THESIS ON INFORMATICS AND SYSTEM ENGINEERING C56

Enhancing Situation-Awareness, Cognition and Reasoning of Ad-Hoc Network Agents

JÜRGO-SÖREN PREDEN

TUT PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Control

Dissertation was accepted for the defence of the degree of Doctor of Philosophy in Engineering on May 20, 2010

Supervisor: Professor, DSc Leo Mõtus, Department of Computer Control,
Tallinn University of Technology

Opponents: Dr. Gabriel Jakobson, Chief Scientist, Altusys Corp., USA

Professor Kuldar Taveter, Department of Informatics,
Tallinn University of Technology, Tallinn Estonia

Defence of the thesis: June 21, 2010

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any academic degree.

/töö autori nimi ja allkiri/

ISSN 1406-4731
ISBN 978-9949-23-010-5

INFORMAATIKA JA SÜSTEEMITEHNIKA C56

Spontaanvõrgu agentide situatsiooniteadlikkuse, taju ja arutlusvõime täiendamine

JÜRGO-SÖREN PREDEN

TTÜ KIRJASTUS

Enhancing situation-awareness, cognition and reasoning of ad-hoc network agents

Abstract

The role of embedded computers, which have been more widespread than conventional computers for decades already, is changing from being closed devices in dedicated applications to being open, networked components of larger systems. Cyber-Physical Systems is the term used for describing the systems that integrate physical processes and systems with networked computing systems.

The formalisms used for describing Cyber-Physical Systems must accommodate projection of the physical and virtual properties of the world to the computing system in a generic way, thereby making the exchange of such data possible across systems.

This thesis suggests that explicit introduction of the concept of situation awareness will help to describe and analyse the interactions and the behaviour of Cyber-Physical Systems. A situation is described by a set of situation parameter values and each component in the system interprets the situation parameter values according to its own rules. The situation awareness concept allows the system's components and the system as a whole to maintain a coherent view of the world (by exchanging situation parameter values) in order to harmonize the actions and responses of the components of the system, and help the system as a whole to behave appropriately in the current circumstances (situation).

The thesis proposes that the situational information should be associated with validity information, starting for example in what area and for what period of time the situational information is valid. This approach allows dealing with the uncertainties inherent to Cyber-Physical Systems. The concept of a situational information mediator is introduced, which uses the validity information associated with situational information for acquiring and providing information relevant to system components.

The thesis also presents a distributed system of systems concept that builds upon the notion of networked situation aware computing systems. A part of information, exchanged in a system of systems is in the form of situation parameter values, which are computed by individual sub-systems and shared with the other system's components.

Spontaanvõrgu agentide situatsiooniteadlikkuse, taju ja arutlusvõime täiendamine

Lühikokkuvõte

Sardarvutid, mis on tavaarvutitest levinumad juba aastakümneid, on muutumas ühele rakendusele pühendatud suletud seadmetest avatud ning võrku ühendatud suurte süsteemide osaks. Süsteeme, mis seovad füüsilisi protsesse ja süsteeme võrku ühendatud arvutisüsteemidega, nimetatakse küber-füüsilisteks süsteemideks.

Formaalsed vahendid, mida kasutatakse küber-füüsiliste süsteemide kirjeldamiseks, peavad võimaldama maailma füüsiliste ja virtuaalsete omaduste esitlemist ühtsel viisil, mis teeks võimalikuks seda tüüpi andmete vahetuse süsteemide vahel.

Antud dissertatsioon käsitleb situatsiooniteadlikkuse kontseptsiooni kasutamist küber-füüsiliste süsteemide puhul, mis võimaldab kirjeldada ning analüüsida sellistes süsteemides toimuvaid interaktsioone ning samuti selliste süsteemide käitumist. Situatsiooni kirjeldatakse situatsiooniparameetrite kaudu, mida iga süsteemi komponent tõlgendab vastavalt kohalikele reeglitele. Situatsiooniteadlikkuse kontseptsioon võimaldab süsteemi komponentidel ja süsteemil tervikuna säilitada ühtset tõlgendust maailmast (vahetades situatsiooniparameetrite väärtuseid), ühtlustamaks süsteemi komponentide tegevusi ning samuti tagamaks süsteemi kui terviku sobivat käitumist kindlas situatsioonis.

Samuti käsitleb antud töö situatsiooniinfo sidumist kehtivuse informatsiooniga, alates ajaintervallist ja piirkonnast, kus situatsiooniinfo kehtib. Selline lähenemine võimaldab hõlpsamini hakkama saada küber-füüsilisele süsteemile omaste tundmatute omadustega. Töös tutvustatakse ka situatsiooniinfo vahendaja kontseptsiooni, mis kasutab situatsiooniinfo kohta käivat kehtivusinfot, hankimaks ning pakkumaks süsteemikomponentidele vajalikku informatsiooni.

Töös kirjeldatakse ka hajutatud süsteemide süsteemi kontseptsiooni, mis põhineb võrgus töötaval situatsiooniteadlikul arvutisüsteemil. Osa informatsiooni, mida süsteemide süsteemis vahetatakse, on situatsiooniparameetrite väärtuste kujul, mida arvutavad välja üksikud alamsüsteemid ning mida jagatakse erinevate süsteemi komponentide vahel.

Acknowledgements

First I express my gratitude towards my supervisor Leo Mõtus, who has been a good mentor through the years, giving me good advice when needed, yet leaving me enough freedom to set and pursue my own targets.

The thesis has evolved over a period of several years (the first draft was created over two years ago), so there are several people and organization to be thanked for contributing either directly or indirectly to it.

I thank Merik Meriste for many fruitful discussions on the many trips we have taken together and also for the advice he gave on my thesis. Without his input the thesis would be much poorer.

My collaboration with Johannes Helander has been productive; I thank him for his trust when employing me as an intern at Microsoft Research in Redmond. I really enjoyed working there and also the collaboration and friendship that followed. Many of the ideas presented in the thesis spawned at the time when I was an intern at MSR. I hope that we can continue that cooperation also in the future.

It is difficult to do good work without a good working environment and I must say that I have had some very good colleagues over the years, which has definitely aided my work. Very much work in the projects that I have led has been performed by Raido Pahtma, who is able to turn fuzzy ideas into reality with much more steadiness than I have. In addition I want to name some of the many co workers and students who have also collaborated with me over the years and helped me develop the ideas presented in the thesis (although they might not be aware of this): Andri Riid, Risto Serg, Tauno Otto, Taivo Lints, Tõnu Näks, Domenico Bianco, Priit Pikk.

I would also like to thank the opponents of my thesis whose scrutiny and comments I respect and in most aspects agree to, it is good to know that my thesis has been reviewed by experts in the field.

I also thank the organizations that have supported me during the studies: Department of Computer Control, Research Laboratory for Proactive Technologies, ELIKO competence centre, IMECC competence centre, Microsoft Research Redmond, Archimedes, EITSA and the Estonian Ministry of Defence.

Last but not least I thank my friends and family who have endured my lack of time over the past years. Some members of my family have granted me use of their summerhouses where I have been able to work on my thesis without any distractions. I thank my parents for providing me a stimulating childhood, my wife Marge and daughter Anna for being supportive of my studies and science related activities.

Contents

List of Figures

List of Tables

Introduction

The ratio between embedded and general purpose computing processors was about one to a hundred (Turley, 1999) already more than a decade ago, which means that for every desktop or server processor there were about a hundred embedded processors built into devices we use daily. All of these embedded systems interact directly or indirectly (if they are part of a larger system) with the physical world. Up until recent years most of these embedded systems have been closed in terms of allowed interactions, performing the task they have been designed to do and not exposing the functionality or computing capability to the outside, except in cases where an embedded system is part of a larger system. In those cases the functionality exposed and processor usage scenarios are well defined and constrained to the set specified by the system designer. It has been predicted (Lee, 2007) that a radical transformation in terms of communication (and therefore interactions that a system is subject to) will come in the near future from networking these devices.

A new term has been coined to describe such networked systems: Cyber-Physical Systems (CPS). Cyber-Physical Systems integrate physical processes and systems with networked computing systems. While Cyber-Physical Systems cover also "classical" embedded computing systems (where the interactions with other computing systems are very constrained) the concept of embedded systems is extended by creating networked systems where the interaction with the physical world plays an important role. This new generation of systems use communicating computers deeply embedded into and interacting with the physical processes, adding thereby new capabilities to physical systems. The Cyber-Physical Systems range from very small (e.g. pace makers) systems with quite limited or no interactions to large-scale (e.g. the national power-grid) systems with complex and dynamic interactions. Creating robust systems that deal with and consist of both, parts of the physical world, and networked computing systems is relevant to a large class of applications, such as consumer electronics, energy efficient smart homes, homeland security, industrial automation, civic infrastructure, and mechanical systems, such as airplanes or automobiles.

In case of Cyber-Physical Systems it is just as important to understand the cyber part (i.e. computing) of these systems as it is to understand the physical processes that the systems are involved in. If both aspects are not considered it is not possible to create systems with desired (and predictable) behaviour. So the methods and tools used for modelling and analyzing Cyber-Physical Systems must be able to accommodate both the cyber (computing) and physical aspects of the system. The formalisms used for describing Cyber-Physical Systems must accommodate projection of the physical and virtual properties of the world to the computing system in a generic way, thereby making the exchange of such data possible across systems.

While the computing systems are becoming more complex, the social endurance to system failures is decreasing. It is hard to overestimate the potential importance of Cyber-Physical Systems to our everyday lives and economies, so every effort should be made to develop reliable and trustworthy systems. Therefore the designers' capability to deal with the complexities, incomplete information about many system's characteristics, rapidly changing environments, and dynamically changing computing systems becomes crucial for success. As part of this capability is the ability to validate and verify the computer application before it becomes operational.

The prevailing practice in the validation of embedded software in the development process relies on testing for concurrency and timing properties. This approach has worked reasonably well, because programs have been relatively small, and because the interactions of the software have been strictly limited – the operational environment and the interaction patterns for the software are fixed at design time so that they cannot alter the behaviour of the software. However, the CPS applications demand various services in a networked environment, so testing only a limited set of the interactions that a system is permitted is not adequate. In a dynamic networked environment, it is not possible to test the system or parts of the system (that are interacting with the physical world) under all possible conditions (Lee, 2007). So as conventional testing has its limitations, it is therefore becoming less useful and alternative methods must be developed. In addition to verifying stationary properties of systems, the customer expects on-line behaviour verification as required in the self-organising systems and in systems which exhibit emergent behaviour.

The temporal aspects of interactions in case of any system that is in direct interaction with the physical world cannot be abstracted away – the type of action that an entity in the physical world performs is just as important as the time instant when that action was performed. So temporal features are an integral part of any Cyber-Physical System. It must be noted, however, that not only timing, but also location is a critical aspect in Cyber-Physical Systems. It can be said that in case of conventional embedded systems timing catered for most unknowns as synchronous operation of the system was expected. In case the coherence of temporal relationships between data items generated by the system components was not maintained problems were likely to arise. In CPS scenarios the temporal relationships are not fixed but in addition the location of system components may be (dynamically) changing. This means that neither the temporal nor the spatial relationships between embedded devices and therefore also data they generate can be fixed at the design time. Therefore Cyber-Physical Systems must be able to cope with variations of data both in time and space.

The CPS technology is clearly one of the enabling technologies for realizing the ubiquitous computing (a.k.a. pervasive computing, invisible computing and calm computing) concept, first introduced by Weiser (Weiser, 1991). Consequently, in order to make any advances in that area, solving the problems related to Cyber-Physical Systems is crucial.

The conventional computer science and many formal models surveyed in the thesis enable to describe and analyse formally various atomic operations but are of very little use for analysing a collection of dynamic interactions of mobile units performing atomic operations. In case of complex Cyber-Physical Systems the designer must be able to express the computation proper, interactions between the peer computing entities and their environments, and the effect of interactions on the current and future computations and interactions.

Using the existing conventional formalisms the algorithms and various aspects of systems' behaviour can be described, but we are stuck on formally describing a deterministic selection of algorithms and trivial behaviours of separate algorithms. In CPS, for feasible selection of actions the state of the external world (including both the physical and virtual worlds), the state of the agent itself (including the results of past interactions which have influenced the state of the agent), and the state of the other agents must be considered.

Sensor networks are one of the emerging technologies that can be classified as being an essential component of Cyber-Physical Systems that support the ability of computing systems to perceive the surrounding world (physical and virtual processes interacting with the computing system). Sensor networks are in close interaction with the physical world, having to react to the stimuli received from the physical world, while the computers in these systems are interacting with each other. As the configuration of systems is not known before the system is operational, these interactions cannot be specified beforehand. Hence the generic difficulties with new ubiquitous computing systems can also be observed in sensor networks. The computation in these systems depends on the current and past interactions, it exhibit emergent behaviour, and is therefore different from that of classical deterministic computing systems.

This thesis suggests that explicit introduction of the concept of situation awareness will help to describe and analyse the interactions and the behaviour of Cyber-Physical Systems. A situation is described by a set of situation parameter values and each component in the system interprets the situation parameter values according to its own rules. The situation awareness concept allows the system's components and the system as a whole to maintain a coherent view of the world (by exchanging situation parameter values) in order to harmonize the actions and responses of the components of the system, and help the system as a whole to behave appropriately in the current circumstances (situation). So (part of the) interactions between the system components cater for exchange of situation parameters. The behaviour of the system is described in relation to situations as defined by situation parameter values. Introduction of situations can be interpreted as a means for weakening the impact of incompletely known non-linear relationships between the parts of the systems and their environment and thus increases the probability of successful operation of those systems. The relationships are incompletely known and non-linear as a complete description of the environment does not exist. Neither is the behaviour of the system and its components completely known as the system specification is only an approximation of the actual implementation of the system. In addition the possible evolution of the environment and the system over time is not known and cannot be therefore considered in the system description.

The thesis extends the concept of a mediator that operates as an arbiter of situational information. The channel in the Q model (Motus, et al., 1994) can be also viewed as a mediator of a kind but in the current thesis the mediator concept is extended so that it can consider other types of meta-information besides temporal. The mediator is also an active/proactive arbiter of information, acquiring and providing relevant information.

The thesis also presents a distributed system of systems concept that builds upon the notion of networked situation aware computing systems. A part of information, exchanged in a system of systems is in the form of situation parameter values, which are computed by individual sub-systems and shared with the other system's components.

The major contributions of the thesis are:

- Introduction of situational information as a design and analysis concept for distributed systems;
- Association of validity (including temporal and spatial) information with situational data;
- Concept of a mediator as the arbiter of situational information;
- Concept of system of systems that builds upon of the notion of situation aware computing system.

Starting from the system's view the thesis presents an overview of different types of systems, categorized by their type of behaviour. Then motivation for introducing situation awareness concept to the distributed systems domain is given. The thesis continues with an overview of situation awareness, firstly in the domain of human factors and then in the context of computing systems. The situation awareness concept is then developed further by introducing the concepts of situation parameters, hierarchies of situation parameters and discussing the validity of situation parameters. The thesis also presents some case studies that are related to situation awareness concept and that the author has been involved in within the past years. In the very end conclusions and some open problems are presented.

Abbreviations and definitions

Computing system – an artificial system that contains one or more computing units; in one end of the spectrum a computing system may be a single-processor system with very limited interactions, in the other end of the spectrum a computing system may be a multi-computer system (a network of systems) with complex interactions

Context awareness – the ability of a system to act based on the state of the world around it

CABP – communication area based positioning

COP – common operational picture

ISM band – industrial scientific medical band

LOP – local operational picture

MANET – (mobile) multi-hop ad-hoc network

Mediator – arbiter of information

MIM – multi-stream interaction machine

POI – point of interest

PTM – Persistent Turing Machine

PWM – pulse width modulation

RDF – Resource Description Framework

RSSI – Received Signal Strength Indication

SA – situation awareness

SIM – sequential interaction machine

Situation – the aggregate of biological, psychological, socio-cultural and environmental factors acting on an individual or group of agents to condition their behavioural patterns (Mir09)

Situation awareness – the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future (Endsley, 1988)

TM – Turing Machine

UAV – unmanned aerial vehicle

UGV – unmanned ground vehicle

UML – Unified Modelling Language

WSN – wireless sensor network

1 Current state of the art and problem statement

In 1936 Alan Turing proposed a universal computing machine which could be used to compute any computable sequence, provided the machine is supplied with a tape containing the description of the task – the algorithm and the input data (Turing, 1936). Although the original objective of Turing's work was to define what a computation is and also to be able to show if a computation is effectively solvable (whether the computation terminates) the practical result of the work was the concept of an automatic computing machine. Machines based on the Turing machine concept are suitable for replacing rooms full of people who solved mathematical equations manually – the algorithms used for solving the equations were well defined as well as the input data. In fact Turing in his paper (Turing, 1936) explicitly depicts the model of his machine on a person doing manual calculations (the people who performed the calculations were called computers at the time). Soon after Turing had presented his ideas on the universal computing machine the development of automatic computing machines was started. The reason for the development of the first computers lay in the fact that the computational tasks at hand at the time were too big to be solved manually. The world's first functional program controlled Turing complete computer (the Z3) came into existence in 1941 and it was developed by Konrad Zuse (Copeland, 2006). The tasks of ENIAC, developed between 1943–1946 (Burks, et al., 1981), the first general purpose electronic computer were calculating artillery firing tables and solving computations required for the development of the hydrogen bomb. Naturally computers have evolved over time and as can be predicted modern computers solve the computing problems much faster than the original computers did as both the computing hardware and the corresponding software have developed substantially over the years.

Gradually people also started using computers for tasks for which they were originally not designed – interacting with the real world, controlling real-world processes, interacting with people and other computers (via a network). The first purpose built industrial control computer system was developed for Texaco Company and installed at the Port Arthur, Texas refinery. The system achieved closed loop control on March 15, 1959 (Stout, et al.). The modern world presents us embedded systems, digital control systems and software intensive devices, all of which fall outside the class of systems that can be correctly modelled using a Turing machine.

So we can conclude that modern computers are used for much more than just computing Turing computable functions (usually such functions are called injective functions) – atomic computations where the output of a function is expected to be always the same given the same input values. The theory of computations stemming from the Turing machine prescribes that the computation runs from start to end without interruptions – once the input data is given and the computation is started the computation should run with no intrusions.

