taking into account the content of this *Chapter 3 Assessment of dependability in critical infrastructures*, we will address the following topics related to Figure 2-1:

- Performance metrics and secure (re)configurations of IP-based communication networks.

- Different state-based modes of coordination related to the three scenarios in Figure 2-1.
- Dependable execution of software connected in networks.

Dependability methods includes:

- Fault prevention
- Fault tolerance
- Fault removal
- Fault forecasting

3.4.1 Management of dependant systems

Dependability, e.g., security is a systemic property (Chapter 3). This means that the concept *neither* is decomposable *nor* composable from the same properties of sub-systems. The traditional engineering principle of functional decomposition and composition hence does apply to dependability and related systemic properties. Put in other terms, maintenance of dependability is a *systemic sustainability criterion* (Section 3.2.3)

The maintenance of a systemic criterion consists of the following four interleaving sub-processes (Section 3.3):

- Prevention: designing and implementing dependable systems in a cost efficient way.
- Detection: detecting flaws in systems and processes.
- Response: taking appropriate measures to limit the consequences of a failure.
- Hardening: measures for fault removal or fault prevention.

In this deliverable we focus on methods for *prevention* of failures, especially software failures since software is the glue of distributed system (Chapter 5).

4 Dependability issues related to Scenarios

This section gives a short list of dependability concerns related to the three scenarios of Chapter 2

4.1 Scenario 1

The following dependability concerns are derived from Experiment B and Experiment C (Section 1.2 and Section 2.1):

1. Secure and safe grid management
2. Dependable embedding of ICT meeting performance criteria

4.2 Scenario 2

Scenario 2 is based on agent mediated demand-supply matching (Experiment A and Section 2.2). The mediation is maintained by auctions at predefined times at in hierarchies. The actual algorithms are described in deliverable D2.2. In this deliverable we focus on dependability issues.

The modelling and algorithms of computational markets are given in *D2.2 Multi-agent Based Simulation Tool – Design Document*.

Dependability issues have two aspects:

1. Related to network security
2. Related to computer and software security
3. Related to information security

The first aspect is addressed in D1.6 *Chapter 4 Information security and network security*, the second aspect is also dealt with in D1.6 *Chapter 5 Layered architectures*. In *Chapter 7 Dependable implementation*, we refine some of the reasoning in D1.6. Issues related to information security, that is to *trusted markets,* are in focus of D1.6. The main concerns are taking appropriate measures to meet threats against:

- Confidentiality
- Integrity
- Availability

- Authentication
- Non-repudiation

The Confidentiality – Integrity – Availability (CIA) issues were addressed in deliverable D 1.6 *Chapter 2 New business models*. In short, the concerns are that information should not be leaked to unauthorised agents, information should be protected against tempering, and the services should be available to every authorised stakeholder of the market. These concerns have also been illustrated in the ELEKTRA game by ECN.

In our setting, it should be validated (verified) that the corresponding market algorithms fulfils those conditions.

The authentication of all stakeholders in an auction should also be guaranteed (this criterion is orthogonal to the CIA-criteria). We propose a variant of the *Kerberos* authentication protocol to that end.

Finally, the issues of non-repudiation are of course important in a trusted market.

4.3 Dependability issues related to Scenario 3

Scenario 3 Figure 2.4 (Section 2.3) illustrates critical interdependencies between the networks of Scenario1 and Scenario 2. From Section 3.3 and Figure 3-9 we suggest to harness this interdependency by *transfer of control* at critical situations in Scenario 1 to coordination mechanisms (computational markets) of Scenario 2. This transfer of control is enabled by transferring into a common coordination model in Scenario 1 and Scenario 2 (Chapter 6). The transfer of control is a high-level mechanism that relies on that we have proper high-level models of the matching algorithms (Chapter 6 and Figure 3-6)

4.3.1 An unified coordination model

We propose as a unified coordination model for Scenario 3 an *extended auction market* introducing privileged agents corresponding to Scenario 1 into the auction model of Scenario 2 (Section 6.4).

4.4 A layered security model

From D1.6 *Chapter 5 Layered architectures*, we copy the following Figure 4-1 illustrating that security measures can be installed at *different layers* of the generic ICT reference model of Figure 2-10.

The placements of the security mechanisms in Figure 4-1 have advantages as well as disadvantages. In general are higher-level protections specific for a certain application, whence protections of lower layers are geared at securing message transport.

The proper choice of protection mechanisms is very *context dependant*. Typically, Scenario 1 has a major concern related to the lower layers, whence Scenario 2 is concern with higher-

layers (user and application oriented information security). In Scenario 3 we have to combine those concerns. Having said that, we need to have a high-level description of the purpose and context of the algorithms but forward in the deliverables behind our three scenarios. That is, Deliverables D2.2 and D2.3 as well as the descriptions of the CRISP experiments (Chapter 1 Background and overview of the deliverable).

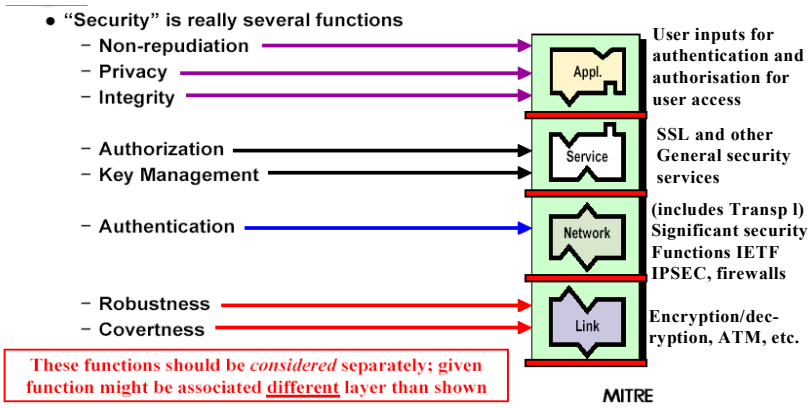


Figure 4-1 Security measures at different layers of the ICT reference model

The Open Grid Service Architecture is given in Figure 4-2.

