Distributed Intelligence in Critical Infrastructures for Sustainable Power

ENK5-CT-2002-00673

Information Security Models and Their Economics

R. Gustavsson, P. Mellstrand, B. Törnqvist

BTH

Identifier:	**BTH_NSM_WP16_001_05**
Date:	**2005-03-21**
Class:	**Deliverable**
Responsible Partner:	**BTH**
Annexes:	**Published papers**
Distribution:	**EC RESTRICTED**
Overview:	**Background material of security issues and model supporting future cell-based utilities and a Framework supporting assessments of cost-benefit measures of asset protections.**

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	2003-07-27	R Gustavsson	Draft dokument
Version 2	2003-09-15	R Gustavsson	Extended draft
Version 3	2003-10-19	R. Gustavsson	Preliminary version
Version 4	2004-03-31	R. Gustavsson, P. Mellstrand, and, B. Törnqvist	Final version related to old description of WP 1.6.
Version 5	2005-03-21	R. Gustavsson, P. Mellstrand, and, B. Törnqvist	Final version according to the new description of WP 1.6.

Document Description

The deliverable *D1.6 Information Security Models and their Economics* includes background material and specifications of a CRISP Framework on protection of information assets related to power net management and management of business operations related to energy services. During the project it was discovered by the CRISP consortium that the original description of WP 1.6 was not adequate for the project as such. The main insight was that the original emphasis on cost-benefit analysis of security protection measures was to early to address in the project. This issue is of course crucial in itself but requires new models of consequence analysis that still remains to be developed, especially for the new business models we are investigated in the CRISP project.

The updated and approved version of the WP1.6 description, together with the also updated WP2.4 focus on Dependable ICT support of Power Grid Operations constitutes an integrated approach towards dependable and secure future utilities and their business processes. This document (D1.6) is a background to deliverable D2.4. Together they provide a dependability and security framework to the three CRISP experiments in WP3.

Table of contents

1 Introduction

During 2003 we witnessed several large-scale blackouts in the world. Among the most published and critical events are the blackouts in US-Canada, Italy, and Southern Scandinavia, (Sweden-Denmark) [US-CAN, IT, S]. The consequences of the US-Canada and Sweden-Denmark blackouts include:

- Huge societal costs and demonstration of interdependencies of critical infrastructures:
    - The North-American disaster2003-08-14 affected about 50 million people and caused 5000 communication network to go down. The societal costs have been estimated by the US DoE to be more than 50 BUS$.
    - The Scandinavian blackout 2003-09-23 was shorter but never the less caused problems for people and societal infrastructures. The estimated societal cost was in the order of 2BSEK (250 MUS$).
- Failures of parts of the infrastructures:
    - Software vulnerabilities. An unknown flaw (creating a "race condition" in interactions between modules) in GE Energy Management System XA/21 at FirstEnergy Corp., Ohio caused the US-Canadian blackout. The particular software module consists of 3 millions lines of code. The software had a record of over 1 million operation hours before the critical event happened!
    - The Swedish - Danish blackout September 2003-09-23 was due to vulnerabilities in the infrastructure (lines and breakers) and in improper procedures before and during the accident.

The [US-CAN] report includes a list of recommendations. These can be grouped in different categories as follows:

- Findings of *institutional issues* related to the causes of the disasters
    - Insufficient investments
    - Lack of training of personnel
    - Insufficient maintenance
- Need of standardisations
    - Establishment of *enforceable* standards
- Need of technological improvements
    - *Specifically the role of ICT- Power Grid Management*
    - *Dependable software - Secure execution environments*

The content of the deliverables D1.6 and D2.4 are addressing some of the challenges and needs reflected in the assessments of the disasters mentioned above. It should be noted that the trends of deregulations of energy systems, integrating grids (including DG and RES), development of new energy-based business processes enabled by introduction of ICT in effect will create new vulnerabilities if careful design and risk analysis is not properly enforced. In fact a goal of the CRISP project is to provide some pointers to that end. Most of that discussion will be included in D2.4.

The scope of this deliverable, i.e., Information Security models and their Economics poses several challenges. Introduction of ICT support of power systems are due to many reasons. Firstly, as we have seen, recent power blackouts in Europe and in the US have illustrated the vulnerabilities of present power grids as such due to inabilities to cope with emerging faults and fault propagation. At the same time there is a political pressure of introducing Distributed power Generation (DG) and Renewable Energy Sources (RES) as well as gas driven generators μCPH at premises of the customer (WP 3). These new demands on power grids and their operations, including fault detection, fault clearing and fault localization, poses new demands on supporting information infrastructures beyond the capabilities of present SCADA systems. Introduction of supporting ICT (Information and Communication Technologies) has to:

- Maintain reliable operations of software at substations and transmission/distribution nets.
- Support transmission of data in a proper a reliable manner, e.g., no susceptibility to EMC (electromagnetic compatibility).
- Protect against malicious intruders, e.g., protecting integrity of parameter settings, message ordering, and, supervisory data.

In short, the old style of hierarchical generation – transmission – distribution networks is in the process of being replaced with cell like structures in which the goal is to maintain local energy balances and system robustness (WP 2). To cope with those challenges we need to replace, or complement, existing SCADA systems with flexible information systems. Simultaneously, deregulation of energy markets introduce new kinds of market models at customer side dealing with buying and selling energy or energy related services (WP 2). Identification of security models that are appropriate for these new aspects of the energy sector is a challenging task, not the least due to critical interdependencies between network supporting power distribution and information exchange (WP 3).

The coupling of security models to economic assessments of, e.g., countermeasures derived by a risk assessment poses a new set of challenges. We propose a framework based on a combination of business modelling techniques derived in the BUSMOD[1] project – Business Models in a World Characterised by Distributed Generation – and risk assessment models based on state-of-the-art in risk analysis of IT-systems. However, it should be noted that we have an additional complexity in the CRISP approach due to the interactions between the electrical grid an information networks.

[1] http://www.dgnet.org:8080/BUSMOD/index.jsp

Interactions between the electric grid and the supporting information network connecting utilities and end users create opportunities as well as vulnerabilities. Models and methods to harness vulnerabilities depend on one side on the types of interactions between the information networks and the grid networks as well as the interactions between the information network and customer networks. Issues of coordination between networks on different configurations as well as coordination between levels of networks are crucial in this context (Annex 2). Furthermore, in order to address the issues mentioned above we need also to identify those business processes and information assets we must protect and the cost/benefit of different protection measures.

Given the background above we describe our CRISP-approach as follows. The generic model is composed by the layered PS-ICT-SO model of Deliverable 2.1 extended with an application layer, in our case, with Power Network Operations (PNO) and Business Operations (BO). The PNO applications are assembled from services from the PS and SO networks connected by the ICT network, c.f., Figure 6 and Figure 13. Similar holds for BO applications. Obviously we will have different choices of services providing the different applications. We return to those issues in Section 5 *Layered architectures*. Figure 1 captures the main components of a layered reference model supporting future utilities and their business processes. The technical details are discussed in D2.4. The present application view of PNO is to 'hardwire' a vertical information 'SCADA stovepipe' thereby creating a system that is hard to maintain and evolve. Furthermore, assessments of recent power blackouts

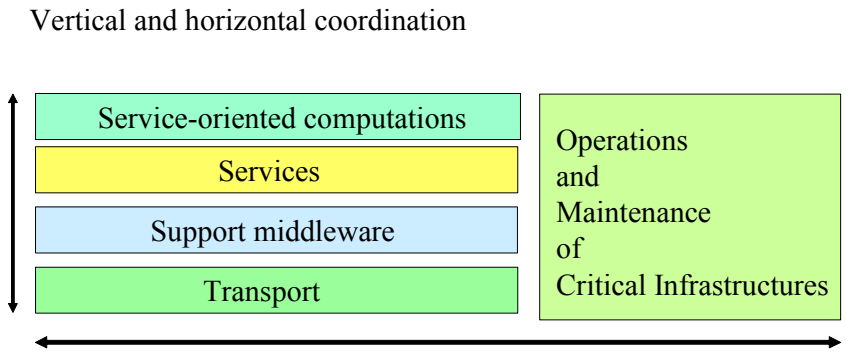

Figure 1. The layered model of CRISP systems. PNO denotes the Power Net Operations application and BO denotes the Business Operations. Both PNO and BO are composed of services.

have revealed software vulnerabilities in existing SCADA systems that have been causing different kinds of breakdowns. Since we have to cope with unreliable software, it is important to design execution environments that are as safe as possible, our attempt in that direction is

outlined in (Annex 1). By introducing applications based on services we avoid the hardwired 'stovepipes' and hence gain in flexibility, however, we now need coordination and 'funnelling' mechanisms to support creation and maintenance of trusted chains of services. Coordination is the topic of Section 5.3 and the 'funnelling' mechanism is addressed in Section 5.

The next generation power grids will be more information-oriented, meaning that there will be large amounts of information transferred between nodes, a high dependency of having correct information for mission-critical decisions. In addition to the data already transferred in the current information network, there will likely be other information, such as dynamic price information, transferred between parties, over some kind of information network, which will affect the power grid.

With many different type of nodes connected to the information network and with more information transmitted between the different nodes it is very likely that the information network will a very heterogeneous network; a network where nodes constructed by different manufacturers, from different hardware, in different generations and with different software configurations should be able to communicate and work together. If, for example, home consumers connect different devices in their homes in a network shared with producers there might be thousands of different types of devices connected in this network where the data transmitted may have a high impact on actual power grid.

With information networks being a central part of the next generation power grids and the information transmitted over these networks being important both for the actual power grid operations as well as for producer-consumer communication there are strong reasons for building the information network as a high-security dependable network. In our view the dependability of such a network cannot be assured solely by using communication-level techniques such as encryption of data transport, but we must focus on techniques for creating stable components; components that can be used to build a network that can survive communications problems, changes in environment, and possible attacks. One technique to create such components that is being investigated at BTH, is how to execute unreliable software safely, something that could be used to increase the dependability of the software used in devices connected to the information network Section 6 *Experimental settings*.

We have produced two papers *On Adaptive Aspect-Oriented Coordination for Critical Infrastructures* [36] and *Safe execution of unsecure software* [16] that complements the presentation in this Deliverable. The papers are for that reason included as Annex 2 and Annex 1, respectively.

The Deliverable is divided into six parts; to set the scene of some underlying security and integrity concerns a scenario of *trusted business cooperation* related to CRISP is given in Section 2. *Emergent technologies* related to the power grid operations are highlighted in Section 3. Section 4 *Information security and network security* includes also and a short background to BUSMOD, i.e., material on *business assessments*. Section 5 *Layered architectures*, gives us the background to *dynamic service oriented 'funelled' applications* contrasting 'stovepiped' old version as stated above. Section 6 *Experimental settings* give an account of methods and tools supporting our proposed *Framework* of Section 7. The Framework Information Security models and their Economics is rather (for reasons highlighted in the Deliverable) a framework on *Information Security Models and Their Economics* in a CRIS setting.

1.1 Scope

The scope of the report is to identify a Framework addressing Information Security models and their Economics. To that end we introduce some relevant background material in the areas of *emergent technologies* of relevance to CRISP in Section 3, issues of *Information security, including integrity, network security*, and *Business modelling* in Section 4. In Section 5 we take a closer look on *layered architectures*, including *'principles of funnelling'* to meet *robustness* and *dependability* criteria of complex distributed systems. The account of *Experimental settings* of Section 6 completes the background to our *Framework* of Section 7.

We have the following inputs from earlier deliverables. Input from D1.1 Functional specification of highly distributed grids, D1.3 Distributed generation as a means to increase system robustness, D1.5 Intelligent load shedding, and D2.1 Requirement specifications of intelligent ICT simulation tools for power applications. In D2.1 the following CRISP services have been identified:

- Fault detection and diagnosis
- Demand and supply matching
- Intelligent load shedding
- New energy based business processes
- Secure operations

This deliverable focus on issues related to secure operations of the three services and the related business processes. The couplings of the material in this deliverable and the CRISP experiments in WP 3 are given in D2.4 *Dependable ICT support of Power Grid Operations*.

2 Scenarios: New business models

Energy markets have up to now been focusing on production, distribution and selling of energy (kWh). Due to de-regulations of the energy market, e.g., introduction of renewable energy sources, the kWh related market becomes more and more volatile and knowledge intensive with an increasing focus on customer demands. However, even more interesting and challenging is the shift from energy based business processes towards new services and business partners for utilities outside the energy sector.

To indicate this new kind of business models we introduce the following scenario depicting two new models of Business to Business (B2B) processes based on energy related services, Figure 2.

In fact, we have a simplified version of the scenario below as described by the *"behind the meter" business model* (e.g., in a CRISP setting) based on introduction of smart meters at customer side (WP 3). The idea being that information from meters can be used for billing or supporting other services, i.e., the smart meter is functioning as a server or gateway to the customer. ENEL is at present fielding in the order of 30 million "smart meters" in Italy with these kinds of applications in mind. In short, issues related to ownership and proper use of information is already an issue between different actors in emerging energy markets.

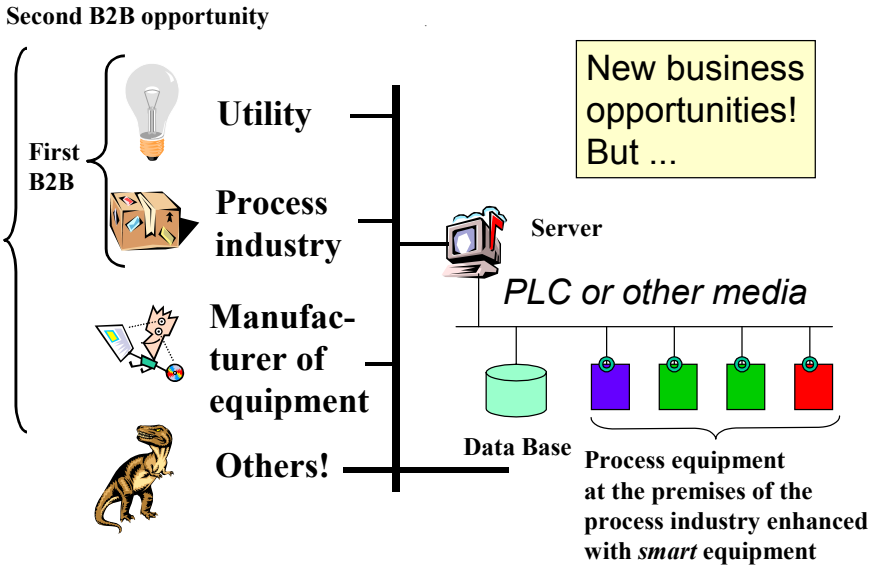

Figure 2. New Business to Business opportunities based on energy related services

In the scenario, we are focusing on *information trading* based on information created by and communicated between smart embedded equipment in a process industry. The business partners are in the first B2B scenario a utility and its customer, e.g., a process industry. In

the second B2B scenario manufactures of equipment are also involved in the business processes. The business processes are based on two principles:

- Energy saving
- Smooth production

The incentive for the utility in this co-operation is by translating selling pure energy (kWh) into the service of smooth production it can 'buy back' energy from its customer and resell it on the energy spot market for a substantial profit simultaneously billing the customer at a higher premium price for providing smooth production. The energy saving is enabled by knowledge of the processes and equipment at the customer side. It is in fact a well-known fact that there is a large potential in saving energy in the process industry if we can utilize the information of the run-time situations.