In (Stepney, et al., 2005) the authors bring many examples (some of which are described below) on how the modern computing systems have departed from the classical Turing computing paradigm. The classical approach assumes that a program has a single well-defined output channel. While in real systems the output channel is not so well defined and we can choose to observe other aspects (as the internal state or the trajectory of the computation) of the physical system as it executes. A computing entity itself does not necessarily know if, where, and how the output is used. The Turing paradigm assumes that the computation is atomic, i.e. computation is discrete in time and space – the end state is preceded by the start state with operations that transform the input to an output between these states. This abstraction is not well applicable to modern systems as the intermediate states of the computation have relevance and the output can be also affected by input data received while computation is executed.

In addition to the non-atomicity of the computations in modern systems they also depart from the von Neumann architecture (von Neumann, 1993) which prescribes a fetch-execute-store model of program execution. Parallel architectures and execution models (e.g. FPGAs, neural nets) are examples of such systems.

A modern computing system is a collection of algorithmic computations, which are, cleverly, tied together, the output of one computation being the input to another one (e.g. superposition of functions in mathematics). At the moment we are not able to describe or construct modern computing systems in such a way – the general system architecture, the components and their interaction patterns must be known at design time in order to construct a complete system from individual parts. In case of many modern systems the components of the systems are not known at design time, neither are the interactions between the system components known. In many cases functions are used which are not Turing computable. For modern systems the algorithm is not the central part but instead interactions connecting algorithms are of importance (although quite often we can achieve practically the same result by applying different algorithms.). These features make the applicability of the compositional method very questionable. Of course systems are constructed also more loosely (without paying due attention to formal restrictions and constraints imposed by the theory based on Turing computable functions) but there is no systematic way of constructing such systems and having any confidence that a given system will work as desired.

The Turing paradigm is based on algorithms and programs (that are superposition of algorithms); however the concept of a program has many interpretations nowadays. A given problem may be solved by a single algorithm contained in a single program or a problem may be solved by a collection of algorithms contained in several programs that run on (non-algorithmically interacting) separate computers. Some of the functionality of a system may be also realized outside of the computer altogether, for example preliminary detection of signal energy can be implemented using analogue electronic circuitry with an output from that circuitry being an input to the computer (and to an algorithm).

The interactive computation paradigm as proposed by (Wegner, 1997) extends further than the Turing paradigm and although there is no widely accepted and used formal model for describing interactive computation, conceptually the interactive computation paradigm can be used to describe and analyse the majority of recently emerged computing systems – e.g. networked embedded systems, networked pervasive computing systems, ad hoc networks. Unlike the Turing computable paradigm which only considers the case of a single computation or a well-specified sequence of computations, the interactive computation paradigm considers any interactions between the computing entities and ongoing computation within a computing entity (including stream computation).

The complexity of interactive computing systems is much higher than the complexity of traditional (Turing) computing systems. In case of an interactive computing system a collection of computing entities interact persistently with each other and with the physical environment, they process streams instead of strings. The output of a single execution of a computing entity depends on its current state (which depends on its previous executions), on its current input from the physical world and on input from peers. So it is very complex if not impossible to design and control the behaviour of an interactive computing system. In addition, time and space are important factors that must be considered when reasoning about the behaviour of interactive computing systems that are directly linked to the physical world and (may be) to autonomously operating other interactive computing systems. In order to have a better understanding of computing systems and formalisms for describing such systems an overview of computing system types is given in the following section and an overview of formalisms is given in Appendix A.

1.1 Classifying computing systems

This section suggests taxonomy of computing systems depending on strictness of requirements to the computing models applicable for analysis of their observable behaviour. Already well-established computing systems can be categorized into transformational and reactive, based on their essential behavioural aspects and methods required for their analysis. There is also a rapidly emerging class of computing systems, further named proactive systems (Tennenhouse, 2000) whose characteristics are discussed below.

The models of computation used for describing those classes of computing systems can be categorized into "classical" algorithmic models (a.k.a. Turing computation models) applicable to transformational (and in many cases) to reactive systems, and into non-classical, non-algorithmic models, which have not yet matured, and are represented by interaction-centred models of computation (a.k.a. super-Turing models of computation) see, for instance (Wegner, et al., 2004) and (Motus, et al., 1994).

As stated in the introductory section the applications of computers have evolved from simple data processing systems, which are obviously transformational, to computers for creating virtual worlds, which are partly reactive, but also transformational systems. Computers have been also embedded into the real world, creating more complex reactive systems, embedded systems, self-X applications (Güdemann, et al., 2006) and finally pervasive computing systems, which in most cases are also proactive applications. When in the first steps of computer technology development the theory was well ahead of the applications then nowadays we can say that theory is lagging behind actual applications – the modern computer applications are far beyond what the current theory is able to describe, verify and validate.

1.1.1 Transformational systems

Transformational systems transform inputs into outputs and the designer of the system only has to specify the transformation function (this function should be Turing computable). Transformational systems (e.g. data processing systems) can be highly complex but all purely transformational systems can be decomposed fairly easily, while conserving all their properties, into sub-functions, each part having a well-defined input and output. Well established tools and theories exist for describing and designing transformational systems. The inputs are presented to a transformational system in an ordered manner and system transforms the presented inputs to outputs as it *is able to*. The main metrics for a transformational system defines the order and correctness of implementing the transformations. If a system is functionally decomposable into smaller fragments then each fragment has a well defined input-output transformation, as does the whole system. The correctness of both the components of a transformational system and the whole system can be validated using, in principle, the same approach.

In essence, each transformation in a transformational system realizes an algorithm and a transformational system as a whole realizes a superposition of algorithms. So the tools and methods based on algorithm theory are sufficient for describing and validating transformational systems. In practice, the interpretation of the notion of algorithms (and algorithm theory) is not very strict (as discussed in for example (Bass, et al., 2003)), since transformational computing systems are usually tolerant to reasonable approximations resulting from neglecting some actual features of applications, or from violating certain axioms of the theory.

1.1.2 Reactive systems

A reactive system must react to external stimuli; the word “external” may have here a different meaning to various parts of the system. From the high level point of view “external” includes everything that is outside the scope of a system and all stimuli generated outside of the system can be considered as external stimuli. In addition, different parts (or components) of a system (which by themselves cannot form an independent system) have to react to stimuli generated by other parts of the system. In this case “external” stimuli originate from the system itself.

The behaviour of a reactive system is determined by the allowable (or desirable) sequences of input stimuli and output responses. During the design of a reactive system various preconditions, actions and timing constraints may be imposed upon the system’s inputs and outputs, since only the fact that a computer reacts to external stimuli does not necessarily suffice to provide the correct reaction in the context of a particular environment-computer interaction.

Embedded systems form a separate subclass of reactive systems that implements a closed loop system with feedback through the environment, meaning that the actual system comprises (a part of) the environment and the embedded computer system. An embedded system is also expected to maintain an ongoing relationship with its environment, providing a persistent service (Harel, et al., 1985) in a timely manner, i.e. satisfying the constraints required by the environment and imposed upon the embedded system by its designer.

Harel and Pnueli state in (Harel, et al., 1985) that the structure of a program in the case of reactive systems (and of any computing system) need not, and cannot in many cases, reflect the true structure of the problem since in a general case one needs to simplify the computational structure of the system in order to be able to apply decomposition for partitioning the complex behaviour effectively among its components. However, decomposition cannot become a regular basis for design of reactive systems. For instance, in case of reactive systems a certain interaction may become a factor affecting systems' behaviour. When decomposing a complex reactive system, quite often the desired behaviour cannot be guaranteed due to several neglected and/or heavily approximated interactions in the environment – in order to maintain composition-based design. Hence, decomposing complex behaviour is still, in the general case, an open research problem which is also tackled in this thesis.

1.1.3 Proactive systems

Proactive systems also react to external stimuli but, in addition to reactive systems, they have local (and/or global) goal function, plus the capability and authorization to apply self-X properties (Güdemann, et al., 2006) in order to modify their reaction – locally and/or globally – so as to (sub-) optimise the goal function(s). On-line application of self-X properties becomes possible only by providing autonomy to the systems' components – so that on-line modification of system's structure, or on-line modification of some of its functional components, or on-line modification of interactions between its components, etc. can be invoked by the system itself. Therefore, the behaviour of proactive systems is remarkably richer than that of reactive systems, and often requires more sophisticated models of computation (e.g. super-Turing models of computation) in order to monitor and control the self-X processes and the emergent behaviour that is characteristic to proactive systems.

For instance, a proactive system has all the properties of a reactive system, plus a proactive system has the freedom to perform (or not to perform) a pre-specified action – depending on the state of its environment as perceived and interpreted by the proactive system (or parts of the proactive system). A system's behaviour is often generated not only in direct response to stimuli from the environment (i.e. from both the physical world and the virtual world – e.g. peer computing systems, the other components of the system), but in most cases the generated behaviour depends also on the internal state of the computing system (i.e. in addition to depending on the system's working memory the behaviour also depends on the system's long-term memory).

Proactive systems are remarkably more exposed to uncertainties (mostly caused by incomplete information, and/or autonomous behaviour of its components) than reactive systems. Reactive systems (usually) implement a closed loop system with feedback through the environment, hence most of the uncertainties stem from incomplete knowledge regarding inner properties of the environment. The behavioural autonomy permitted to components of a proactive system, normally enabled in order to improve robustness and resilience of the system, and to satisfy better the components' goal functions, introduces additional uncertainty in decision-making. The uncertainties may be introduced for instance:

- Due to autonomous and partially asynchronous operation of components which may result in actions that are not perfectly orchestrated to be invoked – which may in turn remarkably complicate control of the system's behaviour; additionally, one might need to consider the impact of:
    - On-line readjustment of (mediated) interactions between the system's components, and/or between system and its environment, or
    - Substantial rearrangements of the systems due to on-line execution of self-X properties, e.g. substitution of components or changing their functionality, modifying the connectivity of components, or the permitted pattern of interactions between the components
- The fact that the environment models used are not sufficiently precise (e.g. some essential interactions are neglected etc) may lead to situations where the in-system interaction patterns cannot match the requirements and constraints imposed by the environment and the goal functions of the system.

In a nutshell, the above listed uncertainties foster the emergent behaviour (Stepney, et al., 2005) of a proactive system – emergent behaviour typically occurs in complex integrated and/or networked systems. It is called emergent behaviour because it occurs dynamically, during operation of the system and cannot be deduced a priori from the static structure of the system and from the behavioural properties of the components of that system; see also (Motus, et al., 2005).

Emergent behaviour is usually a non-desirable property of transformational and reactive systems (although it may appear due to bad system design) since in case of these systems the behaviour of the system should be precisely predictable. Systems that are strictly based on Turing computable functions cannot exhibit emergent behaviour by definition – the underlying axioms and imposed restrictions disable emergent behaviour.

1.1.4 A shift from transformational to reactive to proactive systems

Truly complex systems (e.g. natural systems, networked pervasive computing systems, cyber-physical systems etc.) cannot be adequately modelled by a superposition of Turing computable functions. This has been noted long time ago. There are other properties of system components that are important besides the input-output transformation of these components (be they computing nodes in a computing system, or cells in a living organism). For example, real world processes (e.g. human speech or physical processes) are usually temporally regular which means that the temporal aspects of these processes cannot be neglected when trying to monitor, interpret, and/or control those processes. Clearly the behaviour of computing systems which are immediately interacting with (or embedded into) the physical world is also dependent on the temporal (and/or spatial, and other) aspects of the applied interactions and evolution of the physical world.

In the context of natural sciences P. Anderson (Anderson, 1972) makes a point that biologists, physicists, and others are able to decompose the world into small isolated components and describe the behaviour of each component in a "complete" way (at least in terms of properties observable in an isolated component). However, scientists have encountered serious obstacles when trying to synthesise larger functioning artefacts using the same isolated components and applying their properties and theories, developed in the "divide and rule" based research – meaning that the decomposition process is not always reversible. This claim is valid for example for any multi-cell organisms, recently similar obstacles have been also observed when integrating autonomous smart artefacts into a networked system. It is clear that there are properties of these cells, components, and/or building blocks that were not captured by static observations of isolated components. The aspects of (dynamic) interactions between the components are obviously controlled by these not captured properties. Decomposition into isolated components hides these emergent properties, consequently the higher-level behaviour and properties of the system cannot be deduced smoothly from the known properties and interactions of isolated simple components – partly due to the inbuilt approximation feature in popular "divide and rule" methodology. Anderson notes that "*the main fallacy in this kind of thinking is that the reductionist hypothesis does not by any means imply the "constructionist" one: The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe*." (Anderson, 1972)

By definition, any non-trivial system consists of several interacting components. The components interact with each other, the majority of modern systems also interact with (parts of) the physical world. The algorithmic approach to describing a system as a superposition of Turing computable functions does not handle interactions explicitly – and thus leaves us with an approximate description of systems. Attempts to construct a complex system without considering all the essential interactions (including the temporal, spatial, etc aspects of these interactions) between the system components and the environment will almost certainly provide insufficiently precise match between system that is actually required and its model. While applicable theory and extensive experience is available for designing isolated (transformational) components with well predictable behaviour, the same cannot be said about more complex systems – e.g. those systems comprising components whose interactions are non-transformational. Even when the behaviour of single, isolated (transformational) components is correct, the behaviour of an interacting collection of these components may not satisfy the requirements and expectations – meaning that some other (e.g. non-transformational) aspects of those components, that affect the behaviour of the integrated system, were not satisfactorily described or verified during the analysis of isolated components.

So it can be concluded that we need to enhance the description and analysis power of our models of computation so that they can be used to verify static and dynamic (interactive) behaviour of (proactive) components and their ensemble.

Turing machines serve as a basic model sufficient for describing and analysing transformational systems as the computation performed by a Turing machine implements a transformation– an output value is computed given an input value and a suitable algorithm (Turing computable function). Given a specific input value the output value is always the same – strictly speaking, Turing computable functions do not have a memory of its previous executions. Additional inputs, or any other data, are not accepted before the current computation has been completed (i.e. the Turing machine has reached its terminal state).

The pedantic satisfaction of the above stated restrictions in the case of reactive systems may cause conflicts with the requirements imposed by the environment. Hence in practice those restrictions are satisfied approximately, and the constraints are stretched as long as robustness of computing system and of the environment permits. However, we should not forget that most of the computation in reactive systems is time dependent, and the result of computation usually depends on the data that the system has previously processed. So, strictly speaking, different executions of an algorithm in reactive systems should produce non-identical output data values even if input data values were identical – because the external environment could have memory, and could changing in time. Turing model of computation is applicable as long we can neglect the conflict between the behaviour of computing system and its surroundings (the environment) – as long as we can consider our system closed. Reactive computing systems require, in a general case, a more powerful model of computation – e.g. interaction-centred model of computation.

It is clear that a reactive system is transformational (algorithmic) in certain isolated sections of the computation where an input value is transformed to an output value, however such sections tend to be interacting with each other and with the environment by non-transformational interactions (compare also with a synchronous programming paradigm, (Halbwachs, 1993)). Similarly to reactive systems being transformational in some sections, also proactive systems are reactive in certain isolated sections where an output must be generated in response to input stimuli.

So the interactive computation paradigm (i.e. interaction-centred model of computation, see for instance (Wegner, 1997) (Motus, et al., 2005)) must be used in case of reactive and proactive systems in order to describe and analyse subtle details of their behaviour formally, not just by testing. In case of proactive systems, extending the interactive computation model to a situation aware interactive computation model should be sufficient for describing and analysis the subtle details of behaviour. Interactive models of computation capture the notion of performing a task or providing a service, rather than algorithmically producing outputs from inputs (Goldin, 2006) (Stepney, et al., 2004).

This problem is also apparent in the concept of futures proposed in (Friedman, et al., 1976). The authors propose a concept where an executing program is partitioned into independent execution threads. The main thread initiates the other threads (as futures), they start running and are expected to be ready by the time the computation result is required. Since the exact behaviour (output values) of the futures is not known even when the futures are called the output of the computation cannot be predicted even when the computation is executed. Drawing parallels from the Turing machine (or from the Persistent Turing Machine) the contents of the working tape may change during the computation (the output data generated by the futures affects the contents of the working tape and it changes during the computation), also the state of the machine may change during the computation as the output of the future may change the state of the machine. In addition, since we do not know what exact algorithm is selected to compute a future, the machine itself may change during the computation. The concept of futures can be also compared to distributed computation in an agent network.

In case of complex systems employing the interactive computation model we face several distinct problems which are actually related: a system with many possible states and output combinations, and complex computation in the entities of that system depends on the past inputs to those entities. So the problem with interactive computational model is not whether we are able to compute the output(s) given we know the current state and the input data values, but whether we are able to limit the possible behaviours of the system to an acceptable subset and to what extent are we able to predict the behaviour of the system at a given time instant when we do not have complete information on the system.

1.2 Do we have a model for interactive computation?

Appendix A presents a superficial overview of computing formalisms, describing their expressive power and applicability in the context of different computing system types. Clearly none of the formal models presented in the appendix are suitable for formalizing interactive computation without substantial modifications. In case of interactive computation we need to model the computation itself, the interactions between computing entities, the interaction between a computing entity and its environment, indirect interaction via the incompletely known environment, and the effect of the interaction on the current and future computations and interactions. While we are able to describe the computation formally and also aspects of interactions formally (using various formalisms) we do not have a good method of explicitly expressing the dependency between the past interactions and the current state of the computing system. Some of this knowledge is actually available on lower levels of the hierarchy of models – where detailed descriptions of components are depicted. At the system's level we can approximately describe the knowledge about the impact of the past on the presence and future (e.g. feedback) by defining situations which empirically integrate deep and shallow knowledge about the impact of feedback to the systems behaviour at this moment and in the future.

The situations (and the hierarchy of situations) can be defined empirically and a set of "situational parameters" can be selected that enable to detect the situations and to influence the occurrence or transition of situations. The situation parameter values can be received from the other computing entities or computed locally. The situation parameter values can be combined to define higher level situations, if necessary. The situation parameter values can be used as input data for some computations that occur in a computing system. We can say that situation awareness is a concept that assumes the ability to detect the situations, and to use the situational parameters for improving the functioning of a system, its design, on-line behaviour verification in adaptive systems and/or in systems with self-X properties.

1.3 Why and how situational information has become essential

Modern computer applications comprise computing systems with ever increasing complexity directly interacting with complex environments whereas the cost of system failures or even small malfunctioning may be (depending on the application) remarkable. At the same time the social endurance to system failures is decreasing. Therefore the designers' capability to deal with the complexities, incomplete information about many system's characteristics, rapidly changing environments and dynamically changing computing systems becomes crucial for success. As part of this capability is the ability to validate and verify the computer application before it becomes operational – conventional testing has its limitations and is therefore becoming less useful. In addition to verifying stationary properties of the system the customer expects on-line behaviour verification of the self-organising if a system contains such features.