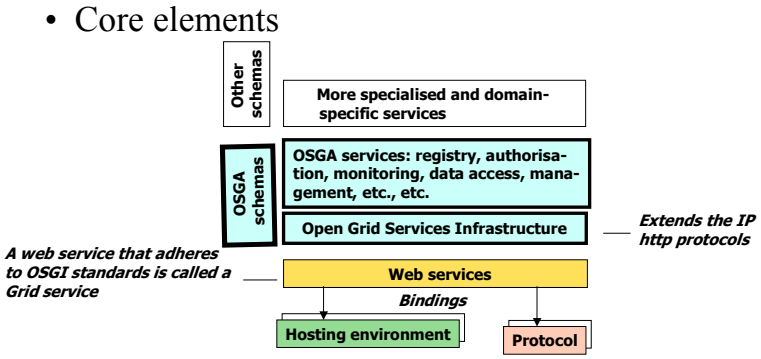


Figure 4-2 The Open Grid Service Architecture

Security in Grid computing is modelled as services (Figure 4-3) [3] [8].

- Security functionality as services

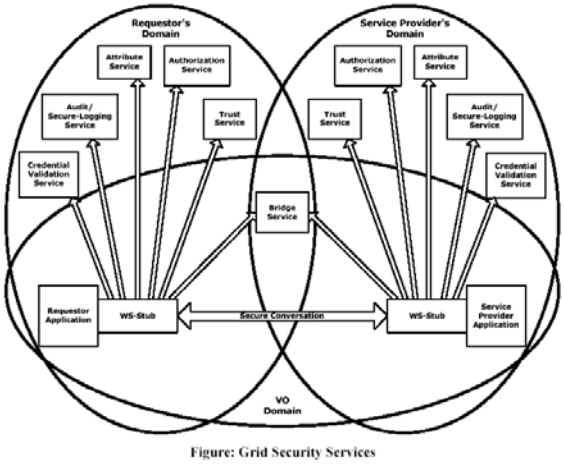


Figure 4-3 Security as services

In the following two Chapters 1 (specifically *Chapter 6 Tasks, protocols and dependable coordination*, in particular *Section 6.3.1 Assessments of algorithms of the scenarios*) we focus on a suitable high-level description of the algorithms involved. Based on this investigation, we return to dependability issues brought up in this chapter and related to Figure 4-1.

5 Coordination in the power grid

We know that there are potential conflicts between the requirements for coordination in grid and business operations, Figure 5-1 shows the coordination inherent in Figure 2-1 and Figure 2-12.

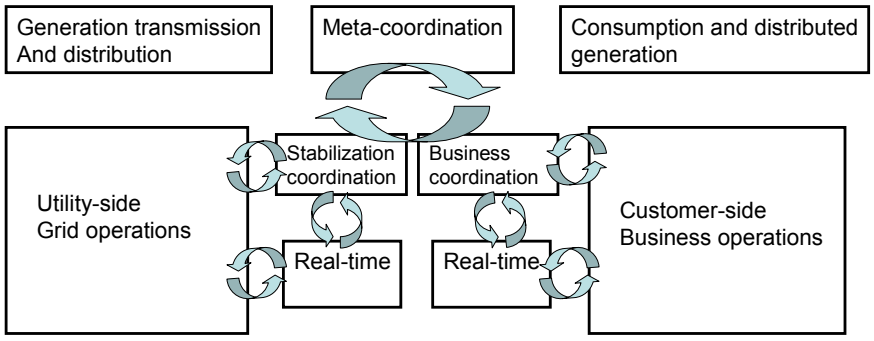


Figure 5-1 Coordination patterns in future virtual utilities (Scenario 3)

In fact, Figure 5-1 could be seen as the coordination necessary in Scenario 3 (Section 4.3). As we have said, Scenario 3 is an *interdependence* between infrastructures of Scenario 1 and Scenario 2 (Section 1.3 and Chapter 2).

Given the different goals of Scenario 1 and Scenario 2, i.e., *technical management* of grids versus *customer-oriented business management*, would indicate that the operations of the two networks should be disjoint in order to maximize their respective performance. Grid and business operations cannot, however, be built independently of each other, as both networks need to share information in order to operate according to their goal. A trivial example of such sharing of information is when the energy market is reacting to actual conditions in the power grid. But it is not enough to just be able to share information between the networks – they must also resolve *potential conflicts* and coordinate their actions.

Our approach is to have minimal (or non at all) control interaction between the two networks in normal situations. The following Figure 5-2 captures the different states of the Scenario 1 (the technical grid operations).

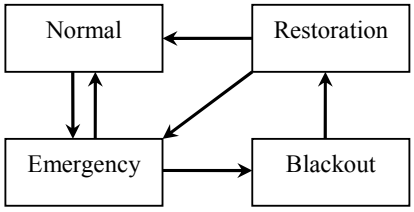


Figure 5-2 Different states of the Scenario 1 grid network

The state of the power grid affects which information is needed where and what actions should be coordinated (Chapter 3). The main phases are normal operation, emergency (e.g. the loss of a line), failure (blackout), and restoration, as depicted in Figure 5-2. The coordination requirements differ between these states, in that normal operation is focused on coordinating wide-area measurements and DSM, emergency operations need quick reactions (e.g. load shedding). In the failure state the network may be incommunicable, and in the restoration wide area consensus and planning is needed so that the system can safely be brought back into normal operation.

5.1 State-based coordination

The coordination model(s) that support the characteristics of normal operation are the blackboard-based ones; such as Linda-like coordination languages and market-oriented models. In blackboard-oriented models a central node is used for all participants. All information on the state of the system is in the blackboard, so consensus between the system nodes is maintained by the model itself. Hierarchal distributions of blackboards is one way to remove the single-point-of-failure that a blackboard introduces (e.g. in the TuCSoN coordination model). Nevertheless, consensus can only be guaranteed for the information that is stored in a single blackboard. The performance penalty of blackboards is noticeable – especially in terms of latency. Since the (sub)system is in effect forced into synchronous operation where all information is passed to/from the blackboard two messages are needed to transfer control to another node in the network, compared to only one for other coordination models [27] [28].