The incentive for the process industry is prospects of lower energy consumption and better process management. However, the data from the equipment potentially also inform the utility about the processes of the process industry. Furthermore, the information also gives valuable insights of the working of the equipment themselves, which is of high value to the manufacturer of that equipment, i.e., the second B2B opportunity in Figure 2. We thus have in this scenario a very interesting emerging system around the generation, processing (manipulation and misuse!), sharing, and trading of *information assets* where possibilities of innovative business processes emerge.

2.1 Trusted cooperation

However, preconditions for *trusted co-operation* between parties in the potential Business-to-Business (B2B) information system, mentioned above, includes trust in that, although information is partly shared, *no harm* to either party can evolve intended or not. In summary, the actors in this scenario would demand a trustworthiness of the information system both with respect to functionality and aspects of security. Note that these trust criteria are *independent of* whether or not the shared intranet is connected to Internet or not. In the latter case the demands remains the same but are enforced by the risk of *possible malicious intruders*. As a matter of fact these concerns of security and integrity of information handling is valid for most non-trivial applications of 'Smart houses/offices' or B2B applications.

The concept of trust has been a topic of investigation in several disciplines from philosophy to engineering. We take in this report a pragmatic point of view, i.e., how to design, implement and maintain Trustworthy systems. A challenge is to operationalise aspects of trust, i.e., aspects of system attributes that can support trust in the behaviour of a system. Well-known examples of operationalisations are support for good explanations of system behaviour and good support at breakdowns of services. These, and similar kinds of operationalisations, can be grouped under the general heading of:

- Trust in a proper behaviour of the artefact. The artefact should have *the intended functionality* with an understandable degree of quality of service.

In fact, this is a common criterion of trustworthiness of artefacts in our physical world. Techniques to achieve these criteria include *validating* selected criteria using different kinds of testing. In certain safety-critical systems we also try to *verify* implementations against design

specifications. However, it is well known that even if we have successfully validated parts of a software system there still remains *vulnerabilities* due to the complexity of the software (could be millions of lines of code, c.f., the US-Canada blackout). Illustrative examples include operating systems such as Windows (NT) or SCADA systems and implementations of complex security functions! These vulnerabilities are detected regularly during the lifetime of the product. Often this detection appears when the product is used in a new context, integrated with other software products or on new platforms. Vendors on a regular basis release repairs 'patches' to address those vulnerabilities. Most of those *known vulnerabilities remain unattended* by most users due to lack of information, resources, or policies. The important point is, however, that complex software will always have known and unknown vulnerabilities throughout their lifetime. We will return to these aspects in Section 6 Experimental settings.

An *adversary can exploit* these vulnerabilities into *serious threats and attacks* towards the owners and users of the system/software. Especially this can happen when the system is connected to larger networks such as Internet. This means that new concerns affecting trust in system behaviour emerges. These concerns can be summarised as:

- Trust in that no *hidden or unintended* side effects appear during the use of the services/products. The artefact should *not do anything else* than the intended function. That is, allow adversaries to harm you, your system, or your electronic assets.

The issues of *electronic privacy, integrity*, and *security* are addressing these kinds of unintended behaviour. It should be noted that these kinds of behaviour are *almost impossible to detect by testing individual components* of the system. This also means that there is a *continuous task* to protect a system against malicious intruders. We will return to this aspect in Section 6.

As a matter of fact, several studies show that distrust in electronic services (e.g. electronic payment) is of a major concern and an obstacle to take-up of e-business applications at this moment. As world wide efforts focus on designing and implementing new services and product of the Embedded Internet both kinds of trustworthiness, mentioned above, in embedded services have to be addressed up-front. Distrust in CRISP services might lead to dramatic business failures due to the role of the power grid as a Critical Infrastructure.

We will address security issues related to business processes in a Section 7. However, before doing that we have to assess current and emergent technologies of relevant information systems for future electric grid operations and business processes, Figure 1.

3 Emergent technologies

In this section we identify some crucial requirements on future information systems in order to support new business models such as those indicated in previous section. Requirements on future distributed information systems often use characteristics such as *open*, *flexible*, *autonomous*, *self-healing*, *reliable*, and *dependable*. Typical applications include *ubiquitous computing*, *ambient intelligence* and *Information GRID* systems. Technologies include high-level concepts such as *Multi-agent systems* (MAS) or middle-ware technologies such as *service-oriented architectures* (including web-services proposed by IBM and Microsoft's .NET). We will return to some of those issues later. The purpose of this section is to give a short account of the ongoing shift in utility based information systems from inflexible bottom-up information gathering – top-down directed action architectures towards net-work oriented open information processing systems.

A typical control system architecture of power systems is given in Figure 3. The hierarchical architecture of the information system closely mirrors the hierarchical structure of the classical power grid. The main drawbacks in today's control systems are as follows. The data and *information exchange* within the system are *structured into a strong hierarchy*. Measurement values, or process data, are transmitted from lower to higher levels, while control information is transmitted vice versa. The communication bandwidth within a station, and that between stations and the system control levels are limited. Several *vendor specific protocols* are used with the need for specific gateways (GW). *Fixed information channels*, e.g., telecontrol systems, are supporting the hierarchy.

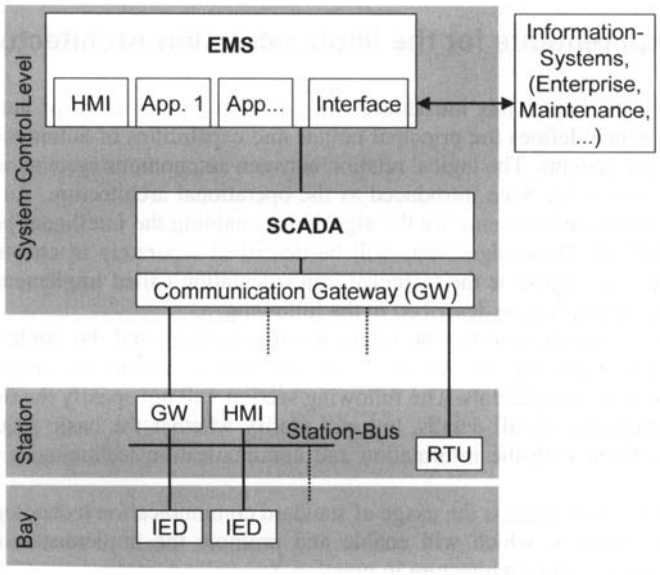

Figure 3. Standard control system architecture in power systems

The applications within the Energy Management System (EMS) are *not* modularised and are

based on the central SCADA database information. Interconnections to other information systems, such as enterprise information or maintenance management systems, have to be handled by *specific* interfaces. The SCADA system itself contains both real-time and configuration data. The *Intelligent Electronic Devices* (IED) like protection devices on the station and bay level are also *vendor specific* with varying functionalities, interface protocols and data models.

As we have seen from the scenarios above, new business models and opportunities will rely on advanced information processing *between components* on the Bay, Station, and System control levels as well as bi-directional flows *between hierarchical levels* and *customer sites*. To accomplish this flexibility we have to introduce network centric architectures to replace the architecture of Figure 2. Furthermore we have to rely on:

- protocol standards, and
- data standards,

to support flexibility. The following Figure 4 introduces a *network-centric* information system for operations of future energy based business such as high-quality generation and distribution as well as new Business to Business processes based on energy services.

The basic topology of the system is outlined in Figure 4. Local area networks (LAN) are used for data communication within a substation. The communication between substations is implemented with wide area networks (WAN) to which the substations are connected via gateways (GW). Some IEDs from substations without a substation automation system can be directly linked with the WAN. To fulfil the requirements concerning bandwidth reservation and security for real time control systems, the WAN for the use of electrical power system is separated from public WANs.

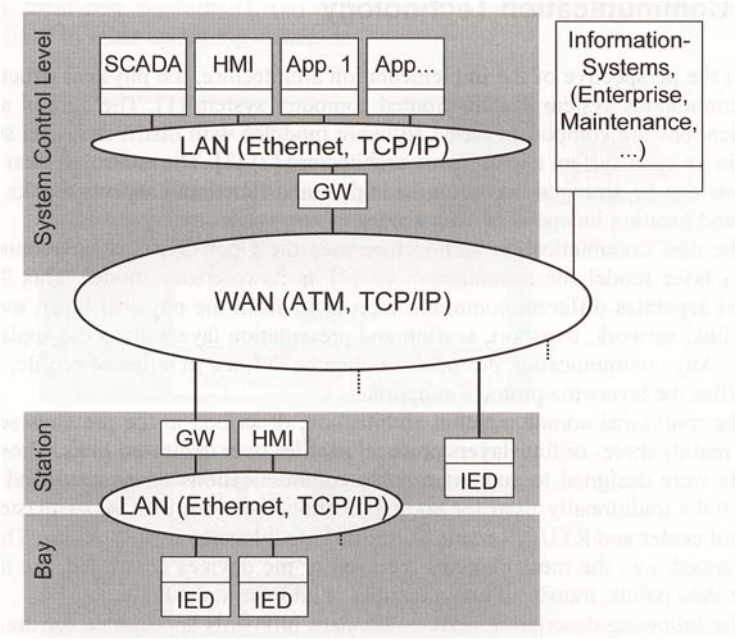

Figure 4. Network-centric architecture for energy based information systems

The hierarchical structure of the information system in Figure 4 has an *orthogonal system level* structure for distributed systems. To illustrate this view we use the ISO-OSI seven layer standard reference model of Figure 5.

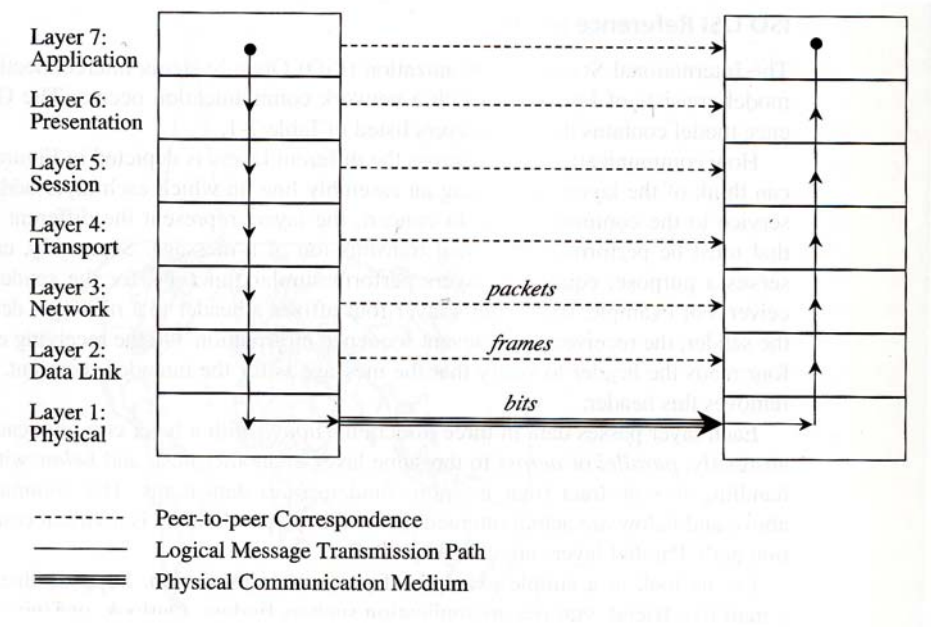

Figure 5. The ISO OSI Reference Model

The communication part of Figure 4 involves the four lowest levels of the reference model of Figure 5, whence the processing components (including the IEDs) relates to the higher three levels of the OSI model. We will complement Figure 5 with refinements in Section 5 *Layered architectures*. The basic principles to achieve higher flexibility and intelligence in the system are:

- Use standard protocols
- Identify appropriate data models
- Introduce 'smart components', i.e., 'agentified components' when applicable on all levels of the hierarchy of Figure 3.
- Introduce service-oriented architectures to support reconfiguration and flexibility of applications.

We will return to most of those issues in later sections such as Section 5.

We conclude this section with a short overview of current trends in standardisation of data models (supported by IEC) that are a crucial requirement for the information processing and management underlying future business processes. We also introduce some basic principles of *agent system, web services* and *network security*.

3.1 Data models for information system related to the energy sector

The communication protocols of the four lower levels of the ISO OSI Reference model enable data to be 'physically' exchanged between modules of the system. Traditional and still widely used protocols usually only implement the lower layers of the reference model. The data model is in this case simply an array containing signal addresses (or data points), and there is no notion of devices at the protocol level. This type of communication is an *anonymous exchange* of data points, because a physical device that receives the value knows neither what the meaning of the value is nor which physical device sent it – it only knows that the signal comes from a given address. The same comment applies to the sending device. To give semantics to the values used by the applications at any end of the communication channel, there must be a mapping facility at each end that has to be *configured* so as to associate data points with meaningful physical objects (i.e., circuit breaker) and their attributes' values (e.g., status = 'open').

This section will focus on the data models specified in the standards [38], [39], which provide for more *abstract* means for enabling communication (i.e., exchange of data), for both real-time and non-real time tasks such as system configuration or inter-application communication (i.e., message sending as needed in earlier discussed scenarios).

In the following paragraph we briefly introduce *three emerging IEC standards* that define more *elaborate data models*. Their application can eliminate the shortcomings of point-oriented, anonymous data exchange, and enable direct interactions between autonomous components. Although all these standards are still only in draft versions, some of their parts have already been adopted as *de facto* standards and even implemented by some device manufacturers and system vendors.

The standards are *IEC 61850* aiming at data modelling for *sub stations and devices*. Another standard in progress is *IEC 61970*: Energy Management System Application Programming Interfaces (*EMS-API*). The third standard is *IEC 61968*: *System Interfaces for Distribution Management*. This work is in progress and is closely related to and coordinated with the efforts on IEC 61970.

The IEC 61850 is basically a *communication* standard but a great effort has been invested in domain analysis. This effort has resulted in an *elaborated domain model*, which contains also the data model. The main abstraction of the *domain model* is the *Logical Node* (LN), which can be seen as an atomic functionality available within the substation, from the substation control system, to protection and control devices, to the process itself. An LN holds data, classified into a number of Common Data Classes, which are the main abstractions of the *data* model. There are efforts to provide a *formal model* of IEC 61850. Each LN encapsulates the data it needs for performing its functionality or behaviour. That is, not only typical operational data but also different configuration data. This implies that the devices can describe themselves to the system.

In the operational context, devices are servers and clients, and thus can be queried or query other devices through ANSI services. For configuration purposes, the IEC 81850 defines *Substation Configuration Language* (SCL). SCL is an XML Schema, with the elements and attributes reflecting the domain model. So, the self-description capability of IEDs can be available in a standard human and machine-readable way, through an XML instance file. Additionally, SCL allows one to configure the communication-related attributes of an IED as

well as to describe the equipment and communication topology within the substation.

The concept of abstract LNs, which model atomic behaviours within devices and systems, encapsulate own data, perform reasoning, and collaborate with other LNs through server-client mechanisms, can facilitate implementations of the IEC 61850 with means of software components, Depending on the *capabilities* of the LN it can in effect be an *agent* and/or a *eService*. We will come back to this issue later.

The standard IEC 61970 (EMS-API) defines means that facilitate 'opening' of traditionally closed and monolithic EMS/SCADA systems, which equip the network control centres. The standard defines a *data* model, called CIM (Common Information Model), and a set of application programming interfaces (APIs) used to manipulate the EMS/SCADA database data.

Contrary to the IEC 61850 for substations, this standard does not specify any particular communication protocol. Rather it specifies concrete APIs, which can realized by a protocol with the OSI profile appropriate for the given execution environment. There are two global sets of APIs. One set is intended for fast access to real-time data, typical used within SCADA applications, such as data acquisition front end or an update of human machine interfaces with the process data. The other set of APIs enables near-real-time access to the full network model or its parts, typically used in EMS on-line applications, even out of EMS for, e.g., long time planning or reliability analysis applications.