At the same time the computer applications tend to become more integrated by networking (earlier) autonomous applications typically using ad-hoc networking technologies, which in turn means that the configuration of such systems is not fixed. Hence, on-line verification and validation is in many cases the only available alternative, in case, for instance, of systems of systems that exhibit proactive behaviour and possess very clearly self-X properties. It is known that the behaviour of those systems cannot be deduced from the behaviour of individual components due to the emergent behaviour inherent in these systems. Therefore we must be able to deal with the emerging behaviour in the form of interactions between the system components (which themselves may be systems with incompletely known properties). This presents us with one of the reasons why so we must look at ways of formalizing interactive computation, and at ways of learning explicit, although approximate, presentation of empirical experience for coping with the incompletely known features. In many practical cases observing and affecting a limited number of interactions in a system are the only ways to manage (or at least influence) system's behaviour.

The conventional computer science and many formal models surveyed in the previous section enable to describe and analyse formally various atomic operations but are of very little use for analysing a collection of interacting atomic operations.

In case of interactive computing we need to express the computation itself, the interaction between a computing entity and its environment or its peers, and the effect of the interaction on the current and future computations and interactions.

Using the existing formalisms we are able to describe the data interpretation algorithms and various system behaviour but we are stuck on formally describing a deterministic selection of interpretation algorithms and trivial behaviours of separate algorithms. In the selection of actions the state of the external world (including both the physical and virtual worlds), the state of the agent itself (including the results of past interactions which are expressed in the state of the agent) and the state of the other agents must be considered. In a complex computing system which itself may be a system of systems there must be way for conveying the current state of the system (the state here having a wider meaning, i.e. the state encompasses the internal state of the system or parts of the system achieved through the normal computation and also the state of the system achieved via interactions with also the physical world) or parts of the system from a component to a component in order to achieve coordinated and desired behaviour of the system.

The behaviour of a complex system (a system of systems) is the combined behaviour of all its components. If all the components also exhibit a certain level of autonomy then it is not possible to describe such a system using algorithmic approaches since a certain input is not guaranteed to produce always the same output, not at the system level and not even at the system component level. This means that the validation and verification of such systems is not possible using conventional methods – there is no desired output corresponding to a specific input, neither are there specific states of system components that is desirable or not allowed when a specific set of input values are presented to the system. The internal state space of the components forming a system may not be identical. Even if the state space overlaps in some sections the internal states of the system's components are not necessarily synchronized but rather each component is responsible for maintaining its own state.

Pervasive computing systems contain several non-terminating processes, where communication is time, selective. It is stated in (Motus, et al., 1994) that pervasive computing processes (comprising of repeatedly activated terminating algorithms (with memory about the history of previous activations)) necessitate the introduction of time-selective inter-process communication. Such communication is not fully analysable in (first order) temporal logic or in timed Petri-nets. The approach suggested in (Motus, et al., 1994) – the Q model – allows describing and analyzing time-selective inter-process communication in stationary mode.

Due to the introduction of mobile computing nodes to current and future computing systems the requirements have become higher: in addition to time selectiveness of inter-process communication it also requires validation of the spatial properties of data (i.e. in addition to the time when the data was generated the location where the data was generated is equally important). So the computing processes are temporally and spatially selective – they can select to consume data originating at desired time instances at desired locations. In addition the temporal and spatial requirements on the data may change dynamically. This is due to the fact the configuration of the system is not known beforehand (not even at the time of deployment, let alone the time of design) and it may change dynamically in an ad-hoc manner during the lifetime of the system. Hence the interaction patterns (data producers and consumers) in the system are not known beforehand. This means that instead of (or in addition to) stationary description and checking (validating) the temporal and spatial properties of data, this must be done dynamically during system runtime, at certain intervals. As the requirements and constraints for data may change at runtime the validation must adapt to these changed requirements.

In order to design a system of systems that behaves in a desirable manner (from the standpoint of the system designer) and also to validate the behaviour of that system, one has to have a concept that is universal within a system and across systems. This concept must be abstract enough to be used across the system, yet the concept should reflect the state of the world (both the physical and computing world) so that the components of a system can utilize the knowledge so as to select the appropriate internal state.

This thesis suggests that the introduction of the concept of situation awareness will help to describe and analyse the interactions and the actions of Cyber-Physical Systems. A situation is an aggregate of factors relevant to a given agent and described by a set of situation parameter values, each component in the system interprets the situation parameter values according to its own processing rules. The situation awareness concept allows the system components and the system as a whole to maintain a coherent view of the world. This in turn allows harmonizing the actions and responses of the components of the system and the system as a whole to be appropriate in the current circumstances (situation). So (part of the) interactions between the system components cater for exchange of situation parameters. The behaviour of the system is described in relation to situations and situation parameter values.

If a system perceives a certain situation the behaviour (actions) of the system (or part of the system) will correspond to some specific patterns. In the same way some specific behaviours may not be allowed in some situations. So checking (and possibly stopping the undesired behaviour) the desired behaviour of the system against the allowed behaviour in the current situation enables the designer to ensure that incorrect behaviour will not occur. Hence this provides a certain additional level of security and quality control for the system's behaviour. Of course the detailed behaviour of the system can not be predicted at a given moment in time but this is not always required in case of proactive systems as long as the system behaves in a way which is not disallowed.

The concept of situation awareness essentially allows softly separating the non-functional part of a system from the functional part. The system reaches a situation after some (temporally and spatially constrained) interactions have occurred. In a specific situation the system's behaviour might have specific constraints corresponding to outside stimuli. In the context of verification and validation we can separate the task in two distinct parts – verifying that a system detects a specific situation correctly and verifying that a system behaves in a certain (pre-specified and fine tuned during operation) way suitable for this specific situation.

1.4 The Problem Statement

This thesis suggests that the introduction of the concept of situation awareness will help to tackle the problems that arise in the design and development of Cyber-Physical Systems. A situation is described by a set of situation parameter values, each component in the system interpreting the situation parameter values according to its own processing rules. The situation awareness concept allows the system components and the system as a whole to maintain a coherent view of the world in order to harmonize the actions and responses of the components of the system and the system as a whole to be appropriate in the current circumstances (situation). Inclusion of temporal and spatial meta-information to the situational information allows dealing with the fact that the temporal and spatial relations between entities and data items are not pre-fixed in a Cyber-Physical System. So (part of the) interactions between the system components cater for exchange of situation parameters. The behaviour of the system is described in relation to situations and situation parameter values.

The thesis focuses on defining and detection of situations, development of situation awareness concepts in the context of artificial systems, and on describing the experimental results.

2 Situation Awareness

Research of context and situation awareness in computing systems has been gaining momentum in recent years. Research of situation awareness was originated in human factors research in the 1970s. But before the discussion on situation awareness can be continued in this thesis, the notions of context and situation awareness must be elaborated upon in order to bring some comparison and clarity to those concepts.

2.1 Difference between context and situation

2.1.1 Context and context awareness

Some authors (Schmidt, 2002) (Dey, 2000) consider context to be the same as situation, assuming that the situation can be inferred directly from the (observable) context or in some cases vice versa (depending on the interpretation of these concepts by the authors). Since the deduction process seems often instantaneous for a person itself, humans generally do not pay attention to the deduction process and therefore they consider the result of a (sometimes long) reasoning process (based on the sensory input) as context. The context is usually the (raw or conditioned) sensory data itself (the state of the world as perceived by the available sensory equipment) about the state of the world.

Humans are able to match basic sensory inputs with low-level situational interpretation without involving (seemingly) any reasoning process. In addition, humans are also able to combine the different situational properties, to infer higher level situations. Again in many cases humans believe that they are not involved in the reasoning process. Because the context and situational information identification is seamless for humans there is also a confusion in the definition and identification of context and situations. It seems to be reasonable to start the discussion on these topics with context and context awareness and from there to move to situation and situation awareness.

Context is defined in the Merriam-Webster online dictionary (Mir09) as "*the interrelated conditions in which something exists or occurs*", so loosely combining the term with *awareness* we can say that context awareness is awareness of the interrelated conditions in which something exists and anybody or anything that is *context aware* must be aware of the interrelated conditions in which it exists.

The term "context awareness" was introduced in the context of ubiquitous computing systems in 1994 by Schilit et al (Schilit, 1994). As many other authors later, Schilit et al view context mainly from the human user's point of view, neglecting the other aspects of context that may be relevant from a computing system's point of view. They describe computing applications which behaviour depends on the configuration and context of the combined computing and physical (the physical also including humans) system.

In other literature many other (sometimes contradictory) definitions can be found for context. In (Dey, 2000) context is defined as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*" and a context aware system is described as follows: "*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*".

In (Yau, et al., 2002) the following definition for context is given: "*Context is any instantaneous, detectable, and relevant condition of the environment or the device*". In (Dey, et al., 2002) the authors again view context only from the human user point of view and in the application described there the context recognition is required only to provide a better service from computing entities to humans. The context is specified as "*information sensed about the environment's mobile occupants and their activities*" and an aware service is a service that is able to take the context into account in its operation. In this use case the set of contexts are unsurprisingly specified from the human user viewpoint and the computing system must be able to detect these contexts. In (Schmidt, 2002) context is defined using the concept of situation "*A Context is identified by a name and includes a description of a type of situation by its characteristic features.*" The reason for such a definition lies in the fact that Schmidt, similarly to other authors quoted above, views both the situation and the context only from the human observer's point and he assumes that the situations are static states of the world from which contexts can be inferred.

2.1.2 Situations and situation awareness

Definitions of situations and situation awareness are even more contradictory than those of context. We can start again for the definition given in the Merriam-Webster online dictionary (Mir09), which defines a situation as "*relative position or combination of circumstances at a certain moment*".

The earliest formal notion of situation (although not situation awareness) was introduced by Jon Barwise (Barwise, 1981) as a means of giving a more realistic formal semantics for speech acts than what was then available. The discussion on situations by Barwise in (Barwise, 1981) offers an interesting perspective to situations and situation awareness, which is rather different from the one prevailing in the approach that most work on context and situation awareness related to computer science offers. Barwise claims, that one can deduce an infinite number of situations from a set of sensor data (regardless of the means of acquiring that data). The questions are, which situations are of interest to the observer, and can the relation between the sensor data and the situation be specified.

Barwise could not find a good formal method (an appropriate logic) for specifying preconditions for a situation. Barwise correctly noted that identical sensor data (perception) can be generated in distinct situations so we must be able to identify what is unique about a certain situation that we want to identify. So we must specify how to distinguish a situation of interest from all the other situations, i.e. what are the characteristics of a situations of interest. In order to express these properties formally we need a formal method that allows us to express the required features.

Barwise stresses that in the context of the study of natural languages (which was the motivation for Barwise's work) we must distinguish between (not interpreted) sign and (interpreted) symbol more carefully than in the study of mathematical theories. This illustrates the point that the same set of data can be acquired in distinct situations and a specific set of data does not always mean that a situation is valid.

Discussing these lower levels of situation awareness (actually acquisition of source data for situation valuation) we can look at the work done by Dretske. In (Dretske, 1969) Dretske argues that what we see "is a function solely of what there is to see and what, given our visual apparatus and the conditions in which we employ them, we are capable of visually differentiating", he also reasons that seeing is independent of "the subtleties if epistemology." This reasoning illustrates well the point that perception and comprehension are two distinct functions and should not be mixed.

A human typically "sees" an object, not the properties that make up the object (Barwise, 1981), he continues saying that "*The most important problem about perception is the relationship between perception and knowledge*", The propositional theories of perception argue that the epistemological and metaphysical questions are really the same – humans don't see an object (say a tree) but rather they see that an object is a tree – humans see generalised facts, not simple physical objects (Barwise, 1981).

In (Hintikka, 1975) the author claims that "To specify what *A* perceives is to specify the set of all possible worlds compatible with his perception … [where] the notion 'compatible with what *A* perceives' is taken as not analyzable.". So Hintikka supports the claim that there is not necessarily a one-to-one mapping between perception and the physical world, so when specifying such mappings one must understand that a specific mapping may be just one of the many possible mappings.

Barwise in (Barwise, 1981) argues that a human "...cannot see a single thing-in-itself, some sort of ideal physical object stripped of its properties and its relations with other objects." Instead a human sees a scene, what is a combination of objects having properties and bearing relations to one another. The properties of objects and relations between the objects are as important as is the idealized thing-in-itself. In the same manner the perception of a specific object or property (that is not completely isolated from its environment) by an artificial agent is not possible, instead the property is perceived in the context of other, related properties. Also the sensory input is not without interferences, i.e. it can't be expected that the perception of a property making up a situation is correct.

Just the fact that an agent is not able to perceive a situation does not mean that the situation does not hold. Barwise distinguishes what a human (or an agent) perceives and what is really going on in the world by calling the (visually) perceived situation a scene. A scene may support the truth and it may not, but the fact that a scene does not support a fact does not necessarily invalidate the truth as the perception capabilities of a human (or an agent) may be limited.

Moving from the conceptual situation awareness problems that were tackled several decades ago to more recent work we can see that the modern implementations take a more simplistic approach to situations and situation awareness.

In (Wang, 2004) Wang specifies situation through contexts "*Situation is a set of past contexts and/or actions of individual devices relevant to future device action*". Supplementing that definition is the definition of context given by Yau, Wang and Karim in (Yau, et al., 2002) "*...any instantaneous, detectable, and relevant condition of the environment or the device*." Both of these definitions imply that a situation is a combination of several aspects (perceived by an observer).

2.1.3 Context and situation

It can be summarized that context is an instantaneous condition of the environment as perceived by an observer by monitoring the observable properties of the world. It is clear that the term “context” carries only meaning when considering the observing entity. Depending on the sensory capabilities of the observer only a limited set of properties of the world can be monitored and therefore only a limited set of contexts recognized. As noted above, humans (being very egocentric) tend to interpret context from their personal point of view, however we must recognize that for each entity the concept of context is different, depending on the properties observable by the entity.

Within the scope of the current thesis the term *context* is used for describing the aspects of the world external to an interpreting entity, not the background or circumstance of an object or phenomena in the world. This interpretation is chosen as the term “context awareness” makes more sense in the prior case.

Within the scope of the thesis a *situation* is a combination of past and present observed contexts (taking into account the temporal and spatial relations between the contexts) and the state of the entity observing the situation is valid in a specific time interval in a specific location.

Situation is an interpretation of the observable contexts, which again are dependent on the observable properties but also on the available interpretation methods. It must be noted that a change in the observable parameters of the world (the context) does not necessarily imply a change of situation. This is best explained by a simple example – a road safety system that (among other parameters) monitors the ambient temperature – a change of temperature from -10 degrees to -20 degrees in Celsius may be observable by the system but does not cause a change of the situation as perceived by the system – it is still freezing, whereas a change from +3 to -2 is a change of situation as at +3 degrees the probability of ice on the road is extremely low, whereas at -2 the probability of ice on the road is quite high. Following a change of situation the system may have to perform some actions, for example notify any parties interested in the condition of the road. Elaborating the example we can consider an intelligent autonomous road safety system equipped with a smart road sign that is able to detect vehicles on the road in addition to the road conditions. So depending on the road condition situation – is the probability of ice on the road high and the situation of cars on the road – is there a car approaching the smart road sign will either display a warning to the driver of the oncoming car or not.

2.2 Human situation awareness

In the following a short overview of work in the area of situation (context) awareness related to computing systems is given. Since a fair amount of work in that area has been done in the context of human situation awareness the overview is started from that area. The overview ends with review of previous work on context awareness of computing devices. It must be said that most of the work on context awareness of computing devices has been concerned with the context as observed by the human user of the computing system, not the context as perceived by the computing system itself.

A human needs good SA in order to perform his tasks effectively. For most of history acquiring good SA has been a matter of experience for humans – what things to take into account in a specific situation in order to know what is going on. However with the advancement of technology the tasks humans had to perform have became far more complex and also the amount of information required for performing the tasks has increased and more information is being provided to the person performing the task.

When in the past a person may have lacked information required to produce good SA, in the present the situation is the opposite – there is a huge amount of information and in many cases the problem is in incoherent or contradicting data, which cause degradation of situation awareness – which in turn may cause accidents. The reason for degradation of SA in the presence of contradicting data lies also in the fact that people are usually not trained for such occurrences and therefore are unable to cope with such situations. With the vast amount of information that modern systems are able to provide to operators the operator may be actually less informed than he was with less information. The reason for this lies in the fact that there is a gap between the amount of information disseminated and the operator's ability to process the incoming information and combine the different bits of information for an objective assessment of the situation.

Therefore the information provided to the operator must be carefully analyzed and foremost, only the information required in a specific situation should be provided to the operator. In order to achieve that some entity must identify the current situation and modify the operator interface so that only the information relevant in the current situation is presented to the human operator. The current state of the art allows to do the latter only offline, i.e. it must be manually specified what information and how to present it to the human user. As machines become more capable a trend that is becoming more prominent (when the legislation allows for it) is that the human is left out of the loop, the human being just an observer and advisor. In this case the situational information provided to the human can be at a much higher level of abstractions since the information is not required for decision making.

For a given operator, SA is defined in terms of goals and decision tasks for that job. An aircraft pilot for example does not need to know everything, but does need to know a great deal of information related to the goal – for instance, safely fly the aircraft in a given situation.

The concept of situation awareness was originally introduced in the human-machine interface research and the main motivation for pursuing research in that direction was the fact that user interfaces for complex systems that relied on computers were able to provide human operators with much more information than the operators were able to process. So in order to create systems that a human operator could operate safely and predictably a selection of information relevant to the human user in a given situation had to be made. The system would then only present the human user information relevant in the current situation in a way most suitable for the current situation. In (Endsley, 1988) the author gives the following definition for (human) situation awareness: "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future".

2.2.1 Three Mile Island accident

In the following an example of a nuclear power plant accident is described to illustrate the importance of good situational information for operators of critical equipment. The Three Mile Island accident is a textbook example of how loss of SA can be the source of incorrect decisions which in safety critical applications can lead to catastrophes.

