TNO Science and Industry

Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek / Netherlands Organisation for Applied Scientific Research

Stieltjesweg 1
P.O. Box 155
2600 AD Delft
The Netherlands

www.tno.nl

T +31 15 269 20 00
F +31 15 269 21 11
info-IenT@tno.nl

TNO report

MON-RPT-2010-01307

Designing Intelligent Systems

Date	18 May 2010
Author(s)	Drs. J.H.C. van Rest Drs. A.K. Bastiaans Dr. Ir. M. Djurica Dr. Ir. L.J.H.M. Kester Dr. Ir. Z. Papp Ir. J.Sijs Ir. M. van Rijn J. F. Levin-Koopman MSc.
Number of pages	142
Number of appendices	4
Projectname	ISN Architecture
Projectnumber	033.22373

Contents

1 Introduction

Our society has enjoyed unprecedented prosperity in the previous decennia. Behind the scenes there is an increasing number and complexity of devices which are making our life easier in many ways, whether it is in running and organizing transport, energy production, security or other technology-moving areas like entertainment and gaming.

Many of the advances in previous decades were enabled by *increasing computer processing power* as dictated by Moore's Law, which enabled powerful and flexible control of processes and development of more powerful and versatile tools and machines. All these developments included processing larger amounts of data and that in ever shorter periods of time.

Moore's Law also holds for *sensors*, which are being developed to help us to measure and quantify the world around us. This means that cheaper and better sensors are available every day. In fact, the first publication of Moore's Law was illustrated with radar sensors [MOORE1965].

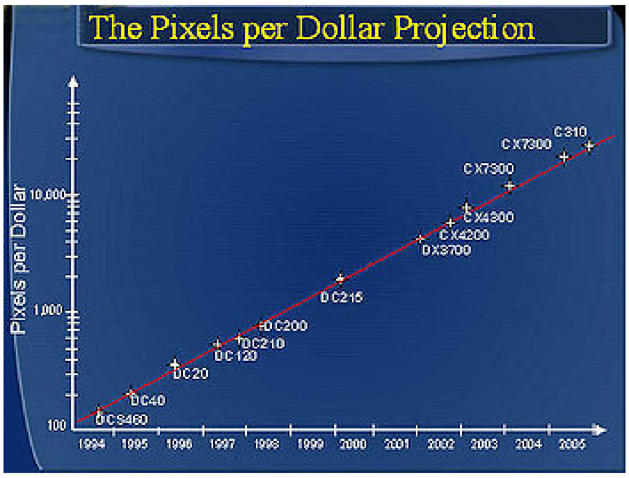


Figure 1 - Pixels per dollar based on Australian recommended retail price of Kodak digital cameras, Barry Hendy, Kodak Australia [WIKIMOORE]

Codifying knowledge about the real world in hard- and software has led to the arrival of *artificial intelligence*: systems that apply knowledge to perform better in the world around them. This intelligence can be applied to the management of system resources such as power and communication, but also to the emergence of higher cognitive functions, such as situation awareness, impact assessment or action planning.

The fourth and final development relevant for this introduction is the arrival of cheap *connectivity*. Although not as spectacular as the development of processing power, the increase of flexible bandwidth per EURO enables the appearance of systems of systems. In 2006, the IEEE Communication Magazine in an 'ON World study' [CHI2004] predicted that in 2010 more than 465 million radio frequency (RF) modules for sensors will ship, which is a significant increase when compared to 3 million in 2003.

As a token example for the combination of these developments, we consider the *intelligent sensor*: a smart agent in a diminutive physical form, able to sense the environment in a responsive and timely manner. It communicates wirelessly to its peers in order to perform collaborative and integrated sensing. It manages its power-, processing- and communication subsystems optimally under the circumstances given. The increasing abstraction level of the information that it generates creates an emerging cognitive intelligence that, when linked to databases and actuators creates the ultimate *Intelligent Sensor Network* (ISN).

This report describes models and methodologies to design and reason about intelligent systems, in an environment where these trends are leading. In this report, these intelligent systems will be referenced to as Intelligent Sensor Networks (ISN), in order to capture as much of the relevant developments as possible in the name for this class of systems.

1.1 Motivation

When applied properly, the developments described in the previous paragraph lead to increased scalability and reliability in systems, which in turn improve efficiency and efficacy. This is the realm of system engineering: to design systems that work efficiently and effectively.

In any reasonable complex system engineering's effort, multiple aspects of the system have to be considered: functionality, costs, physical setup, communication, power, interfaces, etcetera. These different aspects of a system are strongly related to each other, and have to be considered in a holistic manner. We argue that it is hard to reason about the role of (*artificial*) *intelligence* in those aspects.

1.1.1 Enabling technologies to build Intelligent machines

Technological advances in many different research areas have since centuries inspired people to envision a future society in which intelligent machines would coexist with humans. However, only recently the steady exponential nature of these technological advances are convincing many that such a society could become reality before the end of the century. As always there are those who embrace this development with optimism [KURZWEIL2005] while others have a more cautious mind [MARTIN2006]. The main technological developments that feed this belief are, rather arbitrarily listed:

Nanotechnology: Since the famous speech of Feynman there's plenty of room at the bottom [FEYNMAN1959], and much progress is made in exploiting that room. However, plenty of room is still left resulting in even smaller sensors, actuators, computers or complete robots. Other interesting developments are in the area of energy, where thin film technologies may decimate the price of solar cells and nanotechnology is used to develop cheap rechargeable batteries with a high energy density.

Computation: Moore's law is already reigning for decades and is expected to continue to do so for at least the coming two decades. Computational power comparable to the human brain will become available at affordable prices. Even if the methodologies used for artificial intelligence will not improve much, which is not to be expected, a million fold increase in computational power will enhance machine capabilities considerably.

Communication: Our society becomes more and more connected by wired and wireless communication facilities and although advances in this area are more constraint by physical processes, communication capabilities will increase substantially by using smarter (computational intensive) and more efficient methods.

Bio Engineering: Ongoing discoveries about the mechanisms of life including those of the human brain will narrow the gap between artificial and real life and will open up new forms of human machine interaction. Furthermore, the potential power, as already demonstrated by nature, of learning and evolutionary processes applied to many forms of artificial and real life will be substantial.

These technological developments will enable future systems to communicate more easily and co-ordinates their activities over larger areas within shorter time intervals. Also these systems of humans and intelligent machines will -in close interaction with each other and their complex physical environment- be able to adapt their behavior under changing conditions and situations to reach their common goals.

1.1.2 Domain experts

Specifically, we observe a lack in design models and methodologies around the topics of *sensors*, *intelligence* and *systems of systems*. These common assumptions in system design illustrate this:

- The network is reliable;
- Latency is zero;
- Bandwidth is infinite;
- The network is secure;
- Topology doesn't change;
- Transport cost is zero;
- The network is homogeneous;
- Intelligence is central;
- There is one (type of) user;
- Users are human;
- The system ends where the human begins.

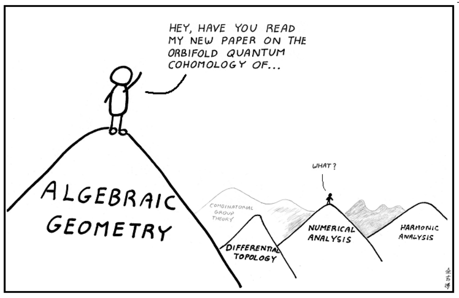

HEY, HAVE YOU READ MY NEW PAPER ON THE ORBIFOLD QUANTUM COHOMOLOGY OF...
WHAT?
ALGEBRAIC GEOMETRY
COMBINATORIAL GROUP THEORY
DIFFERENTIAL TOPOLOGY
NUMERICAL ANALYSIS
HARMONIC ANALYSIS
The Landscape of Modern Mathematics

Figure 2 – Mathematical experts communicating (Abstruse Goose)

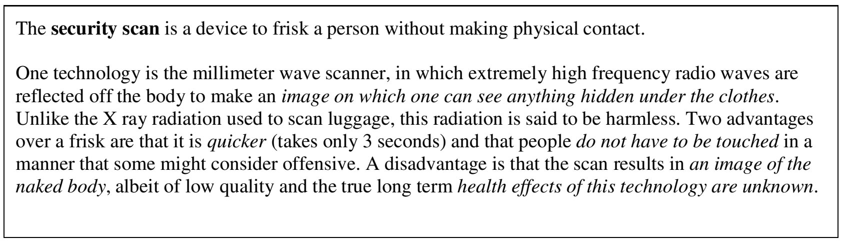
Experienced system engineers will recognize these traps and act to prevent falling in them. However, the engineering domains that are called upon are not used to working together. This is humorously illustrated in Figure 2 albeit only within the domain of mathematics.

The first car could be built based on knowledge of mechanical engineering and the first television based on electro-magnetic knowledge. As systems grew a bit more complex they could not be encapsulated within a single discipline. Instead, innovations took place by combining different engineering know-how. The role of mechanical engineering in current cars for example is merely 50%, while the other part is mainly electrical engineering. Complexity increases when one tries to combine more and more disciplines. Great efforts in this direction can still be found in the 20th century, although their solutions were mainly centralized: a complex system was divided into a number of different tasks done by a central processing unit or parallel units that were all connected together.

1.1.3 Cross-domain challenges

Good system engineers typically have more technologies in their toolbox than less experienced confreres. However, we also observe a trend that problem domains become more interrelated: mobility influences security, comfort and health, for example as illustrated by the discussion about body scanners in airports, Figure 3. This means that engineers are also challenged to step over the boundaries of their comfortable application areas.

It already helps to realise that your engineering effort is most likely merely an incremental improvement on a very small part of the larger problem domain. This suggests that it will pay off to consult a domain expert that has a broader view on the domain.



The **security scan** is a device to frisk a person without making physical contact.

One technology is the millimeter wave scanner, in which extremely high frequency radio waves are reflected off the body to make an *image on which one can see anything hidden under the clothes*. Unlike the X ray radiation used to scan luggage, this radiation is said to be harmless. Two advantages over a frisk are that it is *quicker* (takes only 3 seconds) and that people *do not have to be touched* in a manner that some might consider offensive. A disadvantage is that the scan results in *an image of the naked body*, albeit of low quality and the true long term *health effects of this technology are unknown*.

Figure 3 - Excerpt from Wikipedia illustrating the complex relations between application areas. [WIKISECURITYSCAN]

1.1.4 Design choices

The system engineer plays a pivotal role in the process from problem statement to solution. His (or hers) experience determines the toolbox from which the solution will be built and he is responsible for all major design choices – explicit or implicit.

These are several of the more advanced design choices to be made in current state-of-the-art projects:
- How adaptive is the system to its environment?
- What is the impact of a communication failure on the functionality?
- Can we rely on power harvesting for critical functionality?
- How does the reliability of the data, influence the decisions made by the system?
- Can we use the influence that we have on the environment of the system?

Intuition and heuristics have helped us deal with these questions in the past, but in order to increase the performance of our systems further, we need objective answers to these questions, staved with empiric data.

1.1.5 Wide range of complex systems

The range of types of systems that this report applies to is very wide. These systems comprise on one hand small devices that are limited in energy, computational capability and communication capability, but go to fully powered high data rate systems of systems that monitor complete nation-wide infrastructures, such as dikes, border security and highway roads.

Moreover, sensor network are deployed in a wide range of environments, which can put extremely difficult requirements on the design of the whole system. One such example is a system to monitor water temperature in oceans [REICHENBACH]. That system consists of large number of devices (4000) which are left to float in seas and oceans, and which gather data on water temperature. Maintainers are aware of the location of the devices, but 'hunting' them down in order to for example recharge them is costly.

Another such example is glacier monitoring. The Extreme Ice Survey (EIS) is started by the University of Southampton, UK with the goal to monitor the state of glaciers in Norway [EIS2010]. For this purpose a large number of small devices have been dispersed on glaciers, which will report on the state of the ice and the temperature. Again, a sensor network on a hard-to-reach place and thus a hard place to regularly perform maintenance.

EIS camera setups must withstand winds as strong as 160 mph, temperatures as low as -40°F, blizzards, landslides, torrential rain, and avalanches. The Extreme Ice Survey uses Nikon D-200 digital single-lens reflex cameras powered by a custom-made combination of solar panels, batteries, and other electronics. The cameras are protected by waterproof and dustproof Pelican cases, mounted on Bogen tripod heads, and secured against arctic and alpine winds by a complex system of anchors and guy wires. Each configuration weighs 70 pounds or more.

1.2 KaVoT Intelligent Sensor Networks

The core mission of TNO is to apply scientific knowledge with the aim of strengthening the innovative power of industry and government [TNO2010]. TNO receives part of its funding from the ministry of Education, Culture and Science as part of the KaVoT[1] programme. In the KaVoT *Intelligent Sensor Networks* the specific goal is to enable the development of immensely scalable monitoring infrastructures. The research strategy to accomplish this focuses on:
- artificial intelligence
- distributed systems
- internet of things

[1] Knowledge as an Asset across the Themes: TNO considers knowledge as power to be the enabling role of important fundamental technologies and fields of science. This leads to the creation of research programmes in the themes designated by the Advisory group. This relationship can be explained as followed: a theme is a social matter which is linked to an expert knowledge inquiry. These expert knowledge inquiries form the basis for demand-driven programmes. These programmes should lead to innovations and applications that benefit customers and other parties concerned. Science and technology, especially when developed in an international context, should be used to achieve this benefit.

This report is commissioned as part of the project *ISN Architecture* within this KaVoT, and is the result of an increasing collaboration of three core area's of TNO: ICT, Science & Industry and Defence & Security.

The knowledge in this KaVoT, this project and this report will be put into a master class "Designing Intelligent Systems". In this way, we hope the generated knowledge will boost work in all core area's of TNO, in all business models.

1.2.1 Master class - Designing Intelligent Systems
The master class will be based on the knowledge described in this report. Participants are expected to have:
- knowledge of basic system engineering methodologies (V-model, Waterfall model);
- 2 years working experience in software engineering or systems engineering.

A typical setup for the master class will be three consecutive days, which will be divided into theoretical mornings and practical afternoons. The practical exercises will be given along a hypothetical case, which is described in Appendix B: *Practical case: Balloon case*.

1.3 Report structure

The structure of this report consists of three main parts.

We start with a set of definitions of terms, and a short discussion on artificial intelligence in chapter 2: *Terms* and Definitions.

In chapter 3: *System Views* we introduce the different views on systems. These let us model all system aspects.

Chapter 4: *Design methodology* puts these views in context, and links them to several design methodologies. This chapter also elaborates on more holistic views on system engineering. Design patterns are introduced.

Chapter 5: *Result Cases* finally puts all this in practice in cases that carefully illustrate the application of the models and methodologies.

The report ends with 2 appendices:
A: Questions from research proposal and their answers
B: Practical case: Balloon case

2 Terms and Definitions

When talking about system design, one's experience and background matters a lot. There are many design philosophies, but many already assume a certain vocabulary. Notorious difficult concepts are *intelligence*, *architecture*, *system*, *model* and *performance*. In this section of definitions, the following references were used: [WIKIPEDIA], [RECHTIN2000].

Actuator: A device that converts a signal from a controller or an instrument into a physical quantity.

Data: propositions that reflect reality, such as measurements or observations of a variable.

Distributed: Spatial and/or geographical distribution of people, elements, processes, functionalities, data, information, etc…

Distributed system: Spatial and/or geographical distribution of a system into different subsystems.

Functionality: The aspect of a given utility or goal to an entity. The term functionality supposes the existence of one (purposeful) entity that attributes a function to another entity.

Fusion: The merging of similar or different elements into a union. In the context of this report always in one of these forms:

Data Fusion: combining data from different sources, typically on lower abstraction levels.

Information Fusion: combining information from different sources, typically on higher abstraction levels.

Knowledge Fusion: combining knowledge from different sources.

Sensor Fusion: generating a new, improved signal by combining multiple sensor signals.

Hybrid: Used as an adjective to a noun, stating that the noun has different modes which, within a certain time slot, will perform according to one of these modes.

Based on these definitions one can combine different words for different meanings. In the remainder of this report one will find intelligent sensor, sensor network, intelligent sensor network, distributed system etc. To be clear on their meaning:

Information: input to an organism or a system.

Intelligence: The ability to apply knowledge in order to perform better in an environment. This knowledge can be based on instincts, intuition, reasoning, models, etc…

Intelligent network: A network with the ability to apply knowledge of its surroundings, based on models and reasoning, on its network topology, communication links, routing/communication protocols, etc…

Intelligent sensor: A subsystem with the ability to apply knowledge of its surroundings, based on models and reasoning, on sensor data.

Intelligent Actuator Network: A distributed system with the ability to influence its direct surroundings. Each subsystem communicates via a network link and the intelligence can correspond to the actuator, the network or their interactions.

Intelligent Sensor Network: A distributed system with the ability to observe one or more regions of interest. Each subsystem communicates via a network link and the intelligence can correspond to the sensor, the network or their interactions.

Intelligent Sensor & Actuator Network: A distributed system with the ability to observe and influence its direct surroundings. Each subsystem communicates via a network link and the intelligence can correspond to the sensor, actuator, the network or their interactions. In the remainder of this report, ISN is short for 'Intelligent Sensor & Actuator Network'.

Model: A model is a representation of the essential aspect(s) of a real-life system (or a system to be constructed), which embodies and presents knowledge about that system in a usable form [EYKHOFF1974]. A few notes should be made:

- A model focuses on a particular aspect of the behaviour/characteristics of the system. Consequently (complex) systems are modelled by a set of (inter-dependent) models.
- Modelling is carried out with a purpose. Modelling is meaningful if the knowledge embodied in the model is easily useable for this purpose.
- Even for a given aspect several model types exist: e.g. conceptual, physical, mathematical, etc. models can be built. The model types determine how the model can be used ("executed"). For the purposes of system architecture design conceptual and mathematical models with well-defined semantics are the preferred formalisms because they support convenient and cost effective model building and allow automated execution of the models (evaluation of behaviour).

Network (telecommunication network): is a network of telecommunications links and nodes arranged so that messages may be passed from one part of the network to another over multiple links and through various nodes.

Observation: is either an activity of a living being (such as a human), consisting of receiving knowledge of the outside world through the senses, or the recording of data using instruments.

Performance: a measurement of some output or behaviour.

Sensor: A physical device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.

Sensor network: A number of sensors connected with each other via a network link (wired and/or wireless).

Signal: Output from a sensor.

System: A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities,

properties, characteristics, functions, behaviour and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected.
Alternative: A set of parts (example subsystems) for which it makes sense / helps to view them as a whole, for example, because together they perform a certain functionality. A system only exists in the eye of the beholder.

System's architecture: A (collection of) blueprint(s) for a system - ranging from functional to physical - that is needed to describe a system.

User: Two types of users can be distinguished:

End user: a person or system, which is not part of the system that is being considered, that has some sort of functional relationship with the system. The person or system that is most abstract (highest) in the functional view, and has the information need.

Human-in-the-loop / human-as-sensor: a person that functions as part of the system, for example as operator on any functional level, for example as signal interpreter. This definition is closely related to the area of human interfaces.

2.1 Artificial Intelligence

We observe that the field of artificial intelligence is not unified, and that subfields fail to communicate with each other. Pamela McCorduck [MCCORDUCK2004] writes of "*the rough shattering of AI in subfields—vision, natural language, decision theory, genetic algorithms, robotics ... and these with own sub-subfield—that would hardly have anything to say to each other.*"

However, we need such a common understanding at system engineering level because choices have to be made about adaption, cognition and learning. In this section we discuss a top-down outline for a taxonomy of artificial intelligence.

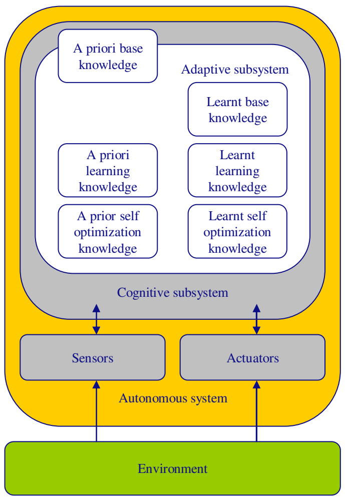


Figure 4 - Taxonomy for autonomous, cognitive and adaptive systems

We build on the terms and definitions from the previous section when we describe *autonomous systems*, *cognitive systems*, *adaptive systems*, *self learning* and *self optimizing*.

An autonomous system is a system that moves in an environment by means of merely its own sensors, actuators and a cognitive subsystem. The only influence the environment has on the behaviour of the autonomous system is through the signals that the sensors gather from the environment.

Autonomy can also be used for a subsystem, where it describes the level of independency of other subsystems in its surroundings. A subsystem that is controlled by another subsystem is less autonomous than a subsystem that changes its behaviour on the basis of a need from other subsystems (delivering a service).

Cognition is used for such subsystems when the focus is on understanding of the situation. Intelligence is used when the focus is on effective decision making. Both are closely related: for effective decision making good cognition is needed. For such systems, cognition and intelligence are synonymous. The term cognition is more specific for such systems and should therefore be preferred.

A cognitive subsystem contains several types of knowledge. We make the distinction between declarative knowledge (“data”) and procedural knowledge (“algorithm”). We also have the distinction in the dimension of adaptivity:

1. Base knowledge about perception and decision making;

2. (Self) optimization knowledge is all a priori knowledge that concerns adapting the behaviour of the system in different situations.

3. Learning knowledge or “knowing how to learn” is knowledge that is used to gather new knowledge by experiencing (new) situations, storing that knowledge and by doing so in potential improving all forms of knowledge.

When we assume that base knowledge is necessary for any system (and therefore present in all subsystems) we can now distinguish these classes of subsystems:

1. Self optimizing subsystems: optimizing the use of base knowledge, choosing algorithms smart;

2. Learning subsystems: learning of parameter settings, etc.;

3. Self optimizing learning subsystems: the learning method is adapted in order to learn as efficiently as possible;

4. Learning self optimizing subsystems: it is learnt how to optimize in which type of situation;

5. Etc.

Subsystems that use self optimizing and/or learning knowledge are called adaptive subsystems. We have to remark here that learning knowledge can be put to use by the system engineer in 2 different ways: in the design phase or in the operational phase. It is the engineers challenge to determine which base knowledge, self optimizing knowledge and learning knowledge is needed to efficiently realise an efficient and effective system.

3 System Views

The architecture of a system acts as a blueprint for future implementation, review and accountability. When considering a future implementation, one starts with an architecture design which is then assessed after a number of pre-defined criteria. The question is to what it should be assessed, because the results are used in the next iteration of a redesign. One way to assess is to model the architecture in different subtypes of models after which a designer can calculate the performance of its design. Depending on the types of modelling-methods that are used it could provide an answer as to which aspects of a design are good and which should be improved. For example, one criterion could be the power need. For a specific design the power need can be calculated, and it can be determined whether this does not exceed the available resources.

Now, having this in mind, we need to look further at the process of designing such sensor networks. While there are some system engineering methodologies, there is also no single ‘silver bullet’ which will provide a recipe for successful design of sensor networks. In the end it is up to designer to make the right choices, given the use case and constraints.

The design process has a number of dimensions and a system under design is subdued to a number of constraints, so the design process should be step-wise. In chapter 4: *Design methodology* we discuss such processes. During the design process, the system designer makes a number of assumptions, for example, depending on your background, it could be assumed that an ISN is a distributed system where:

- The network is reliable;
- Latency is zero;
- Bandwidth is infinite;
- The network is secure;
- Topology doesn't change;
- Transport cost is zero;
- The network is homogeneous;
- Intelligence is central;
- There is one (type of) user;
- Users are human;
- The system ends where the human begins.

Of course, these assumptions are here only because they helped the designer in the past to make some initial choices, in order to deal with the requirements (or constraints) of previous efforts. We argue that these heuristics and instincts based on yesterday’s knowledge are not enough for today’s problems. So we need to develop new insights into the dependencies and relationships between the different aspects of a system, taking new knowledge areas into account.

So, architectures can be modelled with respect to various viewpoints. Different viewpoints are of importance to different applications. The following list describes different viewpoints to model architectures and indicates why they could be important.

- Organisational/context
- Legal
- Costs

- Functional
- Temporal
- Physical
- Communication
- Dependability
- Security
- Human Machine Interface (HMI)

All these views must be taken in account for a given situation, in order to have optimal system architecture.

3.1 The role of modelling for architecture design

In order to make architectural decisions, the "quality" of a particular design should be measured (and then compare it with other designs). It can be done if the system architecture selected is modelled (formally). Then the model can be executed ("used") and the relevant performance figures can be calculated. Consequently the formal modelling of the architecture is a fundamental part of the design process.
Unfortunately the system architecture modelling in itself is a demanding challenge. It is important to realize that there is no single structure, which can describe the system architecture. We are able to talk about architecture of a system if we define our "view". The architectural view defines the "perspective" through which we look into the system. There are multiple relevant views if we want to characterize a design – this characterization is called the model of the system with respect to a particular view. Each model is complete, i.e. it describes the design fully – but from the point of the view it represents. Different designs may result in different models, which can be compared in the context of the view.
As in the case of every at least moderately complex system, the views, jointly describing architecture, are not independent. The dependency has different manifestations:

- Coupling: Changing the design in one view may (and usually does) influence certain system level properties. The magnitude of the change may depend on design choices described by other view(s), i.e. depending on other models in other view(s) the effect of the design change can be amplified or damped.
- Constraining: Models in one certain view can set constraints for valid designs in another view.

Managing the interactions among views is the most challenging aspect of the system design process. The design methodology described in the next chapter addresses this problem domain. This chapter gives the "foundation" to the methodology.
In order to have a "guide light", which helps the designer navigating in the overwhelmingly large design space a method (and related tools) is needed to measure the "quality" of the design alternatives considered. This "guide light" can be realized via a model based design process.

- A modelling formalism should be defined for the various views.
- Formalism should be defined to describe the coupling between the views.
- "Execution semantics" should be defined, which defines the model's behaviour when executed (typically in a simulator).
- A performance measure should be defined, which reflects the "integral quality" of the design expressed by the set of models.

In the following a few characteristic, established system architecture approaches are presented. As it will be also shown none of them in the original form satisfies the "guide light" requirement for ISN related system design. Following the literature overview those views are identified, which play fundamental role in ISN design. Modelling formalism is proposed for each view and the interactions among the views are also explained. The full formal definition of the views, interactions and semantics is far beyond the scope of this report. The intention of the chapter is to provide sufficient insight to the modelling and model usage – which form the foundation of the systems engineering process introduced in the next chapter. However, the modelling and evaluation methodology proposed is independent of the design process to be introduced. Other systems engineering processes, which are capable of incorporating model based, quantitative evaluation (decision making) steps, can also benefit from the modelling framework.

3.1.1 Architecture modelling: the established approaches, shortcomings

With developments in miniaturized sensors, batteries, and processors (microcontrollers) sensor networks has been gaining in significance. Originating in a simple need to observe particular processes, sensor networks have originated from a single deployed sensor at some location, and grew in complexity to large scale, self-organizing, and cognitive networks.

Traditionally, issues of large scale signal processing, of distributed control, and distributed signal processing has been domain of software engineering, and therefore most methodologies for designing ISN are based on those. In this chapter we will present most of those design methodologies, and evaluate their 'fit' for designing ISN-type systems.

3.2 Literature overview

In the following section we will provide an overview of the main design methodologies that are used for designing ISN. Most of them were developed for the design of software systems, but thanks to similarities with issues we might face when designing an ISN, these methodologies can be used as a good starting point.

Note: all graphics and their captions were taken directly from the articles.

3.2.1 4+1 Model view of Architecture

This paper [KRUCHTEN1995] introduces the idea that a single architectural diagram is not sufficient to capture the components of a software system which are relevant to all stakeholders involved. Any diagram attempting to do this for any reasonably complex system would be confusing and messy if not misleading. Different views on the system should be used, where each view corresponds to a particular subset of stakeholder concerns. Each view should be a complete perspective of the system but highlight different aspects, just as one observes an object from different angles. Presumably all views will not be relevant to all stakeholders, but every stakeholder should find his concerns met by at least one view.

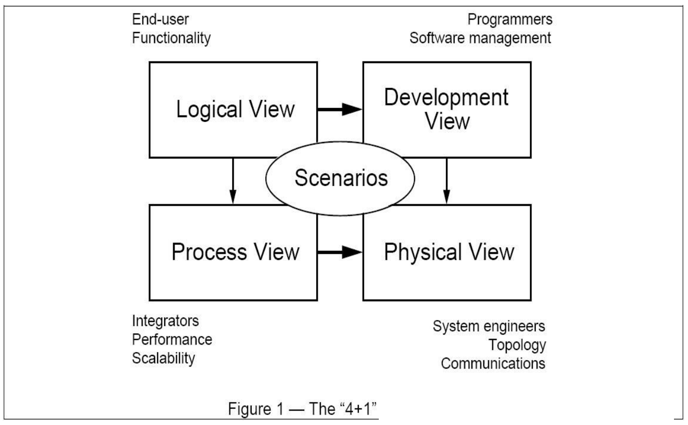


Figure 5 - 4+1 view model

In this article the author states that four separate views of the same system are necessary to convey the most important the aspects of the system architecture. These four views are:

The logical view supports the functional requirements and is primarily (but not necessarily) object-oriented.

The process view focuses on some of the non-functional requirements such as system performance, availability and fault tolerance. The author defines a process as "a group of tasks that form an executable unit. The process view represents the level at which the processes can be tactically controlled."

The development view "focuses on the organization of the actual software modules in the software development environment." The development view is useful for the organization of the project teams in charge of building and developing the software.

The physical view models the hardware that is supporting all of the elements in the other three views, and the objects, processes and software modules must be mapped onto the physical entities identified in the physical view.

Finally the +1 view is a set of example scenarios that illustrate the system in action, and show how the four views describe the working system.

The article explains further the correspondence between the views and describes the 4+1 model as an iterative approach with the scenarios playing a crucial role. This methodology starts with a few carefully chosen scenarios, continues with an architecture sketch and then is organized into the four views. The architecture is implemented and tested against the original scenario and other scenarios, which are then assessed, changed or extended and tested against the scenarios. In this way the architecture and its implementation slowly evolve to become the real system.

3.2.2 IEEE1471 - Recommended practice for architectural description of software intensive systems.

The standard [IEEE1471] establishes a standard terminology for the architecture descriptions of software intensive systems. It formalizes the use of different architectural views as a means of illustrating particular system properties for the relevant stake holders. The standard defines the important concepts such as *architect*, *life cycle model*, *system* and *view*. It gives a conceptual framework for software architecture and then considers different architectural description practices.
The conceptual framework is shown in figure 1 below. The standard carefully defines all the necessary terms and there interrelation. For example the explanation of how system mission and stakeholder relate: “A system exists to fulfil one or more missions in its environment. A mission is a use or operation for which a system is intended by one or more stakeholders to meet some set of objectives.”

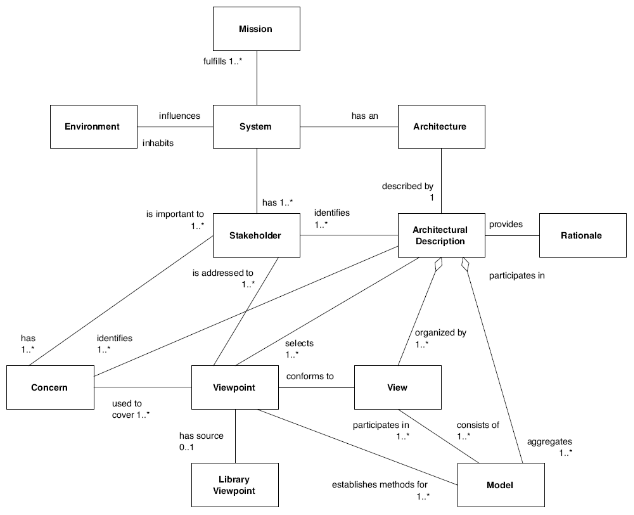


Figure 6 - Conceptual model of architecture description

The standard goes on to address stake holders and their concerns, and architectural activities in the life cycle, and places these ideas within the conceptual framework. The standard also identifies some architectural description practices and lists some of the issues involved in those practices. The practices identified are: architectural documentation, identifying stakeholders and concerns, selecting architectural viewpoints, and architectural rational.

3.2.3 The Reference Model of Open Distributed Processing

This reference model [RM-ODP] establishes a standard language and terminology for traditional distributed systems. The model covers a wide range of topics including distribution transparencies, standard system viewpoints, object modelling, concepts of abstraction and encapsulation.

The distribution transparencies are a valuable tool for conceptualizing the issues involved in distributed processing. Their purpose is to hide a particular aspect of the inner workings of distribution from the user. A few examples of transparencies are: failure transparency which hides the failure and possible recovery of an object enabling fault tolerance measures, location transparency which hides the spatial location of a particular object, and replication transparency which hides a group of objects posing as one object.

The system viewpoints introduced in the article were of the most interest to us and we drew on them heavily for inspiration for the ISN viewpoints. The five viewpoints in ODP are Enterprise, Information, Computation, Engineering, and Technology.

The *Enterprise viewpoint* describes the system in its environment with respect to the stakeholder concerns and policies that govern the system. One of the key concepts in the enterprise view is that of a contract between enterprise objects, members of a community with a shared objective, linking the various roles and mutual obligations of the enterprise objects. The enterprise objects are always real things or organizations in the enterprise view; they are never software abstractions or purely functional objects.
The *Information viewpoint* focuses on the semantics of the information that is exchanged and processed in the system. The information language defines meta-data concepts to facilitate data processing and exchange.
The *Computational viewpoint* is a function decomposition of the system into objects which interact at interfaces.
The *Engineering viewpoint* is concerned with the resources needed (both hardware and software) to support the distribution and interaction of the objects specified in the Computational viewpoint.
The *Technology viewpoint* of ODP focuses on the implementation of the system, in terms of what software and hardware technologies would be used.

The ODP article was very useful as an inspirational tool for the ISN modelling, especially the enterprise view; however an important difference between their approach and ours is the fundamental coupling between hardware and software in the ODP article, where we have taken great pains to keep them separate.

3.2.3.1 Enterprise view in RM-ODP

The Enterprise view on model the system is modelling the system on a business level. The ISN/ODP enterprise Description describes the following entities and their relationship:

- Enterprise objects.
  - Actors
  - Resource
  - Artefacts
- Role. Enterprise objects can have multiple roles. A person can be a car owner and a car driver. Owners may have different objectives than drivers.
- Behaviour
- Action
- Communities (groups of enterprise objects with a common objective)
- Policies
- Contracts
- Objectives
- Process
- Steps

Enterprise Objects can be part of the "real" world or can be part of the system we are building. The state the problem step only defines the Enterprise objects that are part of the "real world"

The ODP Enterprise cannot be applied to ISN applications without modifications. The major differences between the systems that were targeted for ODP and ISN are that ISN involves Real World Objects and Processes and actually may interact with the Real World. The “players” in the ISN world may have opposing objectives. ODP was originally targeted at administrative systems in a distributed environment. In the administrative world there is an illusion that rules are complied with and that the system objectives are aligned. Other views of ODP are more applicable to ISN.

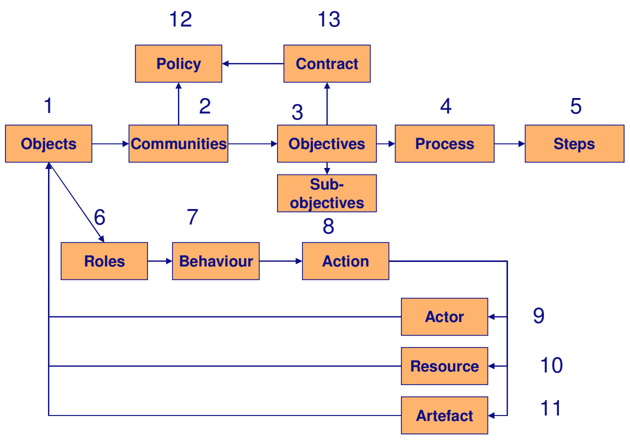


Figure 7 - The relation between the elements of an ODP Enterprise Description

The numbers in the diagram are indication of the order in which the blocks are identified/described (objects → communities → objectives → process → steps, roles → behaviour → action → Actor/Resource/Artefact, Policies → contracts)

3.2.4 Comparing Architectural design styles

This article [SHAW1995] compares 11 different design styles on an example problem of a cruise control system. The design styles are grouped into four architecture categories: Object Oriented, State Based, Feedback-Control and Real Time. The purpose of the article was not only to demonstrate the use of the different styles but also to illustrate that different design styles are not simply a matter of taste but will lead to systems with very different properties.
Figure 8 shows the cruise control system, where the left side of the diagram shows the system inputs and the output is the throttle adjustment.

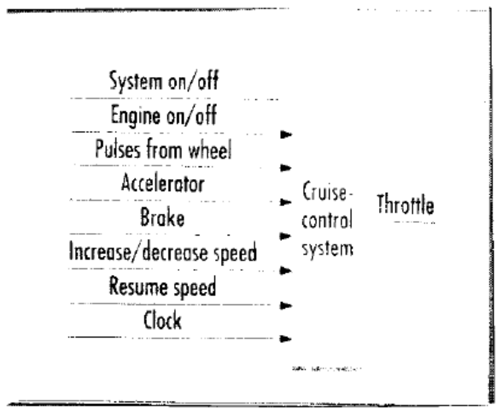


Figure 8 - Block diagram for the cruise control system

Shaw summarizes the different architecture categories and then gives three or four examples per architecture category to illustrate how different authors dealt with the cruise control problem.

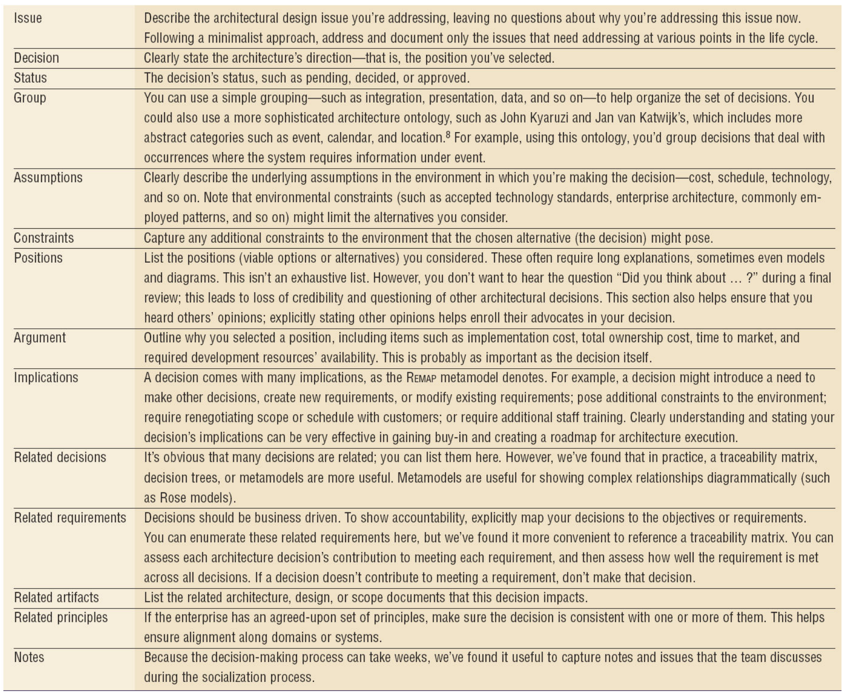
Shaw then goes on to give a criterion for evaluating the end system designs. The criterion include: separation of concerns, ability to analyze the design, and abstraction power. One of the most important ideas in the article is that architecture descriptions (designs) can be scrutinized and evaluated in a systematic way to determine if they are useful to the system developer and/or user.