A notable contribution of this standard is the data model CIM. CIM is an abstract model that describes the domain known to EMS/SCADA systems as a set of objects with attributes and relations to other objects. The model is defined in the standard UML. The CIM model is maintained using a CASE tool. Similar to the IEC for substations this standard also defines the serialization format. CIM and the APIs defined in IEC 61970 represent obvious means that support open EMS/SCADA systems as described above in Figure X2.

The last standard that will be mentioned here, IEC 61968: System Interfaces for Distribution Management is designed for Distribution Management Systems (DMS), which typically must communicate with network operation and process control systems (EMS/SCADA) and substations, respectively), as well as with different enterprise level systems, such as customer management, resource planning and maintenance and outage systems. Therefore, the objective of this standard is to define a set of messages that DMS needs to exchange with other systems within the enterprise.

This standard is complementary to the IEC 61970 (EMS/SCADA) for two reasons. First, it uses CIM as its domain model and extends it with the physical objects and concepts that are relevant for distribution networks only. The second reason is that this standard also uses the APIs defined in IEC 61970, where applicable, and extends them with APIs that allow inter-application messaging. This means that these extended interfaces can be used by EMS/SCADA systems as well.

Messages in IEC 61968 are specified in XML. Since the time scale for inter-application messaging is not real-time critical, the implementations are likely to simply use some XML-based communication protocol such as SOAP. The SOAP protocol itself is a key technology for web services (c.f., IBM and Microsoft's .NET).

3.2 Agent technologies in energy applications

Agent technologies and multi-agent systems have emerged as a suitable technology for *analysis and design* of complex distributed systems. As a matter of fact there are several research agendas worldwide addressing Computational Intelligence and Agent technologies applied to future energy based operations [13], [18], [19], [20]. The following areas have been addressed:

- Multi-agent negation models for power system applications
- Multi-agent approach to power disturbance diagnosis
- Multi-agent approach to power system restoration
- Agent technologies applied to the protection of power systems
- Multi-agent coordination for secondary voltage control
- Agent based power system visualization
- Multi-agent systems controlling islanding criteria
- Operation of quality control center based on multi-agent technology
- Computational markets for energy management

All these multi-agent approaches are aiming at supporting high quality energy distribution and management. So far, very few attempts have been on identifying new business processes as such.

The appeal of an agent approach lies in the natural coupling in high level modelling of distributed systems with autonomous component. In short, an agent is a software module with *own* control and with a natural *communication focus* based on a high level Agent Communication Language (ACL). The state of an agent is often described in high-level concepts such as the BDI-triple (*Beliefs, Desires, Intention*). The high level ACL is composed of two parts (c.f., *speech act theory* of natural languages) the *purpose* of the message and the *content* of the messages. The purpose of the speech act could be; ask, inform, tell, advertise, negotiate, accept/reject, and so on. The content in most new ACL is described using XML. This means that, e.g., the IEC data models above can be communicated between agents. As a matter of fact agents can be naturally implemented on all three hierarchical levels of Figure 1 provided the modules (such as an IED) has own control.

Issues of *coordination* and *negotiation* are at the core of multi agent systems and will be investigated in more details later, Section 5.3.

3.3 Web services and agents

The implicit focus of multi-agent systems is in developing *closed* applications with autonomous components. In many cases this approach is very natural and appropriate. However, sometimes it turns out that closed applications (not only multi-agent systems) uses similar components or have dynamic requirements.

The advance of web-technologies is aiming at reuse of *net-based services* for creating new (*open*) applications. In this context web-services are components that can be *advertised* (by description of the service), found and used using tools for *look-up* and *access*. The implementation of the service is *transparent* from the users point of view. Again, we note that the IEC standards support creation of services.

Key technologies of web services are maintenance tools and tools for integration (*conjunction*) of services. *Service-oriented architectures* are the support structures for creation and maintenance of services and their applications.

Services in the original setting (IBM and Microsoft) are passive components. Obviously we sometimes would like to have active object. To that end, there are ongoing research agendas aiming at combining agent technologies and service technologies into *pro-active services* created and supported by Online engineering. This is in fact the road we are pursuing at BTH, Section 6.

4. Information security and network security

Network security is a branch of the more general topic of *Information security* [2], [3], [5], [14]. Network security is relevant when we have network-centric applications [1]. Security is about protection of assets. *Risk analysis* is a part of a comprehensive *information security strategy* [10], [11]. Information security involves identification and risks analysis of *threats* that can utilize *vulnerabilities* to cause harm, Section 4.2 and Section 6. Information security thus involves *organisations* and *people* on one side and the technologies (*computer security* and *network security*) on the other hand. A living and implemented set of *policies* is necessary to *enforce* and *validate* identified measures.

A rough classification of protective measures distinguishes between:

- Prevention
- Detection
- Response

It is very important to understand that IT security is not only about products; it is in effect an ongoing process [3]. This process is guided by a well grounded and operationalised *security policy*. We introduce the following time based security framework to illustrate the basic underlying principles of a process oriented IT-security model. The model is an extension of a Time Based Security (TBS) model of Schwartau. We have the following basic notations:

P(s(t), ΔP(a, s, t)) denotes the *Protection* of an IT system at time t, where s(t) is the *strength* of the protection at time t, and ΔP(a, s, t) denotes the *duration* of the Protection with strength s under an *attack of type a* at time t.

D(s(t), a, ΔD(a, s, t)) denotes the *Detection* capabilities of an IT system at time t, where s(t) is the *strength* of the detection mechanisms to detect an *attack of type a* at time t. ΔD(a, s, t) denotes the *time interval* it takes to *detect* an attack of type a occurring at time t.

R(s(t), a, ΔR(a, s, t)) denotes the *Response* of an IT system under an a attack of type a, where s(t) is the *strength* of the response measures to *counter an attack of type a* at time t, and ΔR(a, s, t) denotes the *time it takes to effectuate* a proper response of attack of type a occurring at time t.

We can now quantify the *Exposure time* ΔE(a, t) related to an attack of type a on a IT system with protection P, detection D, response P at time t.

$$\Delta E(a, t) = \Delta D(a, s, t) + \Delta R(a, s, t) - \Delta P(a, s, t)$$

From (1) it follows that we will have *control* of the situation of an a attack by an adversary (internal or external) *if and only if the Exposure time* ΔE(a, t) *is non-positive*. If ΔE(a, t) is a positive number (a time interval) we have to address an open exposure to the adversary during that time interval.

We often have the situation that $\Delta P(a, s, t) = 0$, due to, e.g., new types of attacks, bad implementations or neglect of known vulnerabilities. This means that despite installed firewalls we might have a total exposure to certain attacks. A specific case is broadband installations in homes/offices where the general rules is that $\Delta P(a, s, t) = 0$!

We also can have the situation that $\Delta D(a, s, t)$ and/or $\Delta R(a, s, t)$ can be too long (e.g., longer than $\Delta P(a, s, t)$). The strengths of Detection and Protection depend to a high degree on how well security policies are set up, implemented and perceived by people involved. The same situation holds for the strengthening factors (learning and enforcement) s(t) of P, D, and R.

In short: A good protection (a non-positive $\Delta E(a, t)$ of (1)) depends on a continuous assessment involving all aspects of the co-evolution triangle. That is, a choice of appropriate technologies, policies and implementations to meet potential attacks on valuable assets of the parties involved.

As a matter of fact, robustness of the electric grid (overload protection and so on) also follows an equation of type (1) above. In this sense there are several common features of some of the mechanisms involved. An interesting common field is development of smart Detection systems based on patterns and early warning models with learning capabilities (smart sensors). Running smart automatic and continuous vulnerability tests and upgrading can also simplify strengthening of security component, Section 6 Experimental settings and Section *7 A Framework for cost-benefit analysis of securing assets*. Promising technologies here are adaptations of multi agent systems (MAS) and related services.

The following types of threats to an IT system have been identified:

- *Confidentiality* threat. The disclosure threat involves the dissemination of information to an individual for whom that information should not be seen.

- *Integrity* threat. The integrity threat involves any unauthorised change to information stored on a computer system or in transit between computer systems.

- *Availability* threat*.* Prevention of unauthorised modification of information but assure that an authorised user is guaranteed access to the resource at the right time and in the right way.

- *Denial of Service* (DoS) threat. The DOS threat arises whenever access to some computer systems resources is intentionally blocked as a result of malicious actions taken by another user.

- *Network security* threats. These kinds of threats are due to the distributed nature of the systems.

Building and maintaining Thrustworthy systems, Section 2.1, includes a vulnerability analysis, assessment of values to be protected, development and evaluation of threat models followed by a cost benefit analysis to take appropriate security measures at an appropriate risk level. A useful tool in this analysis is the standard ISO 177 99 or Common Critera. Selection of appropriate technologies (cryptography, key management, firewalls, VPN, IPSec, and so on) gives means to protect against unintended disclosure, integrity, DoS, and network threats. These mechanisms, including assessments of the value of assets and models of threats, are the focus of *vulnerability analysis* and appropriate security measures in a particular context. Other crucial factors for Trustworthy systems are

mechanisms supporting *accountability and liability*. Issues of ownership and responsibilities coupled with mechanisms of authentication, non-repudiation and logging are basic in this context. We will address these issues in sections below. Lastly, proper implementations of security measures include a choice on appropriate levels of the computer and communication architectures to address (application oriented versus generic security measures) as well as assessments of other trade-offs.

Besides *time-based security* (as above) we also have to address *defence-in-depth* of our information systems. The latter aspect is due to that systems in Business to Business applications *do not* have a well-defined perimeter. We conclude this section with a simplified security model of our system in Figure 6.

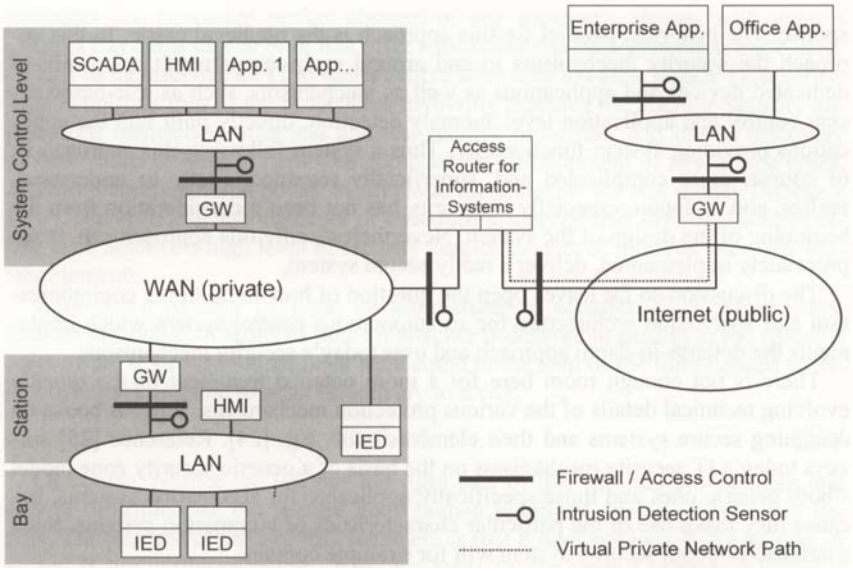

Figure 6. Topology of an in-depth security model related to energy based information systems

It should be noted that multi-agent systems typically operate *within* a security perimeter, whence web services are required to operate *between* security zones [4], [6], [7], [8], [9].

In this section we have addressed technological aspects of future information systems supporting high quality of energy operations as well as supporting new business models based on energy based services. In a following section will identify the CRISP-related aspects of such systems, but first we address some issues concerning protection of information.

We recommend reading of the IEC/TR 62210 Ed 1.0 technical report concerning computerised supervision, control, metering, and protection systems in electrical utilities [41]. Deals with security aspects related to communication protocols used within and between such systems, the access to, and use of the systems. The report also discusses realistic threats to the system and its operation, the vulnerability and the consequences of intrusion, actions and countermeasures to improve the current situation.

4.1 Protecting confidentiality and integrity of information assets

Returning to the B2B scenario of Section 2, on information trading among partners, we face the challenge to provide an information model that fulfils the following criteria:

- Collect information from sensors in 'Smart network components' and 'Smart equipment' to be used in different applications.
- Manage the threats towards disclosure and integrity of information
- Manage proper operations of actuators
- Support trust by enabling liabilities and possibilities of proving violations of rules or norms
- Operations in dynamic environments!

We propose a *combination* of the Bell-LaPadula (BLP) model originating from defence applications and the Clark-Wilson (CW) model building on business practises, [ref] and [ref]. The first model protects against *disclosure threats* whence the second models protects against *integrity threats*, c.f., Section 3.4. The second model also provides means to address issues of *accountability* and *liability*. Furthermore, the models support *scalability* and *change*.

The Bell-LaPadula disclosure model introduces a *level structure* on the information. That is, collecting data from sensors of 'smart equipment' and 'lifting' the raw data into higher-level application specific information objects, Figure 6 and Figure 1. This allows us to re-use data, collected by sensors, in *distinct* higher-level application objects.

- *Abstractions* of low level data from (smart) sensors to high level information belonging to a service or an application
- The abstraction procedures follow the *criteria* of Bell-LaPadula which *guarantee* nondisclosure.
  - NWD - No write down
  - NRU - No read up

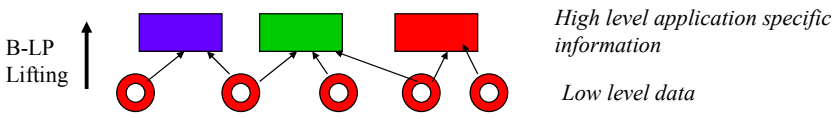

Figure 7. Information model. Step 1, abstraction

The BLP rules of NoWriteDown (NWD) and NoReadUp (NRU) will protect against unwanted disclosure of the higher-level information objects. Returning our scenario of Section 2, we can thus create higher-level information objects *own by* the different actors and *protected against* unwanted disclosure. The well-known short-comings of BLP are not applicable in our case. Furthermore, the BLP 'lifting' also provide us with an information model structure suitable for the CW model supporting information integrity.

The Clark-Wilson integrity model is used on the high-level application data objects created by the BLP model. The CW model has 9 model rules (Transformation Procedures) and a CW triple relation (s, t, d), where s denotes the *subject* acting on the *data* object d using the *transformation* rule t. We extend this model to include a *context* c. That is, our extended CW relation is the quadruple (s, o, t, d). The inclusion of a context c allows us to connect the transformation rule to a context where it could be applied. The context could for instance specify the role or competence for the subject to be allowed to execute the transformation t in the given context.

The data set of the CW model is a union of the Constrained Data Items (CDI) and the Unconstrained Data Items (UDI).

The CW model rules are:

1. Integrity validation procedures (IVP).
2. Application of a Transformation Procedure (TP) must maintain the data integrity of CDI.
3. A CDI can only be changed by a TP.
4. Subjects can only initiate certain TPs on certain CDIs.
5. CW-quadruples must enforce some appropriate separation of duty policy on subjects.
6. Certain special TPs on UDIs can produce CDIs as output.
7. Each TP application must cause information sufficient to reconstruct the application to be written to a special append-only CDI.
8. The system must authenticate subjects attempting to initiate a TP.
9. The system must only permit special subjects (e.g., security actors) to make changes to any authorisation related lists.

Our proposed framework for protection of information integrity and unintended disclosure also provides means for auditing enabling liabilities and assessing compliance of rules and norms (rules 5 and 7 above). In short, we have a tool for addressing basic trust aspects mentioned in Sections 3 and 4 above. Again instantiations of the framework in dynamic contexts might be facilitated by proper use of multi agent technologies (MAS).