Some consider the Three Mile Island nuclear power plant meltdown as one of the most important events to trigger research in the field of situation awareness. In the Three Mile Island meltdown incident in 1979 near the city of Harrisburg (Pennsylvania) one of the reactors in the Metropolitan Edison nuclear power plant was subject to overheating due to failure of main feed water pumps (Nuclear Regulatory Commission, 2009). The high pressure was relieved automatically by a pressure relief valve which however malfunctioned and remained open. As a result the coolant level in the reactor began to drop. The coolant level drop was corrected automatically by emergency fill pumps which started pumping additional coolant into the primary pressure loop.

The experts claim that these automatic actions would have prevented reactor meltdown. Here was first point in the accident where the human factors and human-computer interface issues came into play. One of the operators in the control room at the time of the accident reported later that the control panel was “lit up like a Christmas tree”. Clearly too much information was provided to the operators – more information than was required to make an assessment on the current state (or situation) of the reactor. The design of the control/monitoring system of the reactor was inadequate since it was only possible to monitor the coolant level indirectly via a pressure gauge – a drop in the coolant level would result in a drop of the coolant pressure during normal operation. Since the reactor was not operating under normal operating conditions the pressure reading was not correlated directly to the coolant level in the primary loop since steam bubbles had formed in the primary loop which restricted coolant flow in the system.

The operators were not aware of the open relief valve (since there was no feedback from the valve either) and from the elevated pressure reading they concluded that the coolant level in the primary loop was high enough so they shut off the emergency fill pumps to prevent (in their mind) excess coolant from entering the primary loop. Shutoff of the pumps resulted in even higher temperatures inside the reactor and eventually in meltdown of part of the reactor which everybody was unaware of. Only 16 hours after the incident had started were the designers of the reactor able to get their message through to the plant operators that additional coolant should be pumped to the reactor. Only three years after the accident people realized what had really happened: half of the reactor core had melted, about 20 tons of uranium had melted to the bottom of the reactor vessel.

The Three Mile Island accident is a textbook example of how both lack of information and excess information can lead to incorrect situation assessment. Naturally the US Nuclear Regulatory Commission introduced many changes following the analysis of the accident, motivated by the lessons learned from this accident. In addition to changes in the inspection and monitoring routines and changes in the mechanical design of the plants the new regulations also included *improved instrumentation, improved instruction to avoid confusing signals and identifying human performance as a critical part of plant safety* (Nuclear Regulatory Commission, 2009). In retrospect we can only say that it is a pity that research in some cases (such as this) is triggered by what might have turned into a tragic accident.

2.2.2 Pilot SA

In order to provide the potentially huge amounts of information that a modern fighter aircraft can provide to pilots in a manageable manner, a need aroused for understanding how the pilots gathered, prioritized and interpreted the incoming data. In different situations pilots require a different set of information and similarly priorities of data change, depending on the situation. As can be seen from Figure 1 the potential amount of information available to a fighter pilot is really immense and there are also potentially many sources for providing that information. Some of these sources are independent systems and they may provide data that supplements data generated by other data sources. For example the sensory equipment on board the aircraft may supplement the information provided by the infrastructure-based positioning system (e.g. GPS). There may be a case when the information from these two sources is contradictory and it is up to the computers to determine the actual situation and provide the pilot with the correct situational information instead of (conflicting) sensor data.

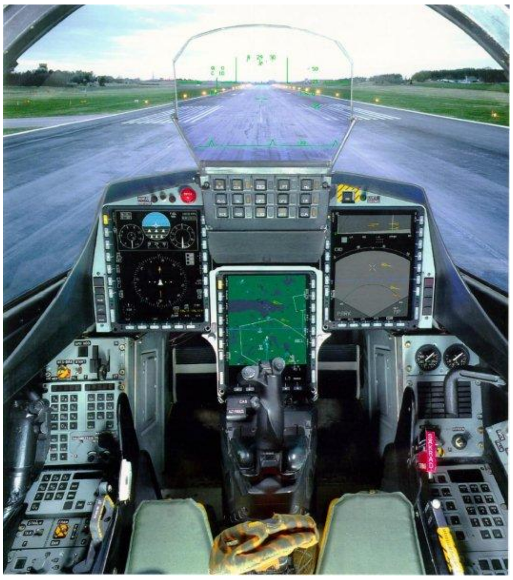


Figure 1 View of a SAAB JAS 39 Gripen Cockpit

US Air Force operational rules define pilot SA as “a pilot’s continuous perception of self and aircraft in relation to the dynamic environment of flight, threats, and mission and the ability to forecast and then execute tasks based on that perception” (Carreta, 1996). Because in air combat the ability to obtain better SA could be a question of life and death (not only for the given pilot), enhancement of pilot SA is of high interest and has been studied pretty thoroughly.

In (Sulistyawati, et al., 2007) the air combat environment is characterized as follows:

- The combat environment is highly dynamic, the pilots need to obtain and process vast amounts of information to develop and maintain awareness of many complex (simultaneous) events relevant for building SA.
- The faster the environment changes the faster one's SA must be updated. Developing and maintaining pilot's SA means maintaining a 3D spatial relationship which is further complicated by the time dimension. Perfect SA at one time instance may be worthless at the next time instance if situational information is not updated with the same pace as the environment changes.
- Air combat situation assessment is based on the following loop: "search – detect – perceive – interpret – project", which is a more detailed version of the OODA (Observe – Orient – Decide – Act) cycle. The pilot must search for the contacts, based on a wide range of data detect the contact of interest, perceive the data from the given contact, analyze the information to obtain further information (relative contact status – speed, distance, closure rate – in relation to his own or friendly aircraft), predict contact intentions and actions.

Although the above points are listed in the context of air combat they are valid for any dynamic situation where situation assessment is required. It can be summarized that in order to maintain good SA a human needs to obtain a vast amount of information, categorize that information based on the entities and phenomenon that the information characterizes and determine the state of every entity of interest. As the situation is changing at a high rate one must update the situational picture at the same rate which means that the steps described above must all be performed also at the same rate. The 3D nature of the air combat environment adds to the complexity as the entities are physically laid out in three dimensions.

2.2.3 Human SA in other domains

From airplane pilots the research in situation awareness has moved into other areas, such as air traffic control, medicine, control rooms, ground transportation, maintenance, space education. A common denominator for all these areas is the requirement for humans to process great amounts of data which arrive at a *high* rate, which is uncontrollable by the receiver, i.e. data must be received and acted upon adequately in a timely manner, regardless of the state of the receiver.

In (Rodgers, et al., 2000) a review on situation awareness of air traffic controllers is presented For air traffic controllers, situation awareness has been used to examine and decrease the number of air traffic control incidents. The use of situation awareness concepts has included for example the following assessments: knowledge of aircraft location, aircraft callsign/altitude/groundspeed/heading/next sector/ type/activity, determination of the aircraft that has been issued incomplete assignments and correct assignments, determination of the aircrafts that are conforming to their assignments, determination of aircrafts that are experiencing an emergency, and determination of aircrafts that are violating minimum requirements.

In some fields of medicine, for example anaesthesiology, situation awareness has become a key component in providing optimal patient care (Gaba, et al., 1995). Situation awareness is being employed to improve the ability of anaesthesiologists to detect, diagnose, and make the correct decisions when treating critical incidents. Medical simulators and analysis of real cases have been used to investigate past cases and train future anaesthesiologists (the topic is discussed further in (Gaba, et al., 1995) (Drews, et al., 2006)).

The authors in (Caserta, et al., 2007) claim that the basis of human situation awareness is conceptual representation, as it is vital to cognitive operations and is intricately tied to memory, perception, decision making, actions, and inductive inferences (Hampton, 2003). A concept in turn is constructed from an infinite set of representations from the environment. Instead of storing full-length recreations the conceptual system organizes a collection of categorical images representing individual components of the experience (Barsalou, 2003). This means that the human conceptual representation system is not storing information on how to detect full situations but instead it deals with components (or parameters) of a higher level situation, which in turn means that situations in the human mind form hierarchies. So although the conceptual system selectively attends to changes in the environment, the information is categorized into components, which are then stored in memory (Schyns, et al., 1998).

The humans' conceptual system's ability to support perception of the situation allows humans to store countless aspects of situations in memory for future use in interactions (Caserta, et al., 2007). The human conceptual system goes beyond representation of experienced events but instead it utilizes relevant components of stored events during unfamiliar, atypical situations. Conceptual representations become vital when heightened awareness of situations is required, because the mental prototype formed from the human conceptual system provides information used in all aspects of situation awareness.

In addition the human perception also adjusts its operation depending on the current situation (Caserta, et al., 2007). The human perception attends selectively to relevant advanced cues, enabling the human to identify the relevant situations which in turn allows an individual to react accordingly. Driving is an example of such behaviour – quick responses are less important when driving a car on a desolate highway, however when driving car in a big city the human perceptive system behaves differently. Attempting to simultaneously attend to or become consumed by all the aspects of city traffic: traffic signs, traffic lights, cars, bikes and pedestrians would be ineffective and dangerous, especially when quick decisions must be made. So the perceptive system is constantly adjusting itself depending on the perceived situation which changes according to the information acquired via the perceptive system.

The declined perceptive capabilities of older people inhibit the situation awareness more than other factors as the capabilities for situation identification do not decline substantially provided the correct data has been acquired from the environment. Whereas perception and comprehension of information can be trained the projection of future that follows from the comprehension can not be directly trained.

2.3 Distributed human SA

Stanton et al (Stanton, et al., 2006) developed a theory of Distributed Situation Awareness (DSA) which is based upon six basic propositions:

- SA is maintained by both human and artificial agents.
- Communication between the agents may also be in the form of non-verbal behaviour, customs and/or practice.
- Application of non-overlapping or overlapping SA depends on the agent's goals.
- There are multiple facets of SA associated with the same scene as observed and interpreted by different agents.
- One agent may compensate for degradation in another agent's situation awareness. SA assists in holding loosely-coupled systems together.

This theory of DSA is applied in the context of a human-machine agent team operating on board of a RAF Boeing E3D Sentry AWACS (Airborne Warfare and Control System) aircraft (Stewart, et al., 2008). Researchers spent time on board of the Boeing E3D to collect data using the EAST (Event Analysis of Systematic Teamwork) methodology (Walker, et al., 2006). The collected data has been analyzed to produce three main representations of a system: a social network, a task network and a knowledge network. All of these representations offer different, but associated, aspects of systems representation. The social network represents the highest abstract level, representing the communication relations between people in the system. The task network expresses the relationships between the goals of different agents in the system and the knowledge network shows the relationships between classes of information in the system that are required to perform the tasks effectively.

The model of DSA proposed by Stanton is extended by Stewart et al. with reference to Endsley's conceptualisation of SA and the results show that some individuals, as part of the team, are engaged in perception tasks (such as the surveillance operators and surveillance controllers), some in both comprehension and in projection tasks (such as the tactical director of the mission and the fighter allocator). It is clear that in such a system it is not sensible for all agents to share all information – the agents involved with lower-level tasks in the SA hierarchy do not need all of the situational information generated at the higher levels. In (Perry, 2003) it is asserted that complex problem-solving systems have their own cognitive properties (including SA) that cannot be identified by individual cognition, and studying a complex agent system at the individual agent level will fail to pick up on these system-level features. It is obvious that there are two aspects of SA for any agent: individual SA for performing the agent's task, and a higher level 'meta-SA' for forming the whole system's DSA required to achieve the goals of the system. Observations made during the study also indicate that there is a lot of collaboration occurring during the mission although there is both an informational and organizational hierarchy in place.

The Boeing E3D crew analysis example is concluded by Stewart et al "*from the study of the crew of the E3D Sentry it can be seen that the proposed theory of DSA works equally well with a large multi-person human–machine system as it does with a single person–machine systems as DSA is concerned with how knowledge is used and parsed between agents when interacting with a system*" (Stewart, et al., 2008). There is no doubt that the performance of an agent system will be most effective when there is 'good' DSA throughout the system as a whole, so care must be taken to ensure that system-level SA is generated correctly and exchanged in a timely manner.

Distributed human SA can be also considered in the context of modelling distributed sociotechnical systems – complex systems that consist of people, devices and software agents in a changing environment. In general terms it can be said that sociotechnical system modelling combines aspects of cyber-physical systems with human aspects, resulting in a more or less complete system description. Sterling and Taveter in (Sterling, et al., 2009) propose the *viewpoint framework* for modelling distributed systems. The *viewpoint framework* proposes three views to a system, namely interaction, information and behaviour view. The *information view* of the viewpoint framework can be clearly associated with situational information generation and storage at agents in the context of distributed situation awareness. In the same manner can the *interaction view* be associated with the exchange of situational information while the *behavioural view* deals with the behaviour of agents in specific situations. As agent modelling frameworks, which the viewpoint framework is, tackle the problem of designing reliable and scalable agent systems the issues of situational data generation, representation and exchange are not addressed in detail. Nevertheless, as situational information is an important aspect of agent systems the problems related to agent system design must be understood when dealing with situational information generation, representation and exchange.

2.3.1 Temporal and spatial aspects of human SA

Temporal aspects of events perceived by the observer play an important role in forming SA. However in the domain of human SA the temporal aspects are viewed in a simplistic way. The temporal and spatial aspects of SA are not explicitly handled in the context of human SA as humans are believed to be able to make the temporal and spatial associations fairly easily. The challenge is mainly in providing the correct data to the human. The human cognitive processes have been analyzed to some extent in relation to temporal aspects of maintaining SA. The temporal aspects are mainly viewed from the standpoint of level 3 SA (projection) – what is the length of time interval until an event occurs and how soon will an element have an impact on the operator’s goals and tasks (Endsley, et al., 2000). In the context of human SA time is also part of level 2 SA (comprehension) as the temporal relations between events allow to understanding the situation from the observed events. The rate at which information changes is also part of SA as it allows to understand the nature of the situation and enables the future projection of situations (Endsley, 1988).

Temporal aspects of SA are important in the context of the dynamics of the situations – as the situation changes over time the person's SA must change accordingly or be rendered outdated and inaccurate. In highly dynamic environments the human must adapt many cognitive strategies for maintaining up to date SA (Adams, et al., 1995).

2.3.2 General approach to human SA

Although the elements of SA vary in different domains, the nature and mechanisms for achieving SA can be generalized (Endsley, et al., 2000). A general definition for SA is "the perception of the elements in the environment within a volume of time and space, the comprehension of the meaning and the projection of their status in the near future" (Endsley, 1988).

The SA can be organized into several levels according to (Endsley, et al., 2000).

Level 1 SA – perception. Perception of information is vital to achieving any SA. Without perception of relevant information it is very difficult, if not impossible to achieve a good SA. In (Jones, et al., 1996) the authors find that 76% of SA errors in case of aircraft pilots are due to problems in the perception of relevant information (which again are caused by problems with the system design or with the cognitive process of the pilot).

Level 2 SA – comprehension. Perception alone is not sufficient for building good SA. Comprehension includes the process of combining, interpretation, storing and retaining information. Clearly this goes beyond perception – comprehension of information means that multiple pieces of information (received from multiple sources) must be integrated and relevance of the information must be evaluated within the scope of the current situation and the goals of the person (Goldin, 2006). According to (Jones, et al., 1996) 20% of pilot errors are related to level 2 SA. In (Flach, 1995) the author claims that "*the construct of situation awareness demands that the problem of meaning be tackled head-on. Meaning must be considered both in the sense of subjective interpretation (awareness) and in the sense of objective significance or importance (situation).*" In order to achieve level 2 SA the situational data collected at Level 1 must be processed and operationally relevant meaning and significance of the data must be extracted from it.

Level 3 SA – projection to the future. The highest level of SA is the ability to foresee the future state of the significant factors and the dynamics of these factors. Essentially it is the ability to project future events from the available past and present data, which experienced operators do on a regular basis.

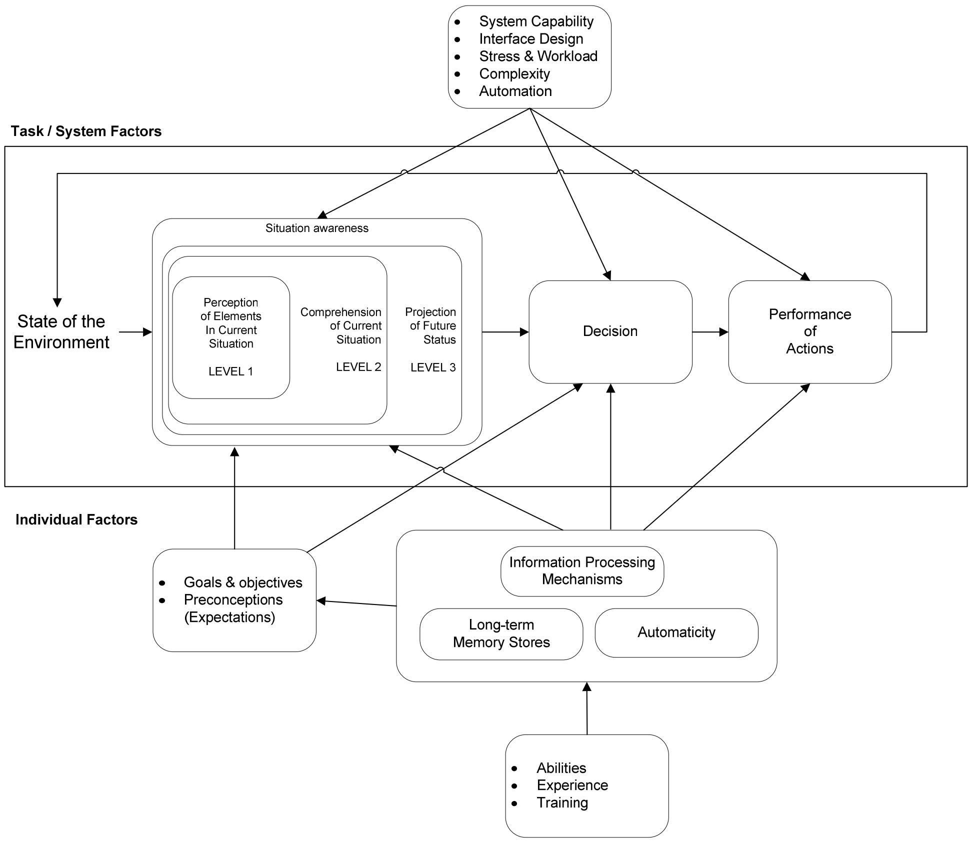


Figure 2 Model of SA in dynamic decision making (Endsley, 1995)

Figure 2 depicts a general model of SA generation and application by Endsley (Endsley, 1995), outlining also the different levels of SA. Clearly both the task / system factors and the individual factors affect the generation of SA, the decision making process, and the evaluation of the performance of the actions. In that respect the situation awareness generation and application cannot be taken out of context, i.e. one must consider the individual and task/system factors when looking at different aspects of SA.