3.2.5 Architecture decisions: demystifying architecture

This article [TYREE2005] presents an additional architecture viewpoint that is not addressed explicitly in the other architecture models called the decision viewpoint, and the authors claim that it is the most relevant viewpoint. They state if it is used by the system architect, it will also be the most used viewpoint by system implementers and future designers who need to make additional changes to the system.

According to the article the traditional approaches such as RM-ODP and 4 + 1, deliver a full architecture, but the rationales for the decisions made to produce the architecture are hidden from the stakeholder who is interested in doing something with the architecture. The stakeholder must take the completed design on faith and the stakeholder does not know to what extent the environment and constraints of the system designer influenced his decisions or even if the constraints are still valid. Fully documented decisions would demystify the architecture and also provide a short cut for the stake holders who don’t have the energy to read through all the architectural views to understand the architecture.

The authors believe architecture decisions should be elevated to first class status and that fully documenting and socializing (convincing others that the decision is the right one) architecture decisions are more important than any architecture view.
Figure 1 presents their template for documenting the decisions. After presenting the template the article goes on to illustrate its use in an example of a large financial organizations IT systems.



Issue	Describe the architectural design issue you're addressing, leaving no questions about why you're addressing this issue now. Following a minimalist approach, address and document only the issues that need addressing at various points in the life cycle.
Decision	Clearly state the architecture's direction—that is, the position you've selected.
Status	The decision's status, such as pending, decided, or approved.
Group	You can use a simple grouping—such as integration, presentation, data, and so on—to help organize the set of decisions. You could also use a more sophisticated architecture ontology, such as John Kyaruzi and Jan van Katwijk's, which includes more abstract categories such as event, calendar, and location.[8] For example, using this ontology, you'd group decisions that deal with occurrences where the system requires information under event.
Assumptions	Clearly describe the underlying assumptions in the environment in which you're making the decision—cost, schedule, technology, and so on. Note that environmental constraints (such as accepted technology standards, enterprise architecture, commonly employed patterns, and so on) might limit the alternatives you consider.
Constraints	Capture any additional constraints to the environment that the chosen alternative (the decision) might pose.
Positions	List the positions (viable options or alternatives) you considered. These often require long explanations, sometimes even models and diagrams. This isn't an exhaustive list. However, you don't want to hear the question "Did you think about … ?" during a final review; this leads to loss of credibility and questioning of other architectural decisions. This section also helps ensure that you heard others' opinions; explicitly stating other opinions helps enroll their advocates in your decision.
Argument	Outline why you selected a position, including items such as implementation cost, total ownership cost, time to market, and required development resources' availability. This is probably as important as the decision itself.
Implications	A decision comes with many implications, as the REMAP metamodel denotes. For example, a decision might introduce a need to make other decisions, create new requirements, or modify existing requirements; pose additional constraints to the environment; require renegotiating scope or schedule with customers; or require additional staff training. Clearly understanding and stating your decision's implications can be very effective in gaining buy-in and creating a roadmap for architecture execution.
Related decisions	It's obvious that many decisions are related; you can list them here. However, we've found that in practice, a traceability matrix, decision trees, or metamodels are more useful. Metamodels are useful for showing complex relationships diagrammatically (such as Rose models).
Related requirements	Decisions should be business driven. To show accountability, explicitly map your decisions to the objectives or requirements. You can enumerate these related requirements here, but we've found it more convenient to reference a traceability matrix. You can assess each architecture decision's contribution to meeting each requirement, and then assess how well the requirement is met across all decisions. If a decision doesn't contribute to meeting a requirement, don't make that decision.
Related artifacts	List the related architecture, design, or scope documents that this decision impacts.
Related principles	If the enterprise has an agreed-upon set of principles, make sure the decision is consistent with one or more of them. This helps ensure alignment along domains or systems.
Notes	Because the decision-making process can take weeks, we've found it useful to capture notes and issues that the team discusses during the socialization process.

Figure 9 - Architecture decision description template

3.2.6 A Survey of Software Architecture Viewpoint Models

This article [MAY2005] explores the different viewpoints of five models and examines the topics that the viewpoints cover. The author has a table of structures, stakeholders and concerns, and the author quantifies the coverage of the models by examining how many of the items in his table are addressed by the viewpoints in the five different models. The author goes on to select a few different viewpoints from the models and declares that this is a collection of viewpoints with maximum, but not complete coverage.

The author's language and framework for comparison were taken from the IEEE standard for software intensive systems IEEE 1471. The main concepts that were used form the standard are shown below in figure 1.

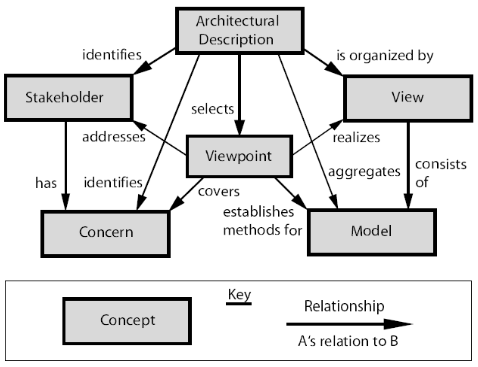


Figure 10 - The concepts associated with viewpoints and their relations.

The framework that the author uses for evaluating the coverage of the different models is shown in Table 1 below. The author examined the different viewpoints in the models to see which aspects of the comparison framework were addressed. An aspect of Table 1 is only covered by a viewpoint if it is explicitly covered, the author does not consider an aspect covered if it is implicitly addressed.

Structures	Stakeholders	Concerns
Decomposition	Architects and Requirements Engineers	Usability
Uses	Sub-System Architects and Designers	Performance
Layered	Implementers	Space
Abstraction	Testers and Integrators	Reliability
Process	Maintainers	Portability
Concurrency	External System Architects and Designers	Delivery
Shared Data	Managers	Implementation
Client-Server	Product Line Managers	Standards
Deployment	Quality Assurance Team	Interoperability
Implementation		Ethical
Work Assignment		Privacy
		Safety

Table 1 - The elements of the initial comparison framework

The models examined by the author were: Kruchten's "4 + 1" View Model, Software Engineering Institute (SEI) set of views, ISO Reference Model of Open Distributed Processing (RM-ODP), Siemens Four view model, and Rational Architecture Description Specification.

The author summarizes the models and then states how they fit in his coverage analysis. The 4+1 model and the Open Distributed processing, are summarized above in this literature review, what follows is a short summary of the other three models.
The SEI model does not have a specific set of viewpoints like the other models; instead it has a complicated process of documenting software architecture in a way that extends the IEEE 1471 standard. The viewpoints arise from the structure of the system itself, and each viewpoint is composed of view packets which are the smallest piece of information that a stakeholder requires. The SEI model is heavily stake-holder centric, instead of offering a predefined set of views for a system, the SEI model offers many different view packets; a few of which are selected and combined into views based on the stakeholder concerns.

The Siemens Four View model is a result of a study of industrial practices of software architecture and the focus of the model is on the software architect, as a result the four view model does not address stakeholder concerns other then the software architect. Like the 4+1 model, the Four View model breaks software design into four categories shown in Figure 11.

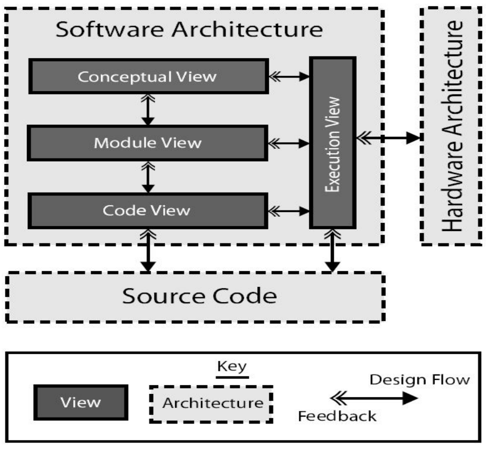


Figure 11 - Siemens four view model of software architecture

The Rational ADS is an expansion of the "4+1" model into a wider range of systems such as enterprise, e-business, embedded systems and non-software systems. Some of the viewpoints have been renamed and four new views have been added. Additionally all nine views have been grouped into four categories shown in Figure 12.

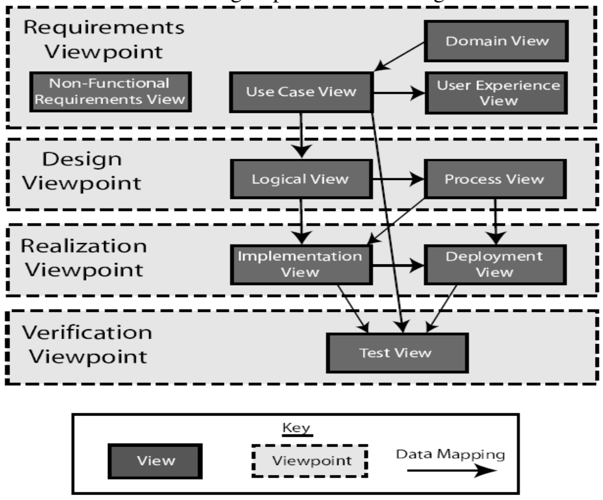


Figure 12 - The rational ADS viewpoints and views

The survey results are shown in Table 2 below. The models received a point for every aspect in Table 1 that was covered, and the points were kept separated by category.

Model	Stakeholders	Concerns	Structures
“4+1”	7	7	11
SEI	8	9	11
RM-ODP	2	9	9
Siemens	1	7	9
Rational ADS	9	10	10
Framework	11	12	11

Table 2 - Concept coverage of the surveyed viewpoint models

The author then selects a few viewpoints from different models and states that this collection of viewpoints is an optimal coverage set. The viewpoints are: the requirements viewpoint from Rational ADS, the module view type from the SEI model, the C&C view type from the SEI model, and the Allocation view type from the SEI model.

The author does point out that the main difficulty in comparing and combining the viewpoints for the optimal set was the fact that the viewpoints are tightly or loosely coupled depending on the various models. For example the viewpoints in the SEI and RM-ODP models are relatively independent while the viewpoints in the 4+1 model have a strong data flow between the different viewpoints and the viewpoints are developed in an iteration sequence.

3.2.7 Model Based Evaluation: From Dependability to Security

This article [NICOL2004] is a discussion of the techniques of quantitative system dependability evaluation and how these techniques could be used to evaluate system security. The article discusses what is necessary for evaluating system security and demonstrates how the existing techniques for evaluating system dependability can be used to fulfil some of the requirements for security evaluation. The authors also discuss the challenges for quantitative security evaluation and suggest some future areas of research.

The authors’ main focus is on stochastic dependability techniques and how these techniques can be used in the domain of system security evaluation.

The authors first discuss measures and definitions of dependability and of security. Dependability analysis in general considers the continuation of service at a particular quality over a period of time during random partial hardware and software failures. Security analysis also considers the continuation and quality of service, however the nature of the failures are different and cannot be assumed to conform to a random distribution as attacks on the system may be connected with system states and attackers can learn over time. There are also additional measures of system security that are not included in measures of dependability. Measure of dependability includes reliability, availability and performability. These concepts are also included in measures of system security but new measures such as data confidentiality and data integrity, which measure the systems ability to inhibit unauthorized persons from gaining access to or changing data, need to be considered as well.

The main focus of the article is on model representation and analysis techniques. The authors exhibit existing techniques for quantitative dependability analysis and then discuss how these same techniques could be used or modified for security analysis.

The dependability analysis technique that most easily translates to the security domain is the combinatorial method called the fault tree. The fault tree is an acyclic graph with internal nodes as logic gates and external (leaf nodes) as system components. If the right combination of system component failures leads to a failure of the root node then the system has failed, or is crippled in some critical way.

The security analysis version of a fault tree is an attack tree; it is structured the same way where a security breach is equivalent to a component system failure in the fault tree. The goal of the attacks is the root node and when that is reached the system is considered to be compromised.

The leaves on attack trees can be assigned continuous variables in addition to Boolean variables. The continuous variables can represent the probability of an attack or the likelihood that an attack will succeed or the dollar cost of defending the node from attack. If continuous variables are used then the method of propagating the cost up the tree is that AND nodes are the sum of the costs of their children while cost of the OR node is the same as the child of lowest cost.

The authors then progress to stochastic techniques of dependability analysis with a focus on Markov Reward models. The authors first note that combinatorial methods such as attack trees only address the issue of compromised system components but does not consider the fact that some system components must be compromised first before others are vulnerable. Markov reward models can address the complex relationships between system components, by assigning rewards to states or to transitions between states in the state space.

After a discussion of the technical aspects of the Markov Reward Modes the authors discuss techniques for dealing with complex systems with large numbers of components where the number of possible system states is too large to be dealt with in a reasonable amount of time. Two categories of techniques are mentioned for dealing with large state spaces. The first is largeness avoidance techniques where several states are lumped into one state or the Markov Chain simply operates on a higher level abstraction of the state space. The second technique is called the largeness tolerance technique. This technique is a class of special algorithms and data structures that are built to handle large state spaces.

After the survey of state based techniques for dependability analysis the authors discuss the current and possible future research in state based techniques for security analysis. In order to apply Markov Reward models to security analysis, the definition of reward must be adapted and whereas system component failures can be assumed to be random in dependability analysis, appropriate distributional properties must be ascribed to attacker behaviours. There have been a few examples of this happening in specific cases, but no generic model of attacker behaviour or accepted definition of reward has been agreed upon. Examples of reward definitions for security modelling include the effort needed by the attacker to make a state transition from “working” to ”compromised” or the mean time needed to make such a transition.

The authors go on to identify promising applications in security analysis, and to identify what is needed to realize these applications. One of these applications is to quantify the intrusion tolerance of a system, i.e. the ability of a system to perform high level functionalities in spite of attacks. What is needed is a system representation that uses state models which capture the system vulnerabilities. Attacker behaviour must be modelled and measurement parameters must be applied to quantify the system vulnerabilities.

Another method of system security analysis is simulation, i.e. evaluating a system by examining individual trajectories or behaviours of the system in a particular instance. Simulation is done as a way to estimate the measurements of system vulnerabilities. The authors discuss some of the statistical issues involving simulation (i.e. how can one be certain that the simulated results match what would happen in reality) and provide examples of simulation in security analysis.

3.2.8 Specification and Modelling of Dynamic, Distributed Real-time Systems

The article [WELCH1998] defines and presents a specification language for distributed real-time systems. The article introduces the language for describing these systems and for describing the quality of service issues that arise in discussing these systems in terms of end to end paths. The authors make the distinctions between static and dynamic systems and present generic models for each. The authors also present techniques for quality of service analysis in dynamic systems. They then present some experimental results for system behaviour on a "benchmark" situation assessment path. The article first introduces and explains the dynamic real time path paradigm in order to set the stage for further discussion. The authors describe a path based real time subsystem and its attributes. Figure 13 displays the real time subsystem.

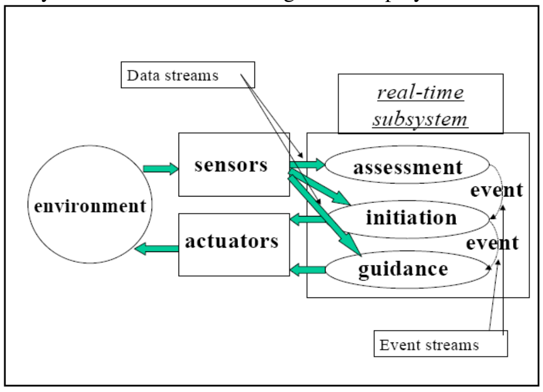
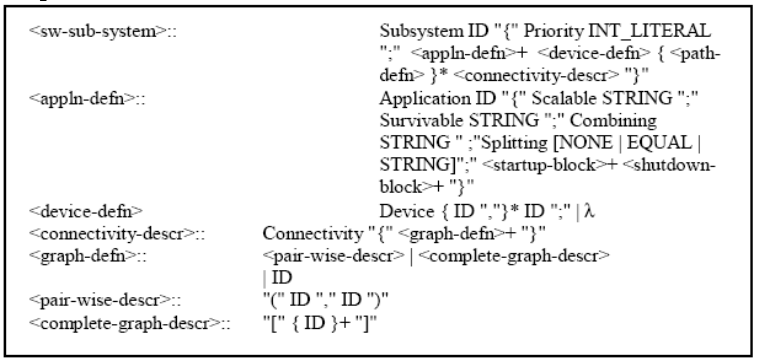


Figure 13 - A real time subsystem

The example they use is that of an air defence system where the threats are assessed which could result in an action being initiated. The initiation action is then immediately followed by action guidance until the action is completed. This example illustrates the three kinds of actions in real time systems: continuous, transient and quasi-continuous. The assessment action is continuous as it is continually running. The initiation action is transient as it is triggered by an event from the assessment action. The initiation action completes a single cycle of actions and then stops. The guidance action is quasi continuous as it is triggered by an external event (just like the initiation action), but then once active it continuously runs its actions, cycle after cycle until the main action,

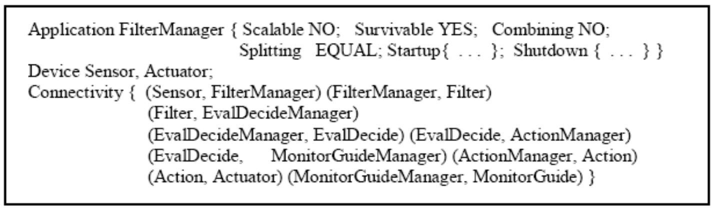
which was initiated by the initiation action, is completed. Each category of actions has timeliness objectives associated with each action cycle. The continuous and transient actions only have cycle timeliness objectives (the transient actions have only one cycle) while quasi-continuous actions have the cycle timeliness objective for each cycle of guidance actions, but also have a timeliness objective for the action it is guiding, i.e. the main action must terminate within a specific time.
With the above concepts in mind the authors turn to their specification language and introduce the grammar for such a language. The authors claim that their language can cover many aspects and relationships of dynamic real-time systems including interconnectivity and run time execution constraints as well as software and hardware structure; however the focus in the article is purely on software and quality of service objectives.

Figure 14 describes the grammar of a subsystem and Figure 15 shows an example of that grammar.



```
<sw-sub-system>::                          Subsystem ID "{" Priority INT_LITERAL
                                           ";" <appln-defn>+ <device-defn> { <path-
                                           defn> }* <connectivity-descr> "}"
<appln-defn>::                             Application ID "{" Scalable STRING ";"
                                           Survivable STRING ";" Combining
                                           STRING " ;"Splitting [NONE | EQUAL |
                                           STRING]";" <startup-block>+ <shutdown-
                                           block>+ "}"
<device-defn>                              Device { ID ","}* ID ";" | λ
<connectivity-descr>::       Connectivity "{" <graph-defn>+ "}"
<graph-defn>::               <pair-wise-descr> | <complete-graph-descr>
                             | ID
<pair-wise-descr>::          "(" ID "," ID ")"
<complete-graph-descr>::     "[" { ID }+ "]"
```

Figure 14 - Grammar for subsystems



```
Application FilterManager { Scalable NO;  Survivable YES;  Combining NO;
                            Splitting  EQUAL; Startup{ . . . };  Shutdown { . . . } }
Device Sensor, Actuator;
Connectivity {  (Sensor, FilterManager) (FilterManager, Filter)
                (Filter, EvalDecideManager)
                (EvalDecideManager, EvalDecide) (EvalDecide, ActionManager)
                (EvalDecide,     MonitorGuideManager) (ActionManager, Action)
                (Action, Actuator) (MonitorGuideManager, MonitorGuide) }
```

Figure 15 - Application, device and connectivity specifications

The subsystem is specified by its priority as well as its end to end path time definitions and a graph of the communication connectivity of the applications and devices. Further there are Boolean variables which specify whether an application can be replicated for survivability and whether the replicas can support load sharing for scalability. The article also has examples of path and data stream specification.

The authors then go on to present an overview of a static model and of a dynamic model with all of the relevant mathematical notation for complete system specification. In the static system model the quality of service requirements are required latency, required throughput of the data stream and data inter-processing time. They allow for the possibility of sampling windows which allow a maximum number of quality of service

violations within a given window. The dynamic system model focuses on providing the mathematical syntax for application replicas and load sharing between the replicas.
The article also contains presentation of dynamic quality of service analysis techniques and adaptive quality of service management. Notation is introduced for the monitoring of quality of service by collecting time stamped events from the applications. Quality of service metrics and their notation are discussed (which are an expanded version of the QoS requirements described in the static model). Finally diagnosis of QoS failures and recovery actions are discussed. The recovery actions are based on reallocation of resources and replication of applications.

3.2.9 4D/RCS - A Reference Model Architecture for Unmanned Vehicle Systems, Version 2.0

This report [4DRCS2002] provides a good and comprehensive introduction to 4D/RCS reference model. 4D/RCS is an offspring of the RCS (Real-time Control Systems) reference model architectures of NIST. The history of the RCS reference model dates back over 30 years and went through 4 major phases of development. The resulting versions are interesting as it shows progress in the field of Real-time Control Systems. The versions are:

1. RCS-1 is a hierarchical cerebrellar model for control systems. It had elements for intelligence linking input to output and able to distinguish between multiple levels;
2. RCS-2 made a split between sensory processing and the behaviour generation in a way very similar to the NAIHS architecture;
3. RCS-3 introduced the concept of a World Model between sensory processing and behaviour generation. The World Model is a virtualisation of the real world the control system operates in and act as a buffer between sensory processing and behaviour generation;
4. RCS-4 added the concept of Value Judgement. As the World Model, Value judgement is positioned between Sensory Processing and Behaviour Generation. Value Judgement is functionality which allows the numeric assessment of situations and plans to cope with the situation. 4D/RCS is the best known implementation of RCS-4

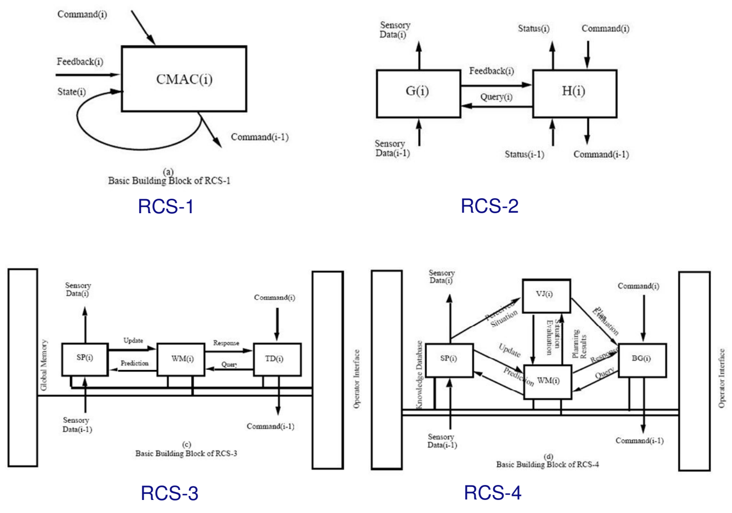


Figure 16 - Historic development of the RCS building blocks

4D/RCS is an implementation of RCS-4 for unmanned autonomous army vehicle as part of a larger army unit.

The building blocks can communicate up, down with building blocks on the same level. 4D/RCS has a strict hierarchy. RCS seems like a good approach to real-time control systems but has some shortcomings when applied as general reference model for Intelligent Sensor Networks. Some significant shortcomings are:

1. The model is purely hierarchical with the ability to communicate horizontally as well. It cannot cope with non-hierarchical structures such as Sensor Networks in a multi organisational environment;
2. RCS assumes unlimited communication and energy which could be the case with unmanned army vehicles but is definitely not the case with many ISN's based on wireless network technology;
3. There is only sensing and control on the bottom layer; a general does not push a button.

3.3 Challenges in ISN architecture design

In the introduction to this chapter we have briefly pointed out why the design of ISN is different 'beast' than distributed software design. As with a lot of other things, it is details, concrete implementations that pose major challenges. When describing them generally, they are similar to what we can find in software design:

a) resource constraints
b) inherently dynamic
c) configuration cardinality

Let's look at the nature of these constraints in ISN context.

Resource Constraints include limitations on all elements of sensor nodes

- Sensing element - its accuracy, reliability, drift (due to changing temperature, pressure, aging,…)
- Processing element - its processing power (number of operations per second)
- Controlling element - as above, control takes also number of computational steps.
- Communication element - capacity of the communication medium, number of participating (communicating) nodes, possible collisions, maximal available data rate, etc.

Inherently dynamic configuration in ISN can come from a few possible sources. It can be either planned - when some sensor nodes are periodically switched off or placed into sleep mode, in order to save energy. This results in changes in ISN topology and paths that can be used for data transport. Another source of dynamic in ISN is communication - when some communication links (fixed or wireless) are failing or are being overloaded, and thus resulting in packets (data) losses. Finally, it can be due to failing or intermittently failing sensor nodes or batteries due to limited available power.

Cardinality: one of the challenges in designing ISN is large number of participating elements, how ISN functionalities are divided between participating sensor nodes and nature of their interactions.

- A large number of participating nodes results quickly in questions of how their communication is arranged, what are possible performance bottle necks (packet losses, delays, limited throughputs). Another issues here is regarding management of sensor nodes - how they are managed, and how much of available communication capacity is taken over by management function(s).

- o Division of ISN functionalities - when designing a system, one normally starts with top-down approach, and goes from high level requirements and in number of steps makes them more concrete until they are mapped to physical (hardware) units. In this process, there is number of choices on how division will be made. Normally this includes trade-offs between what is available, what are requirements on robustness, what are constraints (must-have and must-have-nots), etc.
- o Nature of interactions between sensor nodes - This one results directly from previous one, and what kind of interactions within the system is designed, and has impact on choice of communication technology, including communication protocols.

3.4 Architecture modelling for ISN

In this section a new coherent set of architectural views[2] is presented that consists of view that plays prominent roles in describing ISN designs. After defining the views the modelling formalism for describing the design is introduced.
The table in Figure 17 list the views and aspects. Certain views are already introduced in the literature review section, others are new, some others are adjusted to the needs of ISN architecting. For each view, we intend to describe:
- the role;
- basic building blocks of the model;
- composition rules (“how to connect”, what do the connections represent);
- main attributes (incl. aspects);
- cardinality (what does it represent, how to interpret “vector connections”);
- Illustrating examples.

- **Enterprise view:** The enterprise view (E) is “inherited” from the ODP methodology and describes the interaction between the system to be designed and its environment. The design process considered here assumes that the system requirements are already defined with sufficient details. ODP enterprise view is used to express these requirements, i.e. the model in the enterprise view serves as specification for the coming steps. For more details about the enterprise view, see [RM-ODP]. A more formalized “extract” from the enterprise view could be called the context view (C). It concisely describes the interfaces between the system to be designed and its environment (incl. all players with assigned roles). Similarly to the enterprise view, the context view represents specification, but this view’s emphasis would be on the interfaces, which helps the construction of the functional view. We choose to not have an explicit context view, but let it exist as a sub view of the enterprise view.
- **Functional view:** The functional view (F) describes what the systems “does”. It the shows the functional elements (transformations) and their connections, i.e. a particular functional decomposition of the system. The functional view can be considered as a “master view” as it is related to the “primary purpose” of the system. All other views are “aligned” with the functional view.
- **Process view:** The process view (P) describes the parallelisms in the system and shows the event handling scheme. This view plays crucial role in designing temporal behaviour and communication architectures (see details later).

[2] The selection of the views is an outcome of a – to certain extent – subjective process, provided the set of views defined are consistent. The authors of this report (after tiring but fruitful sessions) reached a consensus about the definitions. This set of views is described in the following. By no mean these views should be considered as “the true set of design views”. Rather they should be considered as a coherent set, which was used by the authors in a number of cases and in the authors’ opinion they can be used conveniently to capture the main characteristics of the design.

- **Physical view:** The physical view (Y) describes the relevant properties of the hardware components composing the system and their connections. In case of wireless communication links a sufficiently accurate channel model should also be provided. The in physical view is the model of the physical reality behind the realization of the design.
- **Execution view:** The execution view (X) defines the context in which the designed system operates. Roughly speaking the execution view captures the use cases (extended with essential non-functional requirements) and serves as set of scenarios for evaluating system performance.
- **Development view:** The development view (D) describes the implementation, i.e. what concrete hardware and software elements (commercially available or other) are used for implementation, how they are interfaced, etc. The implementation process of the system refines this model further, which leads to purchasing (components and/or services) hardware design, programming, etc.

These models expressing the system from certain views are not independent: the complexity of the design process stems from the interactions. These dependencies should explicitly be established as a part of the design. The intentionally created dependencies are called mappings. There are three dependencies, which are especially important during the design.

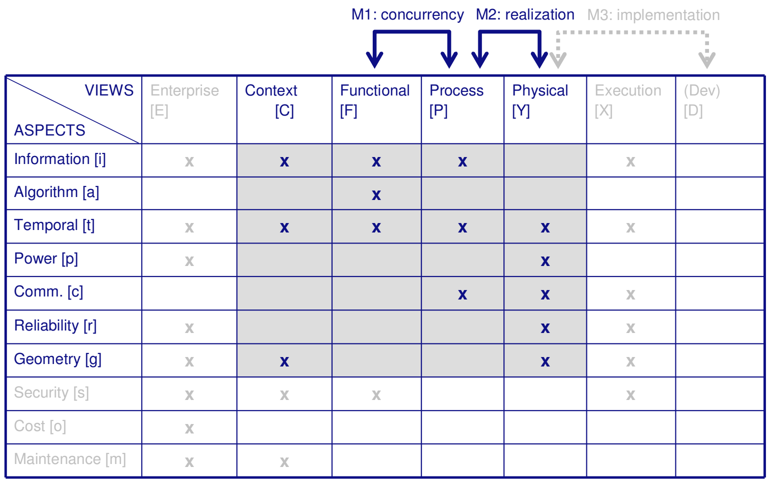
- **Concurrency:** This mapping (M1) associates the transformation blocks of the functional view with the processes of the process view. Indirectly it also defines the requirements for the inter-process communication.
- **Realization:** This mapping (M2) defines which physical system component is used to realize the components of the process view (e.g. which process runs on which processor, which communication link realizes a particular inter-process communication, etc.)
- **Implementation:** This mapping (M3) links the conceptual design to the implementation process.

In this chapter only those views (C, F, P, Y) and mappings (M1, M2) are covered, which strictly belong to the essential systems engineering process: transition from the (well defined) requirements to the implementation (i.e. the implementation design, implementation and to follow-up phases are excluded).

The models representing the design from a view can be characterized from several aspects. An aspect is an “angle” from which only given characteristics of the model is visible. These are the aspects found fundamental in the ISN design process:

- Information aspect (i): characterization of data sets and the data transformation
- Algorithms aspect (a): defines the properties used to implement the functional elements in the functional view
- Temporal aspect (t): defines temporal and execution time properties/requirements
- Power aspect (p): describes the power/energy requirement of the components used for realization of the system
- Communication aspect (c): describes the data transfer and temporal properties of the communication links; channel models are identified and parameterized
- Reliability aspect (r): characterization of the failure modes and failure models of the components of the physical view.

- Security, cost and maintenance aspects: these aspects are not addressed by the current version of the modelling framework and design methodology – though incorporating them into the framework is straightforward.

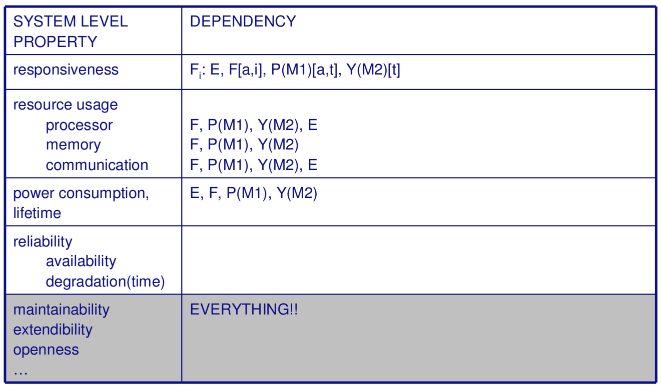


VIEWS / ASPECTS	Enterprise [E]	Context [C]	Functional [F]	Process [P]	Physical [Y]	Execution [X]	(Dev) [D]
Information [i]	x	x	x	x		x	
Algorithm [a]			x				
Temporal [t]	x	x	x	x	x	x	
Power [p]	x				x		
Comm. [c]				x	x	x	
Reliability [r]	x				x	x	
Geometry [g]	x	x			x	x	
Security [s]	x	x	x			x	
Cost [o]	x						
Maintenance [m]	x	x					

Figure 17 - Views and aspects

The models built formalize the features and properties of the design. The mappings make their dependencies explicit. System level properties (e.g. response time, availability, etc.) can be derived using multiple views and the dependencies among them. The figure lists a few essential properties and their dependencies on the models and mappings (the abbreviations used for denoting the views and the mappings are introduced at the definitions above). The ability of deriving system level properties is an essential and fundamental feature of the modelling methodology proposed. Typically these properties are used to quantify the quality of a design, define constraints for a design, etc. After adjusting the system design or making a design decision the models and mappings in the model of the design should also be adjusted and the relevant system level properties should be recalculated. The resulting values explicitly indicate the success (or failure) of the adjustment. In this respect it ca be said that the derived properties are the “interface” between the modelling of the system design and the design methodology, which guides the design process.

In the following the proposed modelling framework will be introduced. Along the chapter an example is used to illustrate the usage of the modelling formalism – so, before we start with the modelling, the example case is described.



SYSTEM LEVEL PROPERTY	DEPENDENCY
responsiveness	F_i: E, F[a,i], P(M1)[a,t], Y(M2)[t]
resource usage	
processor	F, P(M1), Y(M2), E
memory	F, P(M1), Y(M2)
communication	F, P(M1), Y(M2), E
power consumption, lifetime	E, F, P(M1), Y(M2)
reliability availability degradation(time)	
maintainability extendibility openness …	EVERYTHING!!

Figure 18 - Derived system properties

3.5 The Balloon Control (BC) case

A fictional experimental setup is shown in the next figure. Several nodes are placed on a table, each node knows its own position and has a radio, a processor (node could be zigbee), and has a fan that blows air as an actuator. The fan is pointed upward and each node can adjust the intensity of the fan.

There is a balloon that is light enough to be kept in the air by the fans and can be moved about the table by the fans if they work in cooperation. The balloon has no means of staying in the air by itself and if the fans stop blowing it sinks to the ground. The nodes should know where the balloon is at all times (through the use of a state estimation algorithm).
A flat surface is instrumented with fan (air blower) nodes (directed upward) and sensing nodes (to detect the balloon). The nodes are able to communicate with each other. The goal and the specification of the Balloon Control can be summarized as follows:

- The nodes must use their fans cooperatively to move the balloon to a set destination as smooth as possible. (The network can clearly do a better task as the number of nodes increases and will fail if the number of nodes drops below a certain level.)
- Each node must talk with the network of other nodes to find out its own position, where the balloon is, where the other nodes are and how it should use its fan to achieve the network goal of moving the balloon to the correct destination.
- Users can dynamically set the balloon destination

Though the case is very simple, there are a number of system design variants (with different properties), which can satisfy these requirements. In the following we do not intend to find the (in a sense) optimal system architecture. Rather we use this design case to show how the different alternatives can be represented in our modelling framework.

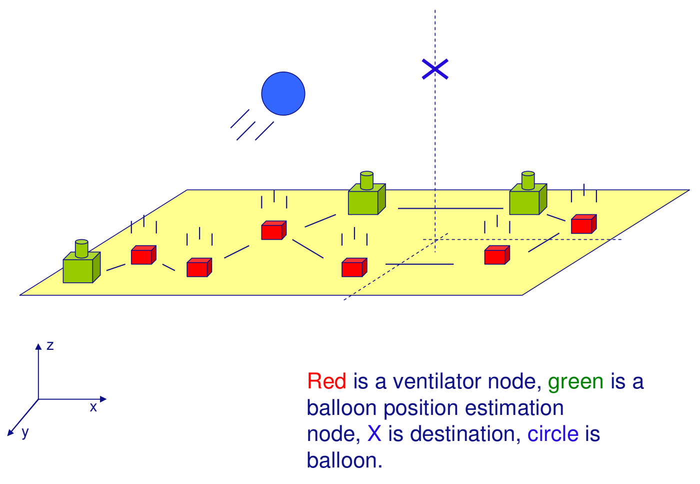

z
x
y
Red is a ventilator node, green is a balloon position estimation node, X is destination, circle is balloon.

Figure 19 - The Balloon Control case

3.5.1 *The enterprise view*

The enterprise view describes the interaction between the system to be designed and its environment. A more formalized context view can be derived from the enterprise view – in this respect it cannot be considered as an independent model view. The context still covers an important gap in the design process: it emphasizes the interactions (information exchange, influencing) between the enterprise objects ("players") and thus makes the interfaces explicit for the functional view (which logically is the next phase of the design process).

Figure 20 shows the building elements of the context view. A particular system can be characterized by multiple context views – depending on the roles the objects actually play. The rectangles represent enterprise objects. In the context view only those enterprise objects, which play role in the use case covered and have I/O relationship with the system to be designed. Typically the system to be designed is represented by one block – i.e. no details are given about the internal composition. The objects are annotated by the role(s) the play in the use case. Figure 21 shows an example context view of a balloon use case. In the use case the user disturbs the balloon's motion by introducing an extra air flow directed to the balloon (flow 2), e.g. by using a handheld fan. The "system" implements the control actions to compensate for this disturbing flow and thus maintaining the required position of the balloon.

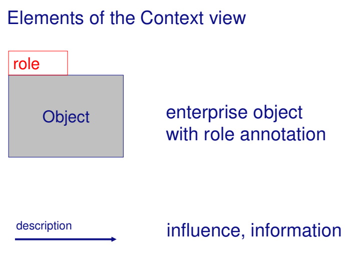


Figure 20 - Context view building primitives

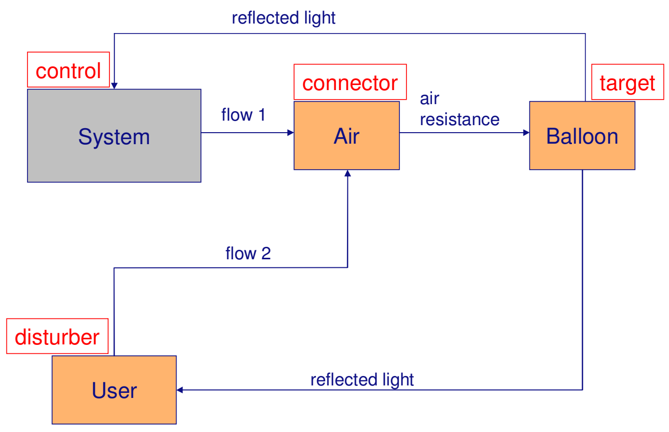


Figure 21 - Context view of a balloon use case

3.5.2 *The functional view*

The functional view is the principle view of the design: it describes a decomposition of the information processing tasks of the system to be designed. The functional view defines what the system does and how the processing scheme is composed of functional elements[3]. The primitives of the functional view are shown in Figure 22.

---

[3] As in any decomposition schemes the "granularity" of the functional view is an important question (i.e. what are the "primitive" building blocks, which should not be decomposed further). This question cannot be answered independently of other views and may strongly depend on the case in hand. General guideline for decomposition is that the functional decomposition should go deep enough

1. to have the allocation units identified for the process view (see details later);
2. to link the elements to the select algorithms, which determine the runtime characteristics of the functional elements.

In the examples and in the methodology section further guidelines will be provided.