4.2 Summary of vulnerabilities, threats, and risks

In order to develop economic models supporting implementation of cost efficient security measures and policies it is of outmost importance to identify potential risks for each business partner in cooperation as exact as possible. Vulnerabilities always appear at *interfaces*, being between hardware, operating systems, software, networks, and people. Firstly, not only *relevant vulnerabilities* have to be addressed. By relevant vulnerabilities we mean those that can be *exploited* by an adversary, that is, those who pose a potential threat. In order to exploit the vulnerability the adversary must have the *means*, *opportunities*, and *motif* (MOM) to exploit the vulnerability at hand. Assessments of the exploitability are termed *security management* and are indeed processes. There are several models of security management based on models from mathematics, economics, and psychology. There is also a clear overlap between models supporting *dependability* analysis and some models for security analysis of IT-systems. As a matter of fact this intersection is most promising for our investigations. However, to be fair, security management for IT-systems is still in its infancy. A state-of-the-art books are [9], [10], [11], [12], [14].

Independent of the methods to handle security management the verdict of our efforts is always given by the market or end user in terms of trustworthiness of services offered, Section 2.1. In short, un-trusted services will not prevail for long.

4.3 The BUSMOD model of business assessments

Trustworthy and cost-efficient protection of business model based on energy services is a core topic of CRISP. Our scenarios in section 2 indicate that the complexity of the supporting infrastructures can be daunting. It is therefore crucial to harness the complexity by carefully chose the type and complexity of end-user services. There are several methods and tools assessing vulnerabilities and risks. However, the connection of risk management (of security issues) with economic measures is less understood. Our staring point in this investigation is the BUSMOD and the e3-Value methodologies.

4.3.1 The goals of the BUSMOD project

The electricity sector in Europe and elsewhere is rapidly changing due to business as well as technology drivers. New players emerge (e.g. wholesalers of electricity) as well as new services (real-time pricing, home services, building management, IP over the electricity grid, demand-side management). A key question for players in the electricity sector is how to find corresponding new, competitive business models. In this section we will demonstrate how our *design*-oriented methodology can be used to explore new business models for the electricity sector, using technological advances in distributed generation technology, and taking into account government motivated re-regulation.

An important trend, occurring on the level of European policy, is a support of the technologies producing fewer emissions. In this connection should be mentioned the current promotion of expensive renewable technologies by government subsidizing schemes. These schemes lead to the emergence of new business models, however, since most of them rely on uncertain subsidizing schemes, the evaluation and sensitivity analysis of such business models is essential.

The goal of the BUSMOD project is to establish a Framework for assessments of business opportunities of future energy based processes. The Framework is itself based on the e3-Value methodology

4.3.2 The e3-Value methodology

The methodology has been developed by University of Amsterdam (UVA) during the last years. The methodology is explicitly addressing issues related to investments and profit-analysis of (potential) Business-to-Business processes [17]. The *e^3-value* methodology provides modelling concepts for showing which parties exchange things of *economic* value with whom, *and* expect *what* in return. The methodology supports feasibility analysis of business opportunities with several stakeholders. Specifically:

- Is the e-business idea at hand expected to be profitable for each actor involved?
- Are the supporting e-business information systems technically feasible?

4.3.3 Approach of cost/benefit analysis of investment in security

Our approach is to extend the e3-Value methodology with cost-benefit analysis of protection measures derived from risk and vulnerability analysis. In this effort we will in the CRISP project take advantage of the efforts done by our partner ECN in the BUSMOD project on adaptations of the e3-Value methodology. A real challenge is to introduce models of assessments of investments related to increase in security or trustworthiness of energy related business processes. First of all we have to define suitable business models including critical assets suitable both for a e3-Value assessment as well as for a security based risk analysis. Secondly, we have to provide a sound cost/benefit analysis concerning relevant countermeasures. Given the timeframe of the CRISP project and the fact that this approach relies on suitable business models and figures of values of assets cost of investments and so on we do not expect to be able to make any substantial progress beyond this rather general framework within the CRISP project itself. We return to those issues in Section 7.

5 Layered architectures

The basic conceptual three-tier layered model of future energy systems from D2.1 has to be further refined in order to support our design approach, see Section 1 Introduction, Figure 1. Specifically, the model of D2.1 represents a design view and has to be complemented with a view of the *running system* in order to support our system observation and maintenance activities, Section 6.5 *Online engineering*.

A layering model is introduced to provide a framework for layering all the communication related functions and protocols, and facilitate communication upgrades as technology advances. Layering can be viewed as a technical architecture, not an operational or systems architecture that is intended to foster understanding and illustrate the “building codes” of the system.

Service oriented distributed systems are recently a core activity within distributed systems worldwide. Of specific interest to us are activities within the Global Information Grid (GIG) system supported by US DoD in efforts on Network-Centric Warfare. The following figure, Figure 8, compares the layering models of the standards ISO-OSI and TCP/IP with the Global Grid layering model. The most differentiating feature of the GIG layering is the explicit introduction of a *Mission layer*. The Mission layer allows us to explicitly address and maintain high-level goals of mission and tasks. Hence we will be able to address *systemic sustainability and global coherence* issues. We will return to those issues later.

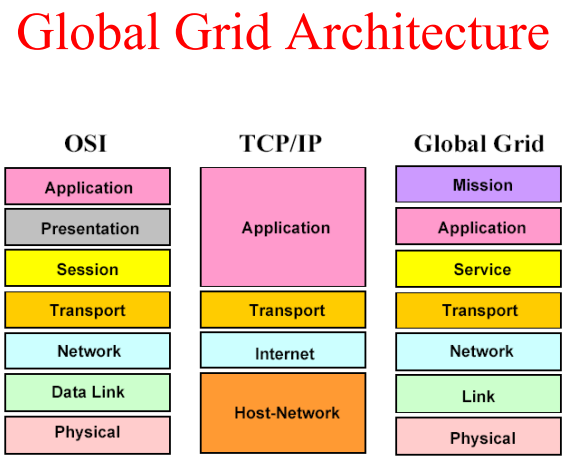

Figure 8. Comparisons of different layering architectures

The following Figure 9 gives an explicit mapping of the ISO-OSI model onto the GIG model.

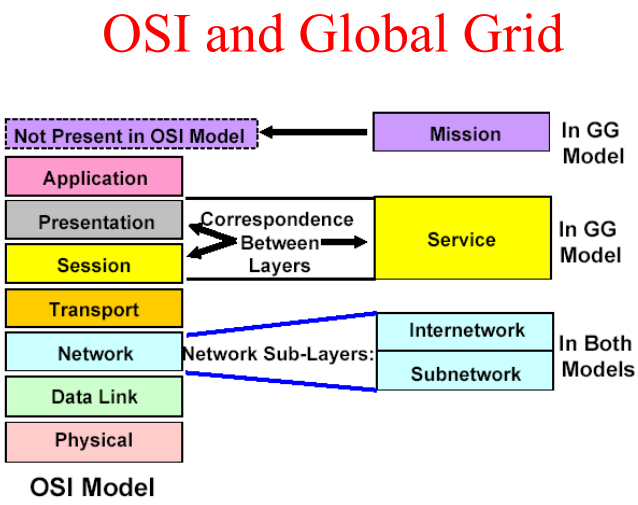

Figure 9. Mapping of the ISO-OSI model onto the GIG layers

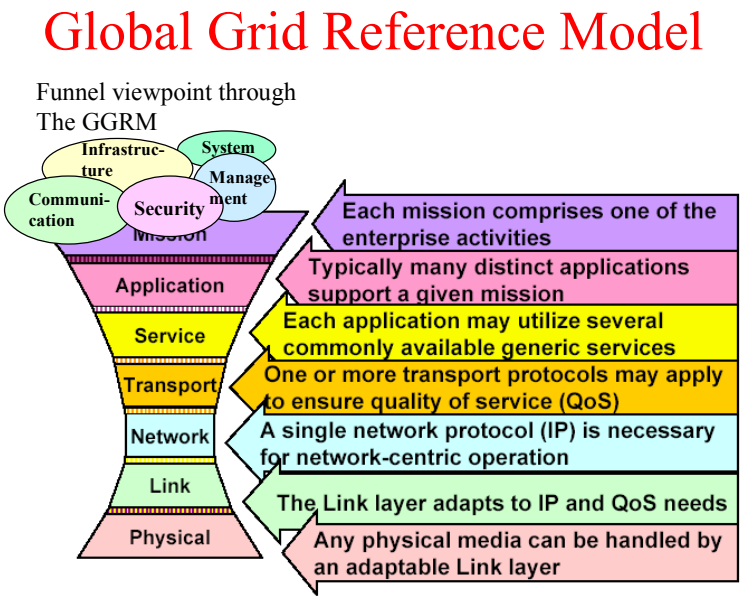

Figure 10. The Global Grid Reference Model

The second most important feature of the GIG, besides the Mission layer, is the explicit goal to account for *maximum interoperability* of networks building on a *single* connectivity protocol, i.e., the network IP. Whence we got connected via IP then we can choose

appropriate protocols below and above the network protocol. The choice of those protocols is determined by the QoS criteria of the applications under a mission. This context dependant choice is denoted *'funelling'*. The *'funnelling principle'* allow us to create the most appropriate choices of services when creating the PNO and BO applications of Figure 1. Furthermore, the Mission layer allows us to *coordinate those applications in a sustainable* way. The mechanism allowing this sustainable funnelling and coordination is denoted *Online engineering*, Section 6.5. Figure 6 above, captures the 'narrow-waist barbell' model of GIG.

Of specific interest for us is the *Security funnelling* function. This function can be illustrated as combining different *security services* on different layers of Figure 6 as indicated in Figure 11, below

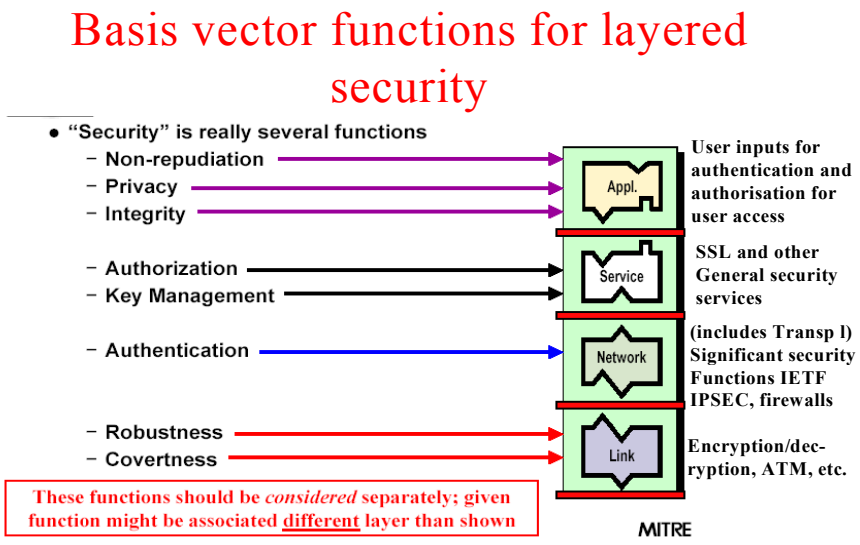

Figure11. Basis vector functions for layered security

The implementation and coordination of those functions to meet 'funneling standards' of CRISP are the topics of Chapters 6 and 7..

5.1 Vulnerabilities of ICT

In Section 4 Information and network security was addressed in general terms. There are several efforts addressing different aspects of network security, a state-of-the-art book is [9]. Most of those efforts are, however addressing detection of intrusions and measures to harness vulnerabilities. Another approach, as we do at BTH, is to focus on developing as secure software as possible [15] and, at the same time develop environments supporting secure execution of unreliable software, Section 6.

5.2 Safe execution of unreliable software

Many complex systems contain software components whose function is critical to the function of the system as a whole. Implementing complex functionality in software has many advantages over other approaches and is the only realistic method for many applications [11], [15]. Unfortunately developing software is error-prone and testing software to assert it fulfils dependability requirements is very difficult if not impossible. The result is that most software contains defects that may be trigged in run-time, either as a result of normal execution with some unexpected data or as a result of a direct attack, causing the software component to fail. In many cases such software failures cause severe security breaches that lead to a compromise of the entire system in which the software operate.

We are currently conducting research and experimentation with tools, methods and techniques for limiting the consequences for a system should a single component or part of a component fail, as well as how to increase the resistance in software against attacks directed at compromising the system security [16], Section 6.4 *Honey nets*. Recent years have seen a large increase in security related attacks on software, especially against networked systems and we believe that construction of a secure information network should begin with securing the components of which the network is built.

5.3 Coordination

Multiple cooperating agents must coordinate their actions in order to achieve their common goal. Traditionally this coordination has been implicitly described and implemented in the communications protocols and within the agent logic. However, the research community is acknowledging that the coordination functionality must be more explicitly defined in order to facilitate looser coupling between design and implementation, and to enable the agent to function in an open environment where it may coordinate its actions with other entities that where unknown at design-time. Consequently, a coordination middleware is necessary [36].

In this section we will examine the major factors that affect the design of a coordination middleware for the power grid. The middleware will be described by funnel viewpoints through the Global Grid Reference Model, Section 5.

5.3.1 Coordination of business operations

The agents involved in business operations will have different coordination characteristics than the agents involved in grid operation. This stems from the fact that the main purpose of the coordination of business operations is to *facilitate trade* whereas the main purpose of grid operations is to *maximize quality of service*.

The coordination that supports business operations will primarily focus on energy and information trade between a buyer and a seller. Consequently, while it is primarily a two-party communication the coordination infrastructure must maintain the global system invariant, G=C. Under no circumstances should two different buyers be given green light to consume “the same” generated energy. There are no hard real-time constraints on the coordination infrastructure in the business operations case, it is however very important that *consensus* is maintained between the business operations.

The business operations coordination is particularly interesting due to the necessity for both cooperation and competition. The businesses must compete in order to gain market share while cooperating in order to protect the whole system from collapsing. This is a standard setup for the Prisoners Dilemma – where sometimes some party must select a less desirable option for the benefit of all parties in the system.

5.3.2 Coordination of grid operations

For grid operation the intelligent nodes in the system will monitor the stability of the power system for any disturbances. Virtually all disturbances are caused by a specific *event* (e.g. a tree tripping a line). In the case of an event a response occurs in the system (e.g. frequency drop). The nodes may either directly detect an event or the occurrence of a response – nevertheless an *alarm* is triggered by the detector.

The quality of service definition with regard to grid operation is typically to provide a steady level of energy to as many customers as possible. Intelligence in the grid operation nodes must react to alarms in a way that maximizes the system's quality of service, e.g. by intelligent load shedding. It may be beneficial for several nodes in the system to take action – and thus maximizing the quality of service throughout the system – by coordinating their responses. Consequently, the coordination that supports grid operation has hard, or close to hard, real-time constraints since action must be taken before equipment is damaged. There are three windows of opportunity available to handle disturbances in the grid [INPG_WP11], each allowing more and more sophisticated control:

0.01 s -> 1 s	Voltage variations
1 s -> 3 min	Short disruption
3 min and more	Long disruption

While it may not always be possible for several nodes in the system to coordinate countermeasures in the shortest timeframe, the overhead in the coordination middleware should be minimal compared to communication.

5.3.3 Differences in coordination requirements

As the grid operates according to the n-1 criteria (meaning that the system should be able to handle loss of any one component without interruption), then so must the grid operation coordination too. Redundant distributed storage of state information is one way to achieve this. However, this may not be acceptable from a business operations perspective, as it could mean that sensitive corporate information is (partly) stored on a competitors system. Consequently, there is a contradiction in need between business operations and grid operations. The two systems cannot be built independently of each other though – both systems need to share information in order to operate optimally.