In the emergency state quick reactions are needed in order to save as much of the power grid as possible. It is more important to save the whole power grid from blackout than to protect a single operator. Consequently, consensus between the ICT nodes is a luxury we cannot afford. The fastest way to coordinate two communicating nodes is direct message passing. This is how control-oriented coordination models function. Latency is reduced and the communication is asynchronous. The drawback of control-oriented models is that consensus is impossible to guarantee, communication paths are nearly static (primary, secondary, and tertiary communication links should be defined for each node upon reconfiguration of the network).

This broad spectrum of requirements cannot be dealt with within a single infrastructure. One option is to have an infrastructure that implements the common aspects. However, this inevitably leads to compromises in the functionality and/or performance.

Instead, we want to allow the system to operate optimally under all operating states of the

power grid. Consequently, we should switch coordination model along with state-changes in the power grid.

There are two ways a node in the ICT network can detect a state-change in the power grid. It can either be informed by the measuring apparatus at the node (e.g. detection of short-circuits), or by another node in the network. This implies that all messages between nodes in the ICT network should include which state the sending node considers the power grid to be in. This is problematic from a security point of view; trust between nodes. To combat this, the first and foremost action should be to isolate the ICT network from access from the public internet.

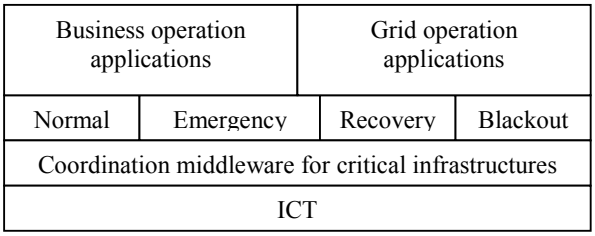


Figure 5-3. A layered coordination model

When the system is under normal operation, it is mainly controlled by business applications interfacing through electronic markets. The different operators of the system typically do not trust each other, and only share the minimum dynamic information required to support the market and the legislation. In emergency operations the situation is clearly the opposite as the stability of the overall power grid is more important than keeping current dynamic information secret (as it will probably be useless when the system has reconfigured). Consequently, it is important that there is no unwanted information transferred between the coordination of normal operations and emergency operations. To that end it may be possible to use barriers for safe execution of unreliable software.

Further research and experiments in fallback from blackboard-oriented to control-oriented coordination models is planned. Figure 5-3 depicts the research approach in coordination middleware for critical infrastructures that we are currently pursuing. The middleware is intended to handle all coordination (but not necessarily streams of data) between nodes in the system while exposing interfaces for normal, emergency, recovery, and blackout states. An application can then behave in a meaningful way given the current state of the system. However, an application should not be forced to implement all interfaces (and in many cases it will not make sense to do so). Consequently, only the applications that are relevant for the current state of the system will be allowed to act in it.

5.4 Policies and maintenance

Trustworthy operations can be assessed using the STAMP++ accident model (Section 3.2). This means , among other things, that we have to define appropriate models and processes to that end (c.f., Figure 3-2). This implies that we have to provide a set of policies and algorithms that allows the implementation of and unified market model (Section 6.4) that meets the constraints expressed above and the architecture of Figure 4-1.

6 Tasks, protocols and dependable coordination

In the previous section we addressed the issue of interdependent infrastructures. In effect Scenario 3 can be seen as a superset of the basically independent Scenario 1 and Scenario 2. However, near a breakdown (potential blackout in Scenario 1) we argue that some critical load balancing or other measures can be taken if we can coordinate, in Scenario 3, the two other networks in order to save the situation. The challenge is, of course; how can we meta-coordinate a technical coordination with a customer-centric business coordination model (Figure 5-1)? The main idea is to identify a high-lever coordination model. Since the market model has a high-level (almost independent of technical constraints) flavour, the idea is to transform the technical coordination at emergency points (Figure 5-2) into a market model.

To that end, we clarify some basic concepts related to distributed applications. The material is a elaboration of material from the EC project CommonKADS, especially from the book on Knowledge Engineering and Management – The CommonKADS Methodology [25]. Specifically, we use notations from the CommonKADS Communication Model.

In effect, agents are engaged in collaboration of performing a task. This collaboration is constrained by a *coordination pattern*. A key component in the coordination pattern is the *communication model* where information exchange *messages* are parts of a *transaction*, that in turn are orchestrated according to a *communication plan*. The control of a communication plan is a *dialogue schema* (e.g., an auction schema). Communication plan control, or *protocol*, can be described by a UML *process diagram.* We illustrate the main ideas by describing a protocol for auctions.

A unified coordination model related to Scenario 3 is described in Section 6.4. Dependable protocols are described in Section 6.5 and we conclude by a short description of the Kerberos authentication protocol in Section 6.6.

In short, in the scenarios of Section 2 we have modelled the resource management task of Scenario 2 as a computational market with *a coordination pattern* of auctions. The control of the auction is described as an auction protocol. The computations of the auctions are described as an algorithm. Specific issues emerge when we address distributed auctions and dependability. For once, the algorithm has to be distributed in a suitable way and also strengthen to withstand disturbances. The protocol itself has to be safe guarded to meet other kind of disturbances. Finally we have to address issues related to security of business processes (auctions in computational markets).

6.1 Coordination patterns

In effect, the communication model focuses on information exchange between tasks carried out by different agents. The concept of an agent is here used in a general setting. We return to situations later when the agents could be software entities as used in multi-agent systems. The following UML-diagram (Figure 6-1) describes the main features of the CommonKADS communication model.

In Figure 6-1 the communication model consists of a Communication plan (dialogue diagram and transaction control) with transactions as subparts and information exchange as parts of transactions. The communication plan is executed by Agents with appropriate capabilities engaged in a task that has to be coordinated. The task structure manages the transfer function between agents and appropriate subtasks of the overall task. The communication model together with task, agents and task structure constitutes a *coordination pattern* in our context.