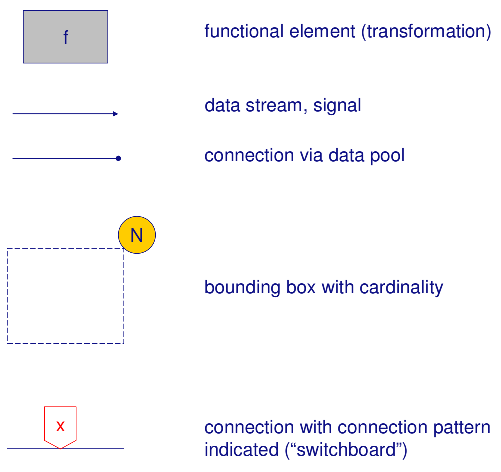

f
functional element (transformation)
data stream, signal
connection via data pool
N
bounding box with cardinality
x
connection with connection pattern indicated (“switchboard”)

Figure 22 - Elements of the functional view

The functional view basically consists of connected functional elements (f), each representing a particular transformation on its inputs and producing outputs. Functional elements can be connected via data streams (like the connections between block in a Simulink model) or data pools. Data pools represent intermediate data storage (typically in memory) with asynchronous access. (This is similar to Simulink blocks communicating via Matlab workspace variables.)

To illustrate the use of these elements the Figure 23 shows the functional view of the System enterprise object of the balloon context view (Figure 21). For simplicity it is assumed that the position sensor (e.g. camera) nodes and the fan nodes are integrated (i.e. they have the same location). The scheme is obvious: the state of the balloon is estimated and passed over to the controller. The controller drives the actuation, which produces airflow. In order to calculate the control command the location of the actuators should be known. This information is derived by the localization function. The setpoint for the controller is given via user interface functionality. The dotted arrows show the links to the enterprise objects of the context view (participating in this particular use case). As the figure shows there is a “signal flow” from the balloon to the air (including sensors and actuators on the interfaces). The setpoint and the location information come via data pools (memory locations) updated and retrieved asynchronously.

This functional view should be further detailed to express the distributed nature of the design (so far the functional view describes a centralized implementation).

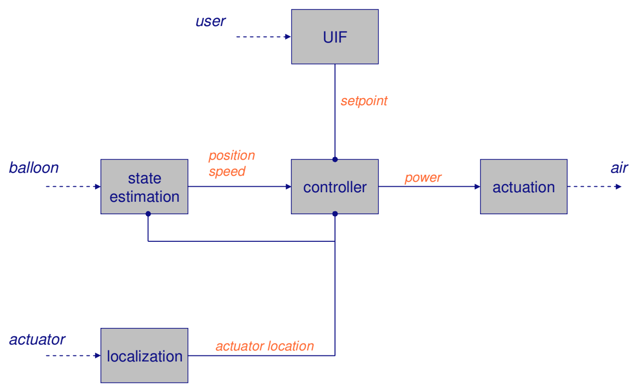


Figure 23 - Example - the functional view of the system object in the balloon case

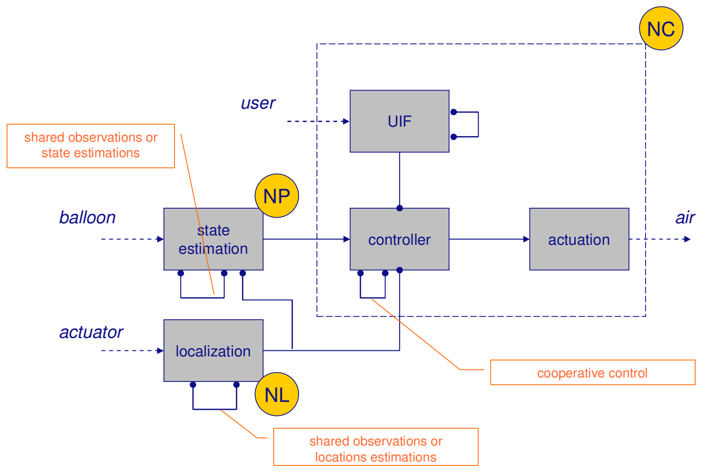


Figure 24 - Example - functional view with cardinality indicated

Figure 24 extends the functional view by indicating the number of instances of the functional blocks (cardinality, denoted by attached circles). The cardinality specification is applied to every functional block inside the bounding box. The figure indicates that NP instances of state estimation, NL instances of localization and NC instances of elements in the bounding box are used to build the systems. In the simplified case, where we assume that all nodes in the balloon setup are of the same type (i.e. nodes contain the sensing, control, actuation and user interface functionalities), NP = NL = NC. Attention should be drawn to a typical detail: in distributed computing (especially in intelligent (wireless) sensor network cases) the instances of the same functional element are rarely independent. Instead they work cooperatively to accomplish the required functionality. In the functional view this feature is reflected in the use of "self-referring" connections. For example the connection loop on the state estimation block expresses that the state estimator blocks exchange information in order to improve the quality of the estimation. In this particular case the information exchange is carried out via intermediate storage (memory pool).

In order to characterize the connections between multi-instance function blocks the connection topology has to be defined. (In reality the topology arises from the features of the algorithms selected to implement the function blocks. For example particular distributed state estimation algorithms may require full connectivity between the individual instances, some others require communication only with the neighbours.) The connections symbol can be augmented by a topology specification as shown at the bottom of Figure 22. The topology is identified by the text field in the triangle ("x" in the figure). The table below lists the commonly used connection schemes[4].

Connection type	Notation	Notes
fully connected	F	
ring	O	
random	R:d	d: the "density" of the connections
constrained	C:nH | C:dD	nH: max. n hops | nD: max distance d
one-to-all	1A	
one-to-one	11	this is the default scheme, can be omitted
all-to-one	A1	

The connection scheme in itself characterizes only the interaction needs among the functional elements. Using this together with the models of the process and the physical views jointly the requirements for the communication platform of the implementation can be determined (see details later).

3.5.3 *The process view and the concurrency mapping (M1)*

The process view describes the parallelisms, the event handling and the resource access scheme of the design. Figure 25 shows the modelling primitives of the process view. These primitives correspond to the concepts used in modelling concurrent systems, thus here only a brief description of the primitives is given.

A *process* is a sequential program (algorithm), which is executed in its own context and can run parallel with other processes. The processes are autonomous and isolated from other processes. The execution state of the process can be influenced by events (i.e. a process can wait for a particular event to happen). The processes can exchange information via messaging or shared resources (e.g. shared memory areas).
An *event* represents significant changes in the state of the environment (external event) or inside the systems (internal event). Passing a particular time instant, an object getting close to a sensor, pressing a pushbutton can be modelled as events. The event is a binary "signal" (like a flag, which can be raised), which becomes cleared when its processing started. Typically events are handled by associated event handling processes.

[4] These are merely the typical connection schemes. Particular functional decompositions may need more dedicated connection topologies.

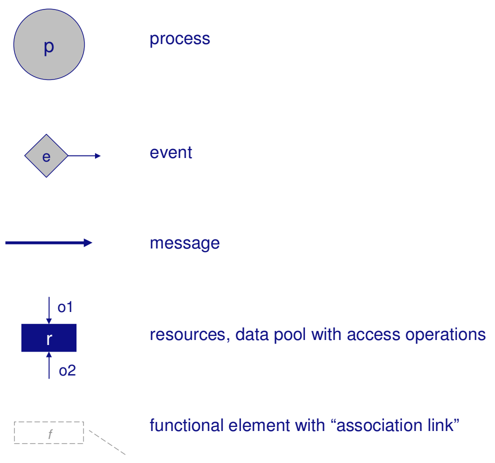

p
process
e
event
message
o1
r
o2
resources, data pool with access operations
f
functional element with “association link”

Figure 25 - The elements of the process view

A *message* is a collection of data, what a process can send over to another process. Messages are assumed to contain serializable data and cannot contain references to the sender’s local memory. In reality messages travel via inter-process communication channels or networks.

The *resources* are passive system components and are used by processes. An ADC, a memory buffer, etc. are typical resources. Resources can be shared by multiple processes. Depending on the type and usage of the resource shared access can be constrained (e.g. mutual exclusion).

The process view
- shows the parallel execution units (processes),
- shows the inter-process communication needs (messages, shared resources),
- identifies events to be handled and assigns “event handlers” (processes) to events: events are either external (for event driven operations) or internal (for process synchronization);
- shows the resources required (especially the shared resources are important to identify);
- assigns the functional blocks to processes (M1 mapping).

Illustrating the use of the process view Figure 26 shows the process view of a variant of the balloon design. Three types of external events are handled: the “keystroke” events are generated by the user (pressing the buttons on the user interface); the two “clock” events trigger the data processing chains (the clock frequencies can be different!). The ppos process implements the state estimation, the results are sent to the control process (pcnt) via a messaging link. The pcnt process picks up the actual setpoint and the its location (which is needed for calculating the control action) from shared memory locations (filled in by the pUI and ploc processes). The dashed grey boxes identify the functional blocks assigned to the particular processes (M1 mapping).

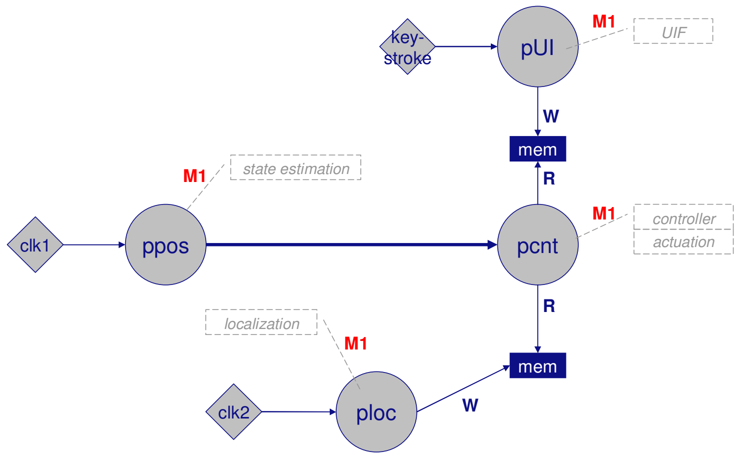


Figure 26 - Balloon case process view - alternative 1

Figure 27 shows an alternative design. This design suggests that the state estimator or the control processes can be merged into one process. (Both alternatives have their merits and drawbacks. Based purely on the process view it cannot be decided, which is the better. It depends on the selected algorithms to realize the state estimation and control, the hardware platform (incl. communication) used for implementation, other non-functional requirements, etc. In the methodology section more details are provided about the use of the models in the design process.)

In typical ISN applications many nodes run in parallel and cooperatively provide the solution. Consequently, as in the case of functional view, many instances of the process in the process view may exist. In the process view it can be reflected by augment the model with cardinality markings.
Figure 28 shows the first alternative of the process models with cardinalities indicated. The following observations should be made:

- As a consequence of the distributed implementation of certain functionalities (see the cardinalities and data flows of the functional view of Figure 24) the process view got a few extra interprocess communication links (marked by the red arrows).
- The type of the interprocess communication is determined by the communication needs of the functional blocks associated with the process. In the figure ploc processes use shared memory based data exchange (constrained by one hop), while others use messaging (assuming this is the requirement of the implementations of the associated functions). Consequently the inter-dependency between the functional and process models is not limited to the M1 mapping (i.e. to the association of functions to processes) but a particular mapping may require introduction of interprocess communication links.
- The cardinality of a process and the assigned functional block should not necessarily be the same: under certain circumstances assigning multiple instances of the same function to the same process may have advantageous properties.

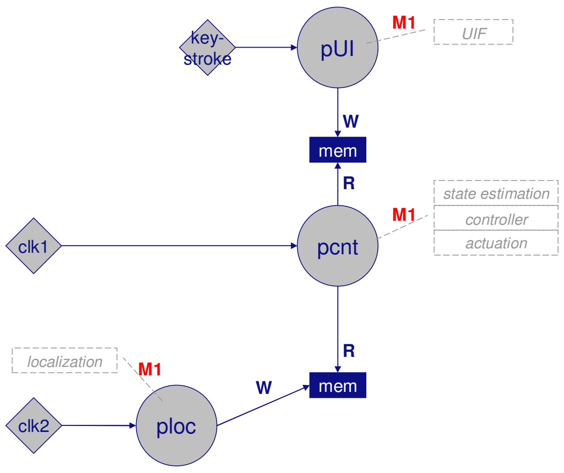


Figure 27 - Balloon process view - alternative 2

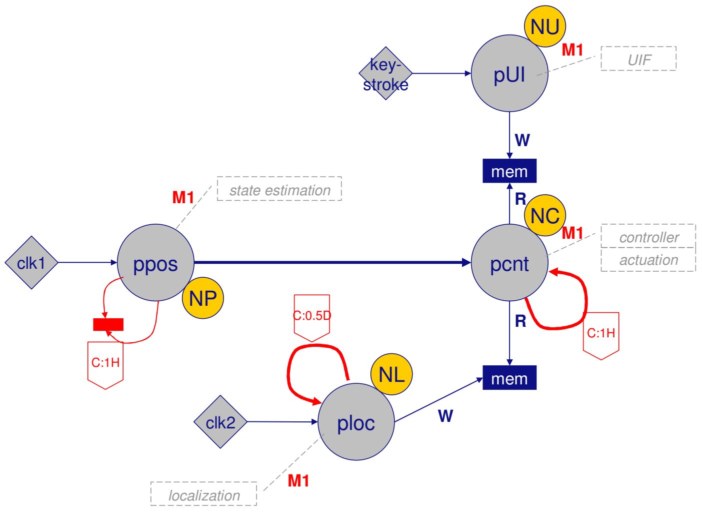


Figure 28 - Balloon process view with cardinalities indicated

3.5.4 *The physical view*

The functional view defined the data (signal) processing scheme of the design, the process view together with the M1 mapping described how concurrence and event handling requirements are addressed by the design. These two views also defined the requirements for the interprocess communication. The remaining questions are around the implementation: How are these designs mapped to hardware? How is the system embedded into its environment?

The physical view and the mapping between the process and physical view attempts to model the designer's answers to these questions. The physical view is an abstract representation of the implementation and the environment – focusing on the most essential elements and interaction forms. Our target is the design of intelligent sensor networks consequently their fundamental features have strong impact on the scope of the modelling. Addressing only the fundamental features makes the modelling easier while the model is still capable of manifesting the most relevant characteristics of the designed system[5]. It is assumed that the designed implementation has the following features:

- The system consists of possibly big number of nodes (computing elements);
- The nodes are loosely coupled (typically via some kind of communication network);
- Beside the nominal computational and communication components the nodes may have additional resources (e.g. sensors, actuators);
- The resources are local, i.e. each resource has an "owner" with exclusive access rights;
- The nodes are spatially distributed and the spatial distribution may have impact on communication topology.

To introduce the modelling concepts used in the physical view, first an example is presented. A design alternative for the balloon case will be modelled, which uses two types of nodes (Figure 29). The position estimation nodes (P) are responsible to deliver the motion state of the balloon; the control nodes (C) calculate the control action and implement the actuation. The circles show the communication radius of the wireless link of the nodes[6]. From this physical layout the connection graph can be derived, which is one of the most essential characteristics of the implementation (see
Figure 30 for the nodes with communication radius). For example the connection graph makes it explicit that for the observations from P3 it takes at least three hops to reach C4 (which strongly influences the response time, reliability and power consumption). If the connection graph is known then allocation of the interacting processes (as defined by the process view) on the nodes can be optimized in order to satisfy the system level requirements (e.g. response time, operational time (on batteries), graceful degradation, etc.). Many times the connection graph is not predefined (at least not fully) but the designer has certain freedom to set it up (e.g. placement of wireless nodes, number of nodes, radio/antenna design, etc.). Consequently the interactions between the models in process view and physical views can be extremely complex and deriving the optimal M2 mapping is the main challenges of ISN design.

[5] The capability of the modelling framework goes beyond ISN. Various types of loosely coupled, distributed embedded systems fall into the category addressed here.
[6] In the example a very simple communication model is used: A node can reach those other nodes, which fall into its communication circle.

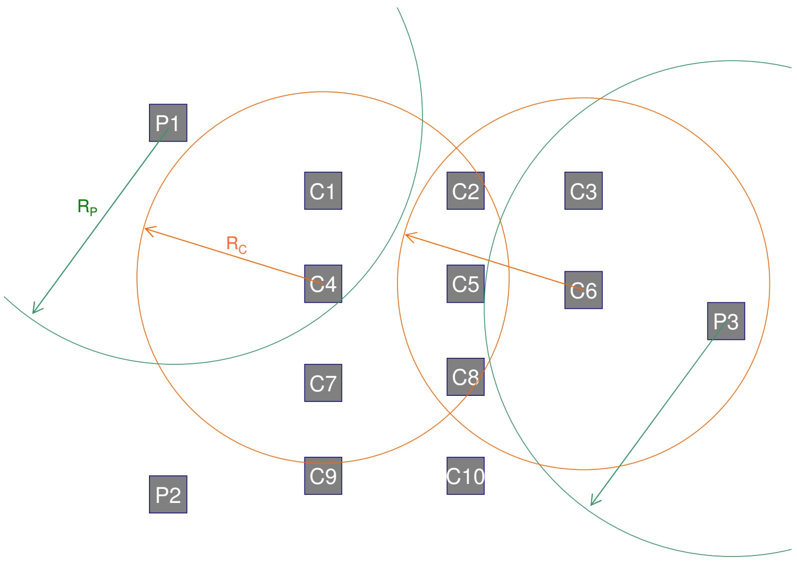


Figure 29 - A spatial layout of the balloon case

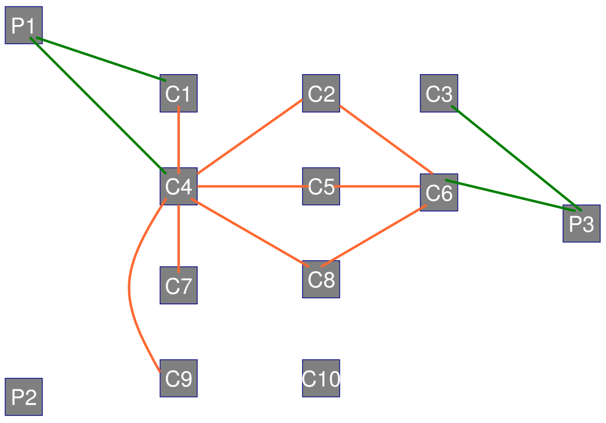


Figure 30 - Partial connection graph of the layout (in previous figure)

The physical view is composed of

- nodes, and
- connections.

The nodes are the active components of the system, capable of "hosting" processes and resources. In general a node consists of the following subsystems:

- programmable processor,
- memory (ROM, RAM, non-volatile RAM),
- other local resources: sensors, actuators,
- communication gear (wired and/or wireless),
- power source.

As mentioned earlier the resources are considered local, i.e. if a node needs a resource of another node (non-local resource), it cannot access it directly, instead the access is to be provided via a service implemented by the owner node.

The meta-model, describing the physical view models, is shown in
Figure 31 (UML class diagram form).

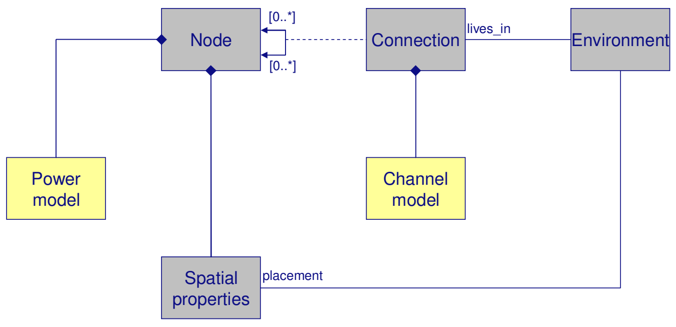


Figure 31 - The meta-model for the physical view

The fundamental building blocks are the nodes ("Node") (as described above). The nodes can exchange information via communication links. The link is characterized by the "Connection". Note that many-to-many connections or no connection at all are part of the model. The actual connection topology is determined by the channel model, the environment which embeds the nodes and the spatial placement of the nodes. Determining the connection topology is one of the most challenging tasks for wireless systems embedded in complex, many times dynamic environment. Consequently the channels models are stochastic models and the topology has to a given extent probabilistic nature. (To isolate the operation of the system from these probabilistic effects it is typical to build redundant topology and implement runtime reconfigurability – which generates runtime overhead, thus should be modelled both in the functional and the process views.)

The "conceptual system design" (as described by the functional and process views) is made real by associating the elements of the process model with the nodes of the physical model (Figure 32). The associations represent the node is responsible for executing the processes and for hosting the required resources. In a proper design after the associations have been made, all interprocess communication links are established and consequently the connections in the functional model are established with the proper characteristics.

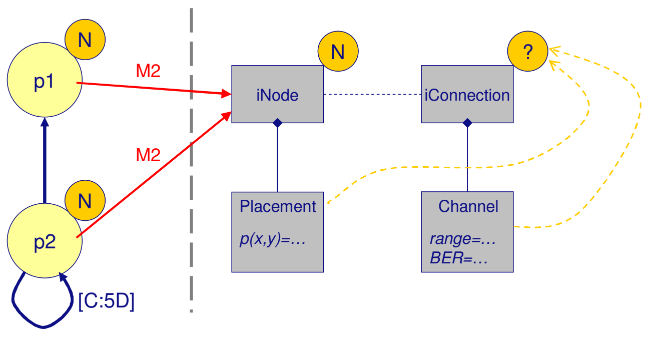


Figure 32 - The Realization (M2) mapping

In the figure an example is shown. In the design each node hosts an instance of p1 and p2 processes. The p2 processes needs p2-to-p2 communication, which is in the current mapping, requires inter-node communication (p2 needs information from other p2 processes in running inside a 5 m circle). The node placement and the channel model determine if this condition is satisfied. This also answers the question if multi-hop communication should be implemented for the particular setup. If the first design iteration is not successful, there are number of ways to adjust the design (e.g. different placement, different antenna design, introducing extra nodes, etc.)

Obviously, in non-trivial cases the design is not a straightforward process, this is why a design methodology is needed: the methodology helps walking through the designs steps and iterations in order to reach a correct solution.

3.5.5 *The execution view*

The execution view places the design into the execution context. Consequently the execution view contains the models that formalize the use cases used for evaluating the design (roughly speaking each model covers one use case, or a parameterized model can cover multiple, but conceptually close use cases).

It is not attempted here to propose even a semi-formal modelling framework for the execution view because the models depend on the actual problem in hand and the tooling of the development to a great extent. Generally speaking the models of the execution view should contain enough details for running the evaluation process of the modelled design. Typically the execution view defines[7]:

- the driving event sequence pattern (incl. temporal features),
- the characteristics of typical input data streams,
- operational modes,
- failure modes, and
- in mobile cases the motion trajectory of the nodes.

If the development process is supported by a simulation environment, which “executes” the models of the relevant views and derive the system level characteristics of the

[7] This is merely of a rough, generic list. Depending on the concrete needs of the case in hand other aspects of the execution context should also be modelled.

design, the models of the execution view is coded in the input language of the simulator.

4 Design methodology

4.1 Introduction

4.1.1 Why a new and uniform methodology is needed?

Despite all enabling technological developments, there is surprisingly little consensus on how to integrate these technologies into systems that are able to meet future challenges e.g. in the field of health care, military defence, crisis management, traffic management, process management and public safety. Several reasons can be given for this situation:

- The field of enabling technologies is very broad which makes it difficult to have a good and unbiased view on their capabilities. The field of artificial intelligence already covers a lot of different technologies where the state of the art is developed by specialists. Their focus is usually on exploring and advancing their technology and less on exploiting it in combination with complementary technologies. Integration of Artificial Intelligence will enable Sensor based systems to become more adaptive. This means design choices can be made run-time instead of design-time. Recent System Engineering methodologies are designed to structure all choices that have to be made design-time.
- System architects usually have a good overview of enabling technologies. Their mind is usually focused on what technology to use to generate a certain performance but have difficulty with integrating artificial intelligence into their methodologies.
- Integration of different areas of expertise requests a crystal clear semantics.
- Man and machine will co-operate instead of operate. Future systems will utilize the individual strengths of man *and* machine. For instance, machine-machine interactions can reach a far higher interoperability (brain swapping) in comparison with man-machine and man-man interactions. Researchers, system architects, operators and users usually have different (philosophical) views on human and artificial intelligence and the interaction between them which makes it difficult to agree on high level system design issues.
- Borders between systems and subsystems, cooperating systems (i.e. GPS) and the “system to be designed” are not that strict anymore. Borders between real environments and simulation environments will disappear. Borders between real (software) components and modelled (software) components are becoming vague (i.e. real algorithm of a tracker vs. a model of that algorithm).

4.1.2 Requirements of an uniform methodology

A design methodology is needed that can

- support intelligent system design in complex and intelligent environments;
- handle different views/backgrounds on system design (software, hardware, infrastructure designers must work with same methodology)
- support model based approach
- support autonomy and runtime optimization and
- handle adaptivity

This methodology should unite the various views on humans and intelligent machines, taking a system designer approach with a focus on how to generate system functionality, trying to have an unbiased view on artificial intelligence and dealing with the difficulties that arise in the systems engineering methodology from the incorporation of artificial intelligence techniques.

4.1.3 Definition of 'The System'

There are many definitions of what a system is in the field of systems engineering. Below are a few authoritative definitions:

- ANSI/EIA-632-1999: "An aggregation of end products and enabling products to achieve a given purpose."
- IEEE Std 1220-1998: "A set or arrangement of elements and processes that are related and whose behaviour satisfies customer/operational needs and provides for life cycle sustainment of the products."
- ISO/IEC 15288:2008: "A combination of interacting elements organized to achieve one or more stated purposes."
- NASA Systems Engineering Handbook: "(1) The combination of elements that function together to produce the capability to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (2) The end product (which performs operational functions) and enabling products (which provide life-cycle support services to the operational end products) that make up a system."
- INCOSE Systems Engineering Handbook: "homogeneous entity that exhibits predefined behaviour in the real world and is composed of heterogeneous parts that do not individually exhibit that behaviour and an integrated configuration of components and/or subsystems."
- INCOSE: "A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behaviour and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected."

In this course, the last one is preferred.

4.2 Systems Engineering

System development often requires contribution from diverse technical disciplines. Systems engineering helps meld all the technical contributors into a unified team effort, forming a structured development process that proceeds from concept to production to operation and, in some cases, to termination and disposal.

Taking an interdisciplinary approach to engineering systems is inherently complex since the behaviour of and interaction among system components is not always immediately well defined or understood. Defining and characterizing such systems and subsystems and the interactions among them is one of the goals of systems engineering. In doing so, the gap that exists between informal requirements from users, operators, marketing organizations, and technical specifications is successfully bridged.

When systems engineers develop systems they frequently rely on current models to guide their way. However, the most prevalent models, waterfall, spiral, and Vee have not been sufficiently explicit regarding the concurrent development of the system architecture and the entities of the same system. On the contrary the Dual Vee model as well as the SIMILAR model does consider the concurrent development of the system architecture and the system entities.

4.2.1 Waterfall Model

A software development method defined by Dr. Winston W. Royce in 1969 [ROYCE1969] to promote a sequentially phased software development process. The model promotes knowing the requirements before designing and designing before coding, etc. The objective was to provide a repeatable process to the then undisciplined (generally ad hoc) software development environment. While the model is accurate for what is depicted it is silent on user involvement, risk management, and most importantly, it is a single solution model that fails to address architecture development that is inherent in all multiple entity systems. Although designed for manufacturing and construction industries, the model can be applied to software, hardware and system development.

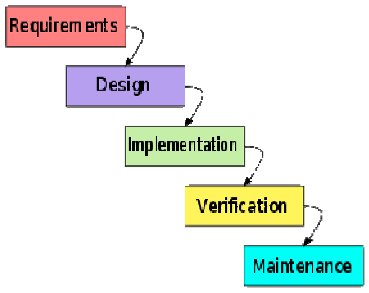


Figure 33 - Waterfall model

4.2.2 Spiral Model

A software development method defined by Dr. Barry Boehm in 1980 [BOEHM1986] to promote the management of recognized risks prior to attempting traditional phased software development. The model promotes resolving requirements, feasibility, and operational risks prior to proceeding with the traditional waterfall phases. The objective is to involve users and stakeholders in resolving recognized software development issues preceding concept development and design. This model is accurate for what is depicted but it also fails to address the problems of architecture development and management. Although designed for software the spiral model can also be applied to hardware and system development.

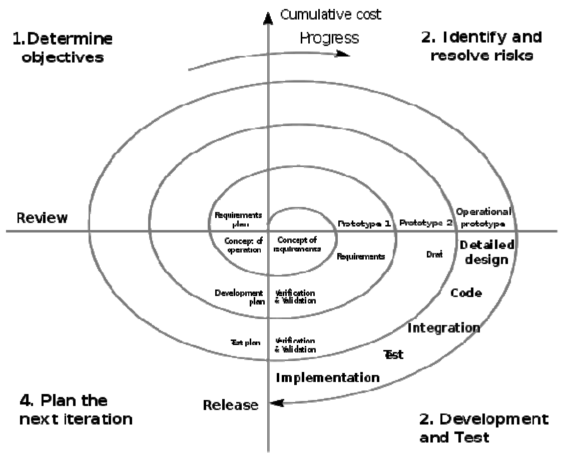
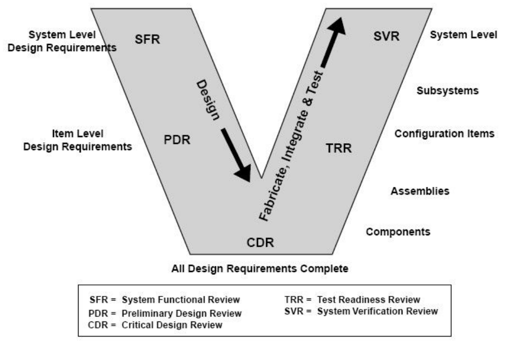


Figure 34 - Spiral model

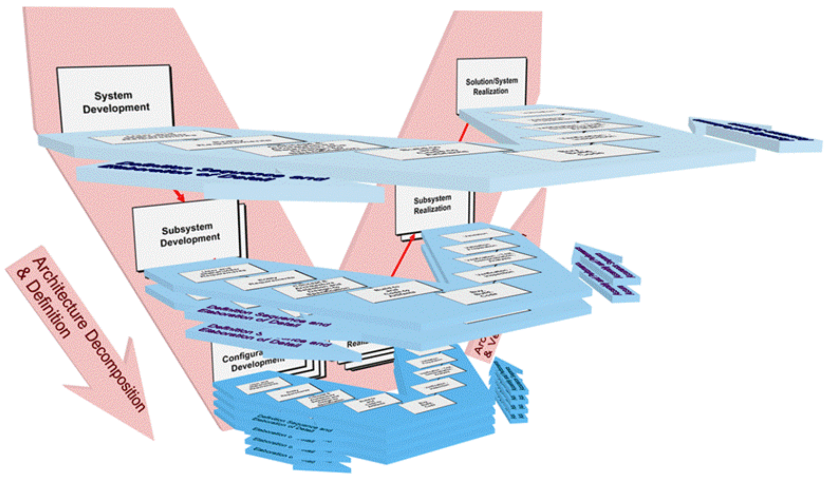
4.2.3 Vee Model

A system development method defined and elaborated by Dr. Kevin Forsberg and Hal Mooz from 1987 to 2005 [FORSBERG1998] to address system development issues of decomposition, definition, integration, and verification. The model also includes user/stakeholder participation, concurrent opportunity and risk management, and verification problem resolution.


System Level Design Requirements
SFR
Design
Item Level Design Requirements
PDR
CDR
Fabricate, Integrate & Test
SVR
System Level
Subsystems
TRR
Configuration Items
Assemblies
Components
All Design Requirements Complete
SFR = System Functional Review
PDR = Preliminary Design Review
CDR = Critical Design Review
TRR = Test Readiness Review
SVR = System Verification Review

4.2.4 Dual Vee

The Dual Vee Model [MOOZ2006] recognizes that there are two types of maturation in system development. The first view is the development and maturation of the system architecture which is composed of a complement of entities that are to be realized at all architecture levels. Architecture development and realization is represented by the Architecture Vee. The second development type is the creation of each entity within the architecture complement. Entity development and realization is represented by the Entity Vee.


System Development
Subsystem Development
Configuration Development
Solution/System Realization
Subsystem Realization
Architecture Decomposition & Definition

4.2.5 SIMILAR

The systems engineering process is usually comprised of the following seven tasks: State the problem, Investigate alternatives, Model the system, Integrate, Launch the system, Assess performance, and Re-evaluate [BAHILL1998]. These functions can be summarized with the acronym SIMILAR: State, Investigate, Model, Integrate, Launch, Assess and Re-evaluate. This Systems Engineering Process is shown in Figure 35. It is important to note that the Systems Engineering Process is not sequential. The functions are performed in a parallel and iterative manner.

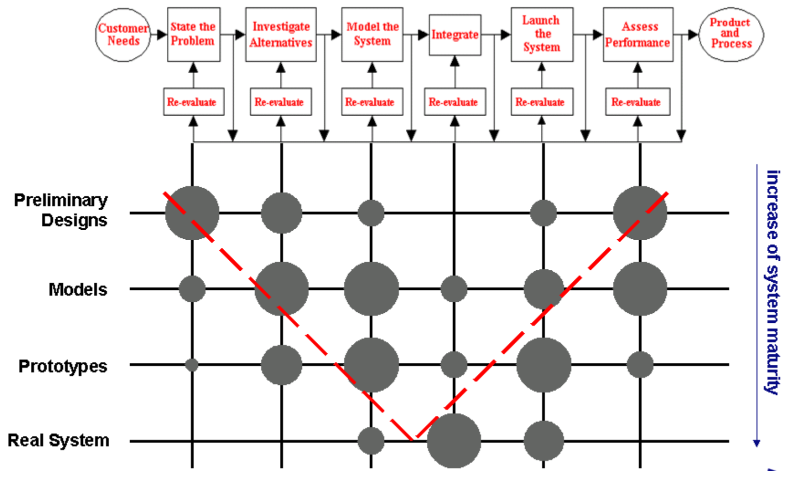


Figure 35 - The emphasis of the process activities within the different maturity stages will eventually lead to a V-shape for typical system engineering processes.

The methodology proposed is based on the system engineering process SIMILAR as recommended by INCOSE [INCOSE2010]. This process comprises the following tasks:

1. *State the problem; Stating the problem is the most important systems engineering task. It entails identifying customers, understanding customer needs, establishing the need for change, discovering requirements and defining system functions.*
2. *Investigate Alternatives; Alternatives are investigated and evaluated based on performance, cost and risk.*
3. *Model the System; Using models clarifies requirements, reveals bottlenecks and fragmented activities, reduces cost and exposes duplication of efforts.*
4. *Integrate; Integration means designing interfaces and bringing system elements together so they work as a whole. This requires extensive communication and coordination.*
5. *Launch the system; Launching the system means running the system and producing outputs – making the system do what it was intended to do.*
6. *Assess performance; Performance is assessed using evaluation criteria, technical performance measures and measures of effectiveness – measurement is the key. If you cannot measure it, you cannot improve it.*
7. *Re-evaluation; Re-evaluation should be a continual and iterative process with many parallel loops.*

An important aspect of this process is that for each task there is a re-evaluation sub-process. This can be used for further optimization as well as engineering the system and system architecture in more and more detail similar to the approach taken by the Dual Vee model.

4.2.6 Conclusion
Dual Vee and Similar both provide space for a kind of system maturity or readiness. In the DUAL Vee model however the emphasis is on system architecture/decomposition related to maturity/readiness in contrast to Similar where the system maturity/readiness is related to activities in the design process.

The Dual Vee model as well as the SIMILAR process can be used for ISN design processes. The SIMILAR process is more open ended where the processes in the Dual Vee model are more tied up. The most important aspect however is that in representations, the readiness or maturity of the system as well as its design or architecture is accounted for.

4.3 Five steps to Intelligent Sensor Networks

In the previous paragraph general models were presented on how to structure design and engineering processes through the entire system development. In this paragraph content is given to these structured processes, tapered on intelligent sensor networked systems. In the first paragraph of this chapter the need of an improved methodology was already given. To deal with complex ISN like systems and run-time optimizations, five critical steps are proposed in which decisions has to be made regarding the system to be designed. These steps can be mapped on to SIMILAR as well as on the Dual Vee model. In both models these four steps will recur at the succession of system maturity levels in SIMILAR and system decomposition levels in the Dual Vee process.

4.3.1 Problem Statement
The system under consideration will be embedded in an environment with which it interacts to reach its goals. As depicted in Figure 36 other systems may be present in the environment. These systems may have different intentions ranging from cooperative,

neutral non-cooperative to hostile. The cooperative systems, e.g. the internet or humans that respond to messages can according to the system definition of INCOSE be considered as part of the operational system but from a designer's point of view, however, these cooperative systems are part of the environment since it is not part of the system that needs to be designed.

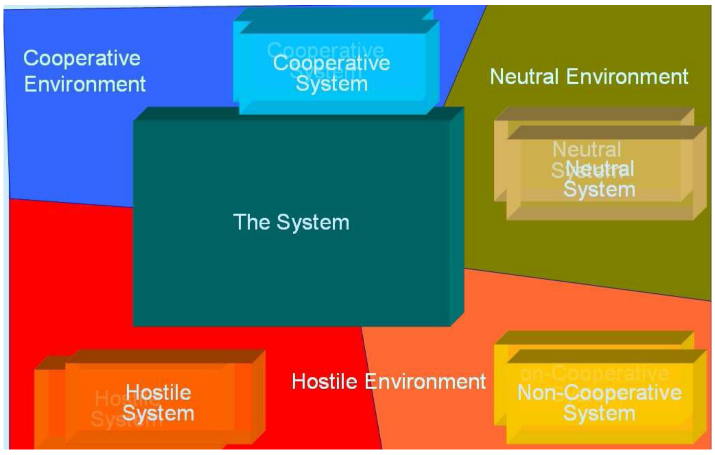


Figure 36 - The system in its environment

The mean characteristics of the system suitable for military, crisis management, traffic management and public safety is that it is Networked, Adaptive to changing and unexpected situations, Interactive within the system and with the environment and consisting of human and machine (Hybrid).

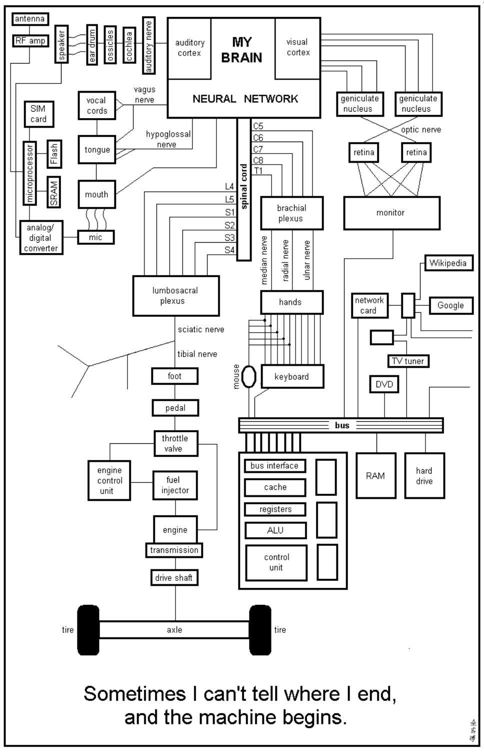


Figure 37 - Physical view of a hybrid system: me driving a car, while having a mobile phone and laptop next to me, illustrating at the same time the importance of choosing a workable scope for the boundaries of the system.

4.3.2 Functional Organization

This section is heavily inspired on the article *Designing Networked Adaptive Interactive Hybrid Systems*, by Leon Kester [KESTER2008].