2.4 Situation awareness of computing system

The research in situation awareness of computing systems originated from the research related to human aspects of computing and human computer interfaces. Therefore much work in the area of computer situation awareness is related to either aiding humans to achieve better situation awareness or adjusting behaviour of computers depending on the situation of the human user. Since the issues of human SA (and how human SA can be improved with the aid of computers) were discussed in the previous section, the overview of SA in this section focuses more on SA relevant to computers. This section starts with an overview of state of the art in computer SA and continues with a new approach proposed by the author.

2.4.1 State of the art in computer SA

As stated above much work in computer SA is related to humans so overview starts with human centric approaches. In (Intille, et al., 2004) the authors attempt to detect the human context with the aid of appropriate sensors and interpretation algorithms. The objective of detecting the human context is the desire to adjust the behaviour of the computing system according to the human user context. The authors claim (quite rightfully) that the research performed in the field of machine learning (computer vision, speech processing) can be utilized to recognize relevant contexts using automated methods. Researchers in many computational perception domains (such as computer vision, speech processing and machine learning) have developed supervised learning algorithms that are highly effective in recognizing complex activities by computer. These algorithms could be also employed in situation detection provided that the contexts of interest are specified suitably and the algorithms are trained appropriately.

A human centric approach is also presented in (Abowd, et al., 2000), where the authors suggest that "*Context-aware applications, however, can use sensors to infer a user's activity to automatically determine good times and places to proactively present or request information*". The suggested approach of adjusting the presentation of information to the user depending on the context is certainly interesting. The suggested method for identifying user context relies on context identification based on data acquired via sensory input. The sensory input acquired at runtime is matched to data collected during training sessions. In order to perform such a matching sensor data is collected during training sessions and then associated with activity labels assigned by the users during the training session.

This is believed to be sufficient to construct computational models that capture the variability in the training examples and the uncertainty in the sensor measurements. It is expected that comparing new sensor readings with the models is sufficient for context classification using maximum likelihood reasoning or other statistically-based matching techniques. Abowd et al (Abowd, et al., 2000) claim that with good training sets, supervised learning techniques can be significantly less brittle than context detectors that employ hand-constructed rule-based models. The authors also expect that supervised learning techniques can be used to create automatic context detection algorithms that can be customized to individual users in the field by training the algorithms on user specific datasets.

In (Gellersen, et al., 2002) the authors attempt to bring some order to the different levels of context abstraction. The authors limit the context recognition only to the information inferred from data gathered from the physical surroundings of the system via sensors. They identify three levels of abstractions for context: 1) the real world surrounding the system, 2) an aspect of a situation and 3) a specific occurrence of an aspect, such as a specific place. The authors claim that they use the term situation in reference to the real world and the term context for a model of a situation acquired by means of sensing and sensor data processing. The authors fail to explain how the information about the real world is acquired, it seems that they assume a god-like view, assuming that their (or human in general) perception of the real world is correct and we should attempt to achieve the same with machines (computers). Gellersen et al (Gellersen, et al., 2002) do anticipate communication between computing systems where one system could obtain context information from another (via the infrastructure).

However only two distinct cases are described – in one the computer acquires all the information required for context identification by sensors attached to it (direct context awareness), in the other case all the context information is acquired from the other computer(s) (indirect context awareness). For some reason the authors do not consider fusion of information collected locally and acquired from peer computers. The authors limit their research to autonomous computers (in terms of context awareness) as the dislocation of context acquisition from the context use is considered a disadvantage – in that case a computer relies on information acquired from external sources for determining its behaviour.

The simplistic approach presented in (Yau, et al., 2003) assumes that a situation triggers an action directly. The authors represent a situation as an expression on previous computer-action record over a period of time and/or the variation of a set of contexts of the computer over a period of time with respect to the application

(Yau, et al., 2002). The process of evaluating a situation is a one-time operation – the situations are described in the form of rules and the validity of a rule can be evaluated at any point in time, the results of the evaluation are not stored.

In (Matheus, et al., 2005) the authors attempt to tackle the situation awareness problem with semantic web technologies. Unfortunately the authors view the problem in a rather limited scope as a data fusion problem. The authors claim that "situation awareness is a fusion problem involving the identification and monitoring of higher-order relations among object-level objects", assuming that a user is able to define the constraints on situations. This in turn means that the user must able to discretely specify situations, i.e. the mapping from the domain of the situation identification function to the co-domain of that function (i.e. the mapping from the input to the output values of the function). Although the authors claim that their approach has an emphasis on handling time the article does not illustrate that. The authors have conveniently neglected the temporal and spatial aspects of situational data. In the context of such oversimplifications the presented approach may even work.

In (Matheus, et al., 2003) a formal framework for the SA is presented. According to that definition SA is the knowledge of the following:

- *A specification of the Goal theory, T_g;*
- *An ontology, i.e. a theory T_O of the world;*
- *A stream of measurements W1, W2… for time instances t1, t2,…;*
- *At each time instance, the fused theory $\tau^t = \nabla_T \left(T_1^t, T_2^t, ..., T_n^t\right)$ that combines all the theories that are relevant to Goal Tg as well as the fused theory $\tau^{t+1} = \nabla_T \left(T_1^{t+1}, T_2^{t+1}, ..., T_n^{t+1}\right)$ that combines all the theories that are relevant to the Goal Tg at some time t+1 in the future;*
- *At each time instance t, the fused model $M^t = \nabla_M \left(\mathrm{M}_{1.1}^t, \mathrm{M}_{1.2}^t, ..., \mathrm{M}_{2.1}^t, \mathrm{M}_{2.2}^t, ...\right)$ that combines all models relevant to the Goal Tg, as well as the fused model M^{t+1} at some time t +1 in the future and*
- *Relations $R^t \subset O^t \times O^t$ relevant at time t, as well as at t + 1, $R^{t+1} \subset O^{t+1} \times O^{t+1}$ among objects (here only binary relations are considered but the formalism can be extended to include relations of higher arity).*

The definition supplied by Matheus et al has some rather substantial simplifications – the authors take a one shot, top-down view assuming that all information is available at the top level where the analysis is performed. The evolution of situations is not considered; rather each situation is evaluated as a standalone entity. In order to perform the situation assessment the authors propose the use of an ontology which describes the critical objects and relations between the objects from the situation assessment perspective.

Probably because the presented approach is application driven the authors take the view that every situation must have a goal, which defines the objective of the situation and also identifies what is relevant in the current situation. The approach of situation specification being based on goals is applicable when the goals can be specified upfront, however the goals of individual computers in a larger system may be contradictory, and the addition of new computers to a system may also add new goals which the existing computers are not aware of.

The authors conclude that while the proposed methodology provides a way for formally deriving higher-order relations in a situation, the applicability of the methodology for real life real-time problems is not clear. They state that "even modelling simplistic dynamics ... becomes a non-trivial task for large ensemble of objects." Awkwardly the authors worry about the relevance of the generated data to the user after the data has been generated (although the main focus of the application they are targeting is improvement of human SA). Maybe starting the analysis from the end objective (i.e. relevance of data to the user) would have provided better results.

Schmidt in his work (Schmidt, 2002) focuses on context awareness of computing systems in the realm of ubiquitous computing systems. Schmidt focuses mainly on recognizing human context, making the context awareness problem essentially a user interface design issue. The reason of identifying the context as perceived by humans automatically by a computer lies in the objective of adjusting the behaviour of a computing system based on the current context of the user and thereby providing a better service improving the user's perception of his work and also ultimately making the process of using a computing system more effective and pleasurable.

An interesting approach to achieving SA using agent in the context of urban threat evaluation is presented in (Lewis, et al., 2009). The authors use a hierarchical approach to attaining situation awareness with lower-level perception handled by event-correlation agents that share the information with situation assessment agents that collectively identify potential threat situations. The authors propose an architecture where the event correlation agents post events inferred from the source data to the higher level situation assessment agents that, based on the received information, attempt to determine potential threats. Also information flow in the reverse direction is possible, in which case the high-level situation assessment agents guide the work of the low-level event correlation agents. This approach bears similarity to the hierarchical approach developed by the author of the thesis and presented in the following chapter.

3 Enhancing the Situation Awareness Concept for Cyber-Physical Systems

This chapter presents the author's suggestions on how the situation awareness concept can be extended to be applicable in autonomous Cyber-Physical Systems. Partitioning situational data into situation parameters and supplementing situation parameters with validity information that makes it clear where and when a situation parameter value is valid makes it possible to construct system of systems that build upon this concept.

3.1 Hierarchical Build-up of Situational Data

As can be concluded from the overview presented in the previous chapters, situation awareness and situation management have been mostly viewed from the context of human requirements and processing capabilities. Traditionally computers have aided humans in achieving situation awareness. This has invoked systematic analysis and formalization of situation awareness since computing systems are capable to process and reason about large amounts of information that the humans cannot process.

An approach inspired by the human factors research can be used to create the concept of situation awareness for artificial agents. In case of humans, situation assessment is achieved by processing incoming concurrent streams of data provided by the sensory equipment. The incoming data is processed using stationary knowledge and algorithms developed over time. In case of artificial agents the same principles can be applied for situation assessment.

I propose the hierarchical build-up of situation parameter values from the lower level parameter values being more or less directly derived from sensor data. A situation parameter reflects a property of a parameter of interest, composing them allows to computing the values of higher level parameters of interest by using the values of lower level parameters and all the other relevant information.

In case of an artificial agent the evaluation of situational parameters is performed by executing suitable algorithms. In order to guarantee the validity of the situation assessment, the constraints are specified on the source data that guarantee the coherence and validity of source data. In order to deliver the data that satisfies the specified constraints to the situation parameter computation algorithm, the concept of mediated interaction (Motus, et al., 2009) must be used. The mediator can be compared to the channel function in the Q model (Motus, et al., 1994), as it transmits only data that satisfies the constraints set by the consumer (i.e. the situation evaluation algorithm) to the consumer (the situation evaluation algorithm). The concepts of the validity of situation parameters and the mediator are elaborated further below.

Figure 3 illustrates the concept of hierarchical build-up of situation parameters with three computing nodes (agents) involved in situational information generation and processing. The black ovals denote the sources of raw data, in case of ovals that are out of the bounds a node the source data originates from the physical world, in case of ovals that are located within the bounds of a node the source data is generated within the node. *Node A* computes the value of Parameter 4 from three situation parameter values: *Parameter 1, Parameter 2, Parameter 3 and Parameter 4*. For *Parameter 1* and *Parameter 2* the source data is acquired from the physical world, for *Parameter 3* the source data is acquired from within the node. As is illustrated in Figure 3 the parameter values can also be shared between the nodes, which do not change the logical build-up of the parameter values.

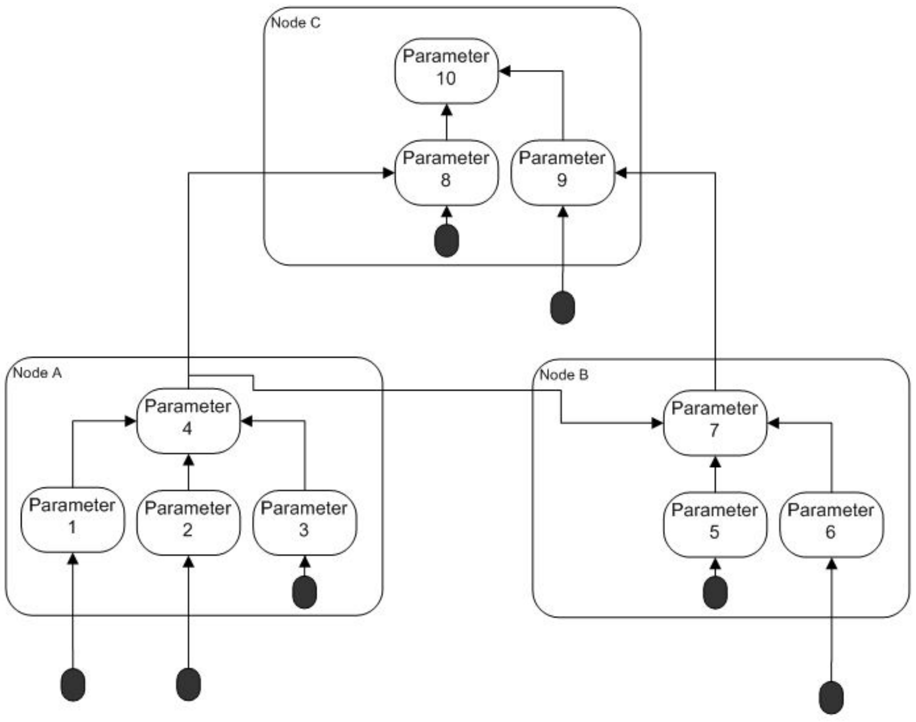


Figure 3 Hierarchy of situation parameters

The methods and hardware used for acquiring data that is the source information for situation parameter evaluation, and the algorithms used for situation parameter evaluation both have an effect on the validity intervals of the situation parameters. The specifications of sensors used, prescribes the accuracy and trustworthiness of the measurements, thereby having an effect on the properties of the situation parameter values derived from these readings. The algorithms used for processing the data have the same effect – depending on algorithms the validity of the output data may vary.

A situation in general, as perceived by an entity or by an agent, in the case of computing systems can be characterized by many situation parameters. In order to elaborate on the usage and processing of situation parameters in a computing system the situation parameters, their properties and origins must be categorized. Naturally in every situational information category there exist numerous parameters that can be evaluated individually, each situation parameter expressing the state of a computing system or of a physical phenomenon. The partitioning of situation parameters can be made based by several properties – firstly, the partitioning could be accomplished by categorizing the situation parameters into virtual (computing) and physical situation parameters.

Another partitioning can be made based on the validity area of a situation parameter. If a situation parameter characterizes the situation of an agent that is evaluating the situation the parameter can be considered local or internal, while all the other situation parameters can be called external. There may be both virtual and physical situation parameters in the internal as well as in the external parameters' set. The question arises where is the border of an agent – which parameters can be considered internal and which are external. For the physical situation parameters the border could be the physical computing system where the agent resides and for the virtual situation parameters everything that is on the processor side of the network interface could be considered local. Of course in case of multiple agents on a single computing device and in case of agents that span several computing devices the border becomes blurry.

The virtual situation parameters characterize properties that describe the features of computing agents. The values for virtual situation parameters are acquired by monitoring the properties of the virtual world. Generally these properties are hidden from the physical world (including human users), although in some cases the properties may be quite apparent – the existence and operation of computing nodes in the vicinity of an entity may be observable. The virtual situation parameters are, for example, the run times of functions on different computing agents, the availability of services on computing agents, the ability to compute situation parameter values and evaluate situations by computing nodes, network delays, utilization of processor or memory of specific computing nodes, etc. The virtual situational information is obtained by the computing nodes themselves by observing their own behaviour, through monitoring the behaviour of other agents and through direct and indirect interaction between agents by forwarding the situation parameter values generated locally by an agent.

The physical situation parameters reflect the state of physical phenomena in the real world. The physical phenomena include both inanimate and animate (including humans) objects, which in some situations may become subjects. The situation of a human user interacting with a computing system that many researchers consider important enough to be a separate category of situations (called the user context in some cases (Schmidt, 2002)) is essentially part of the physical world. Typically computing agents obtain information required for physical situation evaluation via sensors, but there may be other ways to attain such information. Agents can exchange physical situation parameter values and higher level physical situation parameters can be computed, based on lower-level situation parameter values acquired from other agents.

The fact that the values of situation parameters in various groups may be correlated in some cases does not affect the overall partitioning of the parameters. For example a physical event that is reflected in the change of some physical situation parameter value may affect the total network traffic in a network which in turn may result in increased network delays, changing the values of one or more virtual situation parameters, thereby creating a correlation between the physical and computational context properties.

Hence, the parameter values characterizing the basic situations should stem from the direct readings of sensors, direct observations or deductions made by humans, or from elementary data fusion operations. The set of basic situations (i.e. their parameters) forms the basis for constructing more abstract situations whose validity depends on the validity of basic situations.

Two directions of data flow are possible for the situation assessment process, depending on the orientation of the hierarchy of situations either the top-down or bottom up approach can be used (instead of the terms bottom up / top down also the terms forward / backward chaining can be used). The (temporal and spatial) constraints on the source data used by a situation parameter evaluation algorithm must move from the top level down to the lowest level of the situation parameter evaluation hierarchy. The situational data – situation parameter values that satisfy the constraints of a given situation parameter evaluation algorithm should, on the other hand, only move up in the hierarchy of situation parameters. The concept of data flows in the opposite directions can be viewed in Figure 4 where the source data for computing the situation parameter value moves from the left to the right (from the mediator to the evaluator) while the constraints on the source data move from the right to the left (from the evaluator to the mediator).

Backward chaining (or the top-down method) of situation parameter value propagation would not be feasible in a distributed system because of the amount of information exchange that is required in case of this technique. In the backward chaining case the source data for a situation parameter evaluation would have to be collected when the parameter evaluation algorithm is executed which in case of a distributed system essentially means a request for data that is propagated backward (or down) through the logical tree of situation parameters. Responding to the request the data would then move forward (or up) through the logical tree of situation parameters with parameter evaluation functions being executed on the logical tree.

Forward chaining (or the bottom up method) of situation parameter values is a more suitable approach to use for situation parameter evaluation. Using the constraint information propagated down from the top level the lowest level data acquisition is triggered and data on the low level is processed. The situation parameter values computed on the lower level are then communicated forward on the hierarchy of the situation parameter evaluation tree. Figure 4 depicts the lowest level situational data evaluation – the situation parameter evaluation function propagates the temporal constraint information to the mediator of the sensor data acquisition function, which then provides the situation parameter evaluation function with the data that satisfies the constraints.