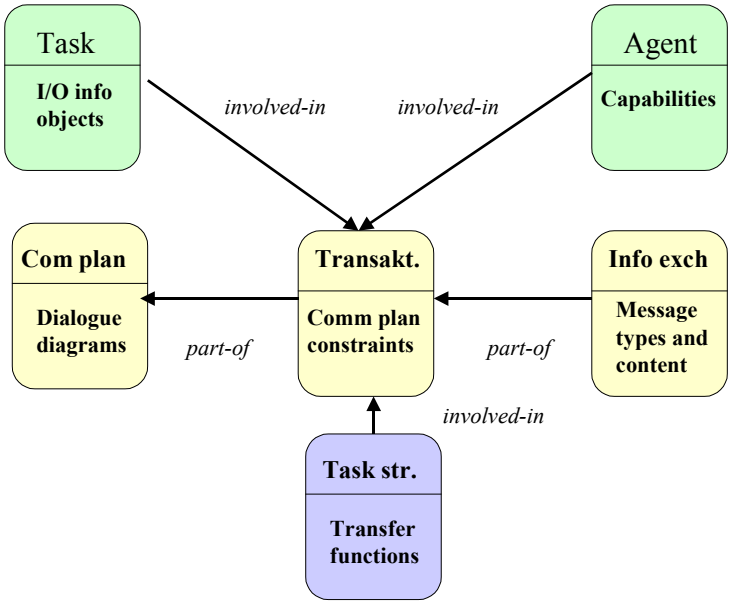


Figure 6-1. Overview of the communication model – coordination pattern

The communication model of Figure 6-1 is taken from the CommonKADS methodology [25]. We need to adopt this communication model into a *coordination model* (pattern). In order to, e.g., discuss security and dependability issues at application levels (Figure 4-1). A way to achiever this is to expand the Task structure component with a richer context (e.g., *underlying communication requirements*). The new coordination pattern thus depends on the task (e.g., coordination as a market), agents (right competencies), a dialogue structure (plan, transaction, information exchange) and the context (e.g., infrastructure requirements). From a dependability point of view we can now determine the appropriate security measures at the higher level of Figure 4-1 (e.g., ensuring non-repudiation, or protection privacy and integrity, or authorisation (right agent with proper credentials and competencies)). The following dialogue diagrams allows us to have similar dependability concerns focusing oh the dialogue and exchange of messages. Finally, Figure 6-4 allows us to introduce, e.g., appropriate inspection points to monitor the auction process.

6.2 Dialogue diagrams and protocols

Coordination of subtasks is modelled as communication based on dialogues and data transfer between the agents involved in a task.

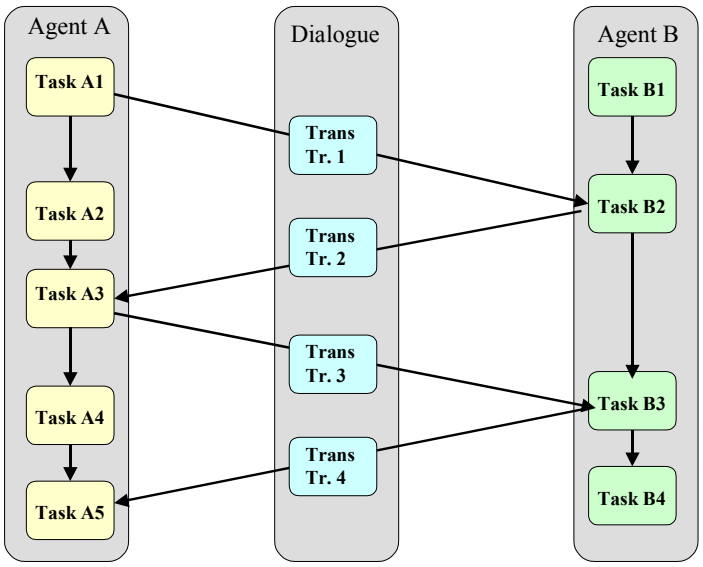


Figure 6-2. The general layout of a dialogue diagram

In the case we have an auction protocol (as in Section 1.2.1 *Experiment A*) we have the following generic dialogue diagram, Figure 6-3 and 6-4.

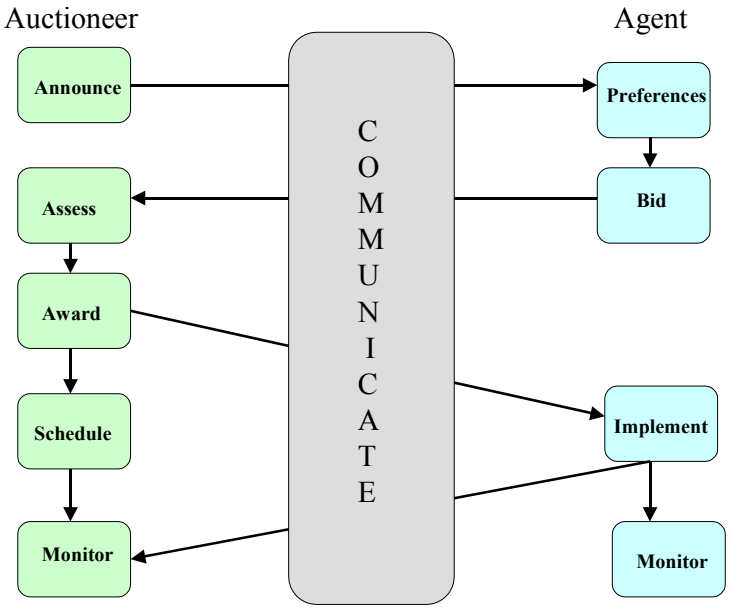


Figure 6-3. Generic dialogue diagram of an auction protocol

Figure 6-2 gives the general layout of a *dialogue diagram* between the agents A and B corresponding to Figure 6-1. The subtasks performed by A are A1, A2, A3, A4, and A5 and similar for agent B. The dialogue consists of *Transactions* between the agents at appropriate points.

In the case the dialogue is related to an auction task (*Supply-Demand matching*), we have in fact a auction protocol as described in Figure 6-3.