In the deregulated market it is feasible that different utilities will upgrade certain parts of their system while maintaining other parts. Consequently, over time the whole system will contain a diversity of both hardware and software components. Still, as the utilities have to cooperate or the network may fail, this diversity must be handled and/or controlled. It can be controlled by introducing more regulation, and handled by designing the system components to be both forward and backwards compatible. A flexible architecture provides several other benefits as outlined elsewhere in this document, and the cost for the utilities would likely decrease due to lower risk of e.g. dependencies to a specific third-party vendor.

In a flexible open system it is beneficial to consider coordination in isolation, realized as a middleware in the system. As long as the interfaces above and below the middleware are constant it will be possible to upgrade and introduce new agents into the system and allow the existing agents to coordinate their actions with the new ones. This is a key feature in any open system; that any given component should be able to interact with new components that where unknown to its designers, Section 6.5.

6 Experimental settings

At BTH we have an environment in which we can do experiments with network coordination and security properties of information networks and their nodes. With input from the consortium we could build a very realistic network modelled after the information flow in a regional power grid, and perform experiments on what would happen to the network in given scenarios.

In order to perform these experiments, we need to model a realistic regional information network. To successfully do so, we need to acquire the following information:

- Physical network to use as a role model.
- Brief overview of the foreseen services in the business and grid operations grid.
- A set of events and situations of particular interest to the consortium.

Some of these scenarios that we anticipate to examine are;

6.1 Availability and access of information assets

In order to achieve robustness in the grid operations, important information needs to be stored at several physical locations in the network. However, businesses will not appreciate if their private information is redundantly stored on a competitor's system. One logical solution to this would be to separate the two systems. However, this cannot be done since business operations need information from grid operations for billing and forecasting, and vice versa for the grid to know which Quality of Service is bought and paid for. Consequently, we need to research cross-compatibility in the coordination middleware.

6.2 System evolution

As parts of one operator's system within the global ICT system will likely evolve at different speed than that of competitors (perhaps due to different foci) we are presented with several options on how to model the coordination middleware. For both grid- and business operations we must ensure that the components in the system can coordinate with each other – even if some of them have been in the system for a decade. The coordination middleware must then be forwards and backwards compatible within components of the system.

6.3 Network attack scenarios

In an open network there is always the risk of attacks against a single node, a group of nodes or possibly even the entire network. While the first two cases might be acceptable under certain conditions, the latter is typically never acceptable. We can perform real attack scenarios against a realistic model of an information network to test different models of communication and network architecture.

6.4 Honey nets

An interesting method to check the interest of potential adversaries interests in your networks is to set up an 'honey net', i.e., a network that is designed to be a 'honey pot' where we can observe the behaviour of intruders. The intruders themselves are not supposed to know that they are attacking a decoy. By designing a honey net mirroring crucial aspects of the real network we can learn about potential threats and also test appropriate countermeasures to be implemented in the real network.

6.4.1 Tools for testing security of open complex systems

One of the projects at BTH is to develop tools and an environment for testing security properties in complex systems. Using this environment and these tools we can deploy systems in a large network, completely separated from public access, and analyze software, network and other aspects of the system and optionally perform full-scale attacks against a system or a part of a system.

Some of the tools developed include; automatic reconfiguration of machines based on a network role, separation of privileges in critical programs, and tools for constructing a dynamic network environment.

6.5 Online engineering

To meet the challenges of designing and maintaining complex software systems we have developed methodologies and tools supporting Online engineering. The basic idea is to provide design information on-line of the existing system during runtime. This approach has proven to be very fruitful in design, development and maintenance of complex knowledge-intensive systems. Applications are in the area of network-enabled capabilities including network-based warfare (NBW) and distributed eHealth [28].

As an operational architecture of the *running system* we propose the following four-layer model, Figure 12. Of specific importance is the *Domain layer* and the *Environment*. Support for run-time *observation, monitoring, construction, and instrumentation* is indicated in Figure 10 as *Funneling* viewpoints. The funnelling tools are cornerstones in connection the design models (as in Figures 10 and 11) to information sources of the run-time system. This connection is denoted Online engineering.

The tools and views of Figure 12 support the funneling functions of Figures 10 and 11 on-

line in the running system. The translation mapping and hence the connection of design criteria and components into on-line tools and support is rather straightforward. In effect the Mission layer of Figure 8 is mapped on the Domain level of Figure 11, the Application and Service layers are mapped on the System level, and the Transport, Network, Link, and Physical layers are mapped on the Fabric level of Figure 11. The *Environment level* of Figure 11 has no counterpart in Figure 8 *but reflects the embedding* of the system in the real world and hence serves as a *validation* tool of the intended behaviour. The Environment level is thus a crucial source for on-line observation and construction.

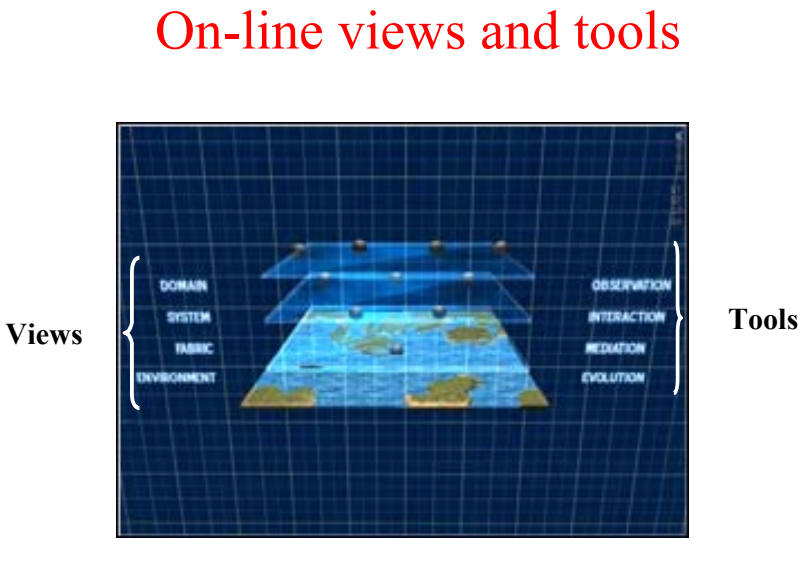

Figure 12. On-line views with supporting tools

We have by now quite good experiences of online engineering supporting sustainability criteria on open distributed systems [21], [22], [23], [24], [25], [26], [27].

7 A Framework for cost-benefit analysis of securing assets

7.1 The ideal framework

In this Deliverable we have emphasized that the task of defining and designing a Framework for assessing Information (Network) Security models and their Economics is indeed challenging [32] and Section 4.3. Many issues are at the moment research topics of several research areas (computer science, computer networks, philosophy, and economics [31], [33], [34], [35], [37], [40]) as has been indicated in the deliverable. We have also to develop practices from experiments and field tests to assess and ground the theories. We advocate such a pragmatic view within the CRISP efforts. The final outcome of this effort within CRISP will then be a *set of guidelines* operationalising a Framework supporting Information (Network) Security models and their Economics. The guidelines can than be evaluated and elaborated by our partners. As we have stated in Section 4.3.3 there will not be sufficient background material (business models, values) to develop and test a full-fledged model for cost/benefit analysis of investments securing assets. We do not, for instance, have any agreed upon set of metrics related to security (how do we know that system A is more secure than system B), hence how can we motivate, or assess the impact of, investments in security in a transparent way?

Given this situation, we propose a pragmatic way to gain more insight in those issues within the time span of the CRISP project.

7.2 A Pragmatic approach towards dependable future energy systems

Technically, the CRISP project focuses on models and methods to introduce DG and RES (e.g., μHCP) as well as new coordination mechanisms such as markets in present power nets in a dependable way. Details can be found in several deliverables from WP1, WP2, and, WP3. The following Figure 13, from Annex 2, complements Figure 1 but highlights the different coordination involved.

As illustrated in Figure 13, the state of grid operations affects the business operations (e.g. the available power affects price), and vice versa (e.g. service-level agreements affect the likelihood that a certain customer will be shed in order to save the system from complete blackout).

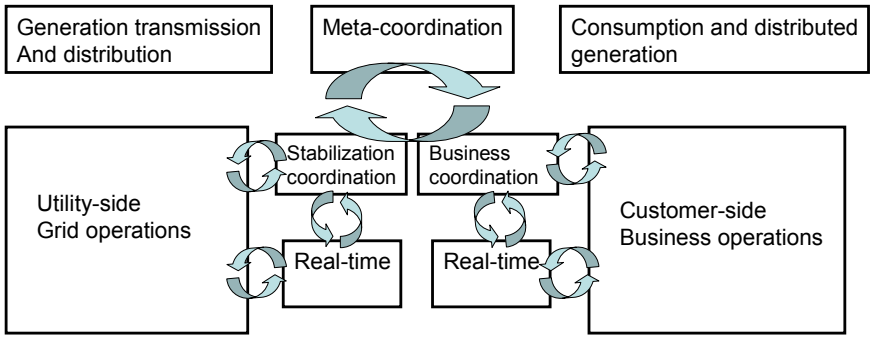

Figure 13. Coordination in the future power grid.

Finding a single coordination infrastructure to support these different high-level coordination goals (Missions of Figure 10) is impossible. We have to introduce a hierarchy of coordination mechanisms coordinated at the top level by a *meta-coordination*.

Our starting points in developing guidelines are mentioned in Section 7.1 are :

- Extensions of the BUSMOD framework, including the e3-Value framework, Section 4.3.
- The exposure time-based Protection-Detection-Response model of Section 4.
- The experimental settings, including honey nets, of Section 6. Honey nets will allow us to detect external threats to a particular network.
- Online engineering, Section 6.5, provides a tool for deployment and maintenance of critical infrastructures.
- Work on dependable coordination mechanisms (c.f., Annex 2).
- Work on dependable environments for execution of software (c.f., Annex 1).

From Figure 1 and Figure 13, we se that we have two *interacting* applications (missions) to protect, i.e., the Power Network Operations (PNO) and the Business Operations (BO). Since the PNO is the most well-known application (mission) at the moment, we suggest the following steps in developing our framework (c.f., D2.4):

1. A mini framework focusing on protection of PNO and connected to Field test C.

When we have a clearer picture of the missions (business cases) of the BO network we can address:

2. A mini framework focusing on protection of BO and connected to Field test A.

3. A framework focusing on protection of PNO and BO connected to Field tests A, B, and C.

It should be noted that advancements of dependable coordination mechanisms and dependable environments for execution of software meet requirements that have been identified in connection to recent vulnerabilities of power nets leading to black-outs costing our societies billions of US dollars. So, by engineering more reliable PNO environments we can implicitly learn more about how to model cost/benefit analysis of security/robustness investments.

8 Conclusions

The task of defining and designing a Framework for assessing Information (Network) Security models and their Economics for business models supported by next generation of DG and RES-networks is generally very hard and involves several challenges. However, we have outlined a pragmatic approach towards that end utilising the CRISP experiments in WP 3 as stepping-stones.

We have also in this deliverable identified and to some length discussed dependability and security requirements coupled to next generation energy-based business processes as a baseline in a suitable Framework. Technical details related this framework and CRISP experiments in WP 3 is presented and discussed in deliverable D2.4.

Needless to say, as our examples from the 2003 blackouts shows, when a energy blackout happens the consequences for the society (in costs and indirect effects) can be huge. It is indeed doubtful if those consequences can be recast into a risk analysis method per se. It is arguable that investments in securing critical infrastructures of the kind we are developing in future energy-ICT systems should be grounded on regulation (as in e.g., air traffic transport) rather than on free market stand-alone principles.

9 References

9.1. Books

[1] Allen, J.: CERT Guide to System and Network Security Practices. Carnigie Mellon Software Engineering Institute, 2001. ISBN 0-201-73723-X.

[2] Amoroso, E.: Fundamentals of Computer Security Technology. AT&T Bell Laboratories, 1994. ISBN 0-13-108929-3.

[3] Anderson, R.: Security Engineering. A guide to building dependable distributed systems. John Wiley Sons, Inc., 2001. ISBN 0-471-38922-6.

[4] Cole, E., Newfield, M., Millican, J.: GSEC Security Essentials Toolkit. Que Publishing, 2002. ISBN 0-7897-2774-9.

[5] Gollman, D.: Computer Security. John Wiley & Sons, Inc., 2000. ISBN 0-471-97844-2.

[6] Kruse II, W. G., and Heiser, J. G.: Computer Forensics. Incident response essentials. Lucent Technologies, 2002. ISBN 0-201-70719-5.

[7] Northcutt, S.: Network Intrusion Detection. An Analyst's Handbook. New Riders Publishing, 1999. ISBN 0-7357-0868-1.

[8] Northcutt, S., Cooper, M., Fearnow, M., and Frederick, K.: Intrusion Signatures and Analysis. New Riders Publishing, 2001. ISBN 0-7357-1063-5.

[9] Northcutt, S., Zeltser, L., Winters, S., Fredrick, K., Ritchey, R.: Inside Network Perimeter Security. The Definite Guide to Furewalls, VPNs, Routers, and Intrusion Detection Systems. New Riders Publishing, 2003. ISBN 0-73571-232-8.

[10] Peltier, T.: Information Sercurity Risk Analysis. Auerbach Publications, 2001. ISBN 0-8493-0880-1.

[11] Pfleeger, C. and Pfleeger, S.: Security in Computing (3rd ed.). Prentice Hall, 2003. ISBN 0-13-035548-8.

[12] Redmill, F. and Anderson, T (eds.): Components of System Safety. Proceedings of the Tenth Safety-critical Systems Symposium. Springer Verlag, 2002. ISNB 1-85233-561-0.

[13] Rehtanz, C.: Autonomous Systems and Intelligent Agents in Power System Control and Operation. Springer Verlag, 2003. ISBN 3-540-40202-0.

[14] Tudor, J., K.: Information Security Architecture. An integrated approach to security in the organisation. Auerbach Publications, 2001. ISBN 0-8493-9988-2.

[15] Viega, J. and McGraw, G.: Building Secure Software. How to avoid security problems the right way. Addison-Wesley Professional Computing Series, 2002. ISBN 0-201-72152-X.

9.2 Journals

[16] Mellstrand, P. and Gustavsson, R.: Safe Execution of Unreliable Software, *Electronic Notes in Theoretical Computer Science*, Vol. 85(3), 2003.

[17] Gordjin, J. and Akkermans, H.: Designing and Evaluating E-Business Models. IEEE Intelligent Systems, Auggust 2001, pp. 11 – 17.

[18] Gustavsson, R.: Agents with Power. Invited paper, in *Communications of the ACM*, March 1999, Vol. 42 No.3, pp. 41-47.

9.3 Book chapters

[19] Akkermans, J.M., Gustavsson, R., and Ygge, F.: Structured Engineering Process for Agent Communication Modelling. In *Knowledge Engineering and Agent Technology*, IOS Press, 2000.

[20] Akkermans, J.M., Ygge, F., and Gustavsson, R.. HOMEBOTS: Intelligent Decentralized Services for Energy Management. In J.F. Schreinemakers (Ed.) *Knowledge Management: Organization, Competence and Methodology*, Ergon Verlag, Wuerzburg, D, 1996.

[21] Fredriksson, M. and Gustavsson, R, and Ricci, A., Online engineering of open computational systems. In Bergeti, F., Glezeizes, M., and Zambonelli, F. (eds*.) Methodologies and software engineering for agent systems*. Kluwer Academic Publishers (To appear).