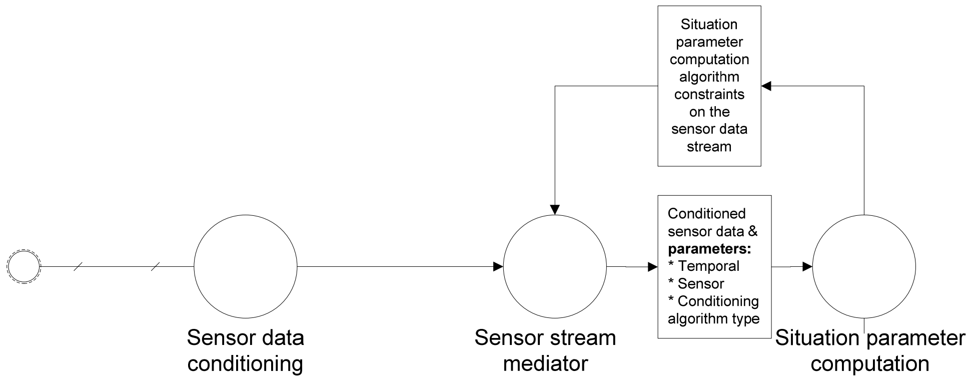

Situation parameter computation algorithm constraints on the sensor data stream
Conditioned sensor data & **parameters:** * Temporal * Sensor * Conditioning algorithm type
Sensor data conditioning
Sensor stream mediator
Situation parameter computation

Figure 4 Situation property evaluation based on sensor signal

3.1.1 A systematic approach to hierarchical build up of situations

As outlined in the previous chapter the situations can be built-up from elementary situations to higher level situations, the principles of situation evaluation (such as propagation of situation parameter values and constraints for the parameters) staying the same. Every situation parameter value has temporal and spatial validity values associated with it. The validity values depend on various aspects, for example the validity area depends on the location of the agent that acquires the data and on the properties of the phenomenon being observed, while the temporal validity interval depends both on the properties of the environment where the agent is located and on the phenomenon being observed. The consumer of the situation parameter values verifies that the validity values of the parameters do match the constraints set on the incoming data. The output of the situation assessment algorithm is the situation parameter value accompanied with the metadata.

Situation parameter is a three-tuple $S=\left(S_p,S_t,S_l\right)$ where S_p is the situation parameter value, S_t is the situation property's validity period and S_l is the situation property's validity area. The situation parameter value reflects the state of a virtual or physical property of interest while the validity period and validity area are metadata on the situation parameter assessment – what is the period for when and what is the area for which the situation parameter value holds. For some situation parameters there is no metadata while for others there may be more metadata than just the location and time. This topic of situation parameter validity is discussed in greater detail in section *Validity of situation parameters*.

A situation parameter value is computed from other situation parameters or sensor data, both of which must be associated with metadata. $S_a = f\left(S_1, S_2, ..., S_n\right)$. The situation parameter values used for computing a higher-level situation parameter value may arrive at different times and at different rates. A specification of an algorithm that computes the situation parameter value must contain the type of input data – the incoming situation parameters. If a sensor reading is used to evaluate the situation it is expressed as follows: $S_b = f\left(s_1\right)$ which should be interpreted that the reading of sensor s_1 is used to compute the value for situation parameter S_b.

The metadata (validity constraints) generation for a specific situation parameter may depend on other situation parameters – for example the location of the agent that is computing the situation parameter values may affect the computation result and the validity values of the situation parameter being computed. The data flow for different configurations for situation parameter composition is depicted in Figure 5 and Figure 6.

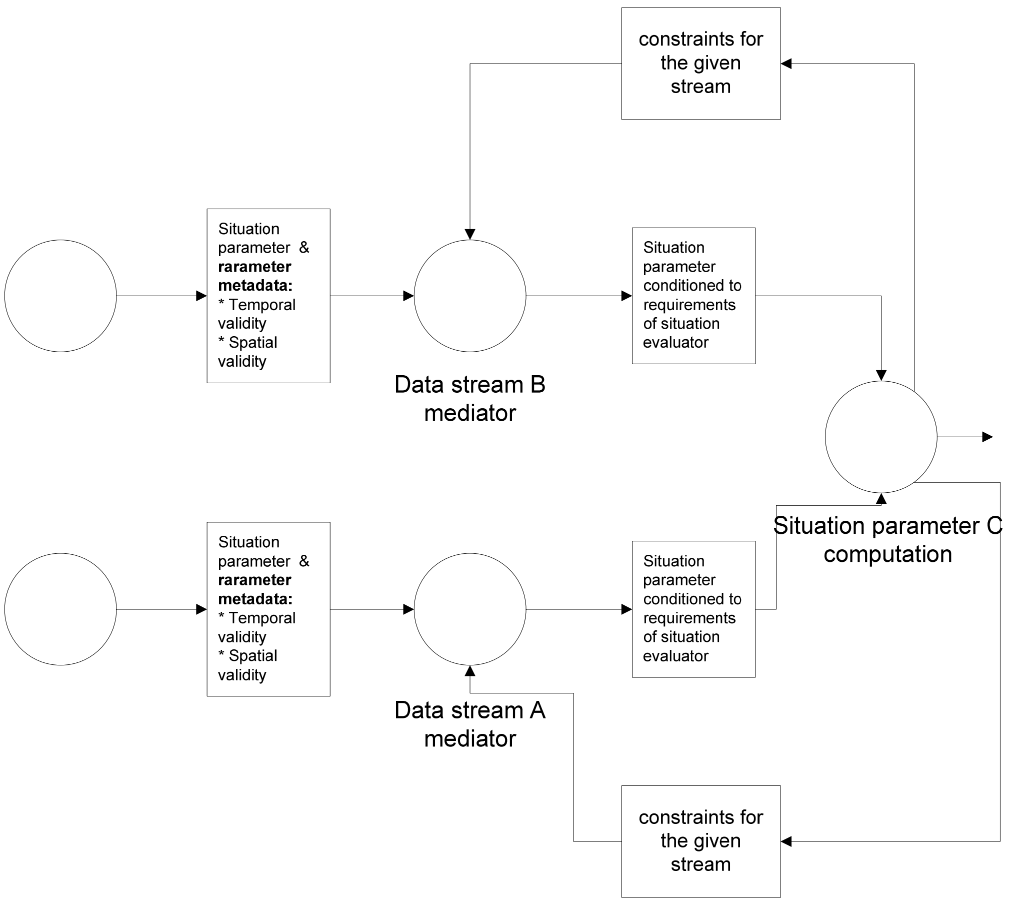

constraints for the given stream
Situation parameter & **rarameter metadata:** * Temporal validity * Spatial validity
Situation parameter conditioned to requirements of situation evaluator
Data stream B mediator
Situation parameter C computation
Situation parameter & **rarameter metadata:** * Temporal validity * Spatial validity
Situation parameter conditioned to requirements of situation evaluator
Data stream A mediator
constraints for the given stream

Figure 5 Situation parameter evaluation based on two situation parameters

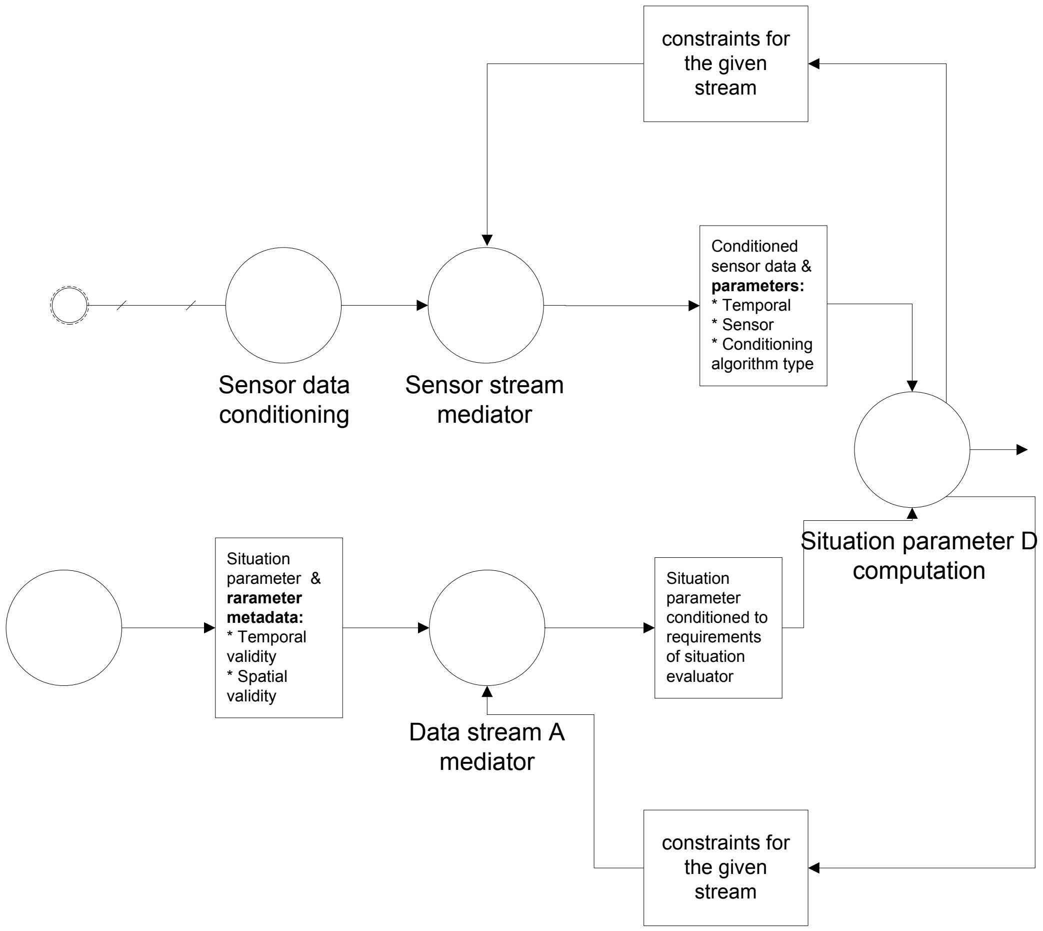

constraints for the given stream
Conditioned sensor data & **parameters:** * Temporal * Sensor * Conditioning algorithm type
Sensor data conditioning
Sensor stream mediator
Situation parameter D computation
Situation parameter & **rarameter metadata:** * Temporal validity * Spatial validity
Situation parameter conditioned to requirements of situation evaluator
Data stream A mediator
constraints for the given stream

Figure 6 Situation parameter evaluation based on a situation parameter and sensor signal

For each situation parameter computation algorithm additional rules (algorithms) are specified for computing the metadata (validity) values associated with that situation parameter.

A reverse transformation is also possible – if it is known that a situation parameter (e.g. parameter *A*) has to satisfy certain validity constraints, it must be possible to determine what the actual values of those constraints are.

So several distinct algorithms are required – one for situation parameter computation, one for computing each situation's metadata item and also one for deriving each of the constraints on the situation metadata items. The situation metadata is derived from situation parameter values and the metadata (validity intervals) of the situation parameters in the input data stream to the given situation parameter computation algorithm.

If data satisfying the constraints is not available the interaction mediator may adjust the data to satisfy the requirements of the situation parameter computation algorithm. The past values of a situation parameter can be utilized to predict the future values of situation parameters. Depending of the phenomenon that a situation parameter describes the methods for utilizing the situational history of the parameter for the prediction of future values of the situation parameter may be different. Even quite thin embedded nodes are able to perform the predictions on situation parameters based on observed situational histories (called context histories in (Helander, et al., 2005) and (Preden, et al., 2006)), since in (Helander, et al., 2005) it is shown how even quite simple mathematical models suffice to predict the future values of situation parameters (context parameters in (Helander, et al., 2005)), such as execution times of scheduled functions, with quite good results. It has been suggested (Preden, et al., 2006) that relatively simple stochastic or statistical methods (when compared to formal mathematical analysis methods) similar to the methods of technical analysis used in economics (Mamaysky H., 2000) should suffice for most cases of situation parameter prediction.

For example interpolation or extrapolation can be used to compute situation parameter values that satisfy the constraints of a situation parameter computation algorithm. The mediator has to warn the data consumer if the constraints are not satisfied. More advanced algorithms can be used if available and required for coping with the problem of unsatisfactory situational information.

3.2 Validity of situation parameters

A critical issue of situation awareness is the validity of situations – when a situation parameter value is computed or when a situation is identified, it must be possible to estimate where and when that situation parameter is valid. This estimation must be performed based on the validity of the source data – starting from the raw sensor data a situation inferred from this sensor data is only valid in the region where the sensor data is valid and also for the period of time when the sensor data is valid. So the properties of the situational parameters (regardless of whether these are parameters of the physical world or the virtual) that are monitored determine the validity of the situations inferred based on these parameter values. An example of this is weather monitoring – for most regions we are able to predict the dynamics of temperature change and the area where the temperature is homogeneous quite well, so based on temperature measured in one spot at one specific moment in time we can quite well estimate the temperature for the adjacent regions for some period of time.

An important factor of situation parameter validity lies in the fact that the use of situation parameter values that are not valid (i.e. do not satisfy the temporal or spatial constraints of the consumer) is in some cases even worse than not having the data if the consumer of the situation parameter values is not notified. It can not be expected that situation parameter values that are computed based on lower-level situation parameter values that are not valid, are correct. If any of the constraints on the source data are not met for the computed situation parameter values should be associated with low confidence estimations. It is up to the consumer of the data to decide if the parameter value can be used or not.

The temporal and spatial aspects of situation validity must be also considered from the perspective of actuation. If the computing system is able to exercise some control over the system that the computing system is part of then that control is exercised in order to change the state of the system (the situation). So there must be a separate class of higher-level (or super) situations that arise in combination with control – the super-situation changes after some control action has been exercised and when the observable situation does not change in the desired direction after some time has elapsed the super-situation changes again to reflect the possible failure of the control action. This actuation example illustrates the problem of synchronizing the internal state of the computing system with the actual state of the (physical) world that the computing system interacts with. Using the situation awareness concept to keep track of both states and describing the relations between these situation allows the designer (or the operator) of the system to manage the irregularities between these parts of a system.

An aspect of situation parameter validity also originates from the fact that the measured values and the estimates stemming from observations originate from different network nodes, different sensors, and from variety of persons and need to be checked for consistency and validity. This can be done automatically only if the measurements and observations are equipped with attributes that foster the on-line validation procedure.

The data acquired from each sensor (or other source of information) forms a data stream, from which the situation parameter values are derived. The temporal (plus spatial and other characteristic) properties of each stream element are unique and determined by the phenomena being measured and/or monitored, by the requirements and/or constraints on the situation parameter, and by the constraints set by higher level situation parameter synthesis algorithms that use the given situation parameter as an input. Any algorithm computing parameter values for higher-level situation may use one or more data streams.

It is intuitively obvious that for each situation its parameter values are valid only for a certain time interval, and/or in a specific location. Hence the temporal intervals, spatial areas, and may be some other attributes must be specified for cross-checking the integrity of parameter values, and thus assess the validity of obtained situational information. The temporal validity interval of the situation parameter specifies the time for which the situation parameter value is valid – the validity interval depends on the time when the situation parameter assessment was made and the known dynamics of the parameter (the property which state that the parameter expresses). The spatial validity interval specifies the area where the situation parameter is valid, which naturally depends on the spatial origin of the source data used for deriving the situation parameter value and the properties of the phenomena that the parameter characterizes. For example a temperature assessment of warm for a room can be typically considered to be valid only for that room not for example the back yard which can be viewed from the window of that room – the validity area of the temperature assessment may ends at the wall of a room.

The validity area of situation parameters can be also characterized in terms of an agent – a situation parameter characterizing the (physical or computing) properties of an agent is only valid for or within that agent, regardless of the location of the agent itself. If a situation parameter characterizes the properties of an agent the parameter can be considered a local or an internal situation parameter while other situation parameters can be called external situation parameters. The internal situation parameters can characterize various aspects of an agent, for example the assessed quality of the measurements made by the agent (which may depend on sensor calibration for example), the accuracy of the algorithms used for data processing and so on. There may be both virtual and physical situation parameters in the internal as well as in the external set. In case of situation parameters internal to an agent the problem arises of what is the border of an agent – a multi-agent system might be viewed as one agent at a higher level of abstraction, so in this case the “internal” situation parameters are actually valid for a group of agents.

The parameter values determining the basic situations are measured or computed by agents in the network. The output information provided by each agent (a node in computer network, or a human) is considered to be a stream. Each element of this stream is to be tagged with the information enabling cross-checking and validity assessment of its value. This information enables the correct interpretation of the value of situation parameters, and properly to assess the impact of this situation on the behaviour of decision-makers.

The tagged information accompanying situation parameter values is derived separately from the parameter value itself. Specific sensors, additional supporting hardware, algorithms, and computing power – e.g. for time measurement and synchronization, for positioning, for estimating stress in humans, and for on-line estimation of communication delays – are often required. This part of computing system deserves more attention in practice.

3.2.1.1 Situation Awareness Levels for Artificial Agents

Now that the principles for the hierarchical build-up of situation parameters and also the basic mechanisms for generating, collecting, validating and exchanging situation parameter values have been outlined we can have a look at how these parameters could be classified. The multi-level approach introduced by Endsley for human situation awareness (Endsley, et al., 2000) can be also employed for classifying the situation parameter hierarchies of artificial agents.

3.2.1.1.1 Level 1 Situation Awareness - perception

Level 1 SA is about data acquisition and perception of the acquired data. *Heisenberg has said* (Heisenberg, 1958) *"We have to remember that what we observe is not nature itself but nature exposed to our method of questioning"*. The issue to note here lies in the fact that in order to perceive anything the right type of data must be acquired first and ultimately an artificial system can only perceive what it has been designed to perceive (even if the system does possess self learning capabilities, learning capabilities of artificial systems are more strictly limited than those of humans). In case of systems that possess learning capabilities the systems is only able to perceive the phenomena that is relevant to the system.

Data from the physical world can be acquired basically from two sources: local sensors directly interfaced to an agent, data acquired via sensors at other agents. Besides the data acquired from the physical world there is data from the computing environment. In addition there is stationary data, which may be inserted into the agents at design time, or acquired at run time – the stationary data does not change over time. The level 1 situation parameter values that are computed by an agent depend greatly on the sensor data that is acquired by that agent itself since communicating raw sensor readings over the network is bandwidth (and therefore also other resource) consuming. Although situation parameter values computed from data acquired from the sensors is communicated across the network, but usually not the raw sensor readings themselves.

Acquisition and perception of data is also dependent on higher level situation parameter values. Depending on higher level situation parameter values and the requests for data from the higher levels specific types of (sensor) data may be acquired. Also different data processing algorithms for the perception level may be selected depending on the dynamically changing requirements. This means that the perception, perception or interpretation of sensor data is not necessarily stationary, i.e. the respective algorithms may be changed when higher level situations or required parameters change.