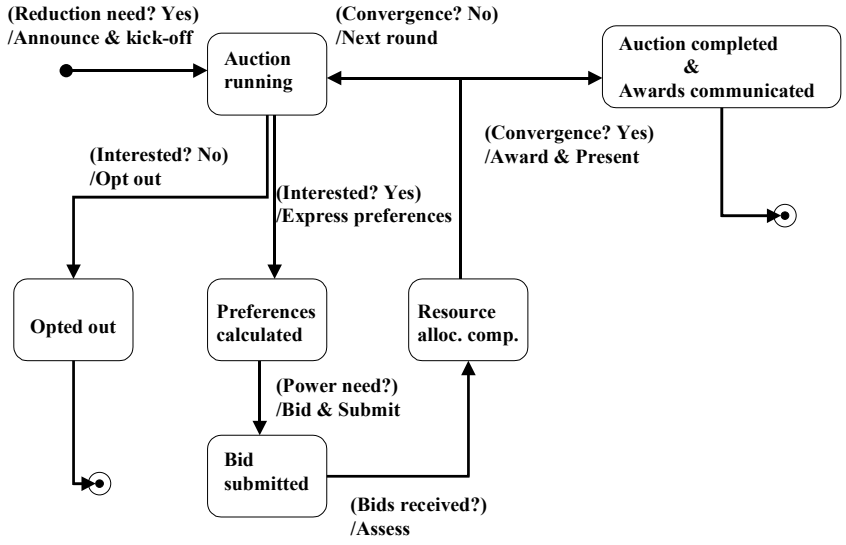


Figure 6-4 An UML state diagram of the auction protocol

In short: The different diagrams of Figures 6-1 to 6-4 allow us to meet information threats at different levels and contexts related to the appropriate abstractions (Chapter 4 and Figure 4-1. We note that standard crypto-graphic techniques usually only protect the *content* of the transactions in Figure 6-2. Fu

Furthermore, the coordination pattern of Figure 6-1 allows us to model change of control between interaction infrastructures to maintain critical operation criteria (e.g., avoidance of blackouts, Section 6.4).

6.3 Algorithms

The algorithms related to the coordination pattern of Figure 6-1 corresponding to the auction protocols of Figure 6-3 and Figure 6-4 describes in a *procedural* way the processing related to the notations along the arcs between states and the processing in the states of Figure 6-4. As we can see there are *several* possible algorithms that implement the same auction protocol. The choice of algorithms is in that sense orthogonal to the choice of supporting

protocols.

In the case we have a distributed system both the algorithms and protocols have to be distributed and allocated to different entities. However the underlying protocols have to be maintained during that distribution.

6.3.1 Assessments of algorithms of the scenarios

The algorithms of Scenario 1 in Section 2.1 and Scenario 2 in Section 2.2 can be recast in terms related to the high-level coordination patterns involved (c.f. D2.3 and D2.2). Furthermore, in doing so we can specify the *competencies* involved to fulfil the tasks appropriately and the *contextual setting* of the coordination (Figure 6.1). This information is crucial when deciding proper dependability measures.

In doing so we have to decide upon at which *level* we what the dependability and security measures installed (Figure 4-1).

6.4 A unified coordination model

From Figure 2-1 and Chapter 2 Setting the scene – Architectures, it follows that coordination in Scenario 3 has to coordinate the needs of grid management and distributed demand supply matching (Figure 5-1). The latter utilize the abstraction of resource management to an auction model of computational markets (Section 2). The most *straightforward unification model* is to translate the needs of grid management into the terms of computational markets. In short, when there is a need to reduce or increase the power in Scenario 1 the corresponding utility acts as a *privileged agent*. The privileged agent can under certain condition enter the auction market of Scenario 1 and either sell energy very cheap or buy energy at very high prices to reduce load or to avoid load shedding.

Obviously, in order to do this abstraction from the technical grid operation coordination to the computational market based meta-coordination (Figure 5-1) we need to *import* the technical constraints of Scenario 1 into the protocol contexts of Scenario 2. We claim that a vehicle to do so is the coordination model of Figure 6-1. Of course, much work have to be done here to that end, but we claim that the CRISP Experiments are a good start in those investigations.

6.5 Dependability protocols

We consider two types of dependability protocols. The first class deals with robust behaviour of algorithms. The second class is focused on security issues.

6.5.1 Protocols for robustness

Issues related to robustness are scalability and proper behaviour at breakdowns. Scalability issues are related to maintenance of the computational markets as well as issues related to power grid maintenance.

6.5.2 Protocols related to security

Of specific interest is protocols securing authenticity of participants engaged in computational markets. We exemplify authentication architectures by a short introduction to *Kerberos* version 5. Kerberos was created by project Athena at MIT in Boston and is based on the Needham-Schroeder Authentication protocol.

6.6 The Kerberos single-signon architecture

In an enterprise environment, a user (agent) is usually entitled to use enterprise-wide distributed information services. These services are usually maintained by various business units in the enterprise. As a result, the various information servers can operate in different geographical locations. For security use these services need the user to provide various credential before a service can be granted. However, it is unrealistic and uneconomical to require a user to maintain several different cryptographic credentials to use.

A suitable network authentication solution for this environment is the Kerberos Authentication Protocol. The basic idea is to use *a trusted third party* to introduce a user to a service by issuing a shared session key between the user and the server.

The following Figure 6-5 captures the architectural components of Kerberos.