When the problem is stated, we can reason about which functional components will be necessary to reach system goals, regarding the known constraints. These functional components can be structured along a three-way coordinate system (XYZ). The first axis (X) represent the physical structure (cardinality) of the functional components, the second axis (Y) represent the space-time hierarchy of the functional components and the third one (Z) represent the (JDL [JDL1991]) information abstraction levels. Below we elaborate on these three principles of the functional organization briefly. Extensive descriptions and foundations can be found in the NAIHS papers of Leon Kester.

The first principle on which the functional organization can be based on is the physical structure. In systems considered here the physical structure is due to the network of platforms and the various collectors and effectors, in other words the situated system. Each ‘perception-action’ component in the two dimensional cognitive hierarchy could therefore have multiple instantiations in the system resulting in a 3 dimensional structure of ‘perception action’ components. This distributed system model is symbolized by Figure 38. Note that each component in this figure may be hybrid and consisting of multiple levels of time abstraction

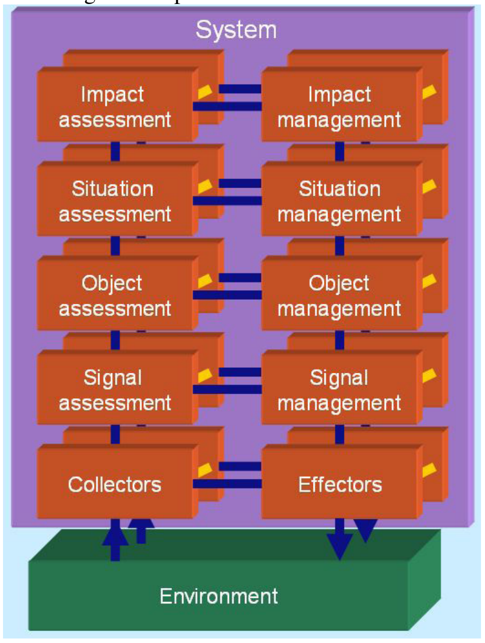


Figure 38 - Functional model of networked adaptive interactive hybrid systems

The second principle on which the functional organization can be based on is the time horizon of the effects the system would like to generate. A decomposition of the process cycle in this dimension has been adopted in various application domains. In the military field a strategic, operational and tactical level is in use. For (business) planning the same levels are in use, however, contrary to the military case the operational level acts on a shorter time scale than the tactical level. In the AI domain Brooks [BROOKS1991], who discards the decomposition in information abstraction, proposes a hierarchical composition of process cycles based on reaction or cycle time. The need for decomposition into temporal abstractions is also acknowledged in the case of decision making processes in complex situations [HAAMAS2006]. The time scale may differ widely for various applications. A suitable decomposition therefore depends on the application.

The third principle on which the functional organization can be based on is a well known model of information abstraction; the data, information, knowledge, wisdom or DIKW [ACKOFF1989], [ZELENY1987] model . Similar to this abstraction the JDL model [JDL1991] uses:

- Level 0: Estimation of States of Sub-Object Entities (e.g. signals, features, data)
- Level 1: Estimation of States of Discrete Physical Entities (e.g. vehicles, buildings)
- Level 2: Estimation of Relationships Among Entities (e.g. aggregates, cueing, intent, acting on)
- Level 3: Estimation of Impacts (e.g. consequences of threat activities on one’s own assets and goals)

We can match these models easily with the high level model of Figure 39 for the part of situation awareness

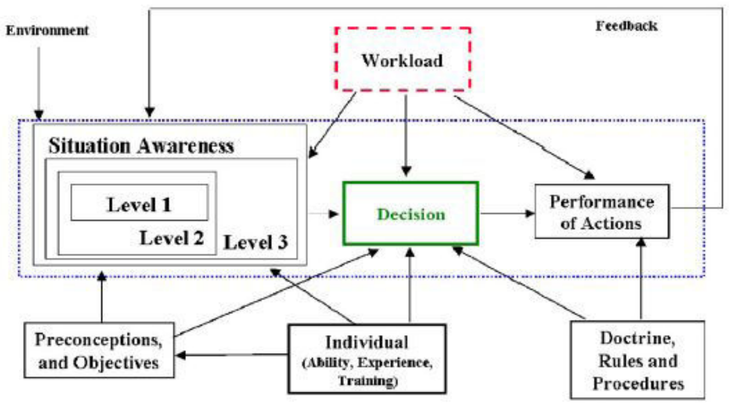

Environment
Feedback
Workload
Situation Awareness
Level 1
Level 2
Level 3
Decision
Performance of Actions
Preconceptions, and Objectives
Individual (Ability, Experience, Training)
Doctrine, Rules and Procedures

Figure 39 - Endsley's [ENDSLEY1988] situation awareness model

Regarding the similarities of the second and third principle (time-space and information abstraction); most models use only one type of abstraction principle, there are some that distinguish these different principles but integrate them in one abstraction hierarchy. Although it is a tempting thought to do so we would like to consider the two principles on which decomposition can be based independently, resulting in a two dimensional decomposition of the 'hybrid mind'. The reason for this is that it is well imaginable that information processing at a high information abstraction level is very fast and relevant for short term decision making or that information processing from one sensor at one platform needs to be decomposed in many components at the higher information levels. Furthermore, results presented in indicate that both abstraction principles play a role in hierarchical behaviour. In the functional model for NAIHS, therefore, the freedom is maintained of accommodating both abstractions.

4.3.3 Interoperability

Now that the dimensions of decomposition for the ISN are analysed, the next phase in the design process is to address how the functional components can be integrated in such a way that the system as a whole is able to reach its goals in the most effective way.

First we reconsider the model of Figure 38 (earlier depicted NAIHS ladder). In this model it is assumed that each functional component only interacts with its direct neighbours, either in the information abstraction hierarchy, in the temporal hierarchy, social interaction or between the assessment and management components. This modular view is disputed, e.g. in the field of neuroscience, particularly at the higher levels of information abstraction. Here we assume modularity keeping in mind that we might have to include additional interactions.

An important characterization related to distributed behaviour is interoperability that characterizes how well the networked components 'understand' each other. A common model for interoperability is the LISI model [LISI1998] that distinguishes five levels:

- L0: no interaction or unintelligible information
- L1: unstructured representation of information

- L2: a common representation of information (syntactic level)
- L3: a common understanding of information (semantic level)
- L4: common understanding of how information is interpreted (pragmatic level)

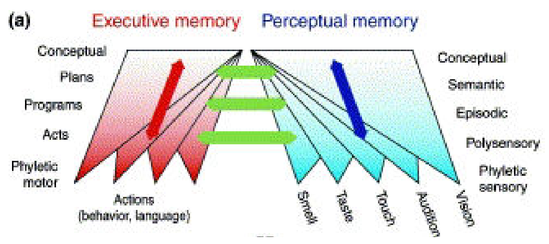
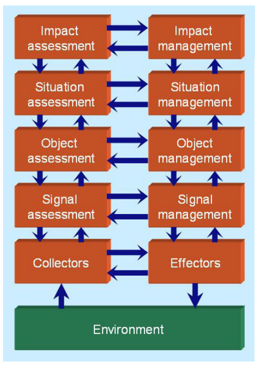


Figure 40 - Fuster's cognitive model

Obviously, the higher the level of interoperability, the better the perspectives
for coherent behaviour. Note that the denomination 'semantic level' here originates from the field of semiotics to characterize that a common understanding of the 'language' is assumed. In the cognitive model of Fuster [FUSTER2004] (Figure 40) 'semantic' and 'episodic' characterize the type of memory involved in different mental processes.

4.3.4 Interactions

Another important concept is that of service orientation in which the components that are in need of a service (information, communication etc) express their need to service providers who try to provide the most utile service.



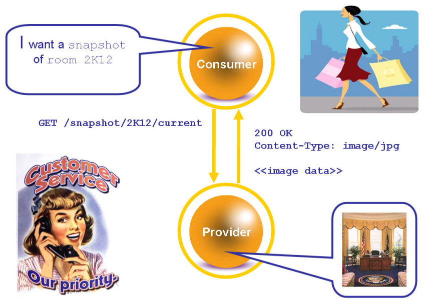


Figure 41 - (Left) Model of NAIHS with service oriented interaction between the components.

Figure 42 - (Right) Representation of service consumer and service provider interaction.

Since we have adapted the 'process cycle' of the original NAIHS model to the information abstraction hierarchy depicted in Figure 40, service oriented interactions have to be added between the assessment and management modules at the lower levels. This adds extra service requests from the management modules to the assessment modules but does influence the coordination mechanism of distributed modules. This is schematically shown in Figure 42.

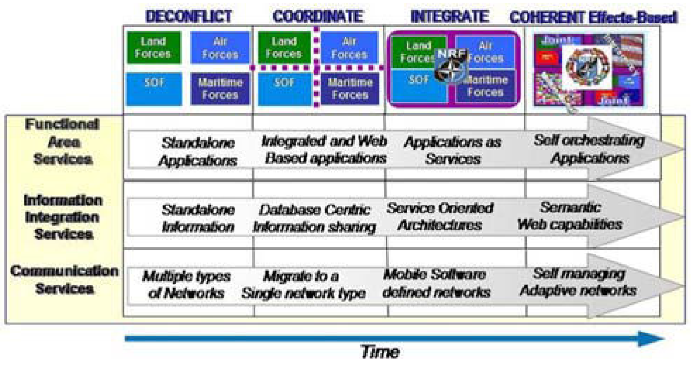

DECONFLICT
COORDINATE
INTEGRATE
COHERENT Effects-Based
Land Forces
Air Forces
SOF
Maritime Forces
NRF
Joint
Functional Area Services
Information Integration Services
Communication Services
Standalone Applications
Integrated and Web Based applications
Applications as Services
Self orchestrating Applications
Standalone Information
Database Centric Information sharing
Service Oriented Architectures
Semantic Web capabilities
Multiple types of Networks
Migrate to a Single network type
Mobile Software defined networks
Self managing Adaptive networks
Time

Figure 43 - NATO NEC maturity roadmap

In the perspective of distributed behaviour it is now interesting to look at a roadmap depicted by NATO on Networked Enabled Capabilities (NEC) in Figure 43. In this roadmap three system layers are distinguished: Communication, Information Integration and a Functional Area. Apart from the confusing use of terminology this corresponds well to the layers in high level system models, such as the OODA loop [BOYD1987]. Each layer delivers services in support of the desired effects.

Furthermore four phases of 'co-evolution' or NEC maturity of the three service layers are identified:

- Deconflict: the services are not conflicting but act separated.
- Coordinate: information and applications are available to the system.
- Integrate: capabilities adjust their services to changing situations.
- Coherent (also known as Agile): the whole system reorganises itself to changing situations.

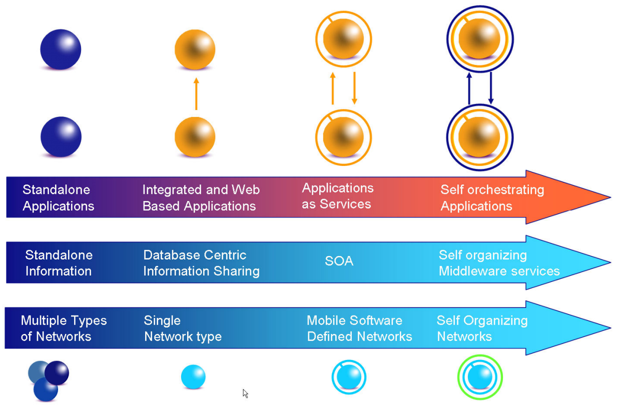

Standalone Applications
Integrated and Web Based Applications
Applications as Services
Self orchestrating Applications
Standalone Information
Database Centric Information Sharing
SOA
Self organizing Middleware services
Multiple Types of Networks
Single Network type
Mobile Software Defined Networks
Self Organizing Networks

Figure 44 - NEC maturity mapped on "core functionalities" (applications), "shells" (information integration services, or middleware) and communication services.

4.3.5 *Physical Deployment*

At last the functional components with their interoperability and necessary interactions, determined in the previous steps, also need some kind of "hardware" to operate. When we assume that in future information intensive systems the needed “hardware” also is provided as a service. We can draw the dependencies as depicted in the Figure 45.

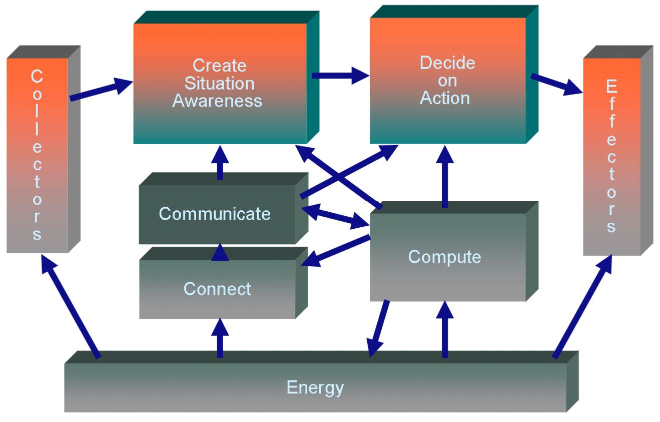


Figure 45 - High Level Functional Model

In the first steps the focus was on the organization and design of the “hybrid mind” consisting out of the “Create Situation Awareness” and “Decide on Action” parts with the collectors and effectors. The orange part in this figure represent the human functionalities, the green part represent software and the grey parts represent the hardware.

In the previous step where we reasoned about the necessary interactions, the network layer was already incorporated. In this high level functional model however this network layer is subdivided in to “Communicate” and “Connect”. Where services provided by the “Communicate” part corresponds (more or less) to the host layers of the OSI model where the “Connect” part corresponds to the Media Layers of the OSI model.

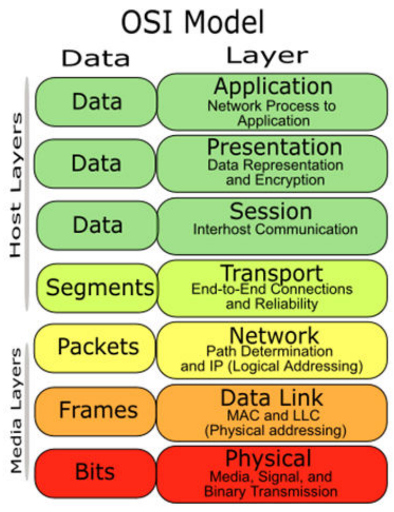

OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Physical addressing)
Bits
Physical
Media, Signal, and Binary Transmission

Figure 46 - OSI model

Besides communication services the “Hybrid Mind” with its collectors and effectors also needs computational power and more or less indirectly, energy. Again, depending on the constraints the system is exposed to, decisions can be made regarding these computational services and energy supplies. This is application bounded but you can imagine that with an ever increase of bandwidth and a decrease in communication costs concepts as cloud computing will gain in popularity rapidly.

4.4 System Maturity and Readiness

The United States Department of Defense distinguishes five stages in a system life cycle: Concept refinement, Technology development, System Development & Demonstration, Production & Deployment and Operations & Support. The first four stages are part of the system design process. This means design choices can be made that affect the performance of the final system.

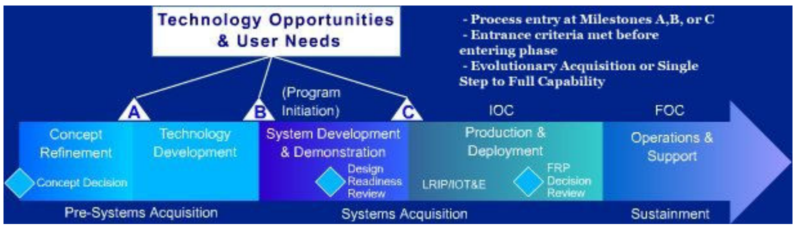


Figure 47 - DoD5000

Bahill and Gissing distinguishes also four stages or phases within the system design process in their SIMILAR paper. They start with ‘Preliminary designs’, “Models”, ‘Prototypes’ and end with the ‘Real System’. They also stated that every stage or phase is a design process by itself, starting with “state the problem” and ending with “launch the System” and “Assess Performance”. However, the last stage in the SIMILAR process as shown in figure X, is only used for Evaluation of the “Real System”. Bahill and Gissing assume that the system is ‘static’ so there is no space for any design choice during operations. However, systems can gain in efficiency or efficacy if they have the ability to adapt to changing circumstances. The next paragraph will dilate upon the consequences of run-time optimization.

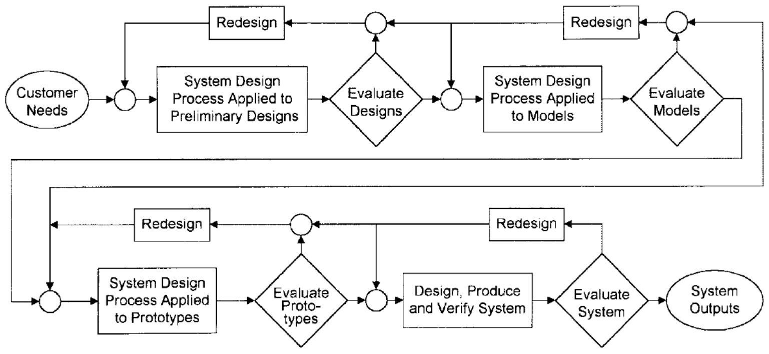

Customer Needs
Redesign
System Design Process Applied to Preliminary Designs
Evaluate Designs
Redesign
System Design Process Applied to Models
Evaluate Models
Redesign
System Design Process Applied to Prototypes
Evaluate Proto-types
Redesign
Design, Produce and Verify System
Evaluate System
System Outputs

Figure 48 - Design portion of the Systems Engineering Process

In Figure 48, the System Design Process is applied to preliminary designs, models, prototypes and to the real system. However, this process is not serial. Each of the loops will be executed many times. Execution of the redesign loop in the upper left is very inexpensive and should be exercised often. Execution of the redesign loop in the lower right is expensive, and should only be exercised when strictly necessary. Whereas, execution of the redesign loop from the lower right all the way back to the upper left is *very* expensive and should seldom be exercised. This just restates the principle of concurrent engineering and integrated product development teams: doing a thorough job early in the design process reduces the amount of change in the production phase, which improves cost and quality.

Figure 48, of course, only shows a part of the Systems Engineering Process. It omits purchasing, manufacturing, logistics, maintenance, risk management, reliability, project management, documentation, etc. It also omits the important feedback loop from system outputs back to the customer needs.

4.4.1 TRL, SRL, IRL

From being a seven level metric instituted by NASA in the 1980s, TRL is now a nine level metric and a concept that is used widely across NASA and DoD's to measure the maturity of a technology and also to compare maturities of different technologies.

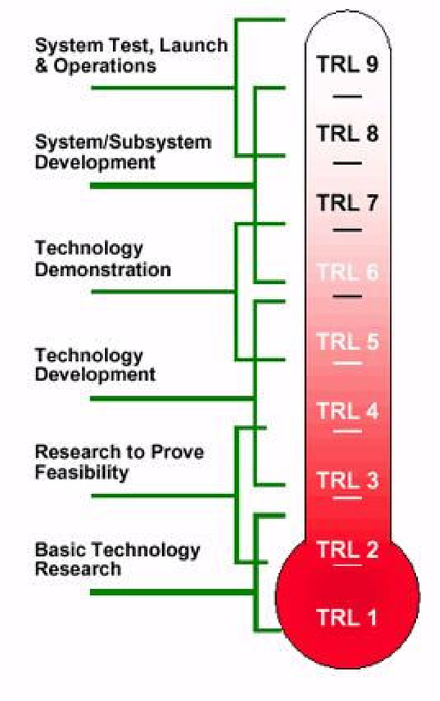
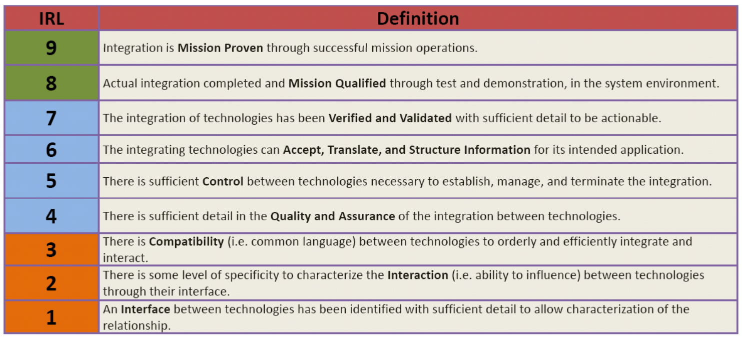

System Test, Launch & Operations
System/Subsystem Development
Technology Demonstration
Technology Development
Research to Prove Feasibility
Basic Technology Research
TRL 9
TRL 8
TRL 7
TRL 6
TRL 5
TRL 4
TRL 3
TRL 2
TRL 1

Figure 49 - Technology Readiness Level

As TRL has been used to assess the risk associated with developing technologies, IRL is designed to assess the risk of integration. Integration is the process of assembling the system from its components, which must be assembled from their specified requirements [Buede2000]. Integration can be a complex process containing multiple overlapping and iterative tasks meant to not only 'put together' the system but create a successful system built to user requirements that can function in the environment it was intended. To measure integration, an index is described (i.e. IRL) that can indicate how integration occurs.



IRL	Definition
9	Integration is **Mission Proven** through successful mission operations.
8	Actual integration completed and **Mission Qualified** through test and demonstration, in the system environment.
7	The integration of technologies has been **Verified and Validated** with sufficient detail to be actionable.
6	The integrating technologies can **Accept, Translate, and Structure Information** for its intended application.
5	There is sufficient **Control** between technologies necessary to establish, manage, and terminate the integration.
4	There is sufficient detail in the **Quality and Assurance** of the integration between technologies.
3	There is **Compatibility** (i.e. common language) between technologies to orderly and efficiently integrate and interact.
2	There is some level of specificity to characterize the **Interaction** (i.e. ability to influence) between technologies through their interface.
1	An **Interface** between technologies has been identified with sufficient detail to allow characterization of the relationship.

Figure 50 - IRL is designed to assess the risk of integration.

Although component TRLs are necessary for evaluating system risk and potential, they are generally insufficient. To address this insufficiency, a new composite metric, called a SRL has been defined as a quantifier to assess maturity for a potential system [SAUSER2006]. An illustration of the rationale under which SRL has been constructed is presented in Figure 51.

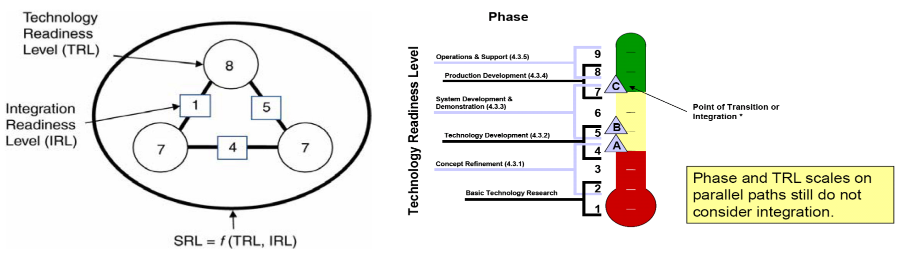

Technology Readiness Level (TRL)
Integration Readiness Level (IRL)
8
1
5
7
4
7
SRL = f (TRL, IRL)
Phase
Technology Readiness Level
Operations & Support (4.3.5)
Production Development (4.3.4)
System Development & Demonstration (4.3.3)
Technology Development (4.3.2)
Concept Refinement (4.3.1)
Basic Technology Research
9
8
7
6
5
4
3
2
1
C
B
A
Point of Transition or Integration *
Phase and TRL scales on parallel paths still do not consider integration.

Figure 51 - An illustration of the rationale under which SRL has been constructed

In a system design process, a common understanding on the system maturity of the system to be designed is indispensable. As mentioned in the previous paragraph system engineering goals can differ in the various stages of system maturity. In early stages for instance the goal of the system is to assess the validity and performance of the system to be designed at the proper level of abstraction. This differ from the goal of the end stages of the process; production and integration of the real system. Especially when complex systems are designed and implemented in mixed reality environments it is very important to keep track on decisions made. Design choices could be made in a technology development phase to *only* proof the concept of a component. It is important not to implement these choices in to the real system. Therefore in complex system design processes, an unstructured and non-standardized view on system maturity could lead to confusion.

4.4.2 Architectural Maturity and Readiness

The mean characteristics of the system suitable for health care, military defence, crisis management, traffic management, process management and public safety is that it is

Networking, Adaptive to changing and unexpected situations, Interactive within the system and with the environment and consisting of humans and machines (Hybrid). In the remaining part of this chapter we will refer to this type of system as NAIHS.

For these complex intelligent sensor networked systems it is improbable to first work out the entire system design and architecture to subsequently build the system according to plan. Like the SIMILAR and Dual Vee models already showed, it is more likely to design the system on a higher level first, then start with the building and assessing the performance

The systems that are considered here are usually very complex. Just building them and assess their performance is therefore too costly or infeasible. Therefore a stepwise approach, also known as model based design, is used where only part of the system is implemented and part of the system is modelled. The situation that is created for experimenting with the system is therefore partly real and partly virtual, also known as mixed reality. If we map this idea on the abstract representation of the system in its environment as depicted in Figure 52 we can see that the setup of the mixed reality experiment is in itself a systems engineering problem but now the goal of this system is to assess the validity and performance of the system under design at the proper level of abstraction of the design process. So, following the system engineering process we need to state the problem; that is what is the purpose of the experiment, which part of our system in its environment do we want to implement and which part do we want to simulate. In addition we have to consider to which level of detail we would like to simulate things to get valid results from our mixed reality experimentation. Note here that since we may deal with hybrid systems, choices have to be made on putting either humans in the loop as part of an implementation or that we simulate (relevant part of) the human functionalities.

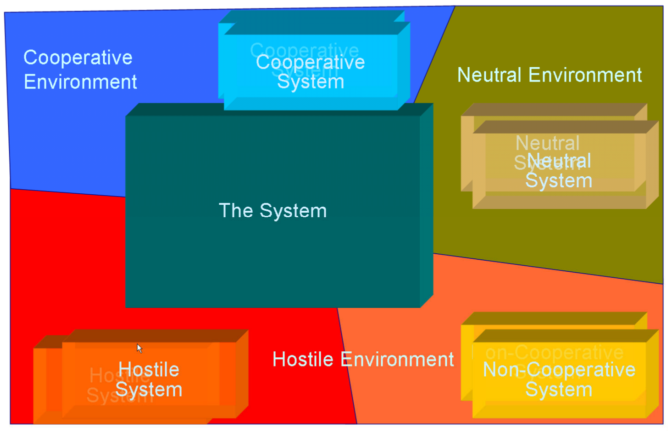


Figure 52 - Every part of the situation can be virtual or real

4.5 ISN Stages of System Maturity

This ISN Course proposes four stages in design processes of complex intelligent systems. These stages have a lot in common with the stages of the DoD and Bahill and

Gissing. However, to anticipate on confusing semantics within the different disciplines of TNO, this course proposes the next four stages:

Paper System: In the paper system stage , the emphasis should be on discovering requirements and defining functions, with some effort devoted to alternatives and interfaces. As described earlier this is the stage where to improve costs and quality of the system to be designed. In this course several tools and representations are proposed to support and structure this process and the thoughts of the engineers and designers.

Proof of Concept: In Proof of Concept stage, the emphasis should be on alternatives, interfaces, and designs, with some effort devoted to rewriting requirements and redefining functions. Especially in this stage mixed reality environments play an important role because the main goal in this stage is to evaluate chosen system concepts and subsystems in a cost effective manner. Subsystems and infrastructures the system to be designed interacts with can be simulated.

Prototype: In the prototype phase, the emphasis should be on integrating the subsystems and evaluating the prototype in a close to realistic setting. Especially the interactions and cooperation between all subsystems is important to evaluate. When designing NAIHS this is the stage where the combined actions of interacting subsystems will lead to an adaptive behaviour.

Real System: Finally, when applied to the real system, the emphasis should be on evaluation of the performance in a realistic setting and corresponding environment. The designers/engineers have no influence on the performance of the system anymore. All design choices are made and implemented, it is up to NAIHS to interact with the environment and adapt to changing circumstances.

The next paragraph 4.6 dilates further on the SIMILAR process steps within the first stage, the Paper System.

4.5.1 IRL vs Levels of Interoperability (LISI)

This IRL metric system described earlier, mixes up the two things: levels of interoperability and the effectiveness and efficiency of the performance of interactions and interfaces.

As mentioned in the previous paragraph the integration is very important. IRL is a method to ‘measure’ and compare the performance of the interactions and interfaces between subsystems. There is one difference between the performance of interactions in relation to their goal (cost effective), and how sophisticated the interactions are. For some subsystems in certain situations, very simple interaction patterns with a low level of interoperability are sufficient. This doesn’t mean the interface and the integration is on a lower level of readiness.

4.5.2 NEC Maturity Levels vs System Maturity

The same applies to NEC Maturity and System Maturity as described in the context of this course. The NATO NEC Maturity Roadmap Model tells something about the structure of systems, in what way services are requested and provided. The System Maturity describes the readiness of the system and the effort it takes to get to the real system.

4.6 Two System Stages

This paragraph focuses on two stages in the design process of a system: the Paper System Stage and the Real System Stage. The goal of the Paper System Stage is to make top level design choices to come up with promising system concepts. Further detailing of the concepts and proofing the value of chosen solutions will be done in the following stages. Preliminary designs and model based concept development will support these top level design choices.

The biggest effort in the paper system stage will be put in stating the problem and investigating alternatives. In Model the System and Integrate, design choices and preliminary designs are refined in system. In Launch the system and Assess performance system concepts are tested against criteria and scenario's or use cases.

In this chapter two tables have been filled in, one for a paper system, and the other for a real physical system. The main difference is that for paper systems application software and physical hardware will have to be replaced with analytical models and simulation models.

In the previous chapter tools and views are described. In this paragraph the use of those tools will be explained according to the SIMILAR process regarding the Paper System stage, and for each of the views: Enterprise [E], Functional [F], Process [P], Physical [Y], Execution [X] and Development [D]. In Figure 53 we illustrate which views get the focus through the SIMILAR process.

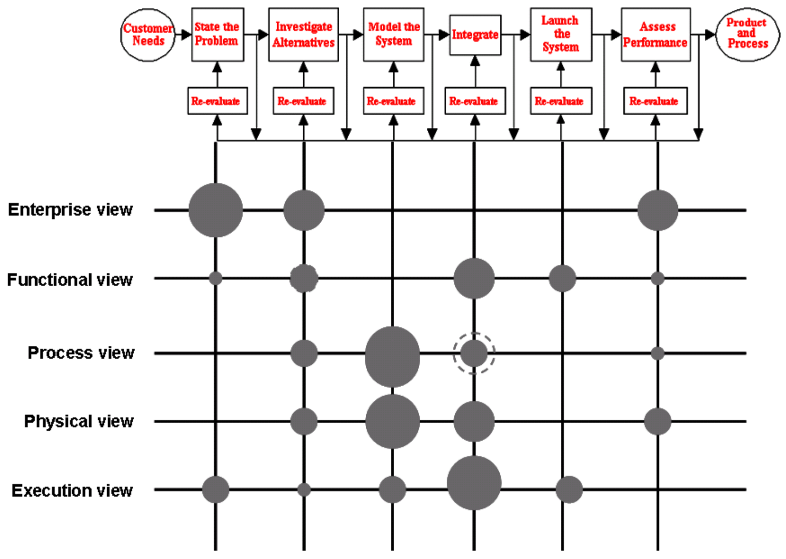


Figure 53 - Views mapped on SIMILAR

The next table gives the activities for a paper system:

Process / ODP / ISN Views	State the problem	Investigate Alternatives	Model the System	Integrate	Launch the System	Assess Performance
Enterprise [E]	ToR (B, C,	Business	ISN (ODP)	Context aspects	Publish	Measure

	P, Q), Purpose (to design a...)	Scenarios	Enterprise Model, SMART targets.	of ISN Real World View	report, present report and animations	KPI's
Functional [F]	Objectives, Functional requirements / constraints (paper / simulation / animation)	Functional design Patterns	Model the NAIHS layers and 4D/RCS building blocks	SW interfaces Distribution across systems Communication patterns HMI	Develop animations/ simulations, Produce report content	Check functional conformance
Process[P]	Interaction requirements, constraints, use cases	Information exchange patterns	Information flows between building blocks	Interfaces between building blocks. L7 protocols	Develop animation/ simulation, produce report content	Run simulations, check use cases
Physical [Y]	Physical requirements and constraints	Physical design patterns (Sensors, actuators, servers, HMI, network, power, processing, storage)	Network power, processing, system SW System configurations HMI	System API's topology, Message bus, network protocols (M, L1 – L4)	Install animation/ simulation tools, run phys. models	Check performance per basic transaction. Conformance to temperature requirements etc
Execution [X]	Use cases. Qualitative and quantitative requirements, NFR's	High and low scenario's	Component performance profile, models	System performance profile, model	Run animation and simulation tests	Run use case simulation
Development [D]	Development requirements and constraints. Project timing	Development platform choices, development methodology (RUP, waterfall, etc)	Specification of dev. env. WP plans,	IF between dev platforms, Project Plan, PM and sponsor	Install dev, PM, test scripts	Cost and time per FP?

Table 3: Typical activities/deliverables of the process steps for each of the views for a paper system

The next table gives the activities for a real system:

Process / **ODP ISN Views**	**State the problem**	**Investigate Alternatives**	**Model the System**	**Integrate**	**Launch the System**	**Assess Performance**
Enterprise [E]	Terms of reference: (B, C, P, Q), Target	Business Scenarios	ISN (ODP) Enterprise Model, SMART targets.	Context aspects of ISN Real World View	Manuals, PR campaigns, training	Measure KPI's

Functional [F]	Functional Requirements and constraints	Functional design Patterns	Model the NAIHS layers and 4D/RCS building blocks	SW interfaces Distribution across systems Communication patterns HMI	Develop the SW of the system, HMI	Check functional conformance
Process [P]	Processes, object interactions requirements and constraints	Alternatives for information exchange	Information flows between building blocks	Interfaces between building blocks. L7 protocols	Develop application information exchange, (message Queues etc)	Check information exchange conformance
Physical [Y]	Physical requirements and constraints	Physical design patterns (Sensors, actuators, servers, HMI, network, power, processing, storage)	Network power, processing, system SW System configurations HMI	System API's topology, Network protocols (M, L1 – L4)	Acquire, install and configure.	Check performance per basic transaction. Conformance to temperature requirements etc
Execution [X]	Use cases. Qualitative and quantitative requirements, NFR's	High and low scenario's	Performance profile	Compound transactions	Develop performance tests / monitoring	Check system performance under realistic load
Development [D]	Development requirements and constraints. Project timing	Development platform choices, development methodology (RUP, waterfall, etc)	Design of development environment, detailed project plans	Interfaces between development environment and equipment. Interfaces with PM and sponsor	Install the development environment, test and acceptation environments. Develop test scripts	Cost and time per FP?

Table 4: Typical activities/deliverables of the process steps for each of the views for a real system

The main differences are marked by yellow shaded cells and concentrate on the “launch the system” process step and on part of the enterprise view and functional view of the “State the Problem” step.

4.7 Process Chains

The precise order in which to tackle a case cannot be given as it can differ per case. The following matrix gives an indication in which the different process step and view combinations can be addressed.

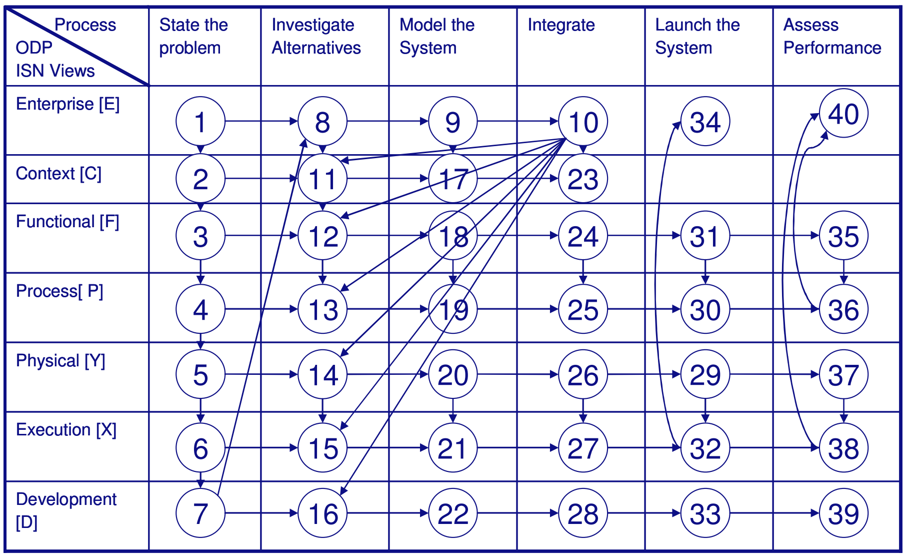


Figure 54 - Typical order in which the process steps are executed for each of the views.

There is some logic. In the matrix some process chains can be recognised. These can run in parallel. They are shown as overlay in Figure 55. Possible chains are:

A. **The state the problem chain**. This chain is handled before anything else. It is extremely important to get all the aspects of the problem in the clear. This does not imply that the state the problem steps cannot be revisited later in the project. This is one of the characteristics of the SIMILAR process.
B. **The Enterprise/top level view**. In this chain the case is modelled at a business level up to the integration step. From then
C. **The Investigate Alternative chain**. This chain is handled next. It is important to get a clear view what the options are. The Enterprise level alternatives are handled in chain B.
D. **The functional chain**. This chain is handled in parallel with the next chain E, the Process chain. This chain is a repetitive sequence of alternatives, modelling and integration. Integration is very similar to modelling in the process view.
E. The Process chain. This chain is running parallel with chain D
F. **The Physical chain**. Again this is a chain parallel to the chain D,E combination. It is also handled in synchronisation with the Execution Chain. E handles how, F handles sizing and numbers and performance aspects of a running system
G. **The Execution Chain**. This chain represents handling the performance requirements of a live system. Run in parallel with F but slight behind.

H. **The Project Management chain**. It handles two aspects of development. The first is the choices for development tools, the second the project management part. It could be worth separating these two chains.
I. **The Construction chain**. This is the chain where the actual construction work is handled. It is evident that the aspects of the different views will be handled very much in parallel with the process and execution aspects slightly behind the functional and physical ones.
J. **The acceptance/testing chain**. This is the steam where the system is tested in all aspects. When the system meets the requirements, the last chain.
K. **The Completion/delivery**. This where the system is handed over to the owner and a final acceptance test is done against the business objectives.

The diagram below visualises the different chains.

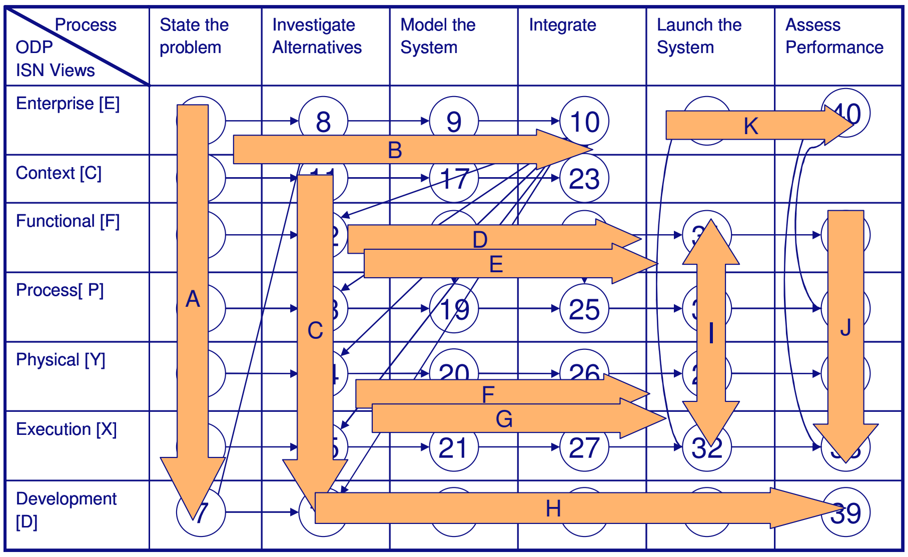


Figure 55: The process decomposed in logical chains of activities.