As outlined in the beginning of the chapter the data generated at all levels of situational data processing must be accompanied with validity information, which specifies constraints on temporal, spatial and any other properties of the data.

3.2.1.1.2 Level 2 Situation Awareness – comprehension

Level 2 SA deals with the comprehension of the data generated/acquired at the lower level. The situation parameter values that are computed at level 2 reflect higher level concepts (as compared to situation parameter values computed at the perception level).

The higher level situation parameters that are computed from lower level situation parameters by using specified algorithms, while the lower level situation parameters are computed from incoming raw sensor or other (not interpreted) data. The level 2 situation parameter values computed in an agent depend on the perceptive capabilities of that agent. The reason for this is economical (considering, for instance the price of consumed communication bandwidth), and any agent tries to make use of the situational parameter values where they are collected. This is not a strict rule – in some cases data perceived by the other agents can be used.

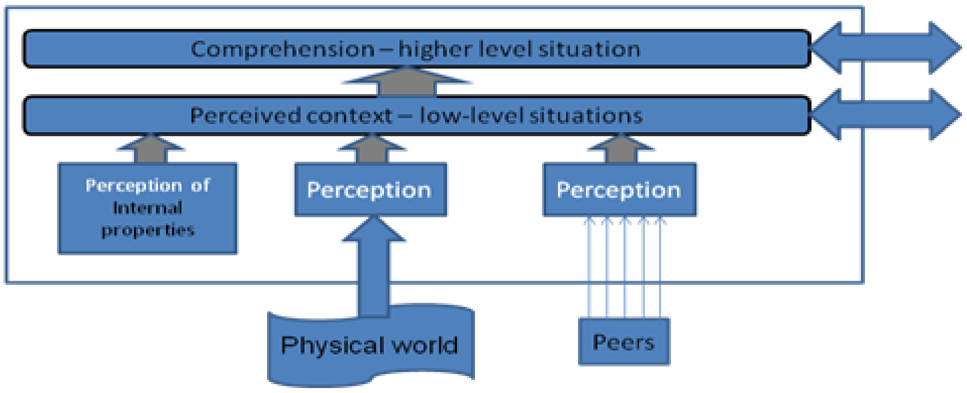

Comprehension – higher level situation
Perceived context – low-level situations
Perception of Internal properties
Perception
Perception
Physical world
Peers

Figure 7 Situation awareness levels

In Figure 7 the configuration of the situation parameter hierarchy and the data exchange required for realizing the hierarchy is depicted. As shown on the figure the agents can exchange situational data (situation parameter values) at different levels, i.e. an agent can send a level 1 situation parameter value to another agent which uses it to compute a level 2 situation parameter value.

Regardless of the situation parameter level for every situation parameter computation algorithm the temporal and spatial constraints are specified. In addition, an algorithm for evaluating the temporal and spatial validity intervals computes validity values for the data in the incoming stream of situation parameter computation algorithm.

It is clear that the situations that are relevant and detectable by an agent depend on the configuration (properties) of the agent. Some agents may not be able to detect some situations because of lack of sensory data, processing capabilities or processing algorithms. An agent should only try to identify situations relevant to its functionality. If the agent does not identify the required situations its functionality will be most likely impaired, if an agent detects more situations than required it is wasting resources. So for each agent it must be decided what are the relevant high-level situations and what are the low-level situations required for identifying these high-level situations. Once this has been decided, the sensory requirements of the agent (hardware requirements), list of required capabilities for each agent, and the properties of the agent federation that the system needs can be specified.

3.2.1.1.3 Level 3 Situation Awareness – projection

Level 3 SA is about projecting the future state of the computing system and its environment based on the past and current states. It is clear that this kind of reasoning is quite complex even for humans and to build such capability into artificial agents is even more complex. Clearly such capability is very useful and even required in some cases. Especially for systems that need to take proactive action i.e. they need to predict the future state of the world and act before that state is reached, or try to find actions that would result in the preferred future state. Research in this direction has only recently started and tangible results can be expected in the future.

3.3 Realizing situation aware spatially distributed mobile computing systems

3.3.1 Team situation awareness

In many practical systems a loosely interacting team of humans and computers has to manage situations composed from a variety of spatio-temporally distributed natural and artificial components. Each entity in such a system can be viewed as an agent on a certain level of abstraction. Situation awareness of an agent, in the current context, comprises the confidence in the available situational information and the ability to use that information in the decision-making process of the agent. In a realistic system it can be assumed that global situations are defined for the whole system – e.g. common operating picture or local operating picture – whereas each agent may have derived its own situations, departing from its own goals.

The existence of several types and levels of situations may be confusing, unless they are (at least partially) harmonized and prioritized. This leads us to the notion of "team situation awareness" (also called "distributed situation awareness"); see, for instance (Salmon, et al., 2007). Formation of team situation awareness needs specific tools, especially when dealing with a mixed team of humans and machines. The first step is to harmonize models of "individual situation awareness" of team members, followed by harmonization of individual decision-making procedures. Typically this process requires negotiations between decisive team members and can be remarkably accelerated by a suitable negotiation medium, such as a specially designed middleware for communication.

The above definition matches with seminal interpretation, given by Endsley (Endsley, 1988), of situation awareness as a product resulting from a process of acquiring situational information and its assessment.

Team situation awareness concept (Artman, et al., 1998) (Salmon, et al., 2007) fits perfectly the problem of creating and distributing COP and LOP at the group and intermediate level. In such complex collaborative environments the focus should be on team situation awareness which comprises team members' individual situation awareness and extends it by imposing rules for harmonizing and prioritizing the individual situations and goals.

The emergence of team situation awareness can be fostered by elaborating the concept of proactive middleware that supports communication between team member-agents (humans, computers, software-intensive devices, smart sensors, etc.). At the same time the middleware should have several advanced features, such as automatic validation of the transferred information, personalizing messages so as to match the processing capability of the addressees, tracking the location of team members, remembering the interest points of team members.

3.3.2 Situation-aware multi-agent system

In case of a cooperative situation-aware multi-agent system the design of a single agent must encompass situation awareness both at the agent level and also at the multi-agent community level. Clearly just collecting arbitrary situation parameter values and determining some situation based on this data is not very beneficial – agent behaviour should be also dependent on perceived situation(s). In case of a mobile agent or a mobile multi-agent, the design of the agent combines several control and location aspects, in addition to usual aspects of situation awareness, so the design of such a system is non-trivial.

It must be noted that there is no universal situation that would be interpreted by all agents in the same way and that would cause the same behavioural decisions in all agents. Two mobile agents can come up with two different assessments of a situation and respond differently to it, depending on their capabilities, goal functions and on their situation assessment algorithms.

An example of this is the interpretation of terrain types by different mobile agents. A swamp may be interpreted as not a suitable terrain for a tracked vehicle while it is very suitable, even preferable (typically a swamp has less obstacles so the speed of movement is potentially higher) for a hovercraft. At the same time the swamp is a suitable terrain for a tracked vehicle if the temperature has been low enough for a long enough period so that it has been enough time for the swamp to freeze solid. Such a reasoning process is inherent for humans and it may seem too trivial for discussion. However, in case of an autonomous mobile agent that must be able to choose its course of action required for the achieving the mission objective autonomously such reasoning is non-trivial as the number of possible aspects that must be considered in the reasoning process may be quite high. The reasoning must take into account the tactical and strategic situation, the information on the mission area available before the mission, the information on the mission area acquired during the mission such as the terrain and other conditions in the area, the past and current movement of adversaries, the movement of peers, etc. The design of such a system is quite complex and it is not possible to verify the behaviour of such a system formally.

3.3.3 An architecture for a distributed situation aware system

The principles for designing situation aware artificial agents was outlined in the in the previous sections. Although useful, the principles by themselves bear little value in terms of system specification and implementation. The concept of an architecture that builds upon the principles described in the preceding sections for a distributed situation-aware computing system is described in this section. This work was done in collaboration with Johannes Helander at Microsoft Research during my internship in the summer of 2006.

In (Preden, et al., 2006) a general architecture for building situation aware computing systems is presented. An architecture that relies on the usage of metadata for describing (among other things) the set of functions involved in a computing scenario, the interactions between the nodes executing those functions and the approaches used for monitoring the execution of those functions has been suggested in order to be able to systematically monitor and predict various situation parameters. The concept of a computing partiture (depicted in Figure 8), which is a collection of metadata about a computing scenario, is introduced as the source of information for the nodes executing the scenario.

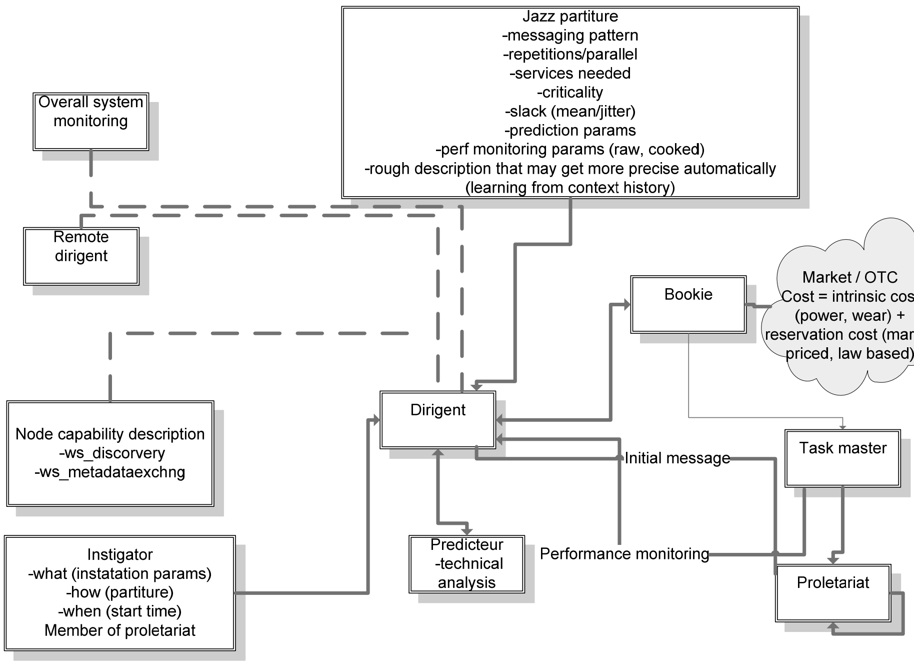


Figure 8 General architecture of a distributed situation-aware computing system

In addition to describing a computing scenario the partiture also allows describing how the context information required for a computing scenario is collected and used.

The partiture does not contain details of the implementation of the functions involved in the partiture – it only describes the functions that are involved and the metadata relevant to these functions. Neither does the partiture contain information on the specific nodes that should execute the partiture but it rather describes the functions that are executed as part of the partiture. The functions described in a partiture can run on one or more nodes depending of the details of the partiture and the availability of resources at the nodes in the given network.

The partiture describes the interactions (messaging patterns) between the functions including the timing constraints of the individual interactions – intervals of execution, mean slack and jitter of the intervals. The partiture also contains information on the possible repetitions and repetition intervals of the partiture.

In order to collect the computing situational information the partiture contains information on how the performance of the execution of the individual functions should be monitored at the nodes. The information describes how execution time of the functions is monitored, which allows a node to locally monitor the execution and later provide the performance information on the execution of functions. Based on the computing situational history the nodes can also make predictions on the future executions of functions on a node and provide these estimations to the nodes that they interact with. The partiture can also be modified according to the recorded situational history if such behaviour has been prescribed by the designer of the system.

As is the case with computing situation parameters the partiture also contains information on how the values for physical situation parameters should be computed and what models (functions) should be used to predict the future values of the physical situation parameters. Formal mathematical analysis methods are most likely not required to predict future values of situation parameters with sufficient accuracy. In addition to being computationally intensive the generation of formal analysis methods requires good information on the physical domain and the creation of adaptive and situational history exploiting systems is much more complex using these methods. Instead stochastic, heuristic, physical models or technical analysis tools are used for predicting behaviour.

As the nodes monitor the situation, add to the situational history and make some decisions based on the situational history they can also update accordingly the partiture of the computing scenario they are executing.

The architecture outlined above allows measuring different phenomena according to predefined patterns and predicting the future values of situation parameters based on past measurements of phenomena. The predicted values are used either directly or indirectly in future computations to improve the efficiency and (user-observable) quality of the systems. According to some sources (Motus, et al., 2005) these features – the ability to anticipate the evolution of its surrounding environment is one of the characteristics of proactive systems, which the invisible computing systems are expected to be.

To execute the partiture every node contains a conductor that can execute a partiture. The conductor is responsible for selecting the nodes that are going to execute the functions described in the partiture and delivering the information required for the execution to the nodes. In addition to the function and interaction information the conductor is also responsible for delivering the information on situation parameter collection and situational history utilization as described in the partiture. A conductor is also responsible for making agreements with conductors on other nodes to execute part of their partiture.

As the conductor reads the partiture and monitors progress, the situation history is also used to update the partiture itself with additional details of the execution flow. For instance the instrumentation of an executed function might reveal that there are two temporally distinct phases in the operation, such as the initial partiture prescribing reading data from a disk, and the monitoring observing that there is some computation leading to the read, then a long pause while the disk is seeking, followed by more computation. Based on the observation the disk read phase can be split into two separate operations. The situational history is thus used to evolve the problem description, allowing the original human author to use rough terms of intent and letting the system discover the details. It seems fitting to call this type of a rough partiture a Jazz partiture, given that the learning and specialization process is akin to improvisation.

The claim that even quite thin embedded nodes are able to perform the predictions on situation parameters is not unsubstantial, since in (Helander, et al., 2005) it is shown how quite simple mathematical models suffice to predict the future values of situation parameters, such as execution times of scheduled functions, with quite good results. In the cited work the authors apply normal distribution using a table based (i.e. pre-computed) approach to predict the future execution times of computations based on observed computation histories.

3.4 Middleware dedicated to exchanging situational information

The partiture concept laid out the principles for implementing situation aware distributed computing systems. Greatly motivated by the partiture concept and the underlying requirements that the partiture has on a computing system a study was initiated on proactive middleware for exchanging situational information. The work on middleware started (and is still continuing) on various aspects of the middleware which together will provide the required functionality.

Middleware is usually understood as a specific software layer in computing system that connects applications, or software components and includes a set of services that allow multiple processes running on one or more computers to interact across a network. In the context of this thesis the middleware should be a proactive “mediator” of situational information and should provide services for forming team situation awareness. Here the team is formed by agents of mixed origin; they can be humans, computers, smart sensors, and/or software-intensive devices.

3.4.1 Concept of a proactive mediator

Middleware is the mediator that supports formation of situation awareness and especially team situation awareness within a team of autonomous interacting agents. Conceptually, the middleware is a smart communication environment that provides services for filtering, partial validation, and distribution of information according to personal access rights of agents, and in preferable message formats for individual agents.

A type of mediator is required even in case of a single agent which collects its situational data from a set of sensors for determining situation parameter values. It is preferable that the situational data is tagged with the information enabling cross-checking and validity assessment of its value, and its consistency and integrity is validated (preferably) before, or at reaching the agent – this simplifies the substitution of the agent, or modification of its inner algorithms. The tagging of sensor readings can be done within a sensor or at the entry point to the mediator. The validity checks are performed by the mediator based on the constraints and requirements provided by the agent that subscribes to the data. In the case of distributed peers-agents the mediation problem is more complex since the mediator operates on multiple simultaneous streams of computation so as each of the interacting autonomous agents is trying to form its individual situation awareness, and after that harmonize it with the team situation awareness.

All the agents are peers by their "social" status, but not by their processing capabilities, and not by their preferences for incoming and outgoing message formats. Hence the basic middleware problem is the flexibility of interfaces between the agents and the mediator. Interactive digital maps can be used as the flexible interface for a wide scope of applications – the information on maps can be represented as alpha-numeric data structure, as separate information layers of a map, and as a conventional digital map.

3.5 Architecture of the middleware

The middleware has service-oriented architecture and enables easy, self-adjusting communication between dynamic collections of interacting autonomous agents – those interactions are required to form individual situation awareness of agents and to develop the team situation awareness that is represented by a dynamically updated common operating picture. The interactions may be divided into multiple groups depending on tasks and goals of interacting partners – for instance, the middleware caters for communication required for sensor fusion, communication between clusters of software intensive devices (e.g. UAV-s, UGV-s, and weapon systems) to coordinate activities, communication to reach decisions between human-human, human-machine and machine-machine groups), and others.

Examples of application-oriented services provided by middleware are:

- validation of information acquired from different sources, assigning tags to data items if necessary;
- transformation (and compression) of validated (and fused, or otherwise processed) information into the interim format defined by the middleware, in our case linking the information with an interactive digital map;
- tracking the position of agents that are linked to the middleware, and storing their position in the interim format;
- keeping track of the access rights of all the agents linked to the middleware and checking the rights during any transaction;
- remembering the preferred formats of messages, specific subscriptions for information from the agents, and processing capabilities of each involved agent;
- delivering the subscribed information and satisfying all the constraints and requirements imposed by the agents.

Another set of services in the middleware is for intrinsic use (for handling the interim data format), these services are required to create, maintain, update, and partition according to the subscriptions from, and position of the agents. In this paper we use interactive digital map as an interim data structure (see map server in Figure 9). This set of services is organized as a multi-agent system. Dedicated agents compose and decompose the designated area in a digital map into respective parts, extract specific layers from parts of the map, add and delete application oriented icons in the map as required, search and link background data about the objects and icons on the map, update the positions of mobile icons, etc.

The peculiarity of this middleware is in its autonomous and smart operation that pays attention to individual properties and requirements of the clients, and in (situation sensitive) on-line validation of the outcome of its services. This becomes possible due to application of the “mediated interaction” concept (Motus, et al., 2009). This concept is built on a situation-aware interactive model of computation (Motus, et al., 2005), with original ideas stemming from (Motus, et al., 1994), and on a well-established message exchange paradigm where consumer has to subscribe to a message. During the subscription process the subscriber specifies the properties to be guaranteed in the message (e.g. time of computing the contents of the message, position of the message sender, validity time of the message, etc).

3.5.1.1 Illustration of middleware functioning

From the application point of view the middleware implements three roles – it serves as a unifying communication media for heterogeneous agents, it enables subscription-based message exchange between (mobile) agents with subscribers' access rights control, and it validates the consistency and integrity of exchanged messages with the requirements and constraints provided by the subscribers.