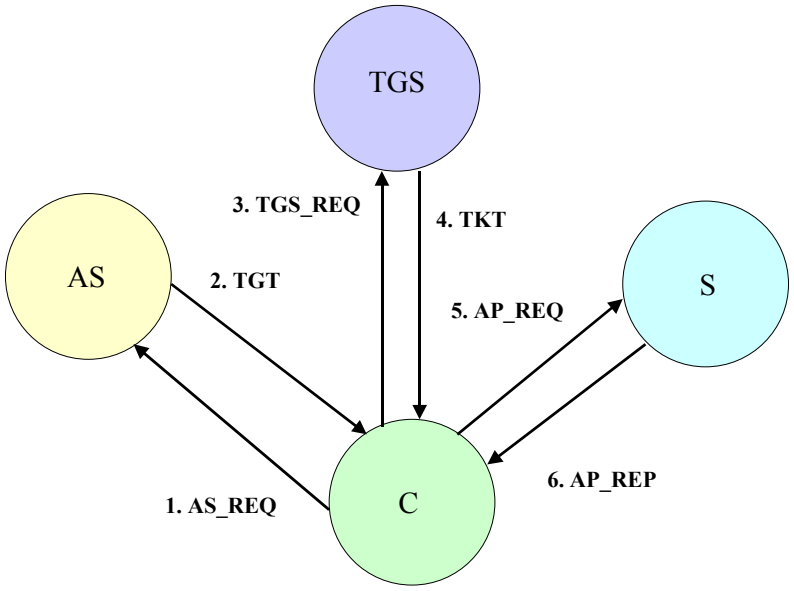


Figure 6-5 Architecture of Kerberos exchanges

The Kerberos authentication protocol consists of a suite of tree sub-protocols called exchanges. These three exchanges are:

1. The Authentication Service Exchange (AS Exchange): It runs between a “client” C and an “authentication server”.

2. The Ticket-Granting Service Exchange (TGS Exchange): it runs between C and a “ticket granting server” TGS after the AS Exchange.

3. The Client/Server Authentication Application Exchange (AP Exchange): it runs between C and an “application server” S after the TGS Exchange.

Each of these three exchanges is a two-message exchange protocol. These exchanges have the sequential dependant relation listen In Figure 8. Kerberos has five principals who operate in these three exchanges and these principals have the following roles:

- U: a User (Agent) whose actions in the protocols are always performed by the agents process; so U only appears in the protocols as a message. Each user memorizes a password as its *single-signon* credential for using the Kerberos system.

- C: a Client (a process) which makes use of a network service on behalf of a user. In an AS Exchange, in which C is initiated by U, C will need U’s Kerberos system credential. This user credential is given to C as it prompting U to key-in its password.

- S: an application Server (a process) which provides an application resource to a network client C. In an AP Exchange, it receives an “application request” (AP_REQ) from C. It responds with “application reply” (AP_REP) which may entitle C an application service. An AP_REQ contains C’s credential called a “ticket” (TKT) which in turn contains an application session key $K_{C,S}$ temporary shared between C and S.

- KDC: Key Distribution Center, KDC is a collectively name for the following two authentication servers:

    i. AS: an Authentication Server. In an AS Exchange, it receives a plaintext “authentication service request” (AS_REQ) from a client C. It responds with a “ticket granting ticket” (TGT) which can later be ysed by C in a subsequent TGS Exchange.

    Initially, AS *shares* a password with each user it serves. A shared password is set up via a single-signon means outside the Kerberos system.

    A TGT supplied to a client C as a result of an AS Exchange has two parts. One part is for C to use and is encrypted under a key derived from a user’s single-signon password. The other part is for a “ticket granting server” (to be described in the TGS item below) to use and is encrypted under a long-term key shared between AS and the latter. Both parts of a TGT contain a ticket session key $K_{C,TGS}$ to be shared between C and a “ticket granting server”.

    ii. TGS: a Ticket Granting Server. In a TGS Exchange it receives a “ticket granting request” (TGS_REQ) (which contains a “ticket-granting

ticket" TGT) from a client C. It responds with a "ticket" (TKT) which entitles C to use in a subsequent AP Exchange with an application server S.

Similar to TGT, a TKT has two parts. One part is for a client C to use and is encrypted under a ticket session key $K_{C,TGS}$ (which has been distributed to C and TGS in TGT). The other part is for an application server S to use and is encrypted under key $K_{S,TGS}$ which is a long-term key shared between S and TGS.

Both parts of a TKT contain a new application session key KC,S to be shared between C and S. The application session key is the cryptographic credential for C to run subsequent AP Exchange with S to get an application service from S.

We must discuss two *warnings* in Kerberos exchanges:

1. The first one is about *careful validation* of a Kerberos ciphertext in a decryption time.

2. The second warning is about "authenticator". In order to prevent an adversary from modifying a Client_time in an authenticator, the cipher block of an authenticator itself need data integrity protection!

This warning applies to all authenticators in Kerberos.

7 Dependable implementations

Issues of Section 6.5 Dependable protocols are related to validation of protocols. Dependable implementations of those protocols are a challenge in itself, especially in open systems. As usual it is a trade-off between complexity and security (Figure 4-1).

7.1 Safe Execution of Unreliable Software

Software plays an important role in several systems used to control the power grid today, and as we are moving to more and more complex coordinated systems for controlling the power grid it is clearly so that even more software than today will be used to control this critical infrastructure. Recent years has shown that software used to control critical infrastructure has many of the unwanted properties of normal desktop software, such as bugs, vulnerabilities and unexpected behaviour, and that software malfunction in systems that control critical infrastructures can have a large impact on that infrastructure, as well as on the society.

There are clearly many advantages of using more intelligence in the control of the power grid, but by introducing this intelligence we also introduce more software, software which we know can fail and cause the system to malfunction. To handle this situation we need to be extremely careful with how software is used to control critical infrastructures and to take good precessions so that a single or a few software failures don't cause major problems for the power grid.

Using IP based networks and software to control the power grid means that other standard protocols (TCP, UDP), network stacks, and operating systems are very likely to be used on nodes in this ICT network. Using standards and standard implementations of network protocols is a fast and inexpensive way to build a useful system, but there are many issues that need to be solved before combining a critical infrastructure with standard software and protocols, especially when it comes to security and dependability issues of the software.

7.2 Software and Vulnerabilities

Much software used today on the Internet and on other IP networks (i.e. Intranets) are written in non type-safe languages, as large monolithic programs that often execute under maximum privileges on a computer system. While there are specialized software for example used in some real-time systems, which is developed using very different methods than most commonly available software, and hence has very different properties, such software is not used on a large-scale basis on IP networks. There are likely several reasons for this, such as the special-purpose real-time software might not be very adaptable to other environments, or that adapting or using such software would be prohibitory expensive.