[22] Fredriksson, M. and Gustavsson, R.: Online Engineering and Open Computational Systems – A unified approach towards theory and practice of agent systems. In Bergenti, R., Gleizes, M-P., and Zambonelli, F (eds.) Methodologies and Software Engineering for Agent Systems, Kluwer, 2004.

[23] Fredriksson, M., Gustavsson, R., and Ricci A.: 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, 2003.

[24] Fredriksson, M., and Gustavsson R.: A methodological perspective on engineering of agent societies. In Omicini, A., Zambonelli, F., and Tolksdorf, R. (eds.) Engineering societies in the agents' world, Lecture notes in artificial intelligence, vol. 2203, 2001. Springer Verlag.

[25] Gustavsson, R. and Fredriksson, M.: 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 series (LNCS), vol. 2603, pp. 127- 142, Springer Verlag, 2003.

[26] Gustavsson, R., and Fredriksson, M.: Coordination and Control of Computational Ecosystems: A vision of the Future. In: Omnicini, A., Klusch, M., Zambonelli, F., and Tolksdorf, R. (eds.): *Coordination of Internet agents: Models, Technologies, and Applications*, pp. 443-469, Springer Verlag 2001.

[27] Gustavsson, R., and Fredriksson, M.: Humans and complex systems: Sustainable information societies. Accepted for publication in Olsson, M. O. and Sjöstedt, G. (eds.) *Revealing complex structures: Challenges for Swedish systems analysis*. Kluwer Academic Publishers, 2003.

[28] Gustavsson, R., Fredriksson, M., and Rindebäck , C.: Computational Ecosystems in Home Health Care. In: Dellarocas, C., and Conte, R. (eds.): *Social Order in MAS* for the series Multiagent Systems, Artificial Societies, and Simulated Organizations, Kluwer, 2001.

[29] Gustavsson, R., and Krejic, M.: Enabling technologies supporting energy related services and products in networked smart homes. In Akkermans, H., and Ottosson, H. (eds.): *The KEES Project. Energy Efficiency in a Deregulated Market*. ISBN: 91-9753567-2-7, EnerSearch AB, 1999, pp. 89-119.

[30] Gustavsson, R.: Knowledge Management through IT support in Distributed organisations. In Ottosson, H., Akkermans, H., and Ygge F. (eds.): *The ISES Project. Information/Society/Energy System. 1996-1998.*ISBN 91-9753567-0-0. EnerSearch AB, 1998, pp. 61-81.

9.4 Papers

[31] Acquisti, A. and Varian, H.: *Conditioning Prices on Purchase History*. University of California, 2003, NSF grant 9979852, 30 pages.

[32] Andersson, R.: *Why Information Security is Hard – An Economic Perspective*. University of Cambridge. http://www.cl.cam.ac.uk/~rja14/econsec.html

[33] Gordon, L. and Loeb, M.: The Economics of Information Security Investment. ACM Transactions on Information and System Security, Vol. 5, No. 4, November 2002, pp. 438 – 457.

[34] Jenkins, B.D.: Security Risk Analysis and Management. White paper, *Countermeasures Inc.*, 1998.

[35] Mercuri, R.: Analysing Security Costs. *Communication of the ACM*, June 2003, Vol. 46, No. 6, pp. 15 – 18.

[36] Törnqvist, B. and Gustavsson R. On Adaptive Aspect-Oriented Coordination for Critical Infrastructures. Submitted to First International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT 2004) at 18th European Conference on Object oriented Programming (ECOOP 2004)

[37] Vermeulen, C. and Von Solms, R.: The information security management toolbox – taking the pain out of security management. *Information Management & Security*, 2002, pp. 119-125.

9.5 Resources

[38] IEC standards, e.g., IEC 61850, IEC 61970, IEC 61968: http://www.wg14.com

[39] IEC 61970: ftp://epriapi.kemaconsulting.com

[40] Economics and Security Resource Page (managed by Ross Anderson): http://www.cl.cam.ac.uk/~rja14/econsec.html

[41] IEC/TR 62210 ED 1.0: http://www.nssn.org/NssnSearch/displayrecord.asp?expand=1&RecordNo=691670

9.6 Reports

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

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

CRISP: Distributed Intelligence in Critical Infrastructures for Sustainable Power

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

Annex 1

Published in Electronic Notes in Theoretical Computer Science, Vol. 85(3), 2003.

Safe Execution of Unreliable Software

Design and Implementation of a Lightweight Privilege Separation Interface

Per Mellstrand and Rune Gustavsson
Department of Software Engineering and Computer Science,

Blekinge Institute of Technology,

PO Box 520, 372 25 Ronneby

Sweden

http://psi.bth.se

<pme@bth.se>, <rgu@bth.se>

Abstract We introduce a method, Lightweight Privilege Separation, enabling safe execution of unreliable software. Our method introduces no new software vulnerabilities and is fairly easy to implement. Further-more, we show by experiments that the execution overhead is in the order of milliseconds per execution of the unreliable process at hand. We compare our method with earlier attempts of privilege separation such as OpenSSH. The paper concludes with a discussion on generalizations of our approach in the form of abstract machines and their interpreters.

1. Introduction

Development of large-scale network centric systems such as those envisioned in, e.g., EC Sixth Framework Program (FP6) on Ambient Intelligence (AmI) or IBM's program on Autonomous Computing, presupposes that we will be able to produce dependable software components and trusted execution environments in the not so far future. In fact, there are calls within FP6 to that end, [1] [2].However, even if we can develop dependable components we still have to face the challenge of safe execution of those components or even less reliable software in dynamic and/or hostile environments, the topic of this paper.

1.2 Main Ideas and Highlights of the Paper

We describe in the paper design and development of LPS, a *Lightweight Privilege Separation* system that allows us to execute non-reliable C and C++ software components in a trusted way. Furthermore, we show that the added overhead introduced is negligible. We also argue that our solution does not introduce any new vulnerabilities as such, and that the added computations are handled in a way that they do not introduce added faults in the execution of the software. Last but not least, our solution does not rely on rewriting on the software and is hence very cost-effective. The paper also includes comparisons with other approaches to techniques of privilege separation and some pointers to further work.

The *Section 2 Lightweight Privilege Separation* introduces the main ideas of our proposed solution. That is, isolating execution of critical function calls in a protected execution environment using the fork system call and proper treatment of execution footprint and function call results. In a following *Section 3 Error Handling*, we address issues related to errors that can appear during execution of the forked processes and during its set up and completion. Our experiments and tests are reported in *Section 4 Experiments*. Comparisons with other approaches of safe execution of unreliable software, notably OpenSSH are addressed in *Section 5 Comparisons*. The paper concludes with *Section 6 Conclusions and Further Work*.

1.3 Background

Much software used today in the Internet infrastructure is written in non typed languages, as monolithic programs that execute partly or fully under super-user privileges. Programs executing under super-user privileges introduces affordances that could be exploited by malicious code to cause harm. Most software executing under super-user privileges require these privileges only for a few operations. Hence, most operations can be carried out without super-user privileges and many carried out even in a restricted environment. It would be useful if we could separate an executing program into a privileged and a non-privileged part, which is the exact purpose of privilege separation.

There are programs available today that uses the concept of privilege separation to protect potentially risky code from the privilege-critical parts of the same program. Typically a vast majority of the code in such programs are moved to the low-privileged part and a complicated client-server relation is kept between the privileged and the non-privileged part. These programs are designed with privilege separation as an initial requirement or has undergone significant modifications later in the development process to get this separation.

In this paper we describe a method, lightweight privilege separation, that can be applied to separate privileges in existing and new programs without the need of large redesigns and that when used properly can eliminate the consequence of many common security attacks.

2 Lightweight Privilege Separation

Functions are the central building block for programs written in the C programming language. The whole life line of a C program is defined in the function main. This function is called by the runtime environment when the process has just started and the process is ended when this function returns. The only way a library can expose its code to a C program is by the mean of functions.

The concept of functions in programming languages are modelled after mathematical functions. A function is expected to produce a return value from zero or more parameters. A function that calculate its return value based only on its parameters and that do not read or write any external state information is called a pure function. Many functions encapsulating mathematical functions are pure functions, such as the abs and sin functions found in the C standard library. Most functions however need to read or modify what we call the global state of the program. With global state we mean a vector of entities that is not non-static local variables, such as global variables, data in pointers or operating system objects (file descriptors, semaphores, etc.) exposed to the process.

Traditionally it has not been an issue which part of a state a function modifies during execution. This is clearly illustrated in older APIs such as *gethostbyname* which modifies a

static buffer in the process as its main return value. With many modern operating systems supporting processes that have several simultaneous execution points (threads) it became important that functions should not modify static buffers that are shared between different threads of the process. Functions designed to work in a threaded environment are limited in which state components they may modify so that other threads are not affected by their execution. There is no way to automatically determine which state a function modifies as this might be dependant of runtime conditions. Such a runtime condition can be a function in a network server which dynamically allocates memory for data which it receives on a socket from the remote host, and stores the pointer to this memory in a value pointed to by a parameter.

2.1 State and Privilege Separation

Privileges in a Unix system are assigned on a process basis. This means that the smallest entity which has the concept of privileges is a process. If a process has several threads they all execute with the same set of permissions, even if they are individually scheduled by the kernel. Because of this a program that execute with separated privileges must execute on several processes, it is the only way to have different sets of permissions of a process in a Unix environment. A process is also the entity by which memory and operating system objects are handled, which means that a file handle or memory address in one process is not valid, or has another meaning, in another process.

By splitting a program in two (or more) processes data written in memory in one process is not reflected in the other unless some kind of synchronisation mechanism between the processes is introduced to perform this. Due to the complex nature of sharing state between functions in C programs a program with separated privileges must have some way of performing this synchronisation or be rewritten in a such a way that it does not rely on sharing state. In general, we believe it would be difficult to rewrite programs to achieve this goal.

2.2 Complex State

There are other type of state components than just memory that can be shared between functions. For example a function can open a file, pass a reference, descriptor, of this file to another function, which read or write data to the file and passes the descriptor to a third function which closes the file. As file descriptors are shared between the different threads in a process the three functions could even execute in three different threads and still share this state. It would however not be possible to transparently use this code in a program with separated privileges as the process boundaries would make the descriptor that is valid in one of the processes invalid in the other. This type of state dependencies are difficult to keep track of and in some cases even impossible. Fortunately this type of state sharing is far less common than sharing memory and can often be avoided by placing the boundary between the privileged and the non-privileged process at a well-chosen place in the code.

2.3 How Privilege Separation is Possible

The key mechanism that makes privilege separation possible is that of cloning a process. This ability is provided in Unix like operating systems through the fork system call. This system call creates a new process that is an exact copy of its parent with exception of the process id and parent process id. [3] The newly created process (child process) has a copy of all attributes from its parent, should any state components be changed in one of the processes the state in the other will not be affected. Let one of the two processes switch

privileges and take the part of executing unprivileged code. The two processes maintain a controlled communication channel so that the main program can continue executing code that need special privileges in one process and code that do not in another and still keep shared state between the two processes.

Due to the complex state that is typically shared between functions and the need for privilege separated programs to execute in several different processes it is not an easy task to create a privilege separated program. We have investigated a method to separate privileges in programs without the need to do large modifications in existing software or make significant changes in software development methodologies.

2.4 The LPS System

The purpose is to create a method that could be used to create privilege separated programs and to modify existing software to use privilege separation without the need of significant changes in development methodology and existing code. We wanted to avoid the use of special heaps to share state between the different processes as used in for example OpenSSH. [4] An assessment of OpenSSH is given in Section 5 Comparisons.

Typically there is a small set of functions that contain most of the vulnerabilities in a given program. For example functions that parse data from a human readable format into a machine readable format (parsers) have been know to contain many vulnerabilities. This fact has been known for quite some time and still there are new vulnerabilities found in parsers that have already been reviewed several times. Many network servers need to parse a user name and a password before they can switch to a lower privilege and hence some parsing of potential hostile input is done under super user privileges. By placing code that we believe is more likely to contain vulnerabilties in a low-privilege access environment we can reduce the consequence if that code in fact should contain a vulnerability and be exploited. By allowing a majority of a program execute under high privileges and place restrictions only on carefully selected parts we can impose a privilege separation on the most risky parts with minimal, if any, impact on other parts of the program.

We have implemented a library that enables the use of lightweight privilege separation in a program executing in a Unix-like environment. The library works as a wrapper between functions on either side of the privilege separation fence. To apply this privilege separation to a process we rely on the abstraction of functions and some basic assumption of the structure of the host program.

When a non pure function is called it is likely that there are some requirements on the global state. For example a function parsing a user name might expect that there is a textual representation of the user name stored in memory and memory allocated where the return data should be placed. When separating privileges in a program we must make sure that all such requirements are still met or the program will not function as expected. There can be very complex requirements on state in a program. It is not uncommon that complex algorithms are broken up in several functions that communicate by passing parameters and by modifying global state. There is no realistic way to capture all requirements that a function have on the environment and transferring this state to a process. However by cloning the process just before the function is called the operating system will copy all of the state that the process has.

By creating a clone we know that all of the state function B would have had in process one is available in process two. Unfortunately this is not enough. As the purpose of executing

function B in a separate process is to run it with lower privilege than it would have had in process A we must apply restrictions to process two before transferring control to function B. Each restriction we place on process two will make the environ-ment slightly less identical to that of process one. We previously mentioned that we make some basic assumptions on the structure of the program. These assumptions are that there are some privileges that can be removed from process two and that functions executed in the restricted environment still will execute as expected.

In our current working version of LPS we apply three restrictions to the unprivileged process;

- Switch of user id (*UID*) that the process execute under,
- Switch of the group id (*GID*) that the process execute under, and
- Revocation of inherited file descriptors.

We believe that there are more restrictions that could be applied to the unprivileged child under some circumstances. Such restrictions could be to change the virtual file system root for the process (*chroot*) or applying other restrictions supported by the kernel, such as jail which is available in FreeBSD and OpenBSD systems. Different situations require different type of permission revocation and we have designed the LPS library so that it should be easy to make modifications to which permissions that should be revoked. The reason for revoking open file descriptors is that by default all open descriptors in the privileged process are inherited to the unprivileged.

Typically connections to data bases, LDAP directories and similar services communicate through a file descriptor. The common practice for such services is that once the client has authenticated (which could have been done in the privileged part) the server keep this authentication state for all following commands. There is a risk that such a descriptor could “leak” into the unprivileged process and to avoid this potential security leak we make sure all descriptors are closed by default. To maximize compatibility with existing code LPS opens /dev/null for standard input, output and error for the restricted process.

2.5 Data and State Transfer

The second main service provided by the LPS library is to capture state modifications and return these to the privileged process in a secure way. Function B could make a vast amount of changes to the state in process two, and we only want to transfer relevant modifications back to process one.

We have identified three different types of state components to handle in a principal way;

- State modified by function B that the privileged process need to continue execution correctly. This state is transferred back to process one.
- State modified by function B that is not needed by the privileged process, such as scrap buffers for temp data and buffers containing intermediate data. This state may be ignored when transferring state back to process one.
- Components that concern security and which must never be transferred to the privileged process.

An example of such a state component is a variable indicating if the user has been

authenticated or not. We must make sure that this type of state does is not transferred back to process A.

Also we need to do this state capturing without placing additional requirements on how function B should be implemented. To provide for all this LPS provides a mechanism for saving state from the restricted process, transferring this to the privileged process and applying it to the state of this parent process.

2.6 Saving State

LPS uses a serialization mechanism to identify which state components to save from the restricted process. Serialization is a technique used in many other programming languages and environments, such as in Java and Microsoft's MFC foundation classes.