The following section provides more detail in the content of the matrix in general. The section after that applies the methodology to two specific cases, the TISNET case and the Smart Grid Islanding case.

4.7.1 State the problem

The following is a template with a table format for the cases

State the problem is a very common process step to all projects. It is basically a classical project definition or a project Terms Of Reference.

View	**State the Problem**	**Re-evaluate**
Enterprise	This view is mainly covered by the Terms Of Reference (TOR) of the project with a description of B(ackground), C(omplexity), P(roblem), Q(uestion), T(arget).	Verify problem statement with sponsor

	In the background section the stakeholders and the object need to be identified. The concerns should be clear from the complexity the problem and the question. The target is the purpose of the system	
Logical/functional	These are the more detailed Objectives of the TOR and resulting Functional Requirements and Constraints	Verify objectives
Process	The Processes involved , object interactions requirements and constraints	See enterprise
Physical	Physical requirements and constraints Geography/ topography Size	Check
Execution	Focus on the size of the problem and the quantities of the environment. Numbers (requirements and constraints). Also identifies quality requirements.	Check numbers
Development	Development requirements and constraints. Project timing o Phasing o Deadlines	Check

Table 5: Views on the "State the Problem" process step.

4.7.2 Investigate Alternatives

This is one of the key activities of developing an ISN as "Identify Alternatives" is about knowing what choices there are and how to make the choices. The alternatives are on all levels. The choices on each of the levels are not independent choices. They are all interrelated. The choice for a particular sensor drives the physical architecture as well as the functional one. Limitations of physical components can be compensated for by software and intelligence. The limitations are often more in the technology (battery power, range) than in the softer components. In the end even the business level choices are linked to physical and function choices and their cost and quality.

Throughout the project the investigate alternatives steps will be repeated to narrow down the choices.

View	Investigate Alternatives	Re-evaluate
Enterprise	Alternative Business Scenario's A standard alternative included in the project proposals of large companies often include the 0 alternative of doing nothing. The costs and benefits of this scenario need to be calculated as well. Doing nothing often costs a lot on things like maintenance of obsolete systems.	Evaluate relevance Evaluate business scenario's
Logical/functional	The choices on the functional level are many as there are many styles in this	Evaluate alternative services and objectives

	field. From classical monitoring and control styles using simple functional relations between input and output to advanced cognitive systems that operate on a virtual model of the environment on different levels of abstraction. Examples of these are NAIHS, 4-D/RCS, AnySense-World that all focus on slightly different perspectives of the system design. NAIHS focuses on a functional decomposition of signal, object, situation and impact, 4D/RCS emphasises on the hierarchical relation between different components and AnySense World focuses on multi-organisational aspects. What applies best depends on the case. The cases below actually indicate that these architecture could very well be combined.	
Process	The process view choices involve what information should be exchanged between the components, patterns like centralised versus decentralised and peer-to-peer and distribution of intelligence across the different components of the system.	
Physical	Alternative technologies Alternative design patterns Physical design patterns (Sensors, actuators, servers, HMI, network, power, processing, storage). The choices in these area's have been worked out in more detail in the section below this table.	Evaluate geographical alternatives
Execution	Choices in the Execution view very often relation to low and high scenarios in the use of the system. Sometimes these choices are outside the scope of the developer and are determined by the outside world. In that cases the scenario's drives the scalability aspects of the physical design. The execution view is also on choices on quality. Systems with better/more dense detection technology could deliver better results but at a cost that is not justified by increased benefits. Cost of maintenance and maintenance scenario's should also be included here.	Evaluate scenario's Evaluate classes of service
Development	Alternative development platforms	Evaluate

	Development view has two major aspects, the development related technology with its choices and the project management aspects. This project will not put much emphasis on these aspects of ISNs. However, in real-life choices in this area can make or break the project. Choices for development platform are partly driven by the choices for sensor technology as sensors often come with development environments for the sensor specific software (Signal processing and sometimes object detection. There are also the usual alternatives of Microsoft .NET versus "Open Source". Project management choices are also on development methodology such as RUP and Waterfall but also include alternative deadlines and project completion dates.	o Phasing o Planning o PM style o Development platforms

Table 6: Views on the "Investigate Alternatives" process step.

4.7.3 Model The System

The "Model The System" step includes the design of the ISN using the components selected in the "Investigate Alternatives" step.

View	**Model the System**	**Re-evaluate**
Enterprise	Create the formal ISN World View (similar to the ODP enterprise view) (see details in section 3.2.3.1: *Enterprise view in RM-ODP*)	
Logical/functional	Per NAIHCS/JDL layer define the functionality Data structures, Algorithms, then per 4D/RCS level do the same for higher layers in the hierarchy. Each of the 4D/RCS function blocks can be decomposed in multiple NAIHS/JDL layers.	
Process	Develop interaction patterns between the NAIHCS function blocks/objects and do the same for 4D/RCS function blocks/object	
Physical	Specify hardware, network technology and power technology using the preferred alternative(s) from the "Investigate Alternatives" process step. Select basic SW components (OS MW) Develop Memory maps (scarce resource mapping) Define the performance cost drivers	

Execution	Develop performance model and simulate/model the performance for the expected load and scale.	
Development	Set up software structures Set up libraries Set CMS system and development platform(s)	

Table 7: Views on the "Model the System" process step.

4.7.4 Integrate

View	**Integrate**	**Re-evaluate**
Enterprise	These are the context and interaction related elements of the enterprise view in "Model the System".	
Logical/functional	In this step the detailed SW interfaces are specified. Furthermore this is where functional components are distributed around, eventually, physical components. It also concerns communication patters between these components. The HMI can also be considered an integration aspect. The Intergate step logical view is very much aligned with "Model the System" Process View.	
Process	This involves Interfaces between building blocks and L7 (application) protocols.	
Physical	This involves System APIs, topology, Network protocols (M, L1 – L4).	
Execution	The execution view for this step involves compound transactions across multiple physical components and scaling aspects of the distributed architecture.	
Development	Interfaces between development environment and equipment. Interfaces with PM and sponsor	

Table 8: Views on the "Integrate" process step.

4.7.5 Launch the System

Launching the system is the actual construction phase. This study mainly focused on the design and less on the actual construction.

View	**Launch the System**	**Re-evaluate**
Enterprise	The Enterprise "Launch the System" involves the interaction with the external world required to get the system running and accepted. It involves manuals, PR campaigns, training and contracts on Service levels and possible commercial aspects such as tariffs.	
Logical/functional	Develop the SW of the system, HMI	
Process	Develop application information exchange, (message Queues etc)	

Physical	Acquire, install and configure.	
Execution	Develop performance tests / monitoring	
Development	Install the development environment, test and acceptation environments. Develop test scripts	

Table 9: Views on the “Launch the System” process step.

4.7.6 Assess Performance

The Assess Performance is the step leading to acceptation of the project and the system. Its is completed with the passing and acceptation of the acceptance tests. Throughout the operation of the system “Assess Performance” migrates to ongoing performance monitoring.

View	**Assess Performance**	**Re-evaluate**
Enterprise	KPI reporting	Does the system meet the business requirements and the customers expectations?
Logical/functional	Check functional conformance	Does it conform? Does it do the job?
Process	Check information exchange	Does it conform?
Physical	Conformance to physical requirements	
Execution	Check performance (realistic load)	Does it perform under realistic conditions? How doe it perform under stress?
Development	Cost/FP time to deliver	

Table 10: Views on the “Assess Performance” process step.

5 Result Cases

The views and methodology from chapters 3 and 4 will be applied to two cases in this chapter. The process used is SIMILAR (see chapter 4) consisting of the following steps

1. State the problem
2. Investigate Alternatives
3. Model the system
4. Integrate
5. Launch the system
6. Assess performance

After each step there is the re-evaluate step to check if the work up to there meets the requirements.

In each of the process steps, the problem will be viewed from the following viewpoints as derived for ISN from the “4+1 view model” and RM ODP (see chapter 3):

1. Enterprise
2. Logical/functional
3. Process
4. Physical
5. Execution
6. Development

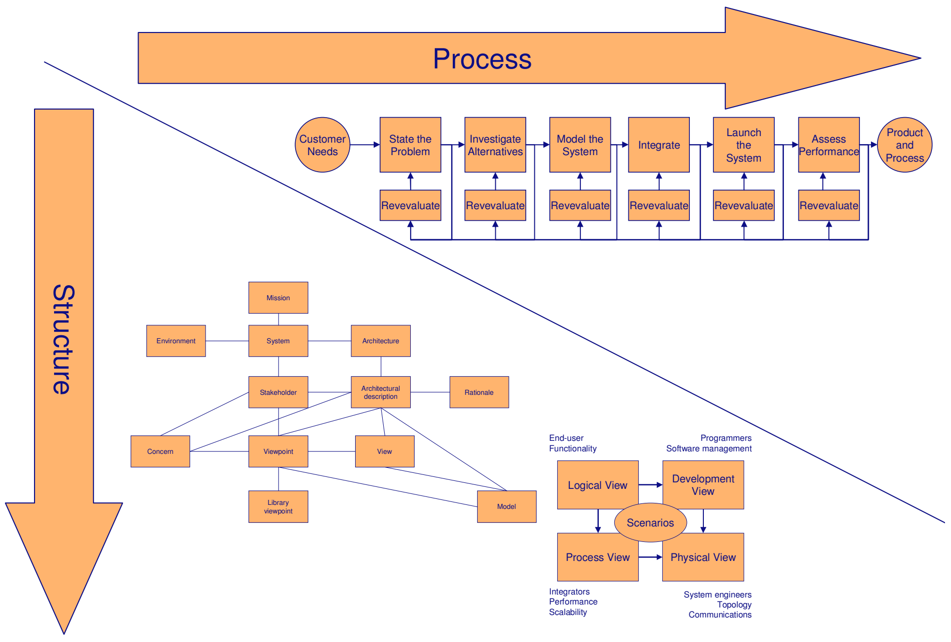


Figure 56: Process and views in perspective

The ISN system to be built can range from a paper system to a mass produced real system. This is more or less in line with Technology Readiness Levels (TRL). Product referred to in TRL are:

1. A paper model with simulations
2. A proof of concept
3. A prototype
4. The real system

It is clear that the emphasis on particular views differs for the different Technology Readiness Levels.

DISCLAIMER: The cases below serve only as examples to illustrate the application of the views and methodology of this report. No factual data should be extracted from these examples.

5.1 TISNET

The TISNET [TISNET2007], [TISNET2009] case is the first case where the ISN methodology has been applied. For TISNET we strictly follow the SIMILAR process steps and cover these for each view. Thereby we apply the design methodology as used with the NAIHCS functional decomposition for the JDL levels and 4D/RCS for the hierarchical levels involving time and space (4D).

5.1.1 State the problem

View	State the Problem
Enterprise	**Background** • There are x kms of freeways in NL. Freeways are main medium and long-distance travel facility. 85% of all commuters use freeways to get to their work. Freeways are also the main arteries for road transport. Freeways range from 4 to 10 lane roads with speed limits ranging between 80 and 120 km/h. There exits, petrol stations, freeway junctions. **Complexity:** • Freeways are getting very crowded resulting in an increasing number of traffic jams and an increasing number of accidents. • Road management has limited control over the freeways (overhead speed signs), limited patrolling etc. • There are advanced technologies to prevent accidents (radar driven cruise control) etc but they have not been implemented in standard cars. Implementation of new technology will generally take years. **Problem:** • Traffic jams causes loss of time, accidents cause jams, material and physical damage. Some of this is caused by “human” behaviour of car drivers and limits in reaction time. Other causes are road conditions, weather, road works, car failure etc. **Question:** • Design a potentially nation wide road mobility system that reduces risk of accidents by 50% and improves traffic flow in the varying road conditions reducing traffic jams by 70% in size and number. Equipment is cars can be considered as long as it allows for gradual implementation and that the benefits are there with a limited numbers of car devices. The system itself will be implemented in a phased manner over a period of up to 10 years. The Question can be translated into a Mission or Purpose statement.

View	State the Problem
	Purpose/Mission The purpose of the TISNET system is implement an advanced traffic control system that reduces accident rate by X% and traffic Jams by Y% in size and by Z% in number.
Logical/functional	The state the problem functional view maps onto the objectives. The objectives of TISNET are: • to implement an automatic traffic control system 1. Using active sensing technology 2. traffic management technology ▪ that can interface to specialist navigation systems in cars ▪ that can use other means of influencing cars and car drivers that have no specialist equipment 3. Provide traffic control with more advanced technology to control traffic flows from the control room
Process	Communication with the cars and car drivers can be through special in car navigation equipment or through external information facilities as roadside information sign/signal.
Physical	There are 38 Freeway totalling 2346 km

Length in km	# of lanes	Average section length in km
	2	
	4	
	6	
	8	
	8+	

	There are [xyz] junctions etc.. The roads can best be described using e.g. digital road maps such as offered by Google, TeleAtlas, NavTeq etc. There are also physical constraints. These are related to the speed of cars and acceptable latency for control and information. Other constraints can be related to the power supply for distributed sensors and processors.
Execution	The execution view will have figures on: • Traffic density distribution • Accident rate distribution • Current travel times
Development	Proof of concept in 12 month. A prototype implementation is to cover the A13 from The Hague to Rotterdam. The prototype is to be implemented within two years. Full system roll-out after evaluation in 10 years, 50% of the Dutch freeways to be covered 5 years.

View	State the Problem
	The system to be developed is subject to the Dutch policy of aiming at Open Source Software.

Table 11: TISnet Views on the "State the Problem" process step.

5.1.2 *Investigate Alternatives*

View	Investigate Alternatives
Enterprise	There are the following alternative business scenarios: • Non-navigation (in-car system) based • Full and partial in-car system based • Forced implementation of navigation gear • Subsidised implementation of navigation gear • Hybrid intra-car and road-side There are also potential scenario's for speed of implementation and financial models (State, Insurance, individual, companies)
Logical/functional	The functional alternatives are (I am not the right person to ask): 1. Vehicle detection 2. Flow detection 3. Density 4. Traditional Monitoring and Control 5. Cognitive self-learning
Process	The Process alternatives are: 1. Central control 2. Hierarchical control 3. Distributed control Other choices are in how information flows to the car/car driver.

Physical	The physical alternatives focus around:
	1. Car/traffic detection alternatives (reasonable ones)
	    a. Magnetic
	    b. Induction loop
	    c. 3-d magneto sensor
	    d. Optical
	    e. Camera's
	    f. Radar
	    g. Acoustic
	2. Car control alternatives
	    a. Overhead information signs (open loop control)
	    b. LED guides on guard rail or in road (open loop control)
	    c. PNA add-on (open loop control)
	    d. From Simple cruise control (speed) to
	    e. Advanced auto-pilot (both closed loop control)
	3. Power
	    a. Mains/Solar (for camera's and road-side sensing)
	    b. Non-rechargeable long-life battery (in road sensors)
	    c. Energy Harvesting (in road sensors)
	4. Network (a and b for in-road sensors, c-> for road-side)

	a. LRWPAN (IEEE 802.15.4, ZIGBEE, ISA100, DUST, Wireless HART) b. Optimised LRWPAN c. WiFi d. Blue-tooth 5. Server topology. I.e. how do we distribute the processing power. In some respect these choices are similar to Process View Alternatives with this difference that it concentrates on computing power rather than information flows. One can try all combination or select some suitable scenario's selected combinations. In this case the choice for sensing is relatively independent of the choices for control For simplicity in this case we select 3-d magneto sensors and camera's for car sensing, for car control we select the PNA and cruise control with external control options like radar or an auxiliary source) for a limited number of cars.
Execution	The execution alternatives are scenario's based on penetration of in-car equipment to support the system
Development	The alternatives on development focus on development platforms and System Development methodology. Options are Incremental development or waterfall etc.

Table 12: TISnet Views on the "Investigate Alternatives" process step.

5.1.3 Model The System
In Model the system the table format used above is no longer suitable and the section have become to large. Hence we switch to sections per view rather than a row in a table per view.

Enterprise View

The Stakeholders are:
1. Road authorities
2. Traffic control
3. Car/fleet owners
4. Car drivers
5. Emergency vehicle drivers
6. Insurance companies
7. Employers

There are also stakeholders related to the system itself. They include development-, operational-, support- and maintenance- staff as well as the supplier and system owner.

The Objects (including people):
1. Cars (and other vehicles)
   a. Controls
   b. Cruise control
   c. Navigator
2. Emergency vehicles
   a. As with cars, possibly more advanced

3. Car drivers
4. Traffic control
5. Road-side/overhead information panels
6. Radio/TMC
7. Communication

It is clear that there are two types of stakeholder and objects, those that have indirect or non-functional (financial/political/emotional) interest and those that participate actively in cars driving on roads. For business case development the indirect stakeholders are important. For development of the logical/technical system the direct (functional) users and objects involved are the key items. However, some choices may be financially rather technically driven. The relation of the direct users for the car are modelled below. Indirect steer to road controllers are kept out of the diagram.

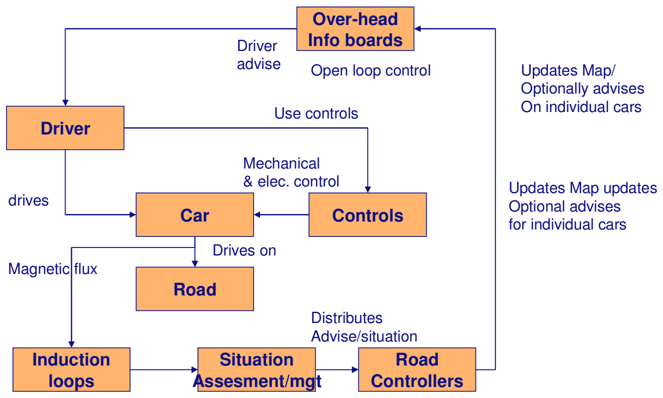


Figure 57: Basic relations of the enterprise objects for the TISnet case

Below is an Enterprise View model of the existing world with a using a TomTom HD navigator. This view actually belong in “Model The System”, Enterprise View. This implies that the Enterprise View modelling should be done before investigating the alternatives.

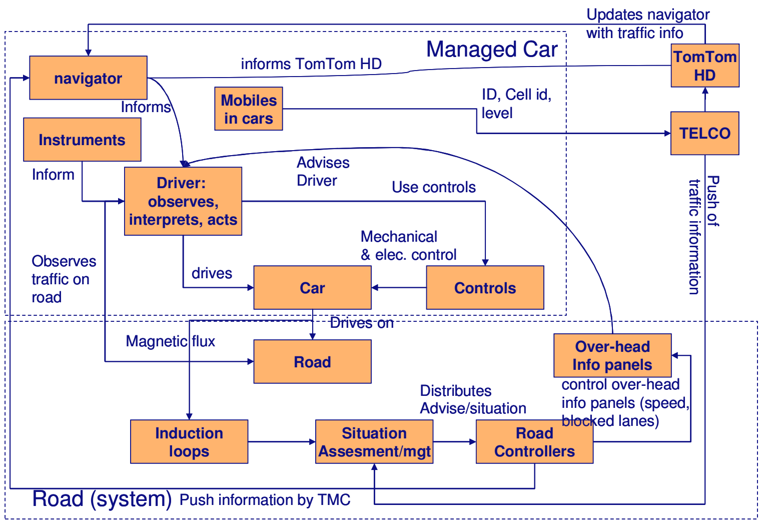


Figure 58: Contextual part of enterprise view of existing situation.

There are two high level scenario's for control. There is the "open loop control" where the driver is part of the control loop and there is "closed loop control" where the car is controlled, normally, without car driver intervention. The closed loop control alternative is in the next diagram.

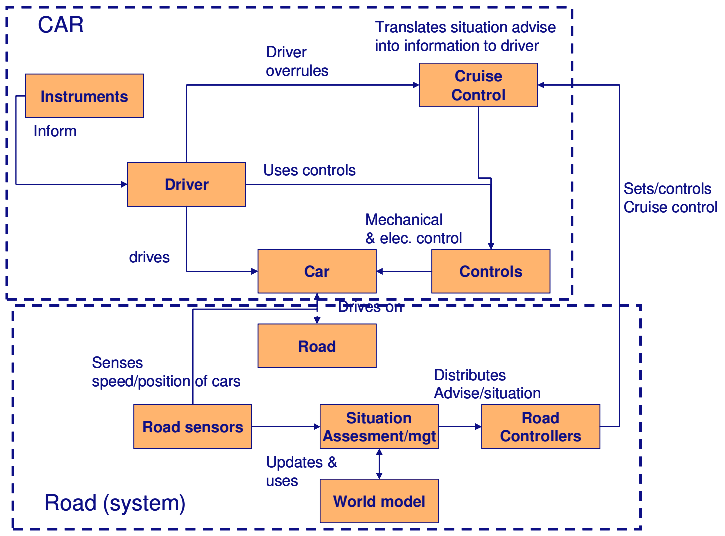


Figure 59: A closed Loop control example

The Open Loop Control alternative in the next diagram mainly differs in the fact that the road controllers inform the driver through the navigator rather than directly control the cruise control. For short term developments this method looks more feasible as a lot less safety aspects of an outside system controlling a car have to be solved. This is not only a technical but also a regulatory/legal problem.

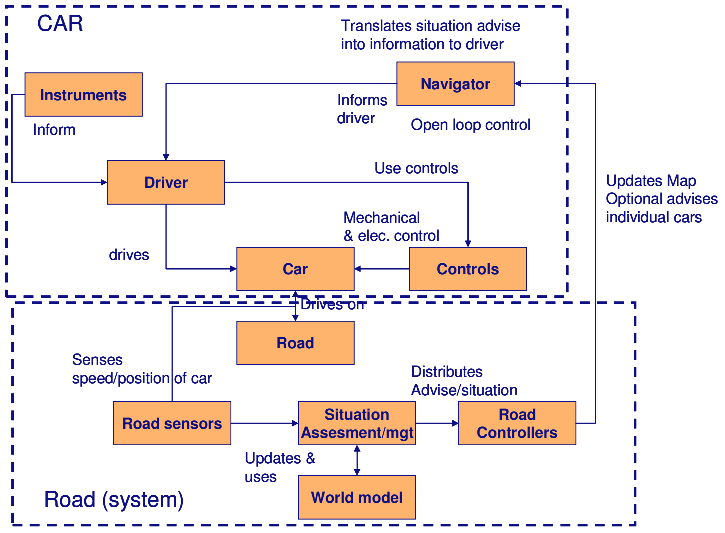
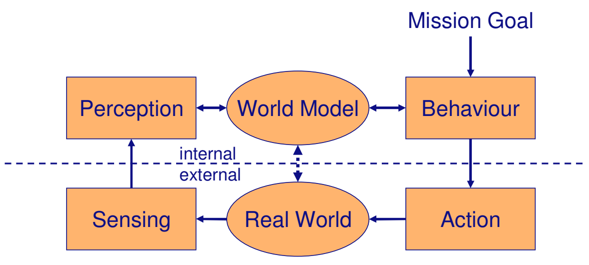


Figure 60: An Open Loop Control Example

Logical View

Whilst on high level the system to be modelled on a functional level it soon diverges depending on the technical alternatives chosen. On a High level we can use the RCS model for Real time Control systems.



On a lower level we have to describe the functional/logical design with a combination of NAIHS and 4-D/RCS. We start with the specification of what sensing, what control mechanism we use. With a sensor network like this we can distinguish between the NAIHS layers (signal assessment, object assessment, situation assessment and impact assessment with the related management blocks, but there is also the physical dimension and time frame. The physical dimension varies from road sections of e.g. 10metres either static or relative to the car up to the national road system and a lot in between. The temporal dimension is equally varying from 10-100ms up to hours or even the year (if going up to governmental/political levels). The temporal dimension is related to the space dimension but the relation is not 100%. The temporal/special view (ranges) also relates to the different stakeholders/actors of the system and their various roles and concerns.

Below is a diagram with a combination of 4D-RCS/NAIHS layering and functional components. No decision has been made yet where to locate what functionality, how to communicate and to what abstraction level one has to go with “world models”

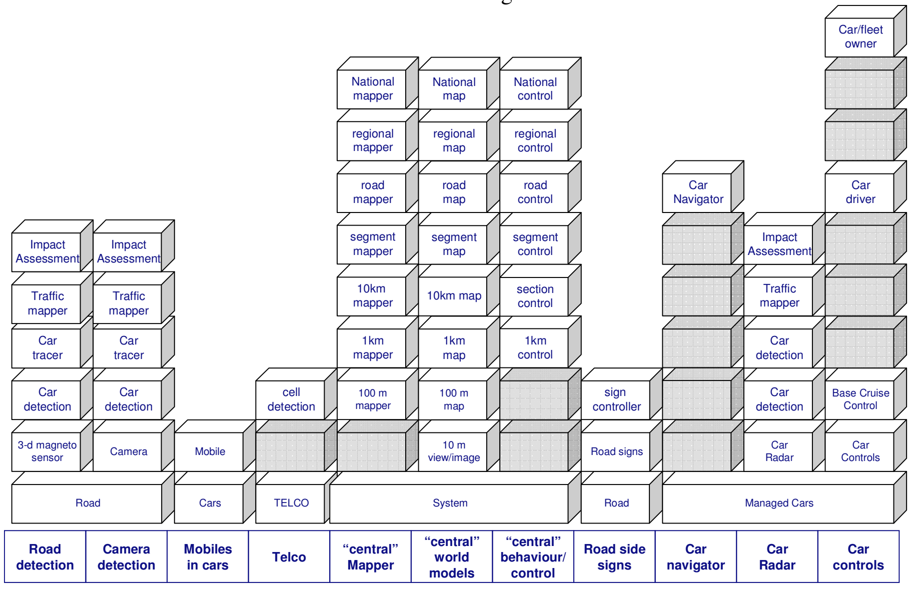


Figure 61: The functional decomposition of all the alternative systems considered.

There are two alternatives for sensing on the road, two sensing technologies for the car (not including the car driver as a sensor). There is a third sensing technology using cell detection of mobile phones. This detection technology does not produce accurate car positions but rather traffic flows through statistics on large numbers of cars. Since recently the Telco pushes the information to central road control. This system is also the sensing base for the TomTom HD service. In the middle there is a more centralised group of columns for more centralised intelligence (in this case intelligence not in the car). This “central” system could be one big centralised system or lots of distributed intelligence, close to road sections. The choice has not been made yet. This should be part of the process view.

There is an independent, commercial, service for providing drivers with congestion information: the TomTom HD service. The components of this service are depicted below. Not that there are far more mobiles involved than navigators with a TomTom HD service subscription.

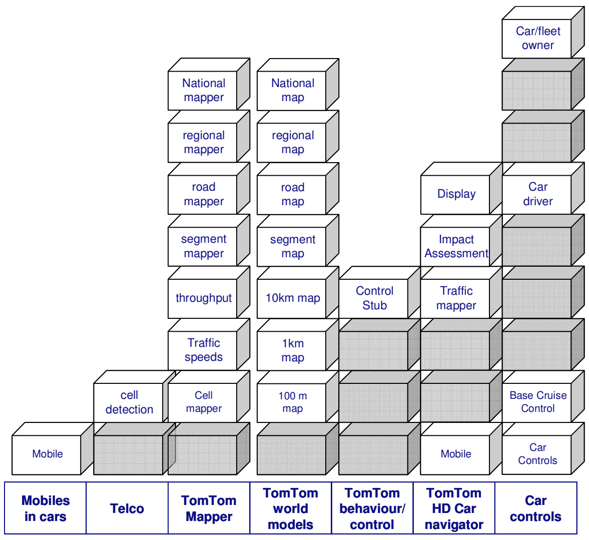


Figure 62: The Functional View of the existing TomTom HD service

Currently the national road controller is also fed with cell position information from a Telco.

In the next paragraphs and diagrams the functional and process views are developed step by step. They do not include the existing TomTom HD service but focus on other that mobile sensing technology.

The first step is to define the essential functional components. One could argue that a central component is not essential but for this case we assume there are central components as there is a central control room in Driebergen today. Note that these components are logically centralised. Physically there is still the option to distribute the functionally across distributed systems. We can recognise the following groups of systems on the basis of there geographical position:

1. Road side sensing. These include induction loops, 3-D magnetic sensors, camera's and radar. For the case of this study we restrict our self to 3-d magnetic sensors and camera's. The other technologies may require a slightly different approach but not significantly so;
2. Road Side Unit (RSU). These are road-side computers that can communicate with road-side sensors, road side information signs and even cars that drive past. Not all alternative require RSU's;
3. A logically centralised group of systems doing sensory process (of remote sensors) maintaining a world view at appropriate abstraction levels and doing control/management at appropriate abstraction levels. For the purpose of this study we have abstraction levels based on geographical scope, starting with 10m sections stepping up in about an order of magnitude to roads, regions and the nation;
4. Road-side control. For this study this includes dynamic information displays over-head on roads such are used now to block lanes and to limit speed;

5. Cars/vehicles. This group includes cars and other vehicles. Cars have controls, a driver, an owner (could be different from driver). Cars can be equipped with a navigator, cruise control and radar controlled cruise control. This is the configuration considered for this study.

In the first diagram the central system has three distinct columns. One for sensing/assessment, one for control/management and one for intelligence that sits between assessment and management. the 3D/RCS models uses three columns, NAIHS only uses two and implies the intelligence in either in assessment or on management. In subsequent diagrams we will only use the two column model in order not to clutter the diagrams too much.

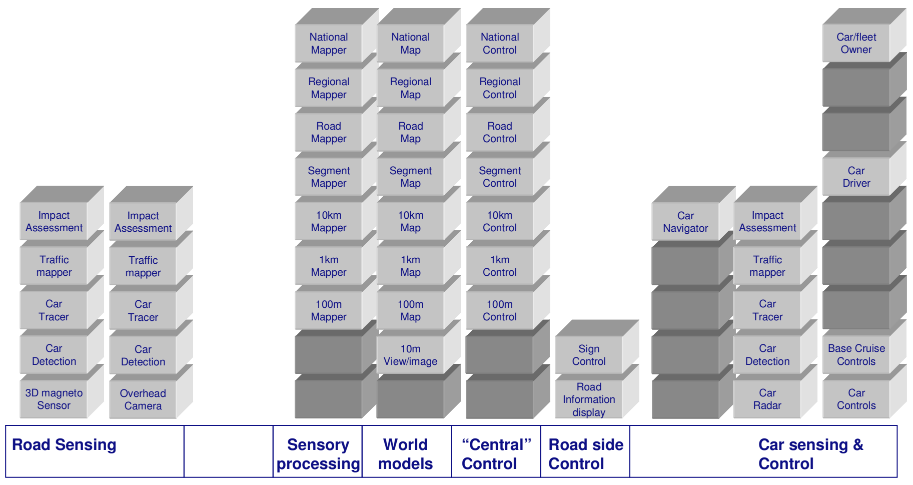


Figure 63: The functional blocks of TISnet based on centralised intelligence.

The next example can use RSU's for local processing of the 3-d magnetic sensor data. The reasons are the response times and scalability. It also off-loads the 3-D magnetic sensor from extensive local processing extending their life time. These sensors are installed in the road surface and are battery powered. The RSU's in this case assess and manage about a 100 m of road. It focuses actually on individual cars on the road. The central systems manage sections of road of 1km+. These systems will more operate on traffic flows than individual vehicles.

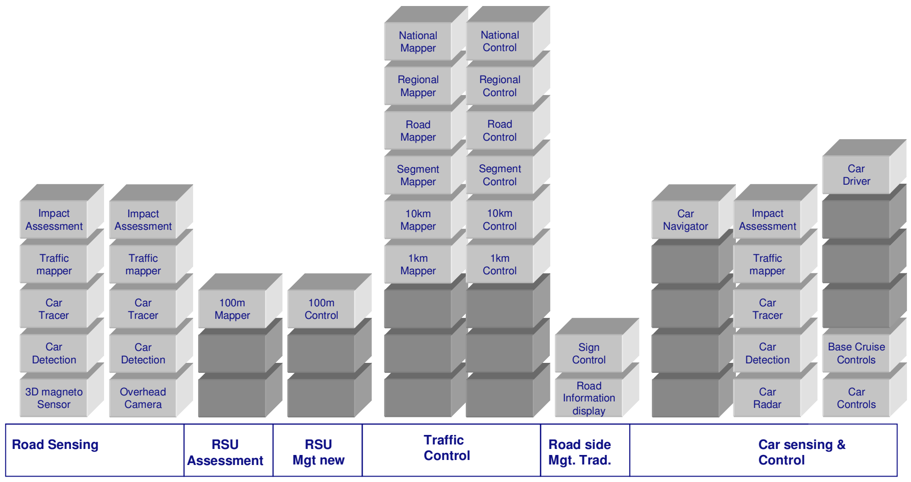


Figure 64: A functional break-down of TISnet with Road-Side Units (RSU) managing about 100 of road.

Process View:
The process view includes cardinality of the components and the interaction between the components.

The next diagram adds cardinality. The cardinality is indicated by extra systems behind the front one. The numbers are symbolic. The 3-D magnetic sensors will be in large number. Perhaps a hundred per 100 m of three lane high-way. Camera's have a larger range than the 3-D magnetic sensors and can typically monitor up to 1000m of multi-lane highway. There are multiple RSU's, potentially 1 per 100 meters. The central system can be a single system or multiple systems covereing a region each. Information displays/panel are typically positioned every 1-2 km. Cars can be in large number. Note that car move along the road and thus have to have some roaming communication capability with the RSU's. Not all cars have navigators and advanced cruise control. The diagram, however, only depicts the ones that have.

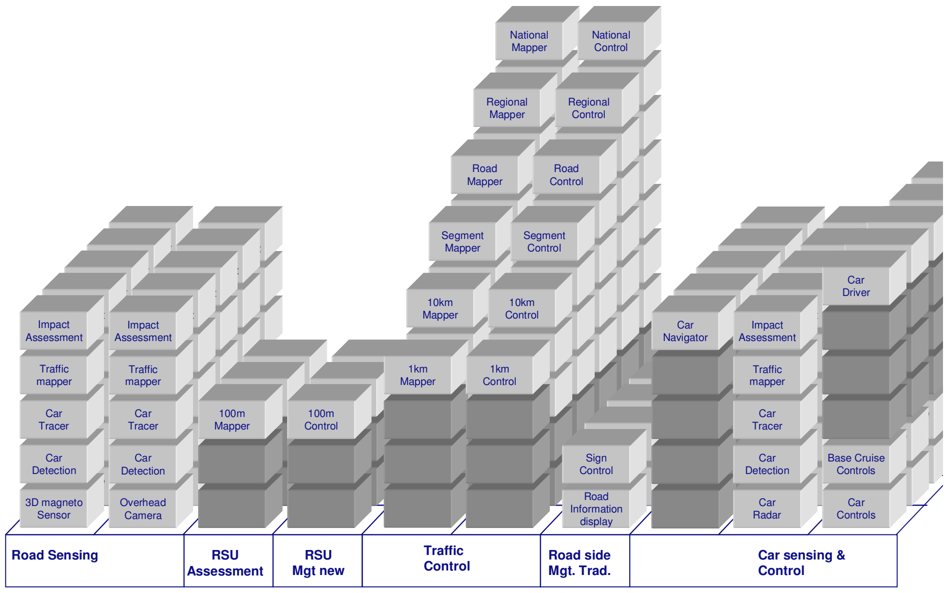


Figure 65: Functional breakdown indicating cardinality and multiple levels of central systems

The next diagram adds information flows. The 3-D magnetic sensor communicate with neighbours notably on passing information on vehicles moving on to next cells. Some of the 3- magnetic sensors communicate with the nearest RSU. The RSU’s pass information to their neighbours very much in the same way. RSU’s also communicate with the central systems passing information on general traffic flow information so that the central systems can build a more comprehensive world view. the central system can inform the RSU’s on more global problems and advise on particular control strategies. The RSUs can also communicate with cars with specialist navigator systems. They can pass information on close and more remote traffic that can significantly influence the local traffic.

The cameras communicate with their neighbours as well passing information on cares detected and moving to the next cell (of the next camera). In this case the cameras communicate with the central system. Communication may not include actual the cameras frames but could be limited to the objects detected, their position, speed and direction.

The central system can construct world views on several levels like a 1km, 10 km, segment, road, regional and national level each with a larger scope and less detail. The number of levels here are based on a scaling factor of about 10. This factor should be determined through modelling, simulation and actual tests.

The central system can communicate with the road-side information displays. This allows limited control over cars that have no sophisticated navigator that can communicate with RSU’s on details. The central system can also communicate with the navigator very much the same way as is now done using TMC or a system like TomTom HD.

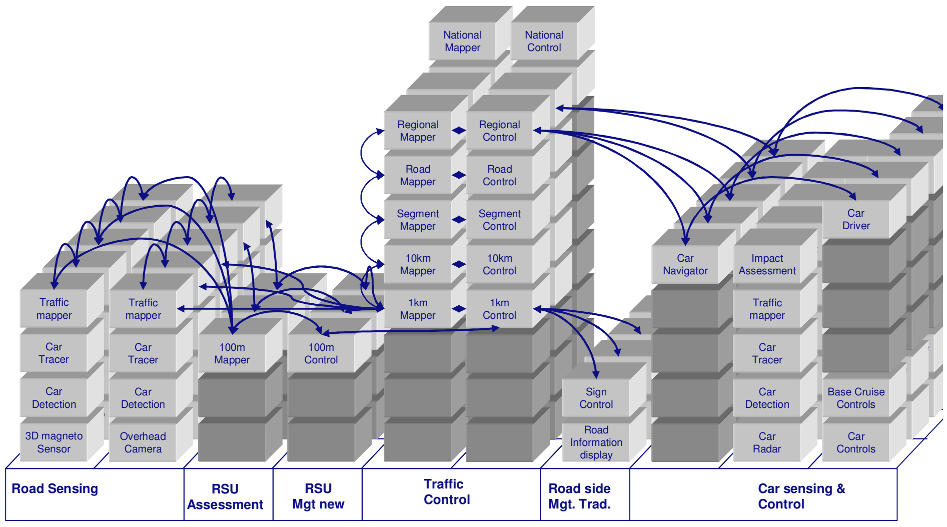


Figure 66: An functional break-down with indication of cardinality and information flows

The multiple levels of the JDL and NAIHS models (signal, object, situation and impact) are very important but do clutter the digram. In the next diagram the functionality of the multiple layer are now grouped per device. The Yellow spheres are for assessment, the purple ones form management/control and the green one for humans in the system (implying both the human and the HMI). The circle around the sphere indicates some form of reasoning. Note that the road-side information displays are dumb. They have no reasoning logic and are driven directly from e.g. a central system.