There will always be room for special-purpose solutions for the most critical tasks, where the risks are extremely high or where money is not a matter, but for all other tasks we need to create a sufficiently dependable solution with good but perhaps not extreme security and dependability requirements, that potentially can fall back to other more secure but less functional systems should a complete system breakdown occur.

For these and other reasons, we must partially use software developed using more conventional methods, meaning that the software is likely to have some defects, and that we must be able to use this software in a secure way. There are several different methods that can be used to increase the dependability of common software, but before we present some of these methods, we take a look at a few of the security-related problems that we a likely to encounter in normal software.

7.2.1 Security problems in non type-safe languages

Software written in non type-safe languages execute in an environment in which there is no notion of which type a piece of data is. This means that an incorrect program can use some data first as one type and then as another. The problems of non type-safe languages are clearly illustrated when an incorrect input to a program can result in the program executing that data. A common variant of this problem is known as a buffer overflow attack, which has caused vast amounts of problems in many Internet programs.

7.2.2 Security problems with monolithic software

Monolithic software is software that execute as one large chunk of program code (a monolith). If one piece of the program should fail, the whole monolith is likely to fail as a consequence. Often are monolithic software developed in such a way that it is difficult to later split the monolith in smaller pieces that communicate.

7.2.3 Security problems with maximal privileges

Many programs require maximal privileges to perform a small number of privileged operations, and for this reason does the entire program execute under these high privileges. The problem with this approach is that should there be a vulnerability anywhere in that program that code will execute with high privileges that could cause severe damage to the system, something that would not be possible if the code had executed under lesser privileges.

7.2.4 Security problems with large state space

A program that only uses one megabyte of RAM still has 2^{1024*8} possible states for this memory. While a vast majority of states can never be reached in a program, and not all states are unique for the program execution, there are still an enormous amount of different states that a program can have.

7.3 Methods for increasing the dependability of software

There are several methods available for increasing the dependability of normal software, so that it can be used for more critical operations. Some of these are classic software quality assurance methods, such as careful testing of a software program and its components while other methods focus on the executing program.

Methods that operate on the source or binary code of a program, without considering the execution of the program are often called static methods. Such methods include classic black-box testing (i.e., verifying the out-value for a function based on a known in and out-value) and source code review. For the latter case one or more security experts read through the binary code or (more often) source code to a program, trying to find and correct potentially dangerous situations. This method has been used on several open systems with quite good results, meaning that several vulnerabilities have been found and corrected, but is very labour intensive and the effectiveness highly dependent on the program and how skilled the reviewer is.

There are also automatic methods that operate statically on programs, but these methods often return vast amounts of false positives or fail to find many vulnerabilities.

Another type of methods, which operate on an executing program, are called dynamic methods. The advantage of using dynamic methods is that the actual state of the program is known, and that different actions can be taken as a direct consequence of the actual state of which the program is in. The disadvantage of dynamic methods is that they typically reduce the performance of the program and that care must be taken so that the tool does not introduce new security vulnerabilities into the executing program.

A dynamic method that is getting popular, but that requires special support in the CPU, is known as NX, standing for No Execute. When using this method the processor is instructed to only execute machine code instructions from certain parts of the RAM memory, but not from others. If an attempt is made to execute machine code instructions from a prohibited part of the RAM the program is stopped. By using this method is possible to prevent some of the attacks known as "stack smashes". The advantage of this method is that it does not impose an execution overhead on the program as all special handling is supported by additional CPU logic. The disadvantages are that relatively few CPUs support this feature today, and that it is not compatible with all programs, but cases incompatible but correct programs to crash.

As a part of the CRISP project we have also done experimentation with two other dynamic methods, namely Transparent Privilege Separation and a general purpose program library protector, Plibc.

Privilege separation is a technique used to let a single program execute different parts under different privileges, and is used on several systems. The purpose of transparent privilege separation is to separate especially dangerous parts of monolithic programs into a temporary low-privileged environment where it can execute without exposing the system to a risk. This is designed specifically to be easy to implement in programs that are not originally designed for separation of privileges.

While privilege separation in general, and also transparent privilege separation, is aimed at reducing the consequences of a potential vulnerability using Plibc gives the administrator of

a system the possibility to determine how potentially unsafe uses of library interfaces should be handled. By intercepting calls in runtime it is possible to analyze the runtime state of the program and using this using this information take a decision of which calls should be allowed.

8 Conclusions

The focus of this deliverable is on dependable ICT support of power grid operation. By recasting the three CRISP experiments into three Scenarios in Chapter 2 we claim that we attained a good notion of benefits and challenges related to future virtual utilities. Among the challenges are securing trustworthy operation from a technical operation side (avoid disturbances such as blackouts) as well as from a user-centric business point of view (value added power related services).

Our investigation on proper means to safeguard operations of future virtual utilities begins with an assessment of lessons learned from recent (2003) big blackouts worldwide in Chapter 3. We propose an accident diagnosis and repair model (STAMP++ in Section 3.2) suitable for the complex socio-technical system we envisage (Figure 1-6).

From this analysis and the background material from deliverable D1.6 Information security models and their economics, we then reassess the dependability concerns related to the CRISP related scenarios of Chapter 2.

The deliverable provides some novel ideas and models that we claim are useful beyond the CRISP project. Having said that, there is much more work to be done along the lines discussed in the deliverable. A good start of further investigations is the planned CRISP experiments.

9 References

Papers

[1] Amin, M. (2004): Balancing Market Priorities with Security Issues. *IEEE Power & Energy Magazine*, pp. 31 – 38, July/August, 2004.

[2] Avizienis, A., Laprie, J-C., Randell, B., and, Landwehr, C., (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. I*EEE Transactions on Dependable and Secure Computing*, Volume 1, Number 1, pp. 11 – 33. IEEE Computer Society.

[3] Berman, F., Fox, G.C., and Hey, A.J.G. , (Eds.), (2004). *Grid Computing. Making the Global Infrastructure a Reality.* John Wiley & Sons Inc..

[4] Bertalanffy, L. (1969). *General Systems Theory: Foundations, Development, and Applications*. Brazillewr, G. (ed.), New York.