The basic idea with serialization is that all state components a program requires can be described in terms of the primitive types provided by the programming language and relations between such types. In the C programming language all primitive types can be copied binary between different processes. When separating privileges at a certain point in a program the programmer must provide a list of state components that the separated function may modify and that should be transferred back to the privileged process. LPS provide mechanisms for transferring all primitive types, raw buffers and C-style NULL terminated strings. We also support serialization of complex structures by callback functions that decompose the structure into its primitive parts.

The LPS library provide both functions and macros for specifying which state compo-nents that should be saved and transferred to the privileged process. Typically only a few lines of code need to be added to a program in order to specify which parts that should execute under lesser privileges and which state that should be transferred. For example, consider the call:

```
int result;

result = func( 23 );
```

where the function func only modifies its return value. The same code, rewritten to take advantage of privilege separation the corresponding code would be:

```
int result, errorcode;

LPS_BEGIN_SEPARATION( ); {

        result = func( 23  );

                                        }

LPS_END_SEPARATION( errorcode,

        LPS_DTYPE_INT( &result )

                                        );
```

The LPS frame work will create a child process, apply security restrictions and execute the function func in this environment. After this function has finished execution the value stored in the result variable will be serialized and transferred to the privileged process.

2.7 Transferring Data and Applying State Changes

The state serialized from the restricted process is transformed into an array of bytes. These bytes are prefixed with information about size and an extended error code, and transferred through an open pipe to the privileged process. This pipe is represented by an open file descriptor in both processes, it is the only descriptor that is not closed by the safety precaution routines.

The final step in returning to a state that the privileged process expects is read state components from the pipe and apply it. Before we can do this we need to check so that the data provided by the unprivileged child is what we expect. We must assume that the child process has been exploited and that the process is transmitting hostile data in an attempt to corrupt the state in the privileged process as well. For this reason the privileged process perform a thorough security validation of the received data.

LPS uses a simple binary protocol to transfer state data. The purpose of this design is to avoid having to place a complex parser in the privileged code and thus introducing unnecessary vulnerabilities. This design also gives performance benefits over using more complex protocols that need more parsing. The data stream is divided in two parts, the head which only contain meta information and a body part which contain the state that should be written. The body and head are read separately from the pipe by the privileged process. After the head has been read the size parameter is verified against a minimum and maximum allowed value. If the size is acceptable the privileged process reads the specified number of bytes and closes the pipe. After this stage the body data is handled by the serializer. The serializer knows which type of data to expect and verifies element by element with the type information in the stream. This might make little sense for primitive types such as char and int but make sense when transferring strings or buffers to make sure the buffers are large enough to store the data received. If these tests succeed the data provided from the stream is written to the state of the privileged process and LPS returns to the calling function. The state is now copied into the privileged process which resume execution as if there had been no privilege separation done at all.

3. Error Handling

The mechanism for calling functions provided by the C and C++ programming languages cannot fail with an error code. The actual function that is called may return an error value, but to the programming language runtime environment this is a return value as any other. There is no way that the actual mechanism for calling the function may return an error value that can be handled by the program. For this reason all programs that we know of assume that the actual calling mechanism will not fail, an assumption that we believe is fair to make when developing programs. If there should be any errors in the environment of a program that prevents the call from being made (such as if there is no stack space left) the program will typically crash, but this is a very rare situation in well-written programs.

The situation of error free function calls are very different when applying privilege sepa-ration to a program. The privilege separation will require resources from the operating system, resources which might not be available. If these resources are not available the operating system will return an error value to the LPS system. The privilege separation will also verify

parameters, tests which may show that the parameters are invalid. If this is the case LPS cannot let the execution continue but must have means to propagate the error into the host program.

We have identified four different situations in the privilege separation interface where errors can occur;

Before Creating the Child

If the LPS system is unable to create a child process (i.e. the fork system call fails) there is no context in which the restricted function can be executed. It is not acceptable to execute the function in the current process as this would circumvent the purpose of privilege separation. In this case the LPS system return an error code to the calling function.

Before Executing the Restricted Function

The LPS system verifies that it has successfully reduced the privileges of the child process before it transfers control to the restricted function. Should there be some error condition that prevents LPS from reducing its privileges the system will not execute the restricted function. In this case LPS transmits an extended error code on the pipe back to the privileged parent process and terminates. The parent process generates an error code and returns this to the calling function.

If the Child Process does not Terminate Correctly

The parent process waits a limited amount of time for the restricted child process to finish and terminate. Should the child process not finish in time the LPS system in parent process return an error code to the calling function. Another, similar, situation is when the child process crashes. In this case the LPS system notices the crash and returns an error code to the calling function.

Received Data is Invalid

After the child has finished executing the restricted function the LPS system transmits state information back to the parent process. This data is thoroughly verified by the parent process and should an error be detected, such as writing more data to a buffer than it can hold, the LPS system will not apply the state in the parent process but return an error to the calling function.

3.1 Integrating LPS Errors in an Existing System

We want to apply privilege separation to programs that are not originally developed with this in mind, so we cannot rely on any existing mechanisms for handling errors that originate from the privilege separation. In all cases of error the LPS system transfers the error information to the privileged process where it is delivered as a return value and an extended error information. Unfortunately there is no universal mechanism for propagating the error information into the host program, as there is not a single universal way of writing programs. When integrating LPS into a host program we require that the engineer that does the integration also integrate the error codes that LPS may return into the error handling system of the host program.

In some cases this might be very simple. At the simplest form the developer can just check if

the LPS call succeeded and if not terminate the program. This strategy is useful in situations where each client is handled by a different instance (process) of the program. Only a single user will be affected by this termination.

If the program uses the same instance to service multiple clients this simple error handling is not recommended, as it could open up for a denial of service attack. In this case the developer integrating LPS must make sure that the LPS error is handled as any other fatal error for the active user. It is important that the error information is propagated in such as way that no part of the privileged program is dependant of state that should have been present only if the function call had succeeded.

4. Experiments

To determine the performance of the LPS system we designed and run an experiment of test cases. The purpose of this experiment was to estimate the time overhead imposed by using the LPS privilege separation system. We wanted to investigate the time used when transferring a small amount state (“dry run”) and when transferring a a significant amount of state back to the privileged process. We also wanted to investigate which impact the choice of operating system and the computer speed had on the performance of the LPS system.

We designed two test cases that we executed under three different operating systems on two different computer systems. Each test case was executed three times on each (operating system, computer system) pair giving us 36 measurement points. We use the mean value from the three tests in this paper.

The experiment was carried out on two different computer systems. The first system (System A) is a few years old but systems of this capacity are still used as servers where privilege separation might be considered. The second system (System B) was selected because it was the most powerful PC system that was easily available to us.

Computer Systems Used in the Experiment:

	System A	*System B*
Vendor Name	Compaq Deskpro EN SFF	Dell Optiplex 260 Gx
Processor	Intel Pentium II	Intel Pentium 4
Processor Speed	400 MHz	2.7 GHz
RAM	160 Mb	512 Mb

Table 1. System descriptions

4.1 Operating Systems

The experiment was carried out on three different operating systems; FreeBSD, Open-BSD and Linux. We installed the operating systems without any graphical environment (X11), but did not tune the installation in any other way. We used the respective standard kernel for each operating system as installed by the installation program (GENERIC for the two BSDs and the standard RedHat kernel for the Linux installation).

The decision to use these three operating systems in the experiment was based on the

following conditions; Linux is the most wide-spread free Unix-like operating system used on millions of computers connected to the Internet. It is important for us to support this operating system. FreeBSD is used by many large web serving farms on the Internet including Yahoo! and others. OpenBSD is a operating system developed with security as one of the main objectives. We believe it is important to support and benchmark this platform for security-related research.

4.2 Test Cases

We used two test cases in the experiment; the first a small function taking an int integer as parameter and returning a value calculated from the function. The payload data transferred back to the privileged process is the return value for the function which on all test platforms is a 32 bit (4 byte) integer. The function was implemented in the C programming language with this following source code:

```
int func( int x ) {

        return x+1;

                }
```

The second test case was designed to modify significantly more state than in the first test case. Here the function writes an exclamation mark in every byte in the first kilobyte of memory pointed to by its in parameter. The payload to transfer back to the privileged process is the modified buffer of 1 kB (1024 bytes). We believe that one kilobyte of state is more than enough for a typical program, but as we will later see the amount of state transferred does not significantly affect performance. The implementation for this function is:

```
#define SIZE 1024

        void func( char * pData ) {

                memset( pData , '!' , SIZE );

                                        }
```

The complete source code for both these tests is available in the LPS package which can be downloaded from the authors' home page.

4.3 Experiment Procedures

For each computer system and operating system the following procedure were followed;

- The operating system was installed on the computer system. For Linux we used the Server profile, for FreeBSD we installed the set Minimal and for OpenBSD we used a default installation. We did not install a graphical environment on the computers.
- The computer was restarted.

We built the executables from the same source code with the default compiler available on the system.

Each test was executed three times. We used the physical console (display and keyboard

connected to the computer) and did not pipe either the input or output of the test program.

We did not separately measure the time taken to execute the actual function from the time taken by the privilege separation, but assumed all time was consumed by the privilege separation. Also we have used real functions (that calculate a value or modify a buffer) rather than dummy functions. The reason for this is that we wanted to make sure that the functions succeed (which is very easy to check from their return values) and that we wanted to avoid possible compiler optimizations that might otherwise have eliminated dummy functions. The two functions that we used as test cases are so simple that we believe it safe to ignore that part of the execution time.

4.4 Results and Conclusions from the Experiment

The results from the experiment is presented in Table 2. We notice that the execution of both test cases take significantly longer time on FreeBSD than on the other systems. We have currently not investigated this difference further, but our hypothesis is that a FreeBSD process has more state that must be copied by the kernel when a process is forked.

We also notice a large difference for OpenBSD and Linux on the different test systems, a difference that we do not see in the case of FreeBSD.

Execution time in milliseconds for LPS

Operating System	*System A*	*System B*
Test Case 1		
FreeBSD	5.179	4.145
OpenBSD	1.881	0.5553
Linux	1.734	0.6540
Test Case 2		
FreeBSD	5.198	4.154
OpenBSD	2.127	0.6407
Linux	1.759	0.6730

Figure 2. Execution time for different operating systems

Overall, however, we were surprised of the low overhead imposed by using LPS. On a modern computer system running Linux or OpenBSD the overhead of using privilege separation is less than one millisecond, even with one kilobyte of state to transfer and without any tuning of the systems. We believe that with this low overhead it is possible to use the LPS on many different platforms and in many different places in a single application. Should there be any doubts of the quality of a particular part of a program we can use LPS to isolate that part without losing much performance.

5. Comparisons

In this section we present a comparison between the privilege separation model used in OpenSSH and the LPS system. We compare the two methods from four perspectives, security, portability, performance and integration time.

One of the most well-known systems today that uses privilege separation is the OpenSSH ssh daemon developed by the OpenBSD team. In their approach a large portion of the program (the ssh daemon) executes under lesser privileges and communicates bidirectionally with a privileged supervisor process. The two processes also have means of communication through a special heap that is stored in memory shared between the two processes. The privilege separation in OpenSSH is described in detail in [5].

5.1 Security

In the OpenSSH model a vast majority of the program code is placed in an unprivileged process. The unprivileged process send requests to the privileged process over a pipe and the privileged part verifies the request and if accepted carries it out. Only two processes are used throughout the life of a session (a single privileged and a single unprivileged process) and these two processes communicate bidirectionally.

With the LPS system only potentially risky functions of the program execute with lesser privileges and when such a part is done executing the child process dies (a single privileged and multiple unprivileged processes). The communication between the unprivileged and privileged process is unidirectional, the unprivileged process transmits state information back to the privileged process just before it dies.

In the LPS system it is up to the engineer that implements privilege separation to deter-mine which parts that should execute with lesser permission, in the OpenSSH case essentially all code is placed in the unprivileged part. We believe that this is an advantage for the OpenSSH model as more code is placed under restrictions than what will typically be done when using the LPS system.

However, by using a model with temporary children (a single privileged and multiple unprivileged processes) that dies when they have carried out a request it is possible to execute the different unprivileged children with different privileges. For example code that requires super user privileges to access to access the systems' password file but does not require file system access can be executed in a child process with these permission set while code that require file system access but not with super user privileges can be set to have these permission. By using temporary child processes we can increase the flexibility of how to restrict a particular unprivileged child process, which we see as a great benefit with the LPS system.

Also by using temporary child processes we can eliminate the consequences of some programming errors, such as memory and file descriptor leaks, that are not strictly security related, but that may result in denial-of-service attacks. Any state in the unprivileged process that we are not interested in is left in that process when it dies for the operating system to clean up.

5.2 Portability

The LPS system currently executes on three different operating systems from the same

code base and is not dependant of complex operating system supported features such as descriptor passing or the use of special heaps, which is used in the OpenSSH model. The LPS system was designed with portability in mind and we expect to port it to even more platforms that support the POSIX API.

5.3 Performance

We have not performed any measurements on the OpenSSH solution, but based on the description of the system that we have, we believe it to be as fast or faster than the LPS system. We do not believe that performance is a problem for either solution though. As we have shown, the LPS system executes with less than one millisecond of overhead time on a modern computer and the additional time required to execute a program with privileges separated is in our opinion negligible. We doubt that any user would notice the slight delay imposed by the use of LPS even if several privilege separated calls are performed in a program.

As the LPS system uses a new child processes for each separation we are unsure if it is suitable in a real time environment, as the time to complete the fork call may differ significantly dependant on the system load.

5.4 Integration Time

The OpenSSH system for privilege separation requires much larger modifications in existing code that what is typically required by the LPS system. We have made very few assumptions on how a program is structured when we designed LPS and this gives us an advantage when applying privilege separation in programs that were not design with this as an initial requirement.

As we have shown in this paper a function call is simply wrapped with a few lines of source code for executing the call with separated privileges through the LPS system. There are no additional requirements placed on the system by the LPS system and no other initialization to be done.

6. Conclusions and Further Work

We have described the LPS privilege separation system, a system that enables us to execute non-reliable C and C++ software components in a trusted way. By using the LPS system it is possible to separate privileges in existing and new software systems with a minimal change required to existing source code and without having to adopt new software design methodologies. The LPS system is a portable and flexible system that does not rely on special features in the operating system, but can be used in a variety of programs that execute on several different operating systems.

Also we have shown that the overhead imposed by using privilege separation is less than one millisecond on a modern computer, an overhead which is negligible in most cases. We have also presented a comparison between privilege separation as implemented in the LPS system and as implemented in the OpenSSH ssh daemon.

We believe that privilege separation and the use of LPS is an inexpensive and fast way to increase dependability in new and existing software.

Our plans on further work include applying the LPS privilege separation system to a larger

software system that is currently executes as a monolithic process with super-user privileges. Also we plan on further investigation of the performance issues that we found when experimenting on FreeBSD.

We also plan on investigating other techniques that can be used to execute unreliable software safely, such as the use of embedded virtual machines.

7 References

[1] IST Advisory Group. Software technologies embedded systems and distributed systems: A European strategy towards an Ambient Intelligent environment. European Commission IST Report – http://www.cordis.lu/ist/istag.htm. June, 2002.

[2] IBM. Autonomic Computing: IBM's Perspective on the State of Information Technology. White Paper - http://www.ibm.com/research/autonomic. 2002.

[3]McKusick, M. K., "The Design and Implementation of the 4.4BSD Operating System", Addison Wesley, 1996

[4] The OpenSSH ssh daemon and documentation is availble from http://www.openssh.com

[5] Provos, N., CITI Techreport 02-2, Preventing Privilege Escalation, http://www.citi.umich.edu/u/provos/ssh/privsep.html, 2002.