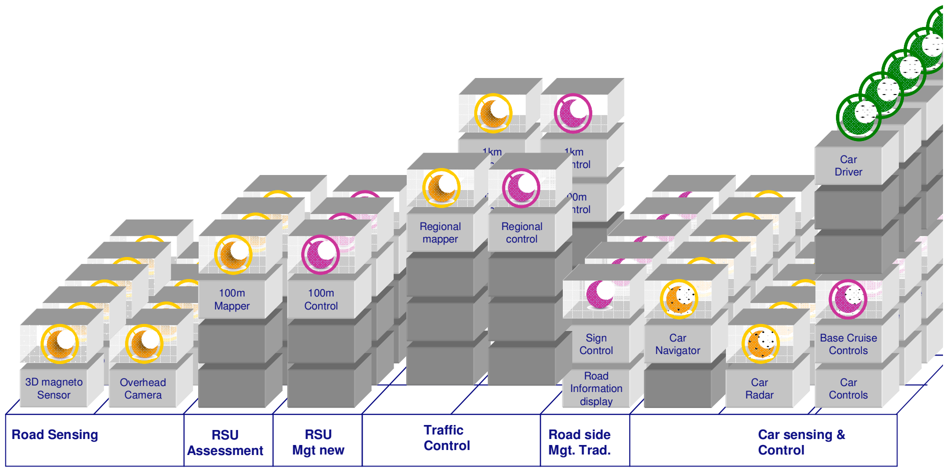


Figure 67: A simplified functional break-down with indication of assessment, management or human.

The next diagram adds the logical information flows. The blue sphere is a communication function. The circle around it indicates an intelligent form of networking. For reasons of simplicity the cardinality in the diagram has been reduced. Two cameras implies there are multiple cameras, the three sensor indicate that there are many more of these sensors. The same with RSU's and cars. There are two central systems signifying there are more than one. E.g. one per region. A grey block indicates that the system has no sensors/actuator of its own but use information from remote sensors.

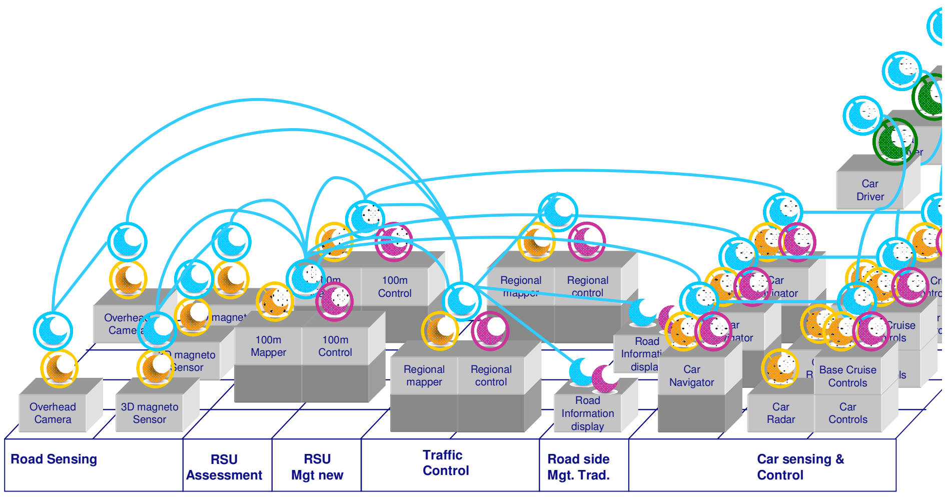
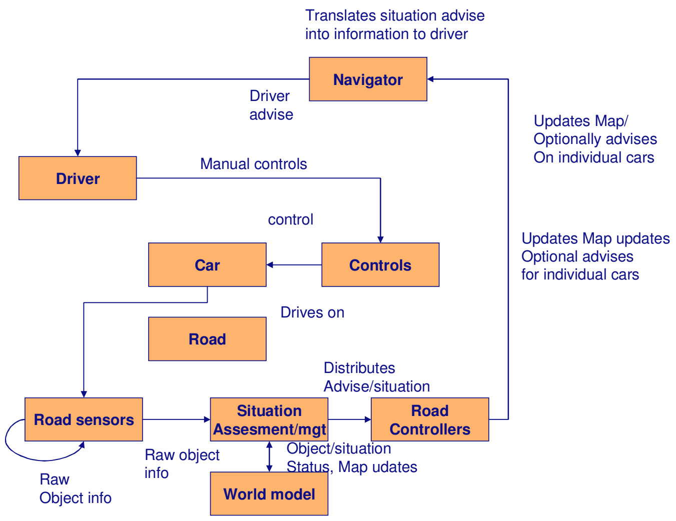


Figure 68: Functional break-down with information flows. (The JDL layers collapsed)

The next diagram depicts the relation between a single vehicle and the system.


Translates situation advise into information to driver
Navigator
Driver advise
Driver
Manual controls
Updates Map/ Optionally advises On individual cars
control
Car
Controls
Updates Map updates Optional advises for individual cars
Drives on
Road
Distributes Advise/situation
Road sensors
Situation Assesment/mgt
Road Controllers
Raw Object info
Raw object info
Object/situation Status, Map udates
World model
Figure 69: The information exchange between a vehicle and the system

Modelling the System: Physical view

As example for alternatives on the Physical, Executive and also Process Level Two alternatives for TISNET have been worked out for the 3D magnetic sensor part.

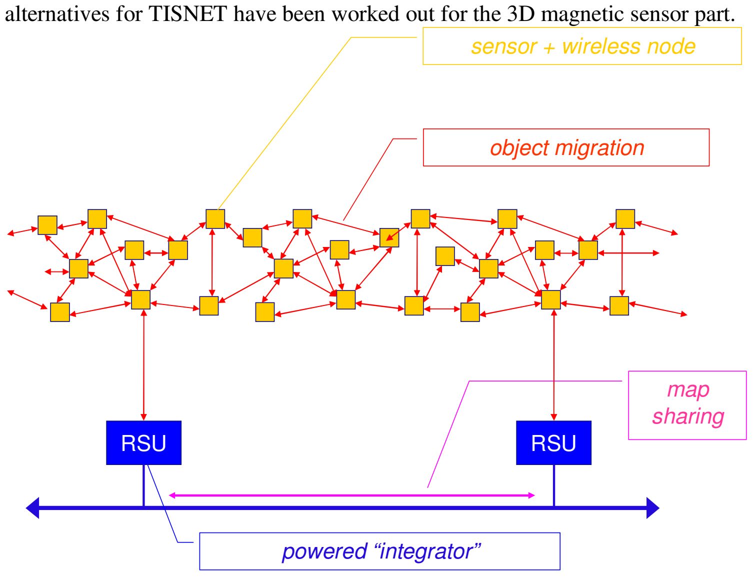


Figure 70: Alternative 1 for communication between 3d magnetic sensors and Road Side Units

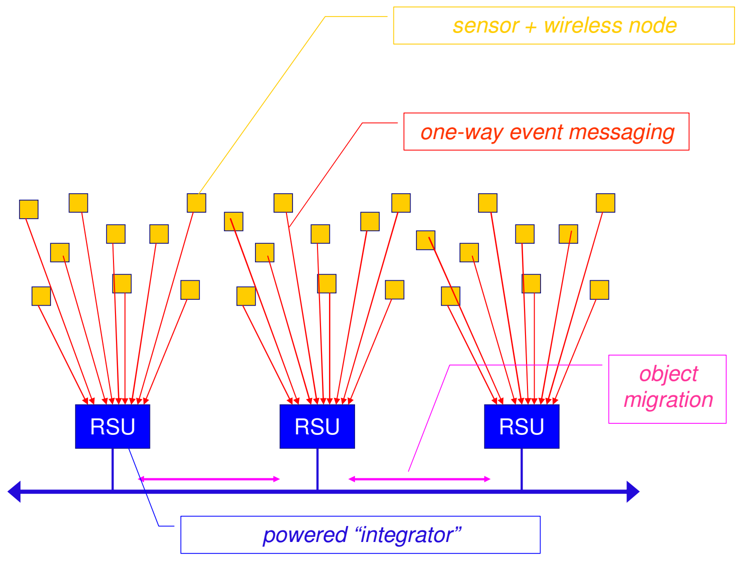


Figure 71 - Alternative 2 for communication between 3d magnetic sensors and Road Side Units

The preferred scenario can be determined using the a modelling/simulation tool simulating battery usage, network communication (topology and range), device density etc.

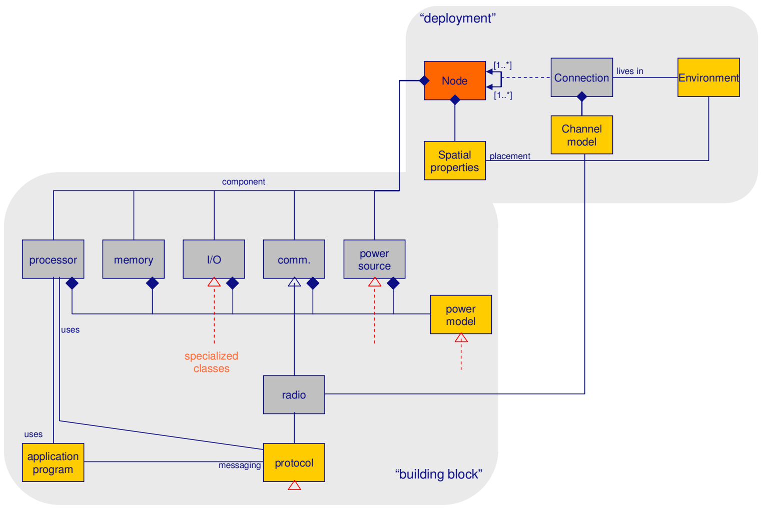


Figure 72: Example simulation tool structure for evaluating configuration alternatives

5.1.4 Launch the System

The construction activities of the system have not been included in detail the TISnet case. It involves:

1. Building the functional modules for the sensor devices (3-D magnet sensors and camera's), RSU for local object and situation assessment and situation management (communication information with the cars). For the central systems for less detailed assessment and management of traffic.
2. Building the communication drivers including some application level protocols for all the physical devices and functional modules that need to communicate with external systems. This also requires cooperation with vendors of navigation devices to agree on a common protocol for information transfer.
3. Acquiring, evaluating and installing the 3-d magneto sensors
4. Acquiring and adapting existing camera's
5. Installing power and communication infrastructure with camera's and RSU's
6. Producing the next generation of navigation devices that can communicate with the system.

For a paper system "Launching the System" involves full scale simulations of the system components and the complete system. The simulation system will have to be built/configured to do so.

5.1.5 Assess Performance

For a paper system it involves:

View	**Assess Performance**	**Re-evaluate**
Enterprise	KPI reporting. The KPI's in this case involve not the system performance but the acceptance of the presentation of the paper system by the sponsor.	

Context		
Logical/functional	Check functional conformance. For a paper system this is a paper assessment.	
Process	For a Paper system simulations have to done using representative use cases.	
Physical	Conformance to physical requirements in this case is a paper exercise where the vendor specifications are checked against the requirements.	
Execution	This is a simulation where the focus is on performance under realistic conditions and under stress rather than a simulation of the functionality.	
Development	Building paper systems are not free of charge. The costs and project throughput should be checked.	

Table 13: Assess performance of a real TISnet system

For a real system it involves:

View	**Assess Performance**	**Re-evaluate**
Enterprise	On the enterprise level one has to assess that the system meets the sponsor's requirements. These requirements should have been translated into high level Key Performance Indicators (KPIs). These KPIs should be measured in a realistic environment. If the system, as it should be, in phases like pilot, phase 1 → phase n to handle scalbility issues the assess performance should be done after each phase and the results should be used to improve the system if required. Troughout the live cycle of the system these KPIs should be measured periodically (weekly, monthly, yearly)	
Context		
Logical/functional	The logical view on assess performance is assessing the functional conformance of the components.	
Process	The process view is about assessing the system function conformance including information exchange.	
Physical	The physical view is about how the physical components behave in test circumstances and conformance to physical requirements as those on the operating environment (temperature, humidity, vibration, mechanical shock, lighting conditions). Especially the sensors and RSU's require extensive testing.	
Execution	The execution view requires a system test under full	

	realistic load and behaviour under stress. It has both a real physical aspect as well aspects of system performance with the selected CPU, IO capacity and battery performance.	
Development	Basically it involves calculating the total projects costs and performance indicators such as: cost and time per function point.	

Table 14: Assess performance of a real TISnet system

5.1.6 Conclusions

Some conclusions can be drawn from the TISnet case:

1. The SIMILAR process steps themselves are useful. The naming is a bit awkward. Terminology like Launching the System has a different connotation in Software Engineering
2. The ODP enterprise view needs some changes to be useful for ISNs. The main drivers for the changes are that ISN’s involves massive parallelism as opposed to far more single threaded sequential behaviour of administrative system. The interaction with a real world which does not always react the way the system expects also requires a different approach.
3. SIMILAR order applies well to the individual views. However, across views, the SIMILAR steps should be executed all in parallel. E.g. the SIMILAR process should be executed first for the Enterprise view aspects (up to Integrate), Then the Investigate Alternatives should be started for the other views (more or less in parallel). Then we have parallel chains where the functional, process, physical and execution aspects should tackled for modelling and integration and launching the system. Assess the System is tackled bottom up until we reach the enterprise level.

5.2 Smart Grid Islanding

The Smart Grid Islanding cases is taken from a challenge done December 2009. In this challenge the ISN methodology and process developed during this project have been applied. In this case the chain approach of SIMILAR for the different views has more or less been used. The output of the challenge, a PPT file with over 60 sheet including appendices have been converted into this word document.

The system is described through the options for the various viewpoints for an energy grid (IEEE, ODP - Open Distributed Processing). In our challenge approach this is consolidated into the major aspects determining the ICT architecture: the Role Model, the ICT Infrastructure and the Energy Infrastructure

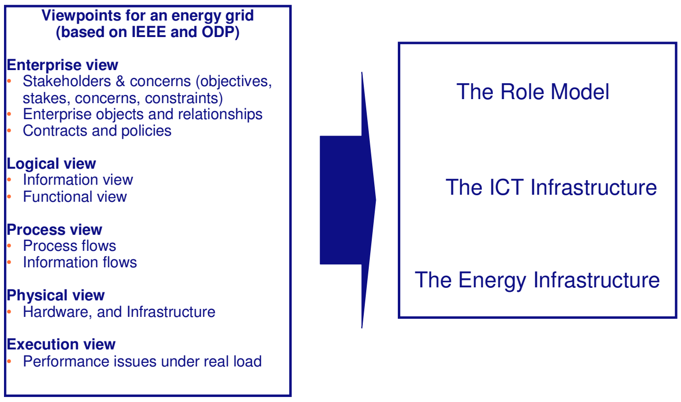

Viewpoints for an energy grid (based on IEEE and ODP)
Enterprise view
Stakeholders & concerns (objectives, stakes, concerns, constraints)
Enterprise objects and relationships
Contracts and policies
Logical view
Information view
Functional view
Process view
Process flows
Information flows
Physical view
Hardware, and Infrastructure
Execution view
Performance issues under real load
The Role Model
The ICT Infrastructure
The Energy Infrastructure

Figure 73: The methodology applied for the Smart Grid Islanding Challenge

Smart Grid Islanding is running part of the Power Grid in isolation during a major power failure somewhere higher in the grid. As a example an Apache Helicopter severed a major High Voltage power causing an 48 hour black-out for 50000 people and numerous businesses. There was quite a lot of capacity for local power generation but this could not be used as current rules and the current Power Grid cannot cope with this (safety and synchronisation issues). Running the area in so called Island mode could have enabled a limited operation in the area. E.g. important electric equipment such as central heating and telephones could have been used provide major energy users such as washing machines and electric water heaters would not be used. The Smart Grid Islanding case has come with a Sensor network based solution where the major sensors/actuator are Smart Meters and domotica.

5.2.1 The state the problem chain

5.2.1.1 Enterprise/context

Islanding is considered a (planned or unplanned) state-of-disturbance with local Energy Power Systems (EPS) energizing a portion of the Electric Power System that is electrically separated from the rest of the network

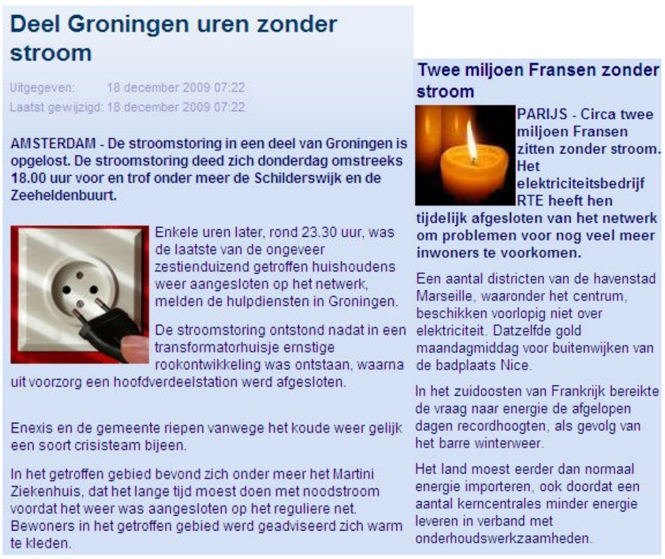
Normal Energy Operation is based on a balance between Energy Consumption, Energy Production and Grid Configuration (topology)

Sometimes Grid Configuration is disturbed
- Planned or Unplanned (Accidental)
- In order to avoid black-outs this may lead to the decision to form smaller operational subgrids: Islanding.

Islanding is defined in Institute of Electrical and Electronics Engineers (IEEE) Standard 1547 as:

> *a condition in which a portion of an Area Electric Power System (EPS) is energized solely by one or more Local EPSs through the associated point of common coupling (PCC) while that portion of the Area EPS is electrically separated from the rest of the Area EPS.*

The need for island mode operation with local EPSs is illustrated by recent examples of (planned and unplanned) disturbances.



Deel Groningen uren zonder stroom

Uitgegeven: 18 december 2009 07:22
Laatst gewijzigd: 18 december 2009 07:22

AMSTERDAM - De stroomstoring in een deel van Groningen is opgelost. De stroomstoring deed zich donderdag omstreeks 18.00 uur voor en trof onder meer de Schilderswijk en de Zeeheldenbuurt.

Enkele uren later, rond 23.30 uur, was de laatste van de ongeveer zestienduizend getroffen huishoudens weer aangesloten op het netwerk, melden de hulpdiensten in Groningen.

De stroomstoring ontstond nadat in een transformatorhuisje ernstige rookontwikkeling was ontstaan, waarna uit voorzorg een hoofdverdeelstation werd afgesloten.

Enexis en de gemeente riepen vanwege het koude weer gelijk een soort crisisteam bijeen.

In het getroffen gebied bevond zich onder meer het Martini Ziekenhuis, dat het lange tijd moest doen met noodstroom voordat het weer was aangesloten op het reguliere net. Bewoners in het getroffen gebied werd geadviseerd zich warm te kleden.

Twee miljoen Fransen zonder stroom

PARIJS - Circa twee miljoen Fransen zitten zonder stroom. Het elektriciteitsbedrijf RTE heeft hen tijdelijk afgesloten van het netwerk om problemen voor nog veel meer inwoners te voorkomen.

Een aantal districten van de havenstad Marseille, waaronder het centrum, beschikken voorlopig niet over elektriciteit. Datzelfde gold maandagmiddag voor buitenwijken van de badplaats Nice.

In het zuidoosten van Frankrijk bereikte de vraag naar energie de afgelopen dagen recordhoogten, als gevolg van het barre winterweer.

Het land moest eerder dan normaal energie importeren, ook doordat een aantal kerncentrales minder energie leveren in verband met onderhoudswerkzaamheden.

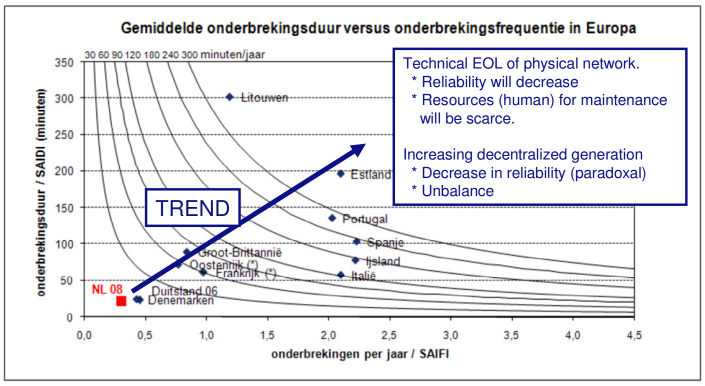
Figure 74: Examples where Smart Grid Islanding could have been of use

The current network in the Netherlands is very reliable with very little need for Islanding. However, there is a trend of more frequent and longer island mode operation due to some major changes in the network.

The changes are:
- A significant increase in local power generation. The current network has not been designed for this. It requires power management on the Medium Voltage and Low Voltage networks. Up to now switching is done in the higher layers of the power grid

- Increase in the use of electric cars. Electric Cars and Hybrid Cars currently have a battery of up to 50 kWh. Charging will change the energy consumption pattern. Furthermore the 50kWh is about what a household currently uses in 5 days.
- Increase in heavy appliances like air conditioning equipment.

The result is that the reliability of the network may degrade and islanding is a way to cope with network failures and contain the damage caused by these.



Figuur 2.8 Onderbrekingsduur versus onderbrekingsfrequentie in Europa, CEER 2008

Figure 75: Statistics on Power Failures. The Netherland currently rank good, but changes in the use of the power network might increase the risk for more and longer power outages..

The current grid is not suitable for Islanding operations. Furthermore if Islanding were possible the Islanding operation and resynchronisation have to go through a period of "Black" (powerless period) for safety and quality reasons.

Can ICT combined with sensor/control network technology facilitate the Islanding Operation with minimal impact on the electricity consumers?

5.2.1.2 Functional

The challenge is to describe the ICT for the various states of the islanding cycle for an energy grid: *a condition in which a portion of an Area Electric Power System (EPS) is energized solely by one or more Local EPSs through the associated point of common coupling (PCC) while that portion of the Area EPS is electrically separated from the rest of the Area EPS.*

5.2.1.3 Process

The challenge is to describe the basic process flow for islanding and the information that has to be exchanged between the different components of the system as it goes through the islanding cycle:

A. Island Initiation (Black-out, from normal to islands)
B. Island Operation
C. Syncing (White-in, from islands to normal)

There are some important safety driven restrictions on the islanding and reconnection process. Currently when the power supply fails all local power generators are switched off. The main reason is that engineers expect the network to be without power. If a local generator would continue to work there would be power on the network and could result in an electric shock for the engineers working on the network. There are also technical restrictions as local power generation is synchronised to 50Hz using the network itself. Without power on the network the inverters of the local power generator cannot be synchronised and no alternating current can be produced. This technical issue, however, can be overcome. Reconnecting the Island current goes through a period of black (all local generators switched off) only after the main power has been restored local power generators can be switched on and be resynchronised before reconnecting to the grid.

5.2.1.4 Physical

Physical requirements and constraints

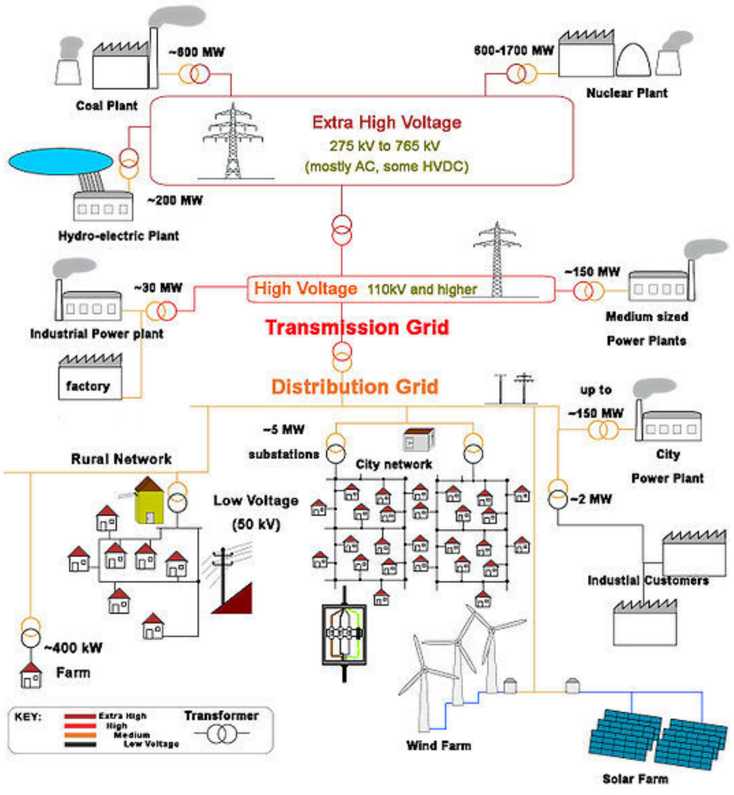


Figure 76: An schematic representation of the USA Power Grid

Energy producers, consumers, buffers and switches can be distinguished at the various layers of the energy grid.

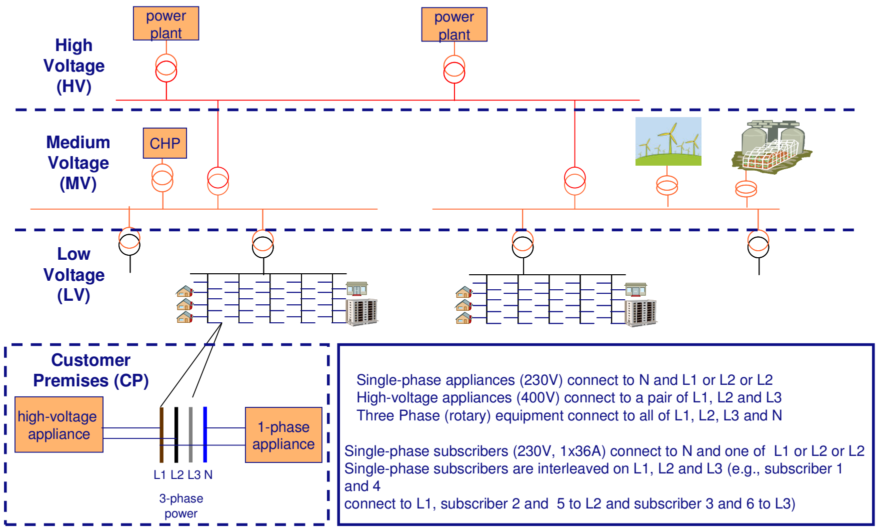

power plant
power plant
High Voltage (HV)
Medium Voltage (MV)
CHP
Low Voltage (LV)
Customer Premises (CP)
high-voltage appliance
1-phase appliance
L1 L2 L3 N
3-phase power
Single-phase appliances (230V) connect to N and L1 or L2 or L2
High-voltage appliances (400V) connect to a pair of L1, L2 and L3
Three Phase (rotary) equipment connect to all of L1, L2, L3 and N
Single-phase subscribers (230V, 1x36A) connect to N and one of L1 or L2 or L2
Single-phase subscribers are interleaved on L1, L2 and L3 (e.g., subscriber 1 and 4 connect to L1, subscriber 2 and 5 to L2 and subscriber 3 and 6 to L3)

Figure 77: Elements of the Power Grid

The end-2-end chain from the main power plants up to the appliances at home is shown below.

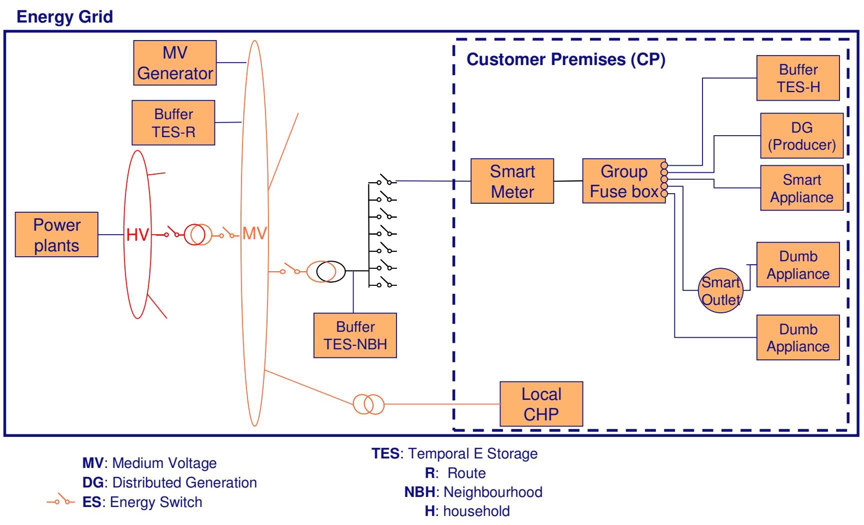
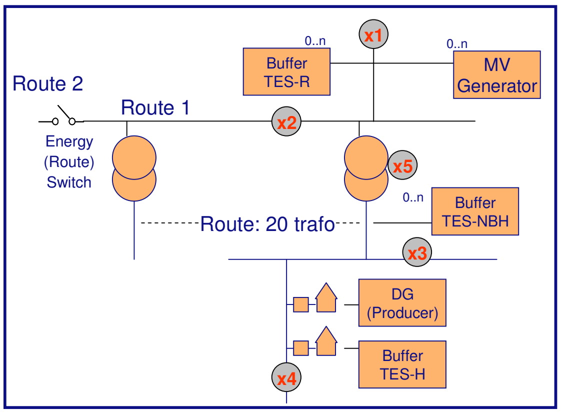


Figure 78: The end-2end chains in the Power Grid

Note that households like all subscribers can have a three phase connection. This means that appliances in the house will be spread across the three phases. Hence a µCHP may not be able to power all appliances in the house during an islanding state. In general each group in the fuse box will be connected to one of the three phases and all the appliances connected to this group.

5.2.1.5 *Execution*

A realistic scenario for the energy grid (topology, producers and consumers) for the year 2020 is depicted below



Island Parameters

Topology:
- Households: 250-5000
- Businesses: 0-5

Energy generation:
- $P_{max,}$ µCHP: 2 kWe
- $P_{max,}$ PV: 3 kWe peak
- P_{max} CHP: 10 kWe
- P_{max} Wind: 250 kWe

Energy Consumption:
- $P_{max, hh:}$ 5 kW
- Daily Household: $E_{daily, hh}$: 10kWh

Energy Buffering:
- TES PV C = 10kWh $P_{max,}$ 3 kW
- TES EV C = 50kWh P_{max} 20 kW

20 Transformers on a Route
Average of 250 Households below Transformer

Energy (Route) Switch: normally open, possible connection to other Route

x1: Island	~5000 households affected
x2: Island	~250-4500 households affected
x5: Island	~250 households affected
x3: Island	~50-250 households affected
x4: Island	<50 households affected

Figure 79: Basic statistics on the Power Grid

The horizon is set at 2020. For 2020 4 scenarios have been developed on penetration of local power generation like µCHPs and availability of local buffering.

Scenario	HH w. µCHP	HH w. PV	HH w. EV	HH cons	P max Generated	Duration	Buffer used
				Watt	% of cons	Hour	
1	10%	0%	0%	800	25%	0	no
2	40%	0%	0%	800	100%	∞	no
3	20%	10%	0%	800	84%	6	yes
4	20%	10%	5%	800	84%	21	yes

Table 15: Scenario's for 2020 on penetration of local power generation and buffer capacity.

Observations
- In scenario 1, the consumption needs to be reduced 4 times to be able to start the Island
- With 40% µCHP, the Island could be powering itself indefinitely if no more power than 800W*)[8] per HH is consumed
- With 20% µCHP (realistic ass. Cogen), and only 10% PV an Island could maintain the 800W for 6 hrs. 84% of needed power can be generated, rest from buffers (assumed full)
- With only 5% connnected EV buffer added, 21 hrs on 800W average could be maintained.

[8] 800 W is based on total E consumption of 10kWh over 12 hrs averaged

A lot can be said of these numbers, but one could state that as long as high peaks in consumption can be avoided (fridge, washers), the island could be self supporting long enough for most outages (<4 hrs) with realistic generators and buffers in the Island. Assuming the µCHP can operate this long.

Basic calculations on (realistic) production / consumption scenarios shows that islands can be self supporting long enough for most outages.

5.2.1.6 Development

The main timing requirement has been that the system should be fully operational by 2020. The development should allow a phased implementation where scaling effect can be fully tested. The development should allow ample time for political and social discussions on the energy allocation rules during emergency situations where Islanding is applied.

5.2.2 The Enterprise/top level view

5.2.2.1 Alternatives

Planned Island Mode

Goal: Maintenance on X1 point of Island

Actors: **DSO**, Prosumer, SME, Island Operator

Preconditions
- Island predefined
- Island consumes power on X1
- Island has 20% µCHP, 10%PV
- 1 SME is connected
- 1 CHP available on MV

Postconditions
- Island operational, SME can do business, fair power scheduling for prosumers, X1 can be serviced

Scenario
1. DSO requests Island operator to create Island
2. Island Operator initiates power balancing in Island
3. Island Operator signals DSO that Island is ready
4. DSO assures island requires no power on point X1
5. DSO disconnects X1
6. Island Operator Manages Energy Balance in Island

Unplanned Island Mode

Goal: Bootstrap Island after calamity on point X2

Actors: DSO, Prosumer, **Island Operator**

Preconditions
- Island predefined
- Island consumes power on X2
- Island has 40% µCHP, 10%PV

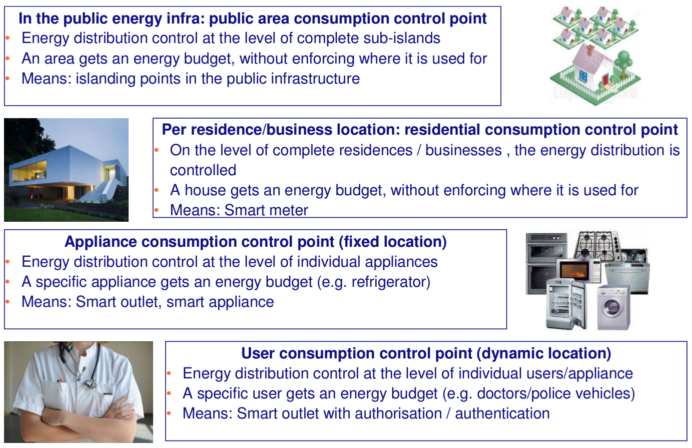
Postconditions
- Island operational, fair power scheduling for stakeholders

Scenario
1. Island Operator receives signal Island is blacked-out
2. Island Operator initiates power scheduling in Island
3. Island Operator Manages Energy Balance in Island
4. Island Operator receives signal Island can be reconnected

Figure 80: Two scenario’s for Islanding, planned (e.g. maintenance) and unplanned (e.g. failure)

For our approach a Planned and an Unplanned Island Mode scenario are distinguished, each with its own use case

Energy Consumption Control Options: Matching demand for Island Mode can (theoretically) be done at various types of “control points”



In the public energy infra: public area consumption control point
- Energy distribution control at the level of complete sub-islands
- An area gets an energy budget, without enforcing where it is used for
- Means: islanding points in the public infrastructure

Per residence/business location: residential consumption control point
- On the level of complete residences / businesses , the energy distribution is controlled
- A house gets an energy budget, without enforcing where it is used for
- Means: Smart meter

Appliance consumption control point (fixed location)
- Energy distribution control at the level of individual appliances
- A specific appliance gets an energy budget (e.g. refrigerator)
- Means: Smart outlet, smart appliance

User consumption control point (dynamic location)
- Energy distribution control at the level of individual users/appliance
- A specific user gets an energy budget (e.g. doctors/police vehicles)
- Means: Smart outlet with authorisation / authentication

Figure 81: Power consumption control points

Energy Consumption Allocation Options: These are expressed by means of three priority levels: “Urgent”, “Fair”, “Market”

Demand priority level 1: “Urgent”
- Social necessity / critical
- Determined by the government
- E.g.: hospital, financials, transport

Demand priority level 2: “Fair”
- Fair usage, distributed amongst households / businesses on the basis of fair policy (e.g. equal shares)
- Determined by the government

Demand priority level 3: “Market”
- Surplus, market-based provisioning
- Based on price-negotiation

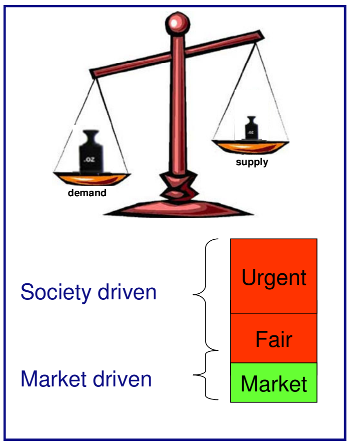
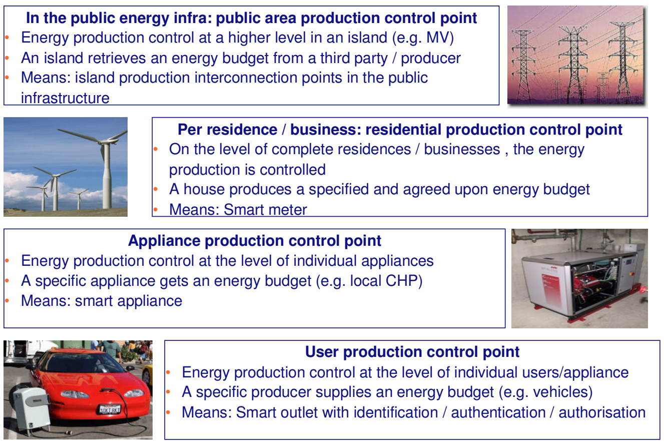


Figure 82: power distribution priorities

Energy Production Control Options: Matching demand for island mode can (theoretically) be done at various types of "control points"



In the public energy infra: public area production control point
- Energy production control at a higher level in an island (e.g. MV)
- An island retrieves an energy budget from a third party / producer
- Means: island production interconnection points in the public infrastructure

Per residence / business: residential production control point
- On the level of complete residences / businesses , the energy production is controlled
- A house produces a specified and agreed upon energy budget
- Means: Smart meter

Appliance production control point
- Energy production control at the level of individual appliances
- A specific appliance gets an energy budget (e.g. local CHP)
- Means: smart appliance

User production control point
- Energy production control at the level of individual users/appliance
- A specific producer supplies an energy budget (e.g. vehicles)
- Means: Smart outlet with identification / authentication / authorisation

Figure 83: Power production control points

Energy Production Allocation Options: These are expressed by means of three priority levels : "Urgent", "Fair", "Market"

Supply priority level 1: "Guaranteed supply"
- Fixed contract for constant (minimal) supply
- Supply minimal guaranteed level of energy
- Determined by the government

Level 2: "Fair"
- Mandated supply of over-capacity based on fair policy, e.g. the excess capacity on top of the minimal-demand need
- Determined by the government

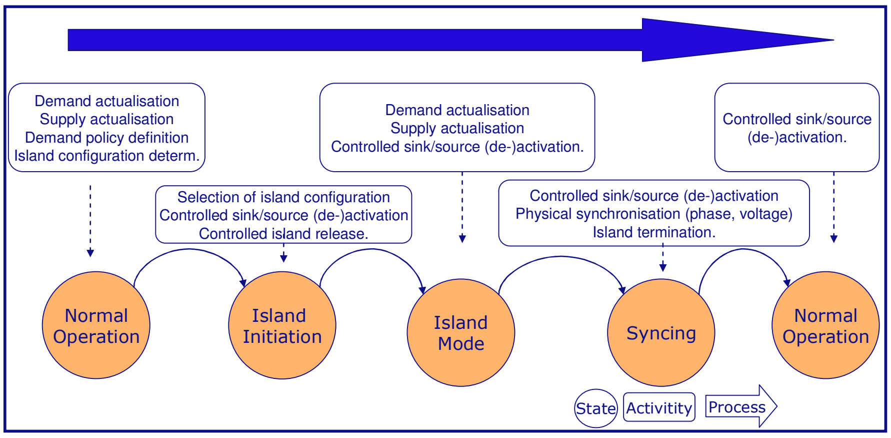
Level 3: "Market"

– Surplus, market-based provisioning
– Based on price negotiation

5.2.2.2 Model the system

ISN (ODP) Enterprise Model, SMART targets.