[5] Checkland, P. (1981). *Systems Thinking, Systems Practice*. John Wiely & Sons.

[6] Costanza, R., Low, B.S., Ostrom, E., and Wilson, J. (Eds.) (2001). Institutions, Ecosystems, and Sustainability. Boca Raton, London, Lewis Publishers.

[7] Cowie, J., Ogielski, A., Premore, B.J., Smith, E., and Underwood, T. (2003) *Impact of the 2003 Blackouts on Internet Communications*. Renesys Corporation, Preliminary Report November 21st, 2003.

[8] Foster, I. and Kesselman C. (2003). *Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, 2003.

[9] Fredriksson, M. (2004) *Online engineering: On the nature of open computational systems.* Dissertation series, no. 2004:05, Department of interaction and system design, School of engineering, Blekinge institute of technology, Sweden.

[10] Fredriksson, M. and Gustavsson, R. (2004) Online engineering and open computational systems. In Bergenti, F., Gleizes, M.-P., and Zambonelli, F. (eds.) Methodologies and software engineering for agent systems: The handbook of agent-oriented software engineering, pp. 377 - 390. Kluwer Academic Publishing.

[11] Fredriksson, M., Gustavsson, R., and Ricci, A. (2003) Sustainable coordination. In Klusch, M., Bergamaschi, S., Edwards, P., and Petta, P. (eds.) Intelligent information agents: The AgentLink perspective, Lecture notes in Artificial intelligence (LNAI), vol. 2586, pp. 203 - 233, Springer Verlag.

[12] Fredriksson, M. and Gustavsson, R. (2002) A methodological perspective on engineering of agent societies. In Omicini, A., Zambonelli, F., and Tolksdorf, R. (eds.) Engineering societies in the agents world II, Lecture notes in artificial intelligence, vol. 2203, pp. 10–24. Springer Verlag.

[13] Gustavsson, R. and Fredriksson, M. (2005) Process algebras as support for sustainable systems of services (invited paper). In *Algebraic approaches for multiagent systems, Journal of Applicable algebra in engineering, communication, and computing*. Springer Verlag. (to appear)

[14] Gustavsson, R, and Fredriksson, M, (2004). Humans and Complex Systems: Sustainable Information Societies. In Olsson, M-O., and Sjöstedt, G. (Eds.) *System Approaches and Their Application. Examples from Sweden*, pp. 269 – 289. Kluwer Academic Publishers.

[15] Gustavsson, R. and Fredriksson, M. (2003) Sustainable information ecosystems. In Garcia, A., Lucena, C., Zambonelli, F., Omicini, A., and Castro, J. (eds.) *Software engineering for large-scale multi-agent systems: Research issues and practical applications*, Lecture notes in computer science (LNCS), vol. 2603, pp. 127-142, Springer Verlag.

[16] Leveson, N.G., (2004). A New Accident Model for Engineering Safer Systems. *Safety Science*, Volume 42, Number 4, Elsevier, April 2004.

[17] Mellstrand, P. (2005). *Design of a Flexible and Secure Experimentation Environment*. To be published.

[18] Mellstrand, P. and Gustavsson, R. (2004) Dynamic protection of software execution environment. In Proceedings of the 2nd International conference on Critical infrastructures, October 25-27, Grenoble, France.

[19] Mellstrand, P. and Gustavsson, R. (2003) Safe execution of unreliable software. In Focardi, R. and Zavattaro, G. (eds.) Electronic notes in theoretical computer science, vol. 85(3). Elsevier.

[20] Muniz de Almeida, I., and Johnson, C.W. (2005) Extending the Borders of Accident Investigation: Applying Novel Analysis Techniques to the Loss of the Brazilian Space Programme's Launch Vehicle VLS-1 03. Submitted for publication.

[21] Reilly, D. and Taleb-Bendiab, A. (2002) Dynamic Instrumentation for Jini Applications. *Third International Workshop of Software Engineering and Middleware*. Lecture Notes in Computer Science ,Vol. 2596, pp. 157 – 173.

[22] Rinaldi, S.(2004): Modeling and Simulationg Critical Infrastructures and Their Interde-pendencies. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, IEEE, 2004.

[23] Rinaldi, S., Peerenboom, J., and Kelly, T. (2001): Critical Infrastructure Interdependencies, (2001). *IEEE Control System Magazine*, pp. 11 – 25, 2001.

[24] Rindebäck, C., and Gustavsson, R. (2005) Why trust is hard – Challenges in e-mediated services. To appear in Special issue of Lecture Notes in AI on Trusting agents for trusting Electronic Societies, Springer Verlag.

[25] Schreiber, G., et. al (2000).: *Knowledge Engineering and Management. The CommonKADS Methodology*. MIT Press. ISBN 0-262-19300-0.

[26] Siewiorek, D.P., Chillarege, R., and, Kalbarcyk. (2004) Reflections on Industry Trends and Experimental research in Dependability. *IEEE Transactions on Dependable and Secure Computing*, Volume 1, Number 2, pp. 109 – 127. IEEE Computer Society.

[27] Törnqvist, B., Fontela M., Mellstrand, P., Gustavsson, R., Andrieu, C., Bacha, S., Hadjsaid, N., and Besanger, Y. (2004) Overview of ICT components and its application in electric power systems. In proceedings of 2nd International conference on Critical infrastructures, Grenoble, France.

[28] Törnqvist, B. and Gustavsson, R. (2004) On adaptive aspect-oriented coordination for critical infrastructures. In proceedings of the 1st International workshop on Coordination and adaptation techniques for software entities (WCAT), June, Oslo. To appear in WCAT´04 special issue of the journal L'Object, 2005.

Reports

[a] U.S –Canada System Outage Task Force (2004): Final Report on the August 14, 2003 Blackout in the United States and Canada. Causes and Recommendations, April 2004.

[b] UCTE, (2004): Final Report of the Investigation Committee on the 28th September 2003 Blackout in Italy, UCTE Report April 2004.

[c] Svenska Kraftnät (2003): Report from the Blackout in Southern Sweden and Eastern Denmark 2003-09-23, November 2003.