Annex 2

Proceedings of First International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT 2004) at 18th European Conference on Object oriented Programming (ECOOP 2004)

On Adaptive Aspect-Oriented Coordination for Critical Infrastructures

Björn Törnqvist, Rune Gustavsson

Department of Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520, SE-372 25 Ronneby, Sweden
{bjorn.tornqvist, rune.gustavsson}@bth.se

Abstract. The future EU power grid must rely on a flexible hierarchy of coordination mechanisms. To that end, we propose a top-down approach to coordination which enables us to introduce meta-coordination as a viable approach. We describe several important aspects of meta-coordination. Software adaptation and aspect-oriented approaches may be a suitable venue for use in meta-coordination. Consequently, we discuss current approaches in software adaptation, aspect-orientation, and coordination. In dealing with these issues we propose a set of open research questions.

1 Introduction

Coordination of services and resources is a key challenge of emerging and future embedded information systems, e.g., as proposed in EC programmes on Ambient Intelligent Systems (AmI). An interesting example of AmI is related to future inter-connections between power grids and information networks. In essence, we have to coordinate two high level goals, i.e., power network operations and customer-centric business models. This high-level meta-coordination is a context or mission oriented task that has to be supported by a flexible hierarchy of coordination mechanisms. To cope with this complex of coordination we propose in this paper a top-down approach to supplement the traditional bottom-up approach of coordination. Our real life example is the future customer-oriented power grids. The power generation and distribution in the EU are being deregulated. This means that the traditional hierarchal network with one operator and a few large generators is changing into a decentralized network with several operators, many of whom have their own smaller generators including renewable energy sources. There are several challenges and opportunities with a de-regulated power market. As a matter of fact there are several EC

supported projects addressing different related topics[2] . It is believed that deregulation will enable greater introduction of renewable energy sources throughout the network, along with more interconnections between countries, which in turn can create a more robust network provided that the corresponding ICT network is properly designed. The power grid and its associated ICT infrastructure is considered a critical infrastructure, and consequently care must be taken to design it correctly. As a matter of fact, interdependencies between critical infrastructures are becoming a major international concern.

We know that the power system will evolve at a faster pace in the future; as distributed generators are added, command and control of facilities is changed when businesses are restructured, and when grid- and business- operations software is updated, just to name a few scenarios (the lifespan of the system is measured in decades, making it impossible to foresee all possible scenarios). Different operators may have different incentives and priorities to invest in different parts of their networks – for economical and political reasons. Meanwhile, the power grids in the EU are becoming more interconnected. Consequently, we know that the networks will become heterogeneous in both hardware and software and we must plan to address this issue from the start so that the heterogeneity is an enabling and not a disabling factor. An operator should be able to upgrade its system without requiring that all operators in the EU implement the same upgrade.

Mechanical breakers will always be present in the power grid, should the ICT net-works fail to correct a failure within a set amount of time. Thus, in principle we have a pleasant precondition in that we can only improve the state of the power network through coordinated ICT networks. But as the recent enormous blackouts have shown, this fallback is something we must try hard to avoid. The actors in the system include grid operators, business operators, embedded agents, generators and consumers. Furthermore, political and legislative factors as well as natural forces (weather) act upon the system.

2 Setting the scene

It is clear that the requirements on the future ICT system in the power grids are complex. We have identified two major subsystems; grid operations and business operations. The main goal of the grid operations is to provide a sustained level of power to as many consumers as possible, and involves the entities related to securing the grid by means of both maintaining a steady state, and responding to unforeseen events. Business operations, on the other hand, have the main goal of facilitating trade both between operators, and between operators and consumers, in the deregulated market.

Due to the high requirements on performance while being prepared to sacrifice consensus it seems most appropriate to use a control oriented coordination mechanism [6] for the grid operations. The business operations will rely heavily on market algorithms and has high requirements on the fact that contractual agreements and non-repudiation are enforced, thus, a blackboard-oriented approach [5] seems appropriate.

[2] E.g., CRISP Distributed Intelligence in Critical Infrastructures for Sustainable Power: http://www.crisp.ecn.nl/

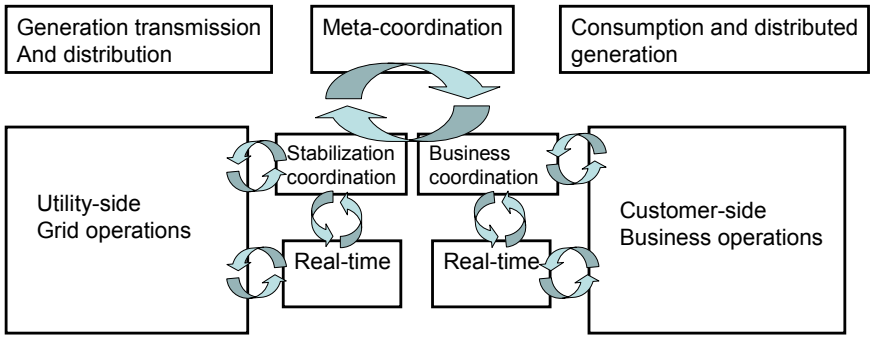

Fig. 1. Coordination in the future power grid.

All would be well if these two subsystems were independent. However, as illustrated in figure 1, the state of grid operations affects the business operations (e.g. the available power affects price), and vice versa (e.g. service-level agreements affect the likelihood that a certain customer will be shed in order to save the system from complete blackout).

Finding a single coordination infrastructure to support these aspects has been hard and without success. As a consequence, we have researched ways to abstract the coordination techniques into *meta-coordination*.

Considering the different ways of coordination in an aspect-oriented framework is another very interesting venue. Consequently, aspect-oriented coordination and adaptation middleware is of great interest to us. We have been unable to find an existing middleware conforming to all requirements and aspects of the ICT network in the power grid.

3. Top-down versus bottom-up approaches of coordination

Coordination is the key technology enabling distributed applications. However coordination appears at different abstraction levels of networked systems. The low level mechanisms supporting coordination are protocols enabling connectivity between components, e.g., middleware. An active and successful approach towards understanding and implementing coordination mechanisms is a bottom-up approach building on extensions of middleware models and techniques. Some recognized drawbacks of that approach are that the available

component-based platforms do not support reasoning and reuse of coordination patterns since they typically only have support for IDL descriptions, i.e., connectivity aspects.

Fortunately, there has been R&D in multi-agent system design and implementation during the last two decades. In effect, a multi-agent system approach is a top-down approach where agents and their coordination are starting points. Agents are in a sense objects with own control implicating that coordination between agents rely on the competencies of the agents and message passing using a high level Agent Communication Language (ACL), where the purpose of the message is clearly separated from the content. It is also inherent in the agent approach that acceptance of or response to a message is context dependant, i.e., dependant on the mental states and assessments of the environment by the agent receiving the message. In this general multi-agent model there is no explicit coordination or coordinator. The coordination is distributed among participating agents due to an agreed upon coordination pattern. Well known coordination patterns include different forms of negotiations and resource management patterns such as computational markets often modeled as auctions. Given these coordination patterns different roles and rules of engagements are agreed upon and followed by the agents.

An obvious drawback with this distributed coordination is that some unwanted behavior might appear or that some inefficiency might violate real time constraints. To cope with these potential drawbacks there have been efforts to make a trade off between local competencies of agents and introduction of a coordinating agent coupled to an instance of a coordinating pattern. Simultaneously, free message passing are replaced by dialogue patterns connected to the coordinating pattern at hand. That is, the top-level approach initiated on the multi-agent level can be narrowed down to coordinating sufficient but not more independent actors in a controlled coordination with a high level description. In the case where the control of the actors is void we have in fact the web service model with explicit coordination models, including Service Level Agreements (SLA).

In our case of meta-coordination we propose a top-down approach to support the earlier mentioned bottom-up approach. An obvious meeting point for middle-out design and implementation would be an aspect-oriented glue between the agent and object oriented approaches at hand.

4 Important aspects of meta-coordination

In any given system, the model of coordination is of course not the end goal. However, which method is used can have great impact on the system behaviour. Likewise, the properties of the system affect which coordination model is the most suitable one.
In this chapter we present some of the aspects for coordination to consider in the future power grid ICT network.

- *Observation*. Some systems provide facilities for an entity to observe the state of the system as a whole. In such a system it may also be possible for the entity to observe the state of the other entities. Due to various factors this may not be possible [4], although it has been shown to be feasible in certain settings [14]. The ICT system in the power grid clearly benefits from having the ability to fully observe the other entities (as a way to achieve consensus), but must be able to adapt the level of consensus in response to sudden drastic events such as short-circuits.

- *Agreements*. In some systems the behavior of the entities is governed by agreements – these can either be statically defined when designing the system or dynamically exchanged. Nevertheless, they give any given entity insight into the current and expected future behavior of another agent. The business operations entities typically require strict immutable agreements while the grid operations entities may benefit from such agreements. One such example of agreements instead of communication as a means to achieve coordination is when power is being rerouted by intelligent breakers in response to a short circuit. The breakers will detect the failure almost instantaneously and can open or close in a pattern agreed upon beforehand. Consequently, their action is coordinated without need of communication.

- *Interaction*. This aspect reflects the level of interaction that is possible between entities in the system. Inherent in this definition is that an entity should also accept that other entities interact with it. Consequently, this is not only an aspect of interconnectivity but also of interoperability. One class of systems that relies heavily on interaction is command and control. In the power grid, the ability to interact with other entities changes as the system is reconfigured in response to events, such as protecting itself from complete blackout if a substantial amount of generation is lost (as would be the case if the wind suddenly abates), or when power (and hence communication) lines are lost.
- *Environment*. An entities' environment can be static or dynamic. Naturally, the environment of entities in the power grid is dynamic. However, for some entities in the system the environment is perceived as static during normal operating conditions. It may be possible to utilise this fact to increase operating performance. Consequently, a mechanism that adapts between the static and dynamic aspects of the environment could be researched.

- *Robustness*. We define a system as robust if its functionality is unaffected by the loss or addition of an entity. Most systems have some level or robustness; even though a sequential process can not function if one of its steps is lost it can still function if new steps are introduced before and after. An auction has very high robustness; in that it is mostly unaffected by the number of bidders at any given time. However, an auction is not entirely robust; e.g. it does not function without an auctioneer. The same entity in the power system may have different robustness criterion and requirements depending on the role it currently is involved in (the aspect of system operation).

- *Performance*. Overall system performance is a combination of two factors; Throughput and Latency. Coordination as a task in itself does generally not need high throughput as there is a very small data exchange. On the other hand, the latency in the communication throughout the system is very important since it effectively stalls the coordination. However, some entities only require high performance when the system is in a certain state, while other entities may not operate fully nor need high performance in that state.

5 Current approaches

As described, the future power grid is a heterogeneous open distributed system. It can also be modelled as a complex adaptive system [17]. As the system changes both endogenously (e.g. as generation and consumption are matched) and exogenously (e.g. changes to infrastructure, natural forces, etc) it is important that the software components are able to adapt to new operating requirements. Recent studies in deploying distributed services in a

network without breaking its consistency [7], while decreasing the computational cost [11], and dynamic adaptation of runtime algorithms [1] are valuable. Furthermore, recent research show that a system can be adapted by ways of an aspect oriented approach [10].

Traditional aspect oriented approaches have been unsuitable for the future power grid as they rely on static aspects introduced at compile time [13]. However, recent work introduces ways to achieve dynamic aspects both for coordination [12] and business [3] logic. As pointed out in [16], aspects are most useful when their scope is well defined.

Several relevant and interesting coordination techniques are emerging, such as field-based coordination in dynamic networks [9] and distributed coordination of teams consisting of both computers and human operators [15]. As the grid operations in the power grid have extreme real-time requirements, coordination techniques could account for the dynamic communication performance in the ICT networks in similar ways to those described in [8] and [2].

6 Open Issues

Some of the open issues include:

- Language independence. The lifespan of the ICT system in the power grid is measured in decades. Consequently, a vast number of programming languages are likely to co-exist. Is automatic generation of adaptors, or language independent coordination protocol most fit for this scenario?

- Aspect-oriented coordination and adaptation middleware. Which future middleware is most likely to fulfil the requirements of our system, and how can they be extended to support meta-coordination?

- Integrating adaptation and aspect-oriented techniques. As mentioned previously, techniques exist for adapting an entity according to a new aspect. We are also interested in how best to either adapt an aspect or handling multiple parallel aspects.

References

[1]. Chen, W.-K., Hiltunen, M., Schlichting, R., “Constructing Adaptive Software in Distributed Systems”, Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-01), 2001.

[2]. Cox, R., Dabek, F., Kaashoek, F., Li, J., Morris, R., “Practical, distributed network coordinates”, ACM SIGCOMM Computer Communication Review, Volume 34 , Issue 1 (January 2004).

[3]. D’Hondt, M., Jonckers, V., “Hybrid aspects for weaving object-oriented functionality and rule-based knowledge”, Proceedings of the 3rd international conference on Aspect-oriented software development”, 2004.

[4]. Fischer, M., Lynch, N., Paterson, M., “Impossibility of distributed consensus with one faulty process”, Journal of the ACM Volume 32 , Issue 2, pp374-382, 1985.

[5]. Gelernter, D., “Generative communication in Linda”, ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 7, Issue 1, ACM Press, 1985.

[6]. Hsieh, C. S., Unger, E. A., “Manifolds: A very high-level conceptual framework of interprocess synchronization and communication”, Proceedings of the 15th annual conference on Computer Science, pp196-204, ACM Press, 1987.

[7]. Janssens, N., Steegmans, E., Holvoet, T., Verbaeten, P., “An agent design method promoting separation between computation and coordination”, Proceedings of the 2004 ACM symposium on Applied computing, 2004.

[8]. Lim, H., Hou, J., Choi, C., “Constructing internet coordinate system based on delay measurement”, Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement, 2003.

[9]. Mamei, M., Zambonelli, F., "Self-maintained distributed tuples for field-based coordination in dynamic networks", Proceedings of the 2004 ACM symposium on Applied computing, 2004.

[10]. Papapetrou, O., Papadopoulos, G., "Aspect Oriented Programming for a component-based real life application: a case study", Proceedings of the 2004 ACM symposium on Applied computing, 2004.

[11]. Pérez, J. A., Corchuelo, R., Ruiz, D., Toro, M., "An enablement detection algorithm for open multiparty interactions", Proceedings of the 2002 ACM symposium on Applied computting, 2002.

[12]. Pinto, M., Fuentes, L., Fayad, M. E., Troya, J. M., "Separation of coordination in a dynamic aspect oriented framework", Proceedings of the 1st international conference on Aspect-oriented software development, 2002.

[13]. Techelaar, S., Cruz, J. C., Demeyer, S., "Design Guidelines for Coordination Components", Proceedings of the 2000 ACM symposium on Applied computing, 2000.

[14]. Urban, P., Defago, X., Schiper, A., "Chasing the FLP impossibility result in a LAN: or, How robust can a fault tolerant server be?", Proceedings of 20th IEEE Symposium on Reliable Distributed Systems, 2001.

[15]. Wagner, T., Guralnik, V., Phelps, J., "A key-based coordination algorithm for dynamic readiness and repair service coordination", Proceedings of the second international joint conference on Autonomous agents and multiagent systems, 2003.

[16]. Walker, R., Baniassad, E., Murphy, G., "An initial assessment of aspect-oriented programming", Proceedings of the 21st international conference on Software engineering, 1999.

[17]. Wilberger, A. M., "Complex adaptive systems: concepts and power industry applications", Control Systems Magazine, IEEE, Volume: 17 , Issue: 6, 1997.