In the process of going into or out of Island Mode, the energy grid can be in several "states" each with its specific activities to be executed


Demand actualisation
Supply actualisation
Demand policy definition
Island configuration determ.
Demand actualisation
Supply actualisation
Controlled sink/source (de-)activation.
Controlled sink/source (de-)activation.
Selection of island configuration
Controlled sink/source (de-)activation
Controlled island release.
Controlled sink/source (de-)activation
Physical synchronisation (phase, voltage)
Island termination.
Normal Operation
Island Initiation
Island Mode
Syncing
Normal Operation
State
Activitity
Process

Note: The activities are similar for both the planned and the unplanned use case
Main difference: In the state "Island Initiation" the activity "controlled island release" is not feasible

Figure 84: The Islanding process

The ICT infrastructure and the information requirements depend on both the Target (energy demand/supply) Options (the "What") and the Implementation Options (the "How")

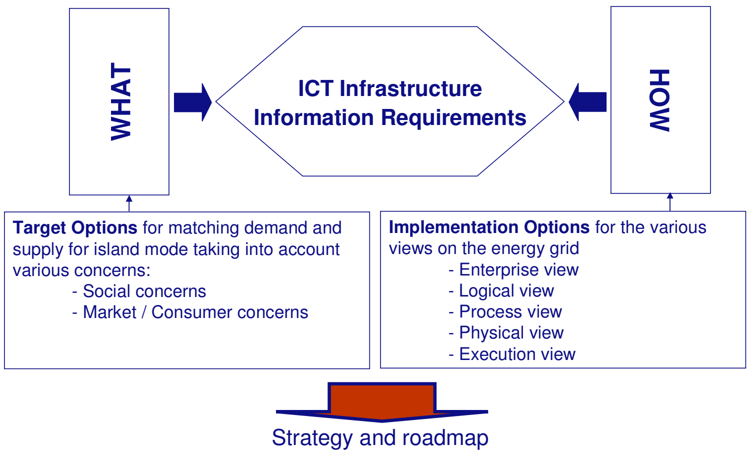

WHAT
ICT Infrastructure
Information Requirements
HOW
Target Options for matching demand and supply for island mode taking into account various concerns:
- Social concerns
- Market / Consumer concerns
Implementation Options for the various views on the energy grid
- Enterprise view
- Logical view
- Process view
- Physical view
- Execution view
Strategy and roadmap

Figure 85: What and How drive the system design

For the “What”, we distinguish between control and allocation options for matching production and consumption for Island Mode. These are evaluated on socio-economic criteria, rather than technological criteria

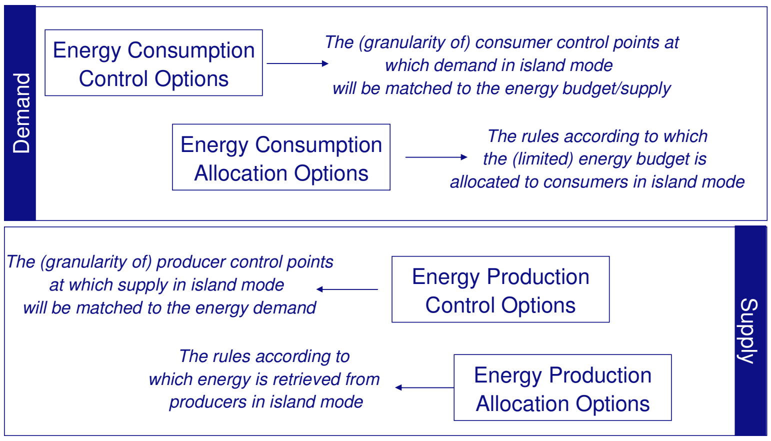

Demand
Energy Consumption Control Options
The (granularity of) consumer control points at which demand in island mode will be matched to the energy budget/supply
Energy Consumption Allocation Options
The rules according to which the (limited) energy budget is allocated to consumers in island mode
The (granularity of) producer control points at which supply in island mode will be matched to the energy demand
Energy Production Control Options
The rules according to which energy is retrieved from producers in island mode
Energy Production Allocation Options
Supply

Figure 86: Demand and Supply

The various control and allocation options for production and consumption accumulate into a multitude of Target Options for Island Mode. A strategy is required.

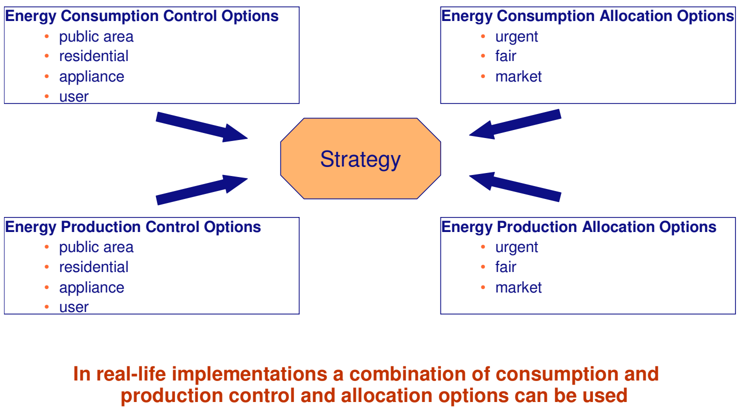

Energy Consumption Control Options
public area
residential
appliance
user
Energy Consumption Allocation Options
urgent
fair
market
Strategy
Energy Production Control Options
public area
residential
appliance
user
Energy Production Allocation Options
urgent
fair
market
In real-life implementations a combination of consumption and production control and allocation options can be used

Figure 87: Combination of production and consumption options

5.2.2.3 Integrate

Integration on the enterprise level mainly involves the role model of the players/stakeholders in the power grid. During normal operations the interaction between the roles involve operation of the network and operation of the commercial aspects of energy supply and demand. These have been recorded in contracts.

Role model: Roles and responsibilities will differ between normal and island operation. In island mode, a (governmental) regulated model has to be adapted. The process and ICT infrastructure have to follow / comply.

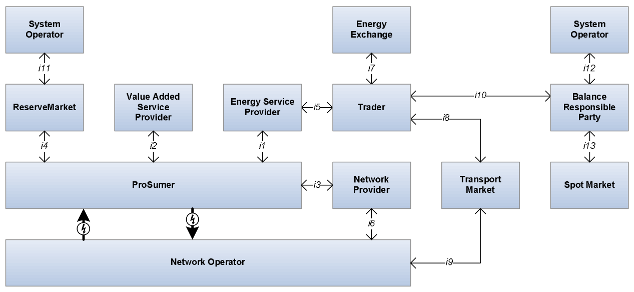


Figure 88: The Power World role model

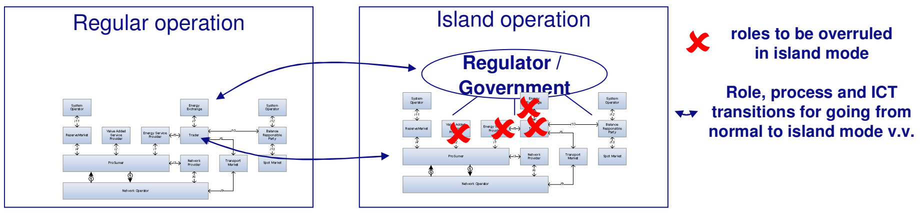
Investigate Alternatives Enterprise view

– In a liberalised energy market many different roles co-exist in the value chain. The distribution of responsibilities among roles may be different between regular operations and island operations. Both modes of operation will require their own supply chain, whilst both have to interact in case of switching between both modes.

– Island mode represents a situation of public / social importance. Therefore, a central role with overall authority on allocating energy budgets has to be implemented, acting on behalf of the government / regulator / society. It requires the functional capability to overrule all energy allocation mechanisms and bilateral interrelationships as deployed by individual service providers, consumers and producers in normal mode. Maintaining privacy imposes an important boundary condition. Probably, financial compensation between parties and/or government must be calculated based on a defined policy and actual happenings.

– More elaboration on the supply chain and their ICT aspects is required, e.g. on the transition between modes of operation.

During Islanding the normal commercial contracts are overruled by social/politically rules how distribute a limited amount of energy fairly.


Regular operation
Island operation
Regulator / Government
roles to be overruled in island mode
Role, process and ICT transitions for going from normal to island mode v.v.
Figure 89: During islanding the role model changes.

Role model: (New) role models have to be implemented for both the normal and island mode. Roles are defined by means of their interfaces and relationships. Rules for implementing the role model apply. The process and ICT infrastructure have to follow / comply.

o The role model represents independent roles

- o Roles are identified by means of their (wholesale) service portfolio
- o The architectures are designed as if roles were independent organisations
- o Although they may be part of same company

Separation of processes and IT per role
- o Enables (open) interface to other organisations
- o Enables fulfilling the roles with maximum flexibility and minimum effort

Separation of (ICT) equipment per role:
- enables unambiguous allocation of (management) responsibility for equipment
- flexibility at additional costs
- Commercial relations
- They describe the agreements on the provisioning and the support of the services
- They contain the conditions, the quality and the payment for the services

Operational relations:
- They represent the real usage and support of the services
- They take place on the physical interface between the roles
- They will involve interactions with respect to:
  - Regular usage (Continuous provisioning)
  - Operational support:
    - Fulfillment (Initial provisioning)
    - Billing (Accounting)
    - Assurance (Service & Maintenance)

5.2.3 The Investigate Alternative chain
Next to the business case scenario's the main alternatives are:
- o The overall design pattern (distributed versus centralised, hierarchical versus peer-peer)
- o The information exchanged
- o The communication network
- o The choices for sensing and control in the Smart Grid;

5.2.3.1 Functional
The alternatives were JDL, NAIHS and 4D/RCS. For this study 4D/RCS has been selected as the main pattern. Within the 4D/RCS the JDL/NAIHS layering is used.

5.2.3.2 Process

Scenario considerations for Island configuration
Pre-defined vs on-demand determined energy balance
- o Pre-defined: At all times, a pre-determined configuration for island mode operation is available. Activation of island mode configuration is triggered by an island mode event (e.g. outage or planned). Island mode is seen as a specific "state" in the regular process, "not as an exception process".
- o On-demand determined. In case an island mode event (e.g. outage or planned) occurs, the process are initiated for demand/supply actualisation and scheduling. The energy balance has to be determined and possibly dynamically adjusted.

Consideration:
- o The following arguments are in favour for "pre-defined":

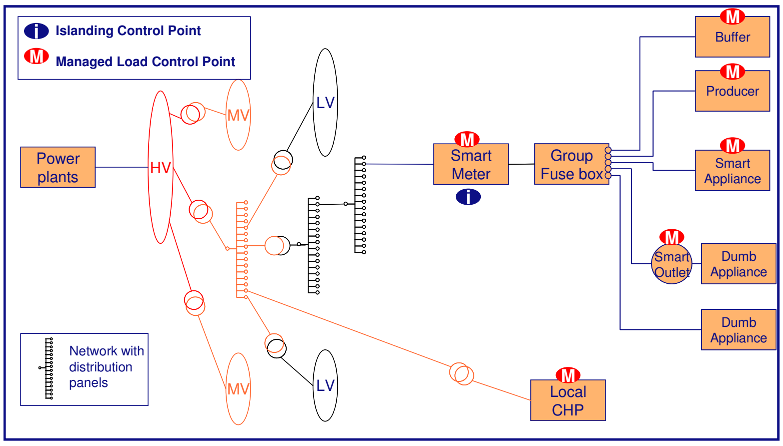
  - o Cost savings in EMS equipment (demand/supply actualisation and scheduling can be done in "batch" instead of at event-time (with quick response required))
  - o Saving time to go into island-mode
  - o Continuously testing of island-mode operation
- o The following arguments are in favour for "on-demand":
  - o Cost saving: less communication and CPU required: Only relevant configurations are determined
  - o better adaptable to actual circumstances

In view of these considerations, the "pre-defined" scenario seem to be the most attractive.

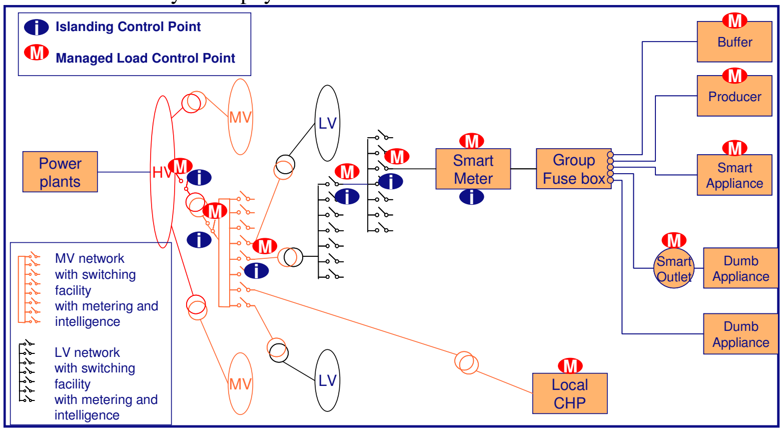
Hierarchically controlled versus peer-to-peer energy management:
- o Hierarchically controlled: a central role has the authority to define and enforce energy balance configurations in case of island mode
- o Peer-to-peer: producers and consumers can make agreements on a bi-lateral basis. This is mainly an administrative / accounting relationship, not a relationship at the transport level (for which the shared, public, infrastructure is used).

Consideration:
- o In normal operation, a peer-to-peer mechanism is allowable. However, island mode represents a situation of public / social importance. In such a situation, peer-to-peer agreements should be overruled. Hence, a hierarchically controlled energy management approach is required. A weaker version of this could be to impose strict rules on allowable agreements for the duration of the island mode ("you can agree as you like, but together you have to achieve at least a production of N kW") with a fallback to full overruling if no agreement is reached.

5.2.3.3 Physical
Energy Infrastructure: In a non-smart grid with only smart meters islanding can only occur on a subscriber level using the Smart Meter switching capabilities. The main control point where power usage can be controlled is on the subscriber premises.



Energy Infrastructure: In a Smart Grid with (remote) switching on all levels of the Smart Grid there are many more Islanding options. Islands can be made on each level, and sections of the same level can be isolated. With intelligence on each of these levels, there are also many more physical


Islanding Control Point
Managed Load Control Point
Power plants
HV
MV
LV
MV
LV
MV network with switching facility with metering and intelligence
LV network with switching facility with metering and intelligence
Smart Meter
Group Fuse box
Buffer
Producer
Smart Appliance
Smart Outlet
Dumb Appliance
Dumb Appliance
Local CHP

5.2.3.4 *Execution*

The horizon is set at 2020. For 2020 4 scenario's have been developed on penetration of local power generation like µCHPs and availability of local buffering.

Scenario	HH w. µCHP	HH w. PV	HH w. EV	HH cons	P max Generated	Duration	Buffer used
				Watt	% of cons	Hour	
1	10%	0%	0%	800	25%	0	no
2	40%	0%	0%	800	100%	∞	no
3	20%	10%	0%	800	84%	6	yes
4	20%	10%	5%	800	84%	21	yes

Table 16: Scenario's for 2020 on penetration of local power generation and buffer capacity.

5.2.3.5 *Development*

Development platform choices, development methodology (RUP, waterfall, etc)

5.2.4 *The functional chain*

5.2.4.1 *Model the system*

Logical View: functional view

The system can be decomposed in the following functional objects:

- Network/grid switch (HV/MV) and (MV/LV)
- MV power lines
- LV power lines
- Distribution/patch panels
- Subscriber
    - Consumer

    - Producer
    - Prosumer
- HV network
- HV/MV transformer
- LV network
- Meter
- Main switch
- Main breakers
- Fuse box (breaker box)
- domestic network
- Power Outlets

On the Domestic/subscriber level we can see the following functional components:

- Breaker
- Throttle
- Smart Meter
- Domestic control unit
- Synchroniser
- Consumers
- Producers
- Buffers
- Sensors
    - Outlet
    - Smart Consumers
    - Smart Producers
- Actuators (status or power)
    - Outlet
    - Consumers
    - Producers

In more detail we can distinguish the following in the “public” network:

- Network/grid switch/control unit
    - Upstream breaker (remote control)
    - Downstream breakers (remote control)
    - Control unit
    - Upstream meter
    - Downstream meters
    - Synchroniser
    - UPS
- Passive Network connection unit
    - Upstream meter
    - Downstream meter
    - Communication unit
    - Optional manual patch capability
- Dumb connection unit
    - Optional manual patch facilities

Consumers and producers do exists in different types:

- Consumer
    - Smart consumer

        - Remote controllable switch
        - Meter
        - Control/communication unit
        - Control panel
    - Power outlet controlled consumer
        - Controllable power outlet (self learning)
        - Meter
    - Uncontrolled consumer
        - direct connection into power
        - Manual dimmer(-switch)
        - Max usage can be registered manually

- Producer
    - DC producer
        - Power generation
        - Inverter (single phase)
        - Synchroniser (optional)
        - meter
        - throttle/shed
        - Control unit
        - Controllable breaker
        - 3-phase AC
            - Control unit
            - Synchroniser
            - Metering

For ISN these are functional units that can each have some form of intelligence linked to it. For this exercise it was decided to stick as close as possible to the real Power World components. Hence the for the ICT – Infrastructure the ICT for supporting Islanding (and Load Balancing) is based on a set of basic requirements on "layering" the physical topology and the communication infrastructure

Layering
- Layering representing logical hierarchy
- Layers: Network, H, MV, LV, Neighbourhood, Street, Subscriber, Home
- Layers do not cross organisational boundaries
- Each logical layer aggregates demand and supply from next lower layers and responds with allocation
- Each layer is capable of managing the layer below (without continuous support from above)
- Business rules representing the policy for allocation of available energy cascaded to each layer
- Distributed control (mapped on physical hierarchy)

Communication infrastructure
- A robust communication network that will continue function with(in) an island even through potential black-out periods
- A standardised application protocol for communication demand and supply and also capable of supporting commercial negotiating aspects of smart grids and supporting communication with the domestic domotica controller

The 4-D RCS system, when adapted for Smart Grids meets these requirements. This system has to be functionally decomposed and "mapped" on the energy grid role model. The process and ICT infrastructure have to follow / comply.

4D-RCS fits well as it is an cognitive real-time monitoring and control system that supports the network hierarchy and the energy role model.

For this challenge we have adapted a version suitable for Smart Grid control.

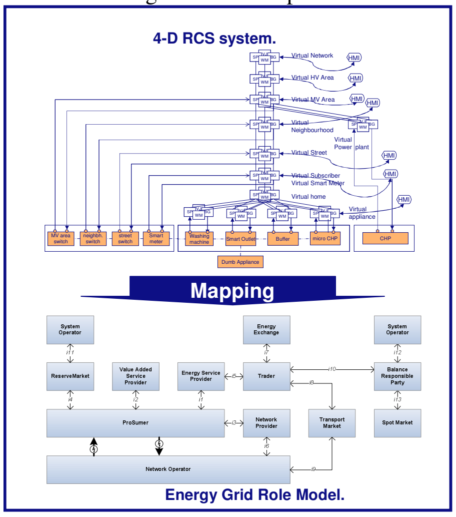


Figure 90: mapping 4D/RCS on the Role model

The 4D-RCS system is a modern implementation of the 30 year old RCS (Real Time Control) architecture of NIST. There are versions for:

- Autonomous army vehicles
- Manufacturing systems
- Space

There are logical control nodes for each Smart Grid layer each with sensory processing, a local world model, a business rule engine and a control section. Each control node can communicate with parents, peers and children.

Below is a description of 4D/RCS and how it was adapted for Smart Grid Islanding.

RCS is based on a Real world and a virtual representation of the world in the World Model. The worlds are connected by sensors and actuators in the real world that are linked to the virtual world through Perception and behaviour generating software. The behaviour is also drive by "the user" (in this case the military) by feeding it with a Mission Goal. For Smart Grid Islanding, Smart Grids Island operation mode is the goal in emergency situations. A normal commercially driven goal applies to the normal situation.

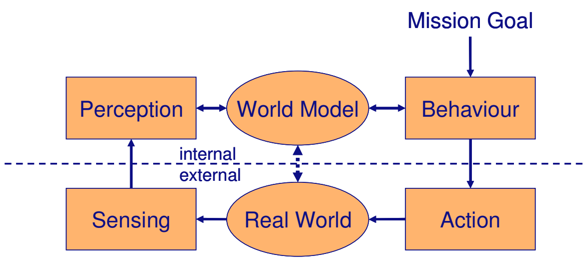


Figure 91: The RCS Real Time Control Model

In more detail The RCS components have the following functional decomposition.

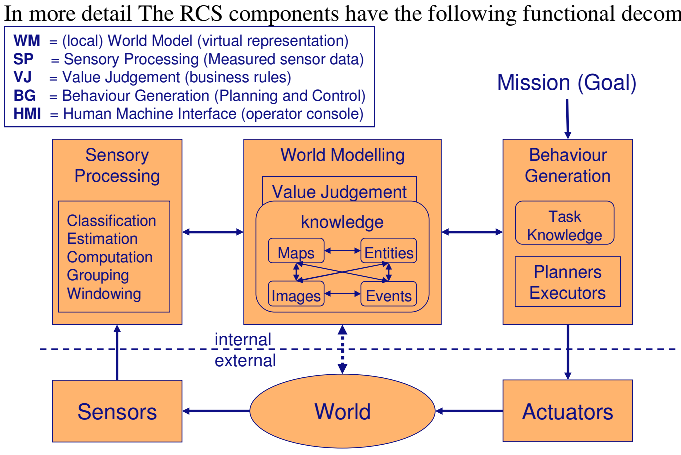


Figure 92: Functional decomposition of the RCS system

4D/RCS is a version of RCS for unmanned army vehicles operating on a (battle) field as part of an Army Battalion. The 4D/RCS reference model breaks down the control system into a hierarchy of controlling layers. Somewhere in the middle is the vehicle itself. Below this are logical layers for sub-systems, primitives (or components) and servo's. The bottom layer concerns non-intelligent sensors and actuators. Above this are logical layers representing station, platoon, company, and battalion level. The 4D relates to the 3 spatial dimensions and one temporal one. The lower layers have a rapid control cycle covering a small geographical area. The higher layers cover larger geographical areas a a slower control rate.

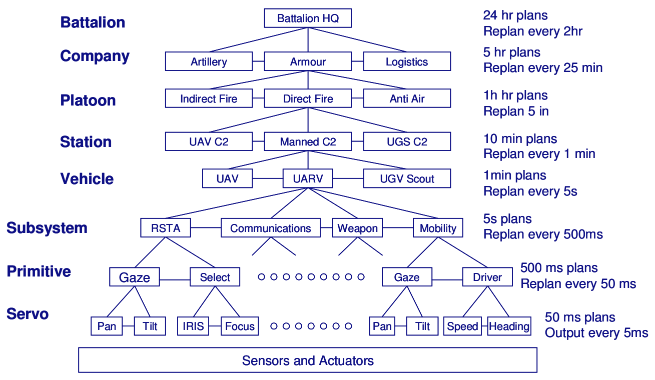


Figure 93: An hierarchical breakdown of 4D/RCS mapped on the army command structure and the control structure of vehicles.

There are 4D/RCS varieties for manufacturing and space robots. Hence it is useful to see if 4D/RCS is applicable to the Smart grid and Islanding.

There are similarities and difference between the unmanned army vehicle and similarities. The similarity is that there is a highly hierarchical organisation as well in the (Smart) Grid. The main difference between 4D RCS for unmanned vehicles and 4D-RCS for smart grids is that in Smart Grids the higher layers can actually do switching. I.e. there is physical switching in the hierarchy. This is not the case with the unmanned vehicles. The actual switching is only done by the bottom layer, the high layer do their control via the lower layers. There is also the difference between civilian applications and military ones.

Applied to Smart Grids the 4-D RCS layers/Hierarchy could look like this.

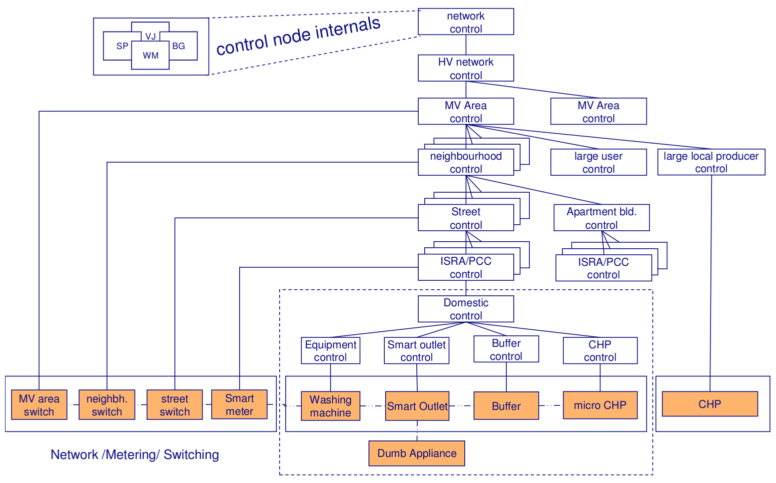


Figure 94: The 4D/RCS hierarchy adapted for Smart Grid Islanding

Each control node has a similar internal structure. For the Smart Grid this internal control node functionality can be described as below.

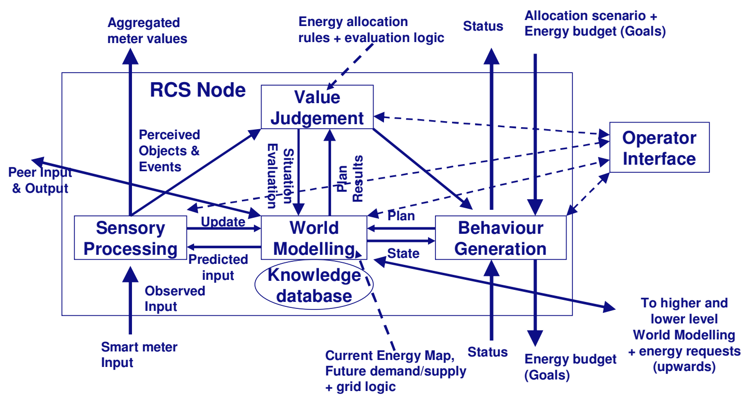


Figure 95: The Internal structure of a 4D/RCS node with information flows between modules.

The essence is that each control node has a virtual representation of the world as relevant on that layer and a rule set in the Value Judgement module to evaluate the value of the plans. For Islanding the Value Judgement module would contain the business rules for allocating energy budgets and an evaluation algorithm to calculate the value of these budgets. Business rules can be simple commercial ones in a normal situation or more socially driven rules based on urgency, fairness and some commercial rules for the remaining budget if any. The World Model would have a view of the current demand/supply situation as fed by sensors and by budget requests. Furthermore it would have historical information and information on future energy production and demands. It would typically have a view over its direct children but an aggregated view on the grand-children. The control node can reside next to the control points (physically highly distributed or in the other extreme as one big central systems with the control node logic processing physically on the central system but logically still distributed. IT is also possible to combine some of the middle layer on logical boundaries such as organisational/role boundaries.

With all the communication in place the 4D/RCS system for the Smart Grid would look like the diagram below.

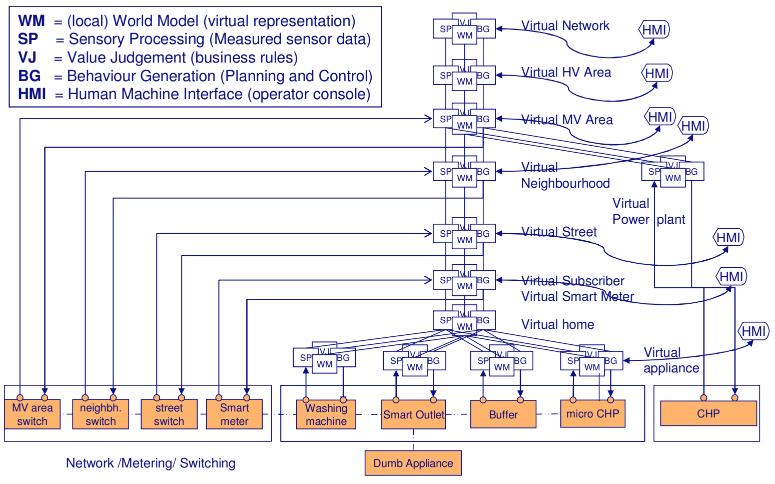


Figure 96: The 4D/RCS system for Smart Grid Islanding

The information in each control node can be:

- (Semi)-Static information
  - Policies and contracts (for normal and for emergency situations)
  - List of subscribers and priorities (or list of "urgent" subscribers)
    - priority
    - energy default profile
  - Network map with status of all relevant nodes
- Dynamic Information
  - Energy Map
    - (node/aggregation level, priority/class, usage, t (history, current, predicted))
  - Energy demand plans
    - (node/aggregation level, priority/class, usage, t (history, current, predicted))
  - Energy supply plans
    - (node/aggregation level, priority/class, usage, t (history, current, predicted))

Modelling/value judgement/algorithms
There are basically two modes: Normal and emergency:

- In the normal mode the normal economical and contractual rules apply.
- In emergency there are overruling policies which allocate energy according the (Urgent, fair, market) algorithms.

Value judgement in these situations is different. (Albeit there are ways of merging these scenario's again into a single parameterised one handling both situations)

For evaluating the solution it is useful to have measures for cost and value not only financially but also socially.

A plan is and energy map with energy production and consumption allocation for current and future time slots. Plans can be constructed using optimising algorithms either with a limited number of iteration or going for the optimal solution (e.g. Simplex methodology)

Each level can do a full optimisation or can only optimise aggregate values.

The diagram below is the information and information-flow 4D/RCS diagram for Smart Grids and Islanding.

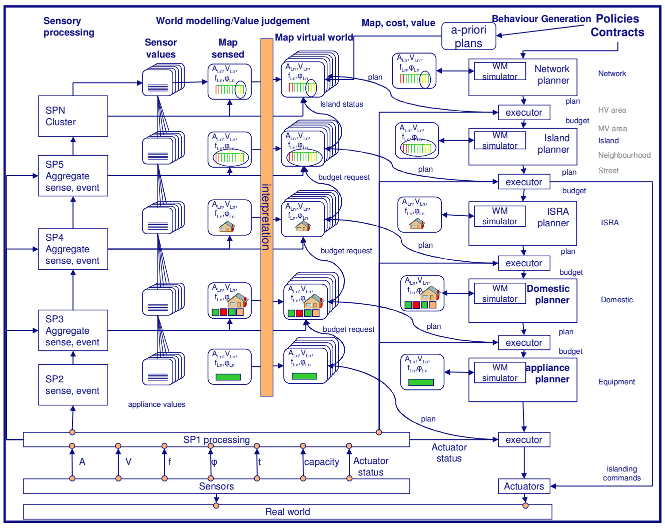


Figure 97: 4D/RCS for Smart Grid Islanding: Information View

5.2.4.2 *Integrate*

This is very similar to the model the system for the Process Chain.

5.2.5 ***The Process chain***

5.2.5.1 *Model the system*

The process of Island Mode operation is considered as a “specific instance” of the regular process (high-level).

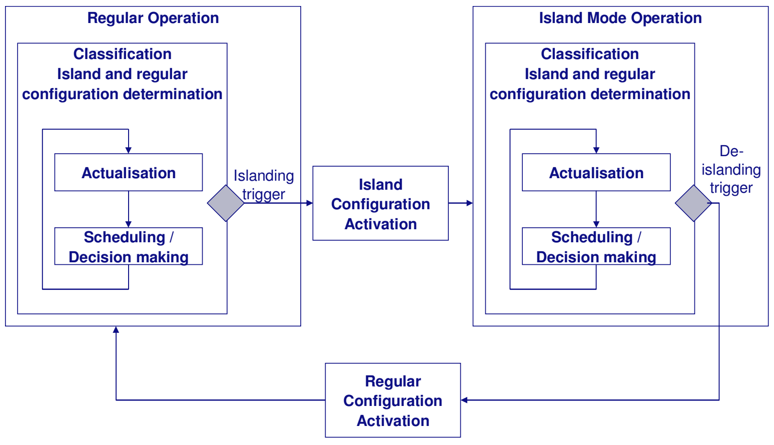


Figure 98: A process view on Smart Grid Islanding

Assumptions for the Island Mode process

(Preparing for) island mode operations is implemented as a regular, continuous, process

- At all times, a pre-determined configuration for island mode operation is available.
  - Activation of such an island mode configuration is triggered by an island mode event (e.g. outage or planned).
- Island mode is seen as a specific "state" in the regular process,
  - not as an "exception process"
  - but differs in operational parameters: contribution of energy from utility service is zero, some restrictions on allowed usage and command of local production will be active
- Hence, processes are similar for island mode and regular mode operation

At island initiation and operation, continuous communication with both demand and supply elements on profiles is possible.

- Before islanding: primarily based on utility service under normal operation
- During initiation: possibly on batteries
- In island-mode: on locally generated energy

Demand and supply profiles are continuously available, also during normal operation

- In island mode, only "delta's" are generated

The Classification Process leads to the determination of the island configuration

Scenario: Planned island initiation on t = t1

- Total area is pre-determined
- Sub-area division is an option
- Hence: demand and supply at both the total area and subarea's must be balanced at t = t1
- This is valid per phase L1, L2, L3

Priority levels

- Pre-defined (demand and supply) priority levels exist
- Demand-side: urgent, fair, market
- Supply-side: most willing to supply, fair prices, market

Each (demand, supply) priority level has its specific requirement w.r.t.:

- The information exchanged between smart-meters and EMS (Energy Management System)
- Protocol used for negotiating and enforcing energy level configurations

In the Classification Process, three demand priority levels are distinguished

Level 1: "Urgent"

- Social necessity / critical
- Determined by the government
- E.g.: hospital, financials, transport
- Pre-defined: requires processes for:
  - registration and status assignment (also i.r.t. supply)
  - authentication at operation in island mode

Level 2: "Fair"

- Fair usage, equally (or according to an agreed "fair" profile) distributed amongst households / businesses

Level 3: "Market"

- Surplus, market-based provisioning
- Pre-defined: requires processes for price-negotiation

The Actualisation Sub Process

- Actualisation of demand: usage per entity – per class
  - Classification of urgency of demands – general situation
  - Privacy issue: to which level can information be autonomously gathered and stored.

- Actualisation of supply: local production per entity
  - Classification of willingness to supply – general situation
  - Privacy issue: to which level can information be autonomously gathered and stored.

- Definition of pre-conditions

The Scheduling Sub Process

- Determine energy generation scheme
  - Per supplier
  - Per phase

- Determine island area and island subarea topology

- Determine the amount of energy to be available per demand priority level
  - e.g.: Urgent: 60%, Fair: 25%, Market: 15%

- Determine energy allocation scheme
  - Per customer
  - Per phase
  - Per demand priority level

5.2.5.2 *Integrate*

Each priority level requires its specific type of inter protocol and information attributes to be provided by the smart meter.

	Priority level	Information attributes Smart Meter <-> EMS	Protocol type
D E	Urgent	#Amperes, Phase, Time Distribution	Subscription / Granting
M A	Fair	#Amperes, Device Energy Characteristiecs, Minimum requested level, Phase, Time Distribution	Distribution
N D	Market	#Amperes, Time-distribution, Acceptable price Profile	Negotiation
S U	Guaranteed supply		Contracting
P P	Fair		Distribution
L Y	Market		Negotiation

Table 17: Information to be communicated for the different priorities

Basically Each node in the system compiles the energy requirements of its children for the next periods and conveys them in aggregated form to its parent. The response is the available budget for the next periods. The node can then switch/inform its children based on the cascaded distribution scheme for islanding.

5.2.6 The Physical chain

There are a lot of physical aspects to be covered. For this challenge there was a detailed study on smart metering. The communication issues are covered by other projects such as the Domestic Gateway projects and will not be covered here.

The Physical Architecture required for Islanding and for Smart Grid operations pivots around the monitoring and switching capability of the network. In the end the majority of the switching is accomplished at the subscriber end. Inside the home switching is done by the inhabitants and by appliances linked to a domotica system. On the boundary with the Grid there is the Smart Meter. Islanding also involves switching at higher layers. The Smart meter also needs to communicate with other systems in the power grid. There are some options there for networking such as: Power Line Communication, Mobile Broadband, Telephony, ADSL, Cable etc. This discussion is very much the same as the discussion on the the Residential Gateway.

5.2.6.1 *Model the system*

Smart Meters

Current Smart Meter Design has limited capabilities.
Current Smart meter designs allow for:

- Switching. Remote control possible for switching off, switch on can only be done manually. (This is not technically mandated, but is a safety policy.)
- Metering on one end of the switch
- Programmable breaker. Maximum value of circuit breakers can be remotely controlled. When break trips through to overload, circuit can only be re-established manually. (Again safety policy.)
- Some communication (essentially, telemetry).

Hence the current smart meter design offers no facilities for synchronisation after islanding or for controlling an island.

Desired modes of operation for Smart Meter

- Smart Meter can be the control point for its own independent island, facilitating
  - Island initiation
  - Island Mode
  - Syncing
- Smart Meter can also act as a local control point within a larger island, answering to an upstream controller to facilitate
  - Sub-Island Mode

- The Smart Meter can communicate with controllers upstream and downstream to establish the policies it and the other controllers have to implement, possibly through negotiation. Downstream controllers include smart appliances, smart outlets, smart producers, and the dynamically adjustable fuse box. Communication is out-of-band, possibly using a different (non-powerline) medium.

(Sub-)Islanding operation is based on policies

- (Sub-)Island Mode means enforcing a given local energy policy by controlling subordinate energy producers and consumers in the presence of an external energy budget. In Island Mode this budget is zero (on the top-level), in Sub-Island Mode it might be non-zero (even negative if the island is to produce power for the grid).
- The policy may contain all elements identified before, such as Urgent, Fair and Market principles in any combination.
- Policies can be dynamically (re-)negotiated between Smart Meter and other controllers, however these negotiations are governed by a meta-policy that is in principle fixed unless explicitly changed by an act of will, e.g. by a regulator.
- A difference between the two modes is that in Island mode an electrical decoupling from the grid must occur. This is relevant during island initiation and syncing. In the ICT view operating in island mode is just a special case of operating in sub-island mode. This will be the focal point in the following slides.

Advanced Smart Meter conceptual model

- In addition to existing functionality, the Advanced Smart Meter contains
- flexible communication facilities, upstream and downstream
- a meta-policy that governs all policy negotiations. An example might be provisions for conflict resolution.
- intelligence to (re-)negotiate with the upstream controller a policy to implement

- intelligence to (re-)negotiate with downstream controllers (including the fuse box) policies they have to implement, consistent with its own policy
- sensors to verify compliance to agreed policies
- actuators to limit or cut off power in case of emergency or unrecoverable malfunction of the island
- a management interface to update the meta policy, operational parameters, etc.