For instance, if the evaluation of a given situation parameter value requires input data tagged with specific temporal and spatial values that should satisfy the constraints specified during subscription to that value, the middleware invokes the check by calling the respective service and referring to the constraints specified earlier. The middleware can propagate the constraints specified during subscription to any middleware related services. Questions about time counting and synchronization in loosely connected mobile distributed computing system are not trivial – quite often we have to deal with multiple independent metric times simultaneously. Some more information about time related problems see (Motus, 2003).

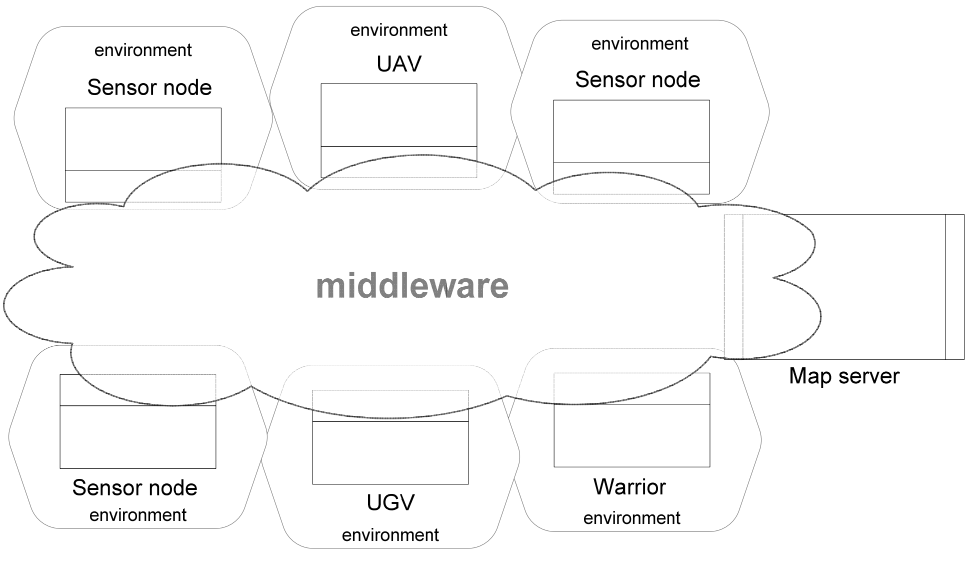


Figure 9 Middleware for forming team situation awareness of agents

An example of this is the acquisition of the weather forecast for action planning. It is clear that weather forecast for last week is not used to plan the actions of tomorrow but instead the weather forecast for tomorrow is used. In addition weather forecast for the target operational area is used instead of using a weather forecast for some place several miles away. In many cases people take this kind of information filtering as naturally guaranteed, but the natural information mediation that occurs in the social network of humans does not occur naturally in the artificial social network. In artificial social network the mediation needs to be pre-specified in details.

In the artificial social network the action planning agent requests weather forecast with fixed temporal and spatial constraints. Such a forecast is not readily available. Hence the agent tries to obtain that weather forecast from the peer agents. After the requested data arrives the validation indicates that data satisfying the constraints 100% is not available but data “close enough” is available. The middleware will return the response to the subscriber of weather forecast with a tag commenting the approximate satisfaction of constraints. The subscriber then has to make a final decision – to be happy with the approximate data, or wait for the better result.

In a multi-agent case the problem is more complex since mediation occurs in a simultaneous multi-stream computing system where multiple computing streams interact with each other during computation (violating thus basic rules of Turing computing). The middleware has to ensure that data exchanged during interactions satisfies temporal and spatial constraint sets defined by the situation parameter evaluation criteria. In order to ensure the above, middleware must provide for bidirectional propagation of situation parameter constraints – if an agent requires situation parameter value that satisfies specific constraints (i.e. a parameter is valid in a specific space area at a specific moment in time) the input data to the agent must also satisfy specific constraints. This, in its turn, means that the data providers for the agent must provide data that satisfies the constraints, and continuing along the chain of data providers, until we reach the data acquired from the environment (or from another independent sources).

The process of introducing new agents to the middleware is initiated by agents themselves. An agent introduces itself, the situations of interest and their parameters, together with the constraints to be used for validation of situation parameter values during the current mission. Then middleware attempts to discover the data providers (agents) that are able to provide the situation parameter values satisfying the supplied constraints. Once the data providers have been discovered from the network, data is requested from those providers and when the data arrive at the requesting agent, the validity of the arrived data is checked against the introduced constraints. If the data satisfies the constraints it is passed on to the agent.

4 Case studies of situation awareness of artificial systems

This chapter describes some of the case studies that utilise the situation awareness concepts described in the previous chapter. The practical examples described in the case studies illustrate how the situation awareness concepts can be applied in various application domains as the case studies range from the interpretation of sensor signals to control of mobile vehicles. The chapter is divided into nine sections that describe different concepts and case studies. The first section illustrates how the situation awareness concepts can be applied in the ubiquitous computing domain, the second section describes a sensor signal acquisition and processing scenario, the third section describes the aspects of applying smart dust motes in monitoring applications, the fourth section proposes a method of how situational information can be encoded and propagated in various networks, the fifth section describes the problems related to indoor positioning and a set of positioning related case studies, the sixth section describes a positioning and navigation method developed by the author, the seventh and eighth sections describes two mobile vehicle related case studies.

4.1 Distributed applications based on smart dust

This case study presents the rationale for building distributed applications based on heterogeneous computers and some issues related to that topic. As wireless sensor networks serve as data providers for Cyber-Physical Systems and since the practical and theoretical problems in wireless sensor networks are very close to these in CPS-s the concepts outlined in the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter are well applicable in the context of wireless sensor networks.

4.1.1 Introduction to distributed applications

Wireless sensor networks introduced in the beginning of the current century are networks of tiny embedded computers that are equipped with a wireless communication interface, some sensors and an autonomous power supply. These devices are also called smart dust motes, implying small size and “smartness” distinguishing them from mere sensing devices. The networks formed by smart dust motes are (mobile) multihop ad-hoc networks (also called MANET networks) where each node also performs the task of a router, forwarding messages from the other network nodes. In Figure 10 the layout of a multi-hop network is depicted. The black dots are the network nodes and the circles around the dots are communication areas of the nodes, the numbers next to the dots denote the identifier of the node. In order for two nodes to communicate the nodes must be within each other's communication areas. So in order for example to nodes one and six in Figure 10 to communicate, nodes three two and five must relay the messages between these nodes.

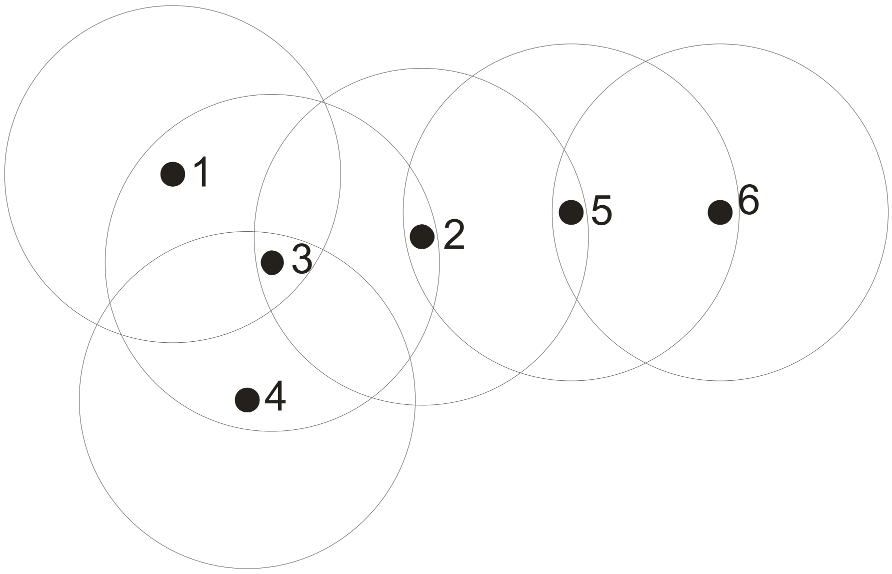


Figure 10 A sample multi-hop network

While such a multi-hop mode of communication may seem complicated the MANET networking approach allows extending the network without increasing the communication range of individual network nodes. The required signal strength for a given distance in case of direct communication and a multi-hop scenario is illustrated in Figure 11. As radio signal strength attenuation is an exponential function of the distance, the power requirement is increased substantially when the distance increases. It is generally accepted that the attenuation exponent is 2 but due to multipath and other interferences the attenuation exponent can be up to 5. As it can be deduced from the figure the total power requirement can be more than three times smaller in case of a multi-hop architecture versus the power requirement in case of direct communication.

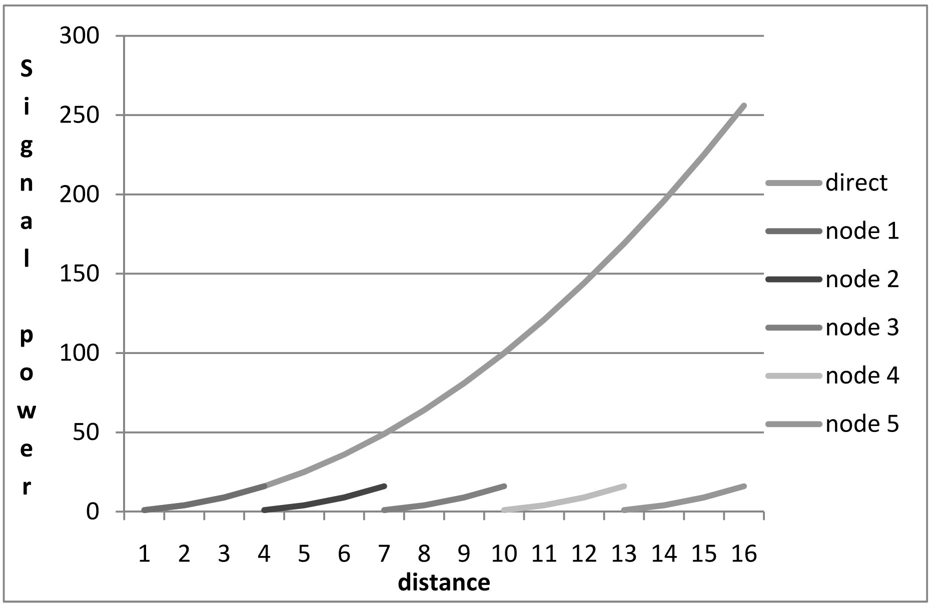


Figure 11 Required signal strength for a specific distance

A commercial MicaZ smart dust node from Crossbow Technology is depicted in Figure 12. Small form factor, low price, simple installation procedure and the offered capabilities seem to call for large scale wireless sensor network installations and wireless sensor networks are used today in a range of monitoring solutions both as research prototypes and also increasingly in industrial applications.

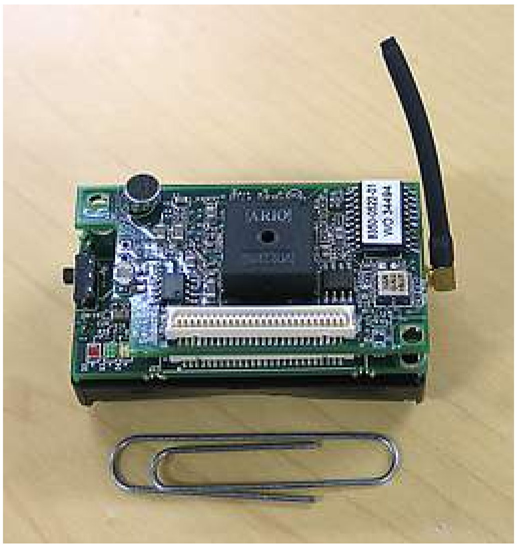


Figure 12 Smart dust mote MicaZ from Crossbow Technology

However, the current use of wireless sensor networks is not as widespread as predictions a few years ago promised. There are several reasons for this, but no single one can be called a major one. Using wireless sensor networks only for monitoring applications is promising and useful; however in most cases the advantages of smart dust do not outweigh the shortcomings when compared to established technologies. It is clear that it will take at least a few more decades before ubiquitous computing environments will be established and available, so this field is not and will not be the driving force behind wireless sensor network development. Regardless of the slow development we can still look at the wireless sensor network technology and discuss what development is needed in order for this technology to enable ubiquitous computing as envisioned by Weiser (Weiser, 1991).

Ubiquitous computing (also pervasive computing, invisible computing) concept was first introduced by Weiser in 1991 (Weiser, 1991). Ubiquitous computing encompasses computing devices in the environment connected (typically via wireless communication interfaces) with each other. Software intensive devices whose functionality is enabled only by the use of appropriate embedded computing software and hardware are already common in our everyday life. Although devices with similar functionality existed before a computer (or several computers) were integrated into them, the non-computerized versions of such devices usually exhibit reduced functionality when compared to their modern counterparts (compare for example modern cars or washing machines to their counterparts thirty years ago). It is envisioned that computing devices will be even more ubiquitous, go far beyond the software intensive devices and personal computing devices in use today. The ubiquitous computing vision foresees that tiny, even invisible computers, embedded into the environment and everyday objects such as tools, appliances, clothing, even the human body, are able to interact directly with the environment and each other, maintaining up to date information on the environment. These smart devices will determine the current situation based on the collected information about their locations, human users in the vicinity, their peers. The devices will use pre-programmed algorithms and information collected via past interactions to determine optimal behaviour for the current situation. A ubiquitous computing system is expected to anticipate the needs and expectations of a human user and act accordingly.

4.1.2 Distributed applications

Most current wireless sensor networks use a many to one data flow model where all data collected by the sensor network nodes is transmitted to a single (or a few) sink nodes (data collection centres) with little or no data pre-processing by the wireless sensor network nodes. Stemming from this centralized approach many researchers still believe that queries are the most important application for sensor networks (Sheng, et al., 2006). Essentially these approaches use WSNs as mere data collection systems where wires have been replaced by wireless links. Such an approach is suitable for data collection systems but even in this application it is not very desirable in all cases since data is accessible only via the sink. Even some of the routing protocol implementations for wireless sensor networks are only able to transmit data to a single sink node.

System architectures that rely heavily on the sink node may be beneficial in a setting where strict control over the data is required but it is quite inefficient in terms of network bandwidth utilization as all (or a very large amount of) the collected sensor data is transmitted from all the sources to a central data collection / data fusion centre via the sink node. The bandwidth usage may be optimized when nodes pre-process (the rules for data processing could be also supplied at runtime as shown in a later case study) the data locally and transmit only critical information to the sink node.

In ubiquitous computing scenarios as suggested by Weiser in 1991 (Weiser, 1991) where the network nodes must realize a distributed application in-network data processing and data exchange between the network agents is clearly required. The configuration of an application in a ubiquitous computing system is not predefined – the nodes and their configurations are not known at design time, nor is this information available at the time of deployment. All the distributed applications realized by a given system may not be known at the time when the computing system is deployed. Applications are formed at runtime and the application components are also discovered at runtime.

A property of such ubiquitous computing systems relying on MANET type networks is that nodes can join and leave (or can be added to or removed from) the network and therefore the nodes (and the lower and higher level algorithms, including applications running in the network) must be also able to adapt to such changes. An example of a dynamically formed network is the deployment of motes from an airplane (which is realistic since motes are used in military monitoring applications for sensing various physical parameters that can be used for detecting phenomena of interest).

The initial configuration of the network (the placement and density of nodes) is to a great extent determined by the way the motes fall to the ground, it can’t be even expected that all the deployed motes will be operational or that the deployed network is even fully connected. If the deployed nodes have different sensing modalities the specific sensor coverage of an area may also vary. The differences in sensor coverage stem from the fact that the area for which a specific sensor reading is valid is limited and not identical for various sensors. Because the sensing areas may be non-overlapping there may be areas for which there is no coverage from a specific sensor. As the configuration of the network is also dynamically changing (nodes may be destroyed, their power supplies depleted), the nodes must adapt to the changes in the configuration dynamically. No central coordinating entity or special nodes with special capabilities can be assumed (as all nodes are equally expendable) and therefore all the tasks in the network must be performed by the nodes themselves.

Such a dynamic system can be also envisioned in the context of future ubiquitous computing systems, for example a smart house – devices built into the house must interact with the devices that the residents of the house introduce to the house, in addition, some of the existing devices are removed as they become obsolete or as the residents become bored with them. A smart house must also interact with computers embedded into or onto the people that live in the house. When the ubiquitous computing system is deployed a resident of the house may not have a cardiac pacemaker but after the pacemaker has been implanted no manual configuration steps should be required. A pacemaker should, if required, be able to discover the coffeemaker in the kitchen and tell it to make weaker coffee because its owner's heart rate is too high.

4.1.3 Data origin

The problem of data origin must also be addressed – if node deployment is not deterministic, exact locations of nodes cannot be known until after the deployment (after a mote has fallen to the ground or after a device has been placed in the house by a resident). Node positioning can be quite complex in situations where no convenient positioning infrastructure (such as GPS) exists, which typically is the case indoors and may be also outdoors in unfavourable or hostile environments.

A common way to overcome these problems is to use some location aware anchor nodes (which location may be determined manually) that are able to position other nodes in the environment. Current positioning algorithms that rely on anchor nodes can be classified into three categories: algorithms that rely on the distance estimations, algorithms that use bearing information and algorithms that use both. The accuracy of the position estimate of all these approaches depends of the accuracy of the distance or bearing estimation. Given the fact that acquiring precise bearing or distance estimations is not possible with the limited hardware of current this is a problems that remains to be solved. A study performed on evaluating the use of received signal strength indication in described in (Pahtma, et al., 2009).

4.1.4 Situation awareness of motes

The premise for the work done involving smart dust motes in distributed applications lies in the belief that motes can be programmed to perform level 1 SA tasks and also limited level 2 SA tasks. This naturally assumes that the communication module architecture in the motes follows the approach suggested in section *3.4 Middleware dedicated to exchanging situational information* of the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter – the motes must be able to process and generate situational information and attach validity information to the computed situation parameters. Some of the aspects of making smart dust motes situation aware are discussed in the other sections of the current chapter – *4.2 Sensor signal interpretation* and *4.4 Data representation for situational information propagation.*

4.1.5 System component lifecycle

The lifecycle of embedded devices is another issue that must be considered in the design and implementation of ubiquitous computing systems. An embedded computer that functions as part of a physical object may have the same expected lifetime as the physical object which may be significantly longer than the lifetime of an average computer. However the distributed applications that an embedded computer is part of may change during the lifetime of the embedded computer. Since the embedded devices