Islanding and the Smart Meter

- Islanding (unplanned)
  - Smart Meter must decouple if outage occurs
  - If communication still functional, informs upstream controller over status
- Islanding (planned)
  - Smart Meter is informed by upstream controller
  - Smart Meter prepares islanding by controlling smart outlets and smart appliances (including power producers) so that no net power is consumed by the household
  - Smart Meter informs upstream controller that it is ready for islanding, and decouples upon receiving acknowledgement

Sub-island Mode and the Smart Meter

- In Sub-Island mode, Smart Meter must act as control point for island (household), instructs smart outlets and smart appliances based on local policy and respecting the agreed energy budget (zero if operating in island mode)
- If communication with upstream controllers functional, policy can be (re-)negotiated, otherwise “own wisdom” applies
- On command from upstream controller, starts preparing for de-islanding
- De-Islanding and the Smart Meter
- Goal is to reconnect to the grid without having to go through dark
- For this, electrical situation (voltage, power, phases) must be sufficiently similar “inside” and “outside”
- The task of the Smart Meter is to create this situation by controlling the smart appliances in the household, e.g., by inducing a phase drift in a µCHP
- Upon sensing that all necessary preconditions are fulfilled and after permission from upstream controller, Smart Meter recouples to the grid and normal operation resumes (possibly in the form of sub-island operation)

Advanced Smart meter and all smart metering switches

For the Advanced Smart Meter, the following would be useful for support of dynamic islanding

- Main upstream switch: status
- Downstream switches: status
- Meter:
- A,V, f, φ up-stream / phase
- A,V, f, φ down-stream /phase
- Throttling breakers: min, max, current A
- Parent, Children
- History
- Digital communication of sensor values
- Digital 50 Hz clock signal (either external through main power or self generated during islanding on that level)

Hence there is very little difference between the switches on the different levels in the network hierarchy!

5.2.6.2 Integrate

Integration involves the links between the Smart Meter and a Domotica controller and between Smart meter and the communication infrastructure supporting the Smart Grid. It requires extensive interaction with Domotica vendors and power network operators an regulator.

5.2.7 The Execution Chain

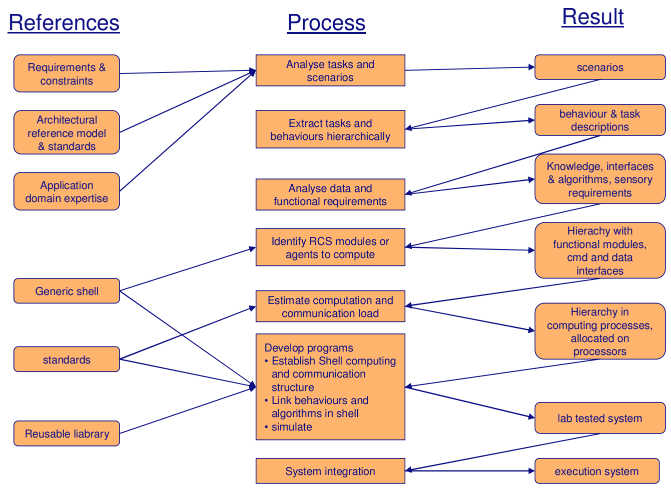
The execution view was outside the scope of the project. However a properly designed system with intelligence close to all important control points will enable scalability. There are also specific execution view aspects considering resynchronisation of an island with the main grid.

5.2.8 The Project Management chain

For this exercise the 4D/RCS methodology has been included.

5.2.8.1 Model the system

Design of development environment, detailed project plans
Develop test scripts
Typically the 4D/RCS design methodology is slightly different from the ISN design one described in this document. It seems to be better suited for a Intelligent sensor network but lacks the finesses of the ISN methodology.


References
Process
Result
Requirements & constraints
Architectural reference model & standards
Application domain expertise
Generic shell
standards
Reusable liabrary
Analyse tasks and scenarios
Extract tasks and behaviours hierarchically
Analyse data and functional requirements
Identify RCS modules or agents to compute
Estimate computation and communication load
Develop programs
• Establish Shell computing and communication structure
• Link behaviours and algorithms in shell
• simulate
System integration
scenarios
behaviour & task descriptions
Knowledge, interfaces & algorithms, sensory requirements
Hierachy with functional modules, cmd and data interfaces
Hierarchy in computing processes, allocated on processors
lab tested system
execution system

5.2.9 Conclusions

From the Smart Grid Islanding exercise can learn the following:

1. The Chain approach to on the SIMILAR/VIEW matrix worked well for the Smart Grid Islanding Challenge. The challenge did follow the order is all aspects, but in hindsight it would have been better to do follow the order. Investigate Alternatives

for the functional part had been included in the Modelling section rather than the Investigate Alternatives itself. This also means that Either that state the problem functional view should have some more details on requirements or that Investigate Alternatives should do so.

2. The 4D/RCS can be adapted to cover Smart Grid Islanding
3. Separation between Executive, functional and process view can be fuzzy.

6 Discussion

This report structures existing information in a way that helps to reason about the design of intelligent systems. As such, it is by no means “complete”. In this concluding chapter we would like to touch on some open issues and briefly suggest an approach for each of them, based on the approach of this report. Any of these issues alone would already be enough to warrant the pre-1.0 version of this report.

6.1 Innovation processes

When trying to convince someone to innovate, it is paramount to realise that there is an existing system –in the broadest sense of the word- in place for the part you are trying to innovate. This existing system is integrated with other systems, and this integration is rooted in all views on that system: process, physical, enterprise, etc. This means that business intelligence is not that different from traditional R&D: the former mostly describes the existing system in it’s context, the latter describes the new system –as seen by your system engineers- and is mostly concerned with the inner workings of that new system. A successful innovation policy efficiently integrates both, over all views.

This supports the idea that in R&D organizations it is wise to integrate the internal R&D capacity as much as possible with the external business intelligence. The contents of this report might help to give them a common language.

6.2 Business Cases and Costs

A proposal for an innovation process is usually motivated by a business case. This business case ties actors, actions, intentions and goals, and is supported by contracts about services and goods. This supports the notion to make the business case part of the enterprise view of a system.

However, the investment costs are related to the process of developing the innovation. Therefore this has strong ties to the development view of a system.

We suspect that other views might also benefit from having a financial paragraph.

6.3 Reliability, safety and security

These terms are notoriously vague, so we have to start with their definitions in order to make our points.
Reliability is defined as the ability of a person or system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances.
Safety is defined as the freedom from danger.
Security is defined as the freedom from danger where the threat is coming from hostile systems or persons.

Put this way, a working definition could be that **reliability** governs the superset of situations that somehow negatively impacts the functioning of the system. **Safety** is concerned with a subset of those situations, where the impact is so big, that it becomes a *danger*. **Security** is a subset of safety, where in addition to the presence of a danger, the threat is also coming from hostile persons or systems.

For example, malfunctioning hardware is a typical case for reliability, unless it turns out to be sabotage by a rivalling company. In any case, it is a safety hazard if the associated impact is considered a danger.

We recommend making risk management on all these three levels an integral part of the design process. For now, we leave it to further study and discussion how this should be implemented.

7 References

7.1 Chapter 1 - Introduction

1.	[MOORE1665]	*Cramming more components onto integrated circuits*, G.E. Moore, Electronics Magazine 19 April 1965
2.	[WIKIMOORE]	http://en.wikipedia.org/wiki/Moore%27s_law
3.	[CHI2004]	*Wireless Sensor Networks: Mass Market Opportunities*, C. Chi and M. Hatler, ON World Inc., 2004.
4.	[KURZWEIL2005]	*The Singularity is Near*, Kurzweil, R., Viking, New York (2005). ISBN 0-670-03384-7
5.	[MARTIN2006]	*The meaning of the 21st century*, Martin, J., Penguin Books, London (2006). ISBN 1-57322-323-9
6.	[FEYNMAN1959]	*There's Plenty of Room at the Bottom*, Richard P. Feynman, At the annual meeting of the American Physical Society at the California Institute of Technology (Caltech), December 29th 1959
7.	[WIKISECURITYSCAN]	http://en.wikipedia.org/wiki/Security_scan
8.	[REICHENBACH]	*Monitoring the Ocean Environment with Large-Area Wireless Sensor Networks*, Frank Reichenbach, Matthias Handy, Dirk Timmermann, Institute of Applied Microelectronics and Computer Science, University of Rostock
9.	[EIS2010]	*Extreme Ice Survey*, http://envisense.org/glacsweb/index.html
10.	[TNO2010]	*TNO,* http://www.tno.nl

7.2 Chapter 2 - Terms and Definitions

1.	[WIKIPEDIA]	*Wikipedia,* http://en.wikipedia.org
2.	[RECHTIN2000]	*The art of Systems engineering*, E. Rechtin, M. Maier, CRC Press, second edition 2000
3.	[EYKHOFF1974]	*System Identification,* P. Eykhoff, John Wiley & Sons, 1974
4.	[MCCORDUCK2004]	*Machines Who Think*, (2nd ed.), McCorduck, Pamela, (2004, Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1)

7.3 Chapter 3 - System Views

1.	*[KRUCHTEN1995]*	*Architectural Blueprints—The "4+1" View Model of Software Architecture*, Philippe Kruchten (Rational Software Corp.*),* IEEE Software Nov. 1995
2.	[IEEE1471]	*Recommended practice for architectural description of software intensive systems.*, IEEE1471
3.	[RM-ODP]	*The Reference Model of Open Distributed Processing*,

http://www.rm-odp.net/

4. [SHAW1995] *Comparing Architectural design styles*, Mary Shaw, IEEE Software, Volume 12, Issue 6, November 1995

5. [TYREE2005] *Architecture Decisions: Demystifying Architecture*, Jeff Tyree, Art Akerman, IEEE Software Volume 22, Issue 2 (March 2005), Pages: 19 – 27, ISSN:0740-7459

6. [MAY2005] *A survey of software architecture viewpoint models*, Nicholas May, 2005

7. [NICOL2004] *Model Based Evaluations: From Dependability to Security, IEEE Transactions on Dependable and Secure Computing,* January-March 2004 (vol. 1 no. 1), pp. 48-65, David M. Nicol, IEEE, William H. Sanders, IEEE, Kishor S. Trivedi, IEEE

8. [WELCH1998] *Specification and Modeling of Dynamic, Distributed Real-Time Systems*, L.R. Welch, 19th IEEE Real-Time Systems Symposium (RTSS'98)

9. [4DRCS2002] *4D/RCS: A Reference Model Architecture For Unmanned Vehicle Systems Version 2.0*, James Albus, 2002

7.4 Chapter 4 - Design methodology

1. [ROYCE1969] *Managing the Development of Large Software Systems*, Dr. Winston W. Royce, 1969

2. [BOEHM1986] *A Spiral Model of Software Development and Enhancement*, Boehm B, ACM SIGSOFT Software Engineering Notes, ACM, 11(4):14-24, August 1986

3. [FORSBERG1998] *System Engineering for Faster, Cheaper, Better*, Forsberg, K., Mooz, H. (1998). Center of Systems Management.

4. [MOOZ2006] *The Dual Vee —– Illuminating the Management of Complexity*, Mooz, Harold; Forsberg, Kevin (July 2006), Rochester, NY: Proceedings of the International Council for Systems Engineering (INCOSE) Conference, Ibid, pp. 341-352

5. [BAHILL1998] *Re-evaluating systems engineering concepts using systems thinking,* Bahill, A. T. and Gissing, B., IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 28(4), 516-527, 1998.

6. [INCOSE2010] *A Consensus of the INCOSE Fellows*, http://www.incose.org/practice/fellowsconsensus.aspx

7. [KESTER2008] Kester, L.J.H.M., Designing Networked Adaptive Interactive Hybrid Systems, IEEE MFI 2008

8. [BROOKS1991] *How to build complete creatures rather than isolated cognitive simulators*, R.A. Brooks, in K. VanLehn (ed.), Architectures for Intelligence, pp. 225-239, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991

9. [HAAMAS2006] Proceedings of Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS), AAMAS'06, 2006

10. [ACKOFF1989] *From Data to Wisdom*, R.L. Ackoff, Journal of Applied Systems Analysis 16, 1989

11.	[ZELENY1987]	*Management Support Systems: Towards Integrated Knowledge Management*, M. Zeleny, Human Systems Management 7, no.1, 1987
12.	[JDL1991]	*Data Fusion Lexicon*, US DoD, Subpanel of the Joint Directors of Laboratories, Tech. Panel, 1991.
13.	[ENDSLEY1988]	*Design and evaluation for situation awareness enhancement*, M.R. Endsley, Proceedings of the Human Factors Society 32nd annualmeeting Santa Monica, CA, Human factors and Ergonomics Society,1988, 97-101
14.	[LISI1998]	*Levels of Information Systems Interoperability (LISI),* US Department of Defense C4ISR Architecture Working Group, March 1998.
15.	[FUSTER2004]	*Upper processing stages of the perception–action cycle*, J.M. Fuster, Trends in Cognitive Sciences, 8:143-145, 2004
16.	[BOYD1987]	*A Discourse on Winning and Losing*, J. Boyd, Maxwell AFB Lecture, 1987.
17.	[BUEDE2000]	*Engineering Design of Systems - Models and Methods*, Buede, Dennis M. © 2000 John Wiley & Sons
18.	[SAUSER2006]	*From TRL to SRL: The Concept of Systems Readiness Levels*, Sauser et al, 2006

7.5 Chapter 5 - Result Cases

1.	[TISNET2007]	*TISNET: Veiliger Verkeer, minder Files*, Joris Sijs, Bart Driessen, TNO Magazine, December 2007
2.	[TISNET2009]	*TISNET Future in Traffic Management,* http://www.ieee.tue.nl/aes2009/presentations/ Sijs%20TISNET_IEEE_symposium_6_4_2009.pdf

8 Appendices

A Questions from research proposal and their answers

The research proposal that led to this report included a list of explicit questions to be answered. Based on the structured content in the main report, we answer these questions here one by one.

A.1 How do you know which architecture is better than another?

The quality of an architecture can be defined as the level of which it addresses the needs that are put to it. These needs can range on all aspects of the architecture, most often on functionality and costs. Needs are typically defined by a complex collection of buyers, maintainers, end-users, operators, etc. The quality of an architecture is therefore a subjective aspect: it depends on the point of view. This means that all sorts of compromises have to be made when determining an architecture. The method of making these compromises is often the determining factor for the quality of the resulting architecture, both objectively as subjectively.

A.2 How do you know which design method is better than another?

The quality of the design method can be defined as the level of which it addresses the needs that are put to it. These needs consist of quality of the architecture and the costs of obtaining that architecture.

Methods such as Rapid Application Design are an extreme form where the costs are important. This leads to concessions in the quality of the architecture. This is typically taken care of by allowing a cyclic process in which design phases are iterated multiple times.

A.3 Which criteria play a role in the decision to build intelligence centralized or to build it distributed?

The question suggests a binary choice: centralized or distributed. However, intelligence can be put in any part of the system, in any quantity, so the question needs rephrasing: **which criteria play a role in designing the intelligence of a system?**
Any trade-off between different aspects of a system might be an incentive to improve a part of the system, including making it more intelligent. Typical criteria to be considered are:

a. *Cost of communication or storage versus scale*: this could lead to a need to filter raw sensor data as soon as possible. The intelligence can be used to filter erroneous, redundant or irrelevant data at lower functional levels.
b. *Functionality versus speed*: communication between subsystems takes time, which may not be available. Moving intelligence closer to the source of the data can be used to reduce the need for communication.
c. *Privacy versus functionality versus costs*: pointing a sensor at humans directly introduces privacy issues. Keeping object knowledge in the sensor prevents a technical need for a highly secured centralized (more costly) database and secure

communication to this database. The data-breach for a node might be less of a problem than the data-breach of the central system.[9]

d. *Robustness versus costs*: distributed intelligence in a system can be more robust against errors, malfunction or removal of parts of the system.

A.4 How do you, independent of the type of ISN, express the scale of a sensornetwork?

The question is equivalent to "how do you express the scale of a tree (independent of its kind)?" The scale can be measured in many dimensions. Typical dimensions for ISN's are covered area or resolution (scale), some quantity of intelligence or robustness (dependability).

A.5 At which scale of an ISN do the challenges start to become so big that the results of this project are necessary?

This remains an open question. People that are aware of the views and methodology offered in this report tend to find their own rules of thumb to answer this question.

A.6 Is an ISN without actuators (i.e. without a feedback loop) still an ISN? Which typical ISN challenges emerge when including actuators?

The definition of ISN is irrelevant. The question should be: does it help to think about the actuator (management)-side of a system? Including that part of a system presses one to consider actuator management, signal management, object management, situation management and impact management. A system that does not include any of this, is a strict monitoring (observing) system. We have to realize at this point, that the strict objective observer is merely a theoretical entity. In practice, the real observer always causes an (unnecessary) perturbation of some kind.

A.7 Which are the relations between existing architecturemodels: JDL, OODA, Cognitive models, etcetera?

In his paper Kester describes these relations [KESTER2008].

A.8 Which are good methods and principles to get a good functional decomposition?

This is described in chapter 3 of the main report.

A.9 Which are the types of interaction between functional components, and why are these relevant?

This is described in chapter 2 of the main report, and also in the paper by Kester [KESTER2008].

[9] This is slightly related to de-perimeterisation, as described by the Jericho Forum.

A.10 Which level of (description of) interoperability between sensors, and between sensors and central systems, if any, is desirable: syntactic, semantic or pragmatic?

Many protocols exist, either designed by TNO or by others. As far as we know, three good starting points exist to explore such protocols. The first is the OSI model [WIKI2009OSI], the dominant model of how to look at physical communication. Virtually all existing communication protocols can be linked to this model.
The second is Levels of Interoperability [WIKI2009CI]. This model describes a more generic way of looking at data abstraction and communication. It helps to think about how system components can work together.
The third is Sensor Web Enablement [OGC2009], which is a very practical initiative of the Open Geospatial Consortium.

A.11 Which relations exist between existing modelling methods: UML, SysML, etc?

This question was not addressed in this report.

A.12 How do different aspects (energy, dependability, functionality) of an ISN interact with each other?

This is the main topic of chapter 3 of the main report.

A.13 At which abstraction level should we define the performance indicators? Should this be at the level of individual components, end-to-end performance or should we rely on the intelligence of the system, and steer only on the highest abstraction level?

This report does not directly answer this question. However, we hope it gives the methods and tools to design the performance indicators that are needed.

A.14 What is the best way to model the system aspects? Why?

This is the main topic of chapter 2 of the main report.

A.15 What is the matching multi-aspect design framework that is capable of dealing with the large amount of design choices? Are those heuristics? What is the role of interactive design?

The 4+1 model that is described in section 3.2.1 of the main report is such a framework, and section 4.6 - *Two System Stages* elaborates on that. We applied it to two cases in chapter 5.

A.16 Which methodology should be applied to design and deploy very large scale ISN's?

Chapter 3 of the main report recommends a methodology that is suitable for any scale of ISN.

A.17 Which are usable methods to come from a functional decomposition to a design of an ISN?

This is the main topic of chapter 3 of the main report.

A.18 How can knowledge with respect to a usable architecture model and design methodology effectively be transferred to a larger group of system engineers within TNO?

We designed a master-class for our colleagues, based on the content of this report.

A.19 How do we deal with data- and information security in ISN's?

Although the topic is important, we have been unable to address it properly yet. Based on our work so far, we suppose it starts by identifying two sources of threats:
1. Neutral external systems/parties that pick up data or information by accident, and handle it without care.
2. Hostile external systems/parties that actively seek out data and information.

All views on the system have to be considered from the viewpoint of both these cases. Please also note that *data- and information security* is probably not the only form of relevant security. Physical access control to key actuators (such as guns) might be just as relevant.

A.20 Which are the necessary trade-offs between robustness of the ISN and it's performance (such as retransmissions in communication), delay tolerance, etc.

The main theme of this report is that the answers to such questions depend on the case. We hope to offer you the models and methodologies with which you can answer such questions yourself. In this case we suggest modelling elements of the process view (retransmissions, delay tolerance), the physical view (cardinality, power, layout) and the enterprise view (use cases and costs).

A.21 Which parameters of an ISN must be observed in order to judge its performance is according to specifications? In other words, which are the key performance indicators (KPI), how do we get them from the network, how must they be interpreted and what kind of management is possible?

The KPI's should be given in the specifications, and the specifications should be expressed in the language of the system-views.
The question behind this question is, which are typical good KPI's over many types of systems? To be able to answer that question, we may start by looking at a general model of a system, and determine which are the key indicators that relate to a common sense idea of *performance*:
- A monitoring system is part of a sense-think-act control loop. The speed with which data and information can move through this loop might be an indicator.
- The resources that are used per iteration of such a loop could be an indicator of the efficiency.
- The amount of effect in the external world related to the resources used (including those used to gather data and information) could be an indicator.

A system is comprised out of multiple subsystems. This subdivision can be done along any of the views on a system (physical, process, etc.). For each view, and for each subsystem in that view, performance indicators might be proposed.
It may help to be aware of types of indicators, as seen from a companies' point of view [Wikipedia: Performance Indicator]:

- *Quantitative indicators* which can be presented as a number.
- *Practical indicators* that interface with existing company processes.
- *Directional indicators* specifying whether an organization is getting better or not.
- *Actionable indicators* are sufficiently in an organization's control to effect change.
- *Financial indicators* used in performance measurement and when looking at an operating index

A.22 What is the relation of this report with the TNO KaVoT MiReCol, specifically with the subtheme MiReCol Architecture?

The MiReCol (Mixed Reality for Collaboration) programme is another KaVoT besides ISN. It started in 2009, 2 years after ISN started. One of the theme's in MiReCol concerns the design of a MiReCol architecture (MA). The programme focuses on linking models (mixed reality) for collaboration. Virtual reality plays a big role when visualising such a mixed reality. The MA is designed to facilitate this.

ISN Architecture (ISNA) follows a model-based approach, i.e. all relevant aspects and interactions among the aspects will be handled via explicit models. In this respect virtual reality and human machine interaction (and other MA related features) can be handled via specific model types, though these are not in the focus of the ISNA activities. As such, MA can be seen as a *case* for ISNA where these aspects play an important role.

During the parallel execution of both projects we examined the feasibility of a joint case: the Secure Haven case. However, we found out that MiReCol used this case in the pre-design phase: a case where different stake-holders could interact using virtual reality to set the system requirements. As such, the MiReCol process ends where the ISNA approach starts. Therefore it was decided to abandon the idea of simultaneously working on this case.

The main guideline for the MiReCol Architecture development is to establish an implementation platform, which enables easy and extendible connections among various sensing, simulation, visualization and HMI components. The result is a generic, but relatively low level publish-subscribe mechanism, which does not address the needs of the ISN application domain (e.g. dependability aspect is not addressed, temporal aspect is only implicit, etc.). From ISNA point of view the MA can be considered as one particular class of system architectures, which is suitable for the MiReCol application domain. It can be said that the instantiation of the generic MA for a particular MiReCol application needs a design process to be developed in the ISNA.
We propose that when customizing the MiReCol Architecture to a particular application, the ISNA methodologies will be applied.

B Practical case: Balloon case

A hypothetical case was made up that allows the teachers to gradually introduce the concepts and knowledge taught in the master class: the Balloon Control (BC) case.

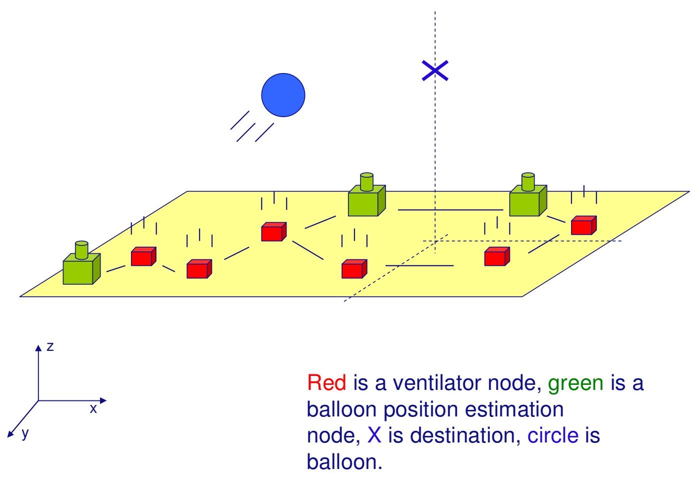

z
x
y
Red is a ventilator node, green is a balloon position estimation node, X is destination, circle is balloon.

Figure 99: The Balloon Control case

A fictional experimental setup is shown in Figure 99. Several nodes are placed on a table (flat surface), each node knows its own position and has a radio, a processor, and has a fan that blows air as an actuator. The fan is pointed upward and each node can adjust the intensity of the fan. There is a balloon that is light enough to be kept in the air by the fans and can be moved about the table by the fans if they work in cooperation. The balloon has no means of staying in the air by itself and if the fans stop blowing it sinks to the ground. The nodes have to determine by themselves where the balloon is at all times.
The specification of the Balloon Control can be summarized as follows:

- The nodes must use their fans cooperatively to move the balloon to a set destination as smooth as possible. (The network can clearly perform better as the number of nodes increases and will fail if the number of nodes drops below a certain level.)
- Each node must talk with the network of other nodes to find out its own position, where the balloon is, where the other nodes are and how it should use its fan to achieve the network goal of moving the balloon to the correct destination.
- Users can dynamically set the balloon destination

Though the case is very simple, there are a number of system design variants (with different properties), which can satisfy these requirements. In this report we do not intend to find the (in a sense) optimal system architecture. Rather we use this design case to show how the different alternatives can be represented in our modelling framework.

C Physical Design Patterns

Changing a system on paper is easier than changing it in real life. Therefore it makes sense to include a brief summary of physical design patterns that have somehow proven there value in relevant cases.

C.1 Typical intelligent sensor network hierarchy

In a typical network hierarchy we can recognise three types of systems:
1. The sensor systems/motes
2. The edge systems
3. The application systems

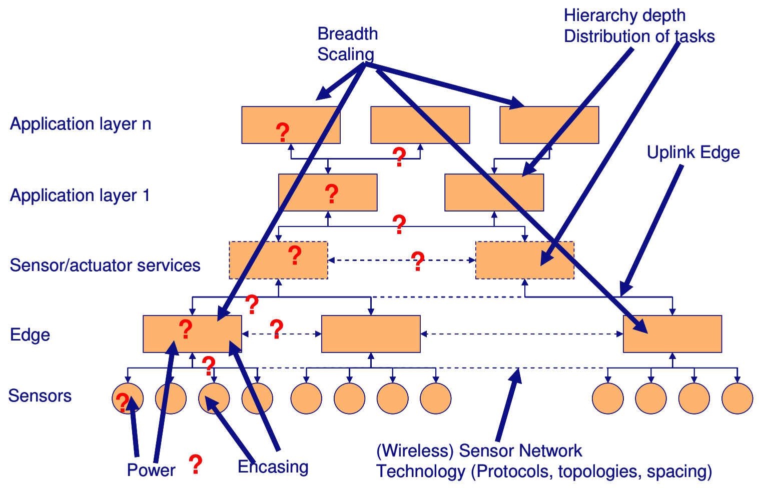


Figure 100: Where choices need to be made in the physical architecture

Below these systems have been worked out into the relevant components.

C.2 Archetypical intelligent sensor

When we talk of sensor networks it sometimes helps to consider the token example: the intelligent sensor. This device, or any number of devices has as goal is to gather data from the environment, then process this in some way and pass it on to the user (human), or to the system which will then perform some action on that data. The physical subsystems are typically:

Transducer - this is sensor in its pure sense of the word. It converts one type of energy into another type of energy. In our case the measured physical quantities are converted into an electrical signal, which we can further process.

Controlling unit - Heart of the sensor node as it controls the gathering of the measurements from the transducer (including processing of those signals - like filtering, amplification, …) and passes on this data (note ‘data’ - not anymore raw measurement since it has been processed) to the communication unit. It is also responsible for interpretations of actions and their execution regarding commands that come via the communication modules.

Communications module - responsible for the communication of the sensor node with other entities (which can be other sensor nodes, but also other entities).

Power source – power source is often required for the sensor node to be able to perform the different activities: sensing, controlling and communicating. There are different types of power sources (e.g. power lines, wind power, water or solar power sources) that can be used in sensor networks.

This physical form of an intelligent sensor is typically called a *mote*[10].

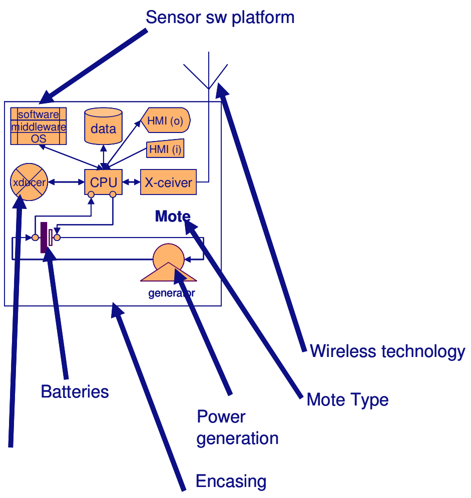


Figure 101: The typical components of an Intelligent Sensor.

Below are some examples of intelligent sensor devices. The is an example of a Pan & Tilt camera, a dome camera and an example of a "Mote" a ZIGBEE or similar device that has houses e.g. acoustic, vibration, acceleration, temperature or other such sensors. The processors are low power processors with limited processing capacity. Energy harvesting could be vibration, delta-t or solar. Batteries are typically Lithium –ION with a very high energy density. Vibration power harvesting is often accompanied with a super capacitor for energy buffering rather then a battery.

[10] http://en.wikipedia.org/wiki/Motes

Figure 102: Examples of Smart Sensors.

The important choices to be made are:
- What sensor
- What Mote
- What software platform (Sun-spot, Tiny OS, ZIGBEE, etc)
- What middleware
- The communication network/protocols (E.g. Smart-dust, IEEE 802.15.4 like Zigbee, Wireless Hart and ISA100)
- Power provision (batteries, harvesting etc)

C.3 EDGE Systems

Edge systems are used to link sensors to a general IP infrastructure (sensor multiplexor) and can be used to do some local processing like signal enhancements and object recognition. Edge system could be separate from the sensor as is in the case where large numbers of simple sensors are used or when the sensor itself has limited intelligence or could be embedded in the sensor as is often the case with complex sensors. With Intelligent camera's the edge functionality is often embedded in the camera itself.

Logically an edge system has the following components:

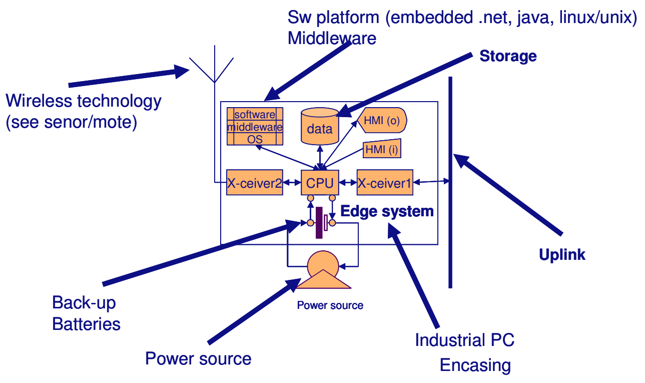


Figure 103: The typical components of an Edge Server

Physically Edge systems are industrial pc's running real-time and embedded versions of UNIX, LINUX or Windows. One example can be found are in the figure below.

Figure 104: example of an Industrial PC that can be used as Edge Server

The choices to be made for Edge system are:
- Encasing (Industrial, water/dust proof, vibration proof etc)
- Software platform (UNIX/LINUX, Java, (Embedded) Window, …)
- Sizing (CPU, RAM, disk storage)
- Up-link (WiFi, UMTS, wired, ..)
- Power (230V AC, 24V DC)

C.4 Application Servers

The application servers are normal application servers that run standard versions of UNIX, LINUX or Windows albeit optimised for real-time performance.

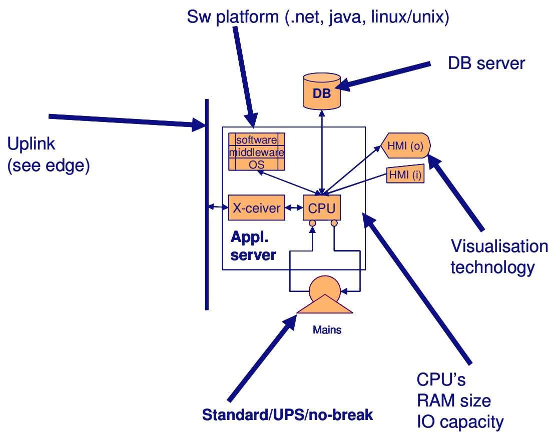


Figure 105: The typical components and choices for an application server in an ISN

The choices to be made are mainly:
- OS (UNIX/LINUX versus Windows)
- Functionality (DB, application, ….)
- Virtualisation
- DB system and other middleware
- Development environment and tool boxes
- Sizing (CPUs, RAM, storage)
- Encasing (cabinet, rack-mount, blade, ….)

C.5 Networking

Some other physical components are already implicitly mentioned above. They mainly involve the networking. The closer to sensor the more specialist is the network. The new generation of wireless sensor networks are often based on the IEEE802.15.4 family of Low Rate Wireless Personal Area Networks (LRWPANs). The physical layer tends to be very much the IEEE802.15.4 standard, the higher layers (MAC and upward) has some diverging developments to include frequency hopping in the 2.4 GHz band to increase reliability. Some older networks are based on older proprietary protocols on the 450MHz, 863 MHz ISM bands that offer more range but less capacity. The important aspects determining the choice for network are:

- Availability in COTS components
- Range
- Energy consumption.

The hardware used, the protocols, the sample rate and the amount of internal processing determine the total energy requirements.

The diagram below is an indication how these choices can be driven.

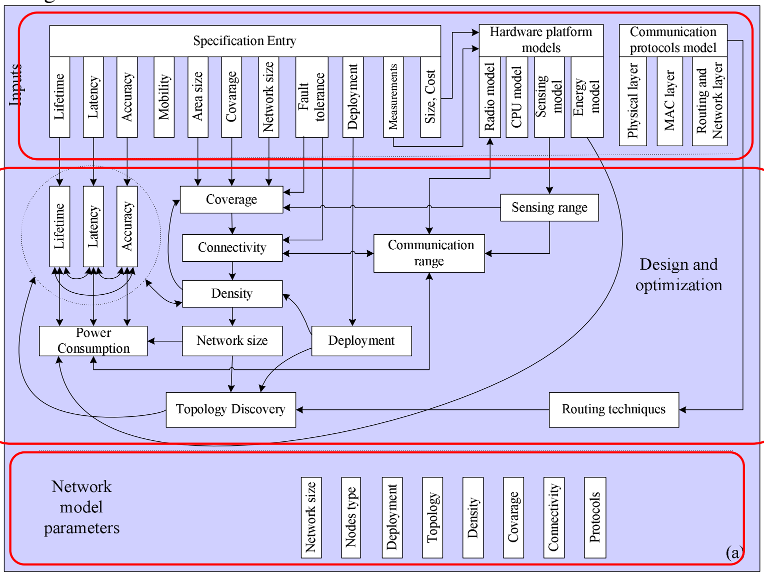


Figure 106: A design methodology for wireless sensor networks

D The AnySense Framework

The AnySense Framework is described in more detail in the following draft white paper: *TNO's AnySense – an architecture to realize productive use of sensor data and information; From single domain sensor network stovepipes towards a common information space*; Authors: B.D. van der Waaij, E.A.F. Langius, W. Pathuis. This paper was not written for peer review but rather as an informal description of the framework. The work on AnySense is still very much in progress. As a result of this project AnySense will be more clearly positioned in relation NAIHS and the 4D/RCS reference models.

TNO ICT is developing the AnySense Framework. This AnySense Framework is constructed around a design pattern of multiple heterogeneous sensor & control networks serving multiple applications in multiple organisational domains. The AnySense design patterns are the opposite of the traditional stove pipe patterns where Sensor & Control network are used for a single application in a single organisation. It allows developers and designer to focus on the innovative aspects of complex Intelligent Sensor Network applications that fit the above design pattern.

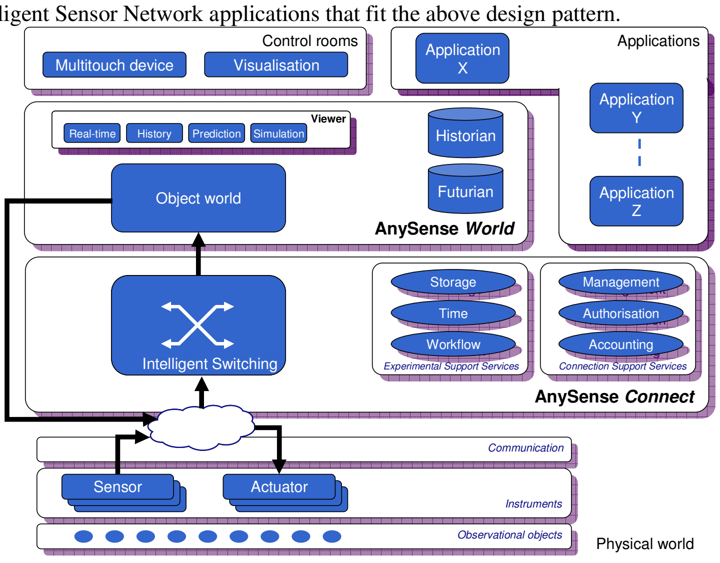
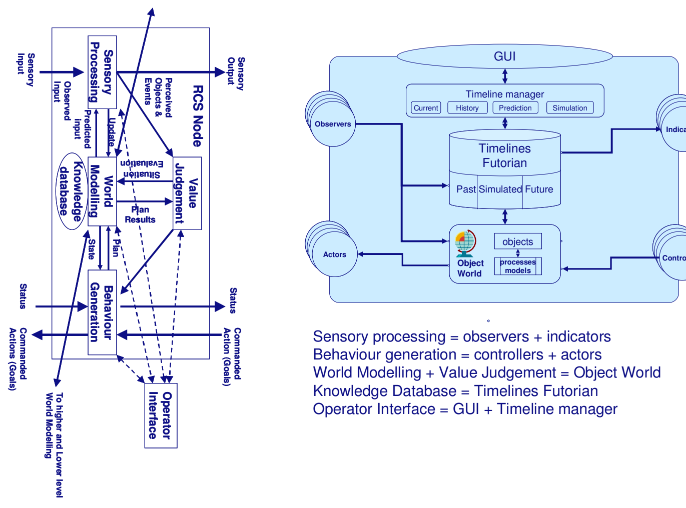


Figure 107: The AnySense Framework

AnySense consists of two main components. The first is AnySense Connect that is aiming at efficient data acquisition from multiple sensor sources, at (pre-)processing the data, very much according the bottom layers of the NAIHS model, storing the data and distribution of the data to a broad spectrum of different destinations. The destinations can be multiple applications in multiple organisational domains including start-up companies and Small and Medium Es developing new services that can be added on top of the existing AnySense Connect infrastructure without the need to adapt the infrastructure. AnySense Connect offers designers and developers tools to handle the different aspects of the sensor data logistics such as data transport, frequency, storage and retrieval and security. Part of the AnySense connect is the AnySense Datamodel in which not only sensor data is handled but also the meta data description of the sensor data such as type and the way it has been measured. This allows unambiguous interpretation by multiple parties.

The second component is AnySense World. AnySense World is the design pattern for the informational component of multiple sensor & control applications in multiple organisations. The AnySense World Domain Container very much resembles the 4D/RCS control node and mostly operates on the higher layers of the NAIHS model.



Sensory processing = observers + indicators
Behaviour generation = controllers + actors
World Modelling + Value Judgement = Object World
Knowledge Database = Timelines Futorian
Operator Interface = GUI + Timeline manager

Figure 108: The 4D/RCS control node (left) next to the AnySense World Domain container (right)

AnySense World allows designers and developers of the sensor applications conforming the AnySense design pattern to deliver a system of which the process chain crosses organisational boundaries. The world views offer a near real-time view on the real world from the proper perspective of the organisations participating in the process chain and the appropriate abstraction level without the need for all to see all of it. Designers and developers using the AnySense framework are encouraged to publish their view on the sensor data to other interested parties as if it were data produced by virtual sensors in the virtual world. The other parties can use this virtual sensor information to add value with their own services. The virtual sensors allow data fusion of sensor data without requiring detailed domain knowledge of the domain of the underlying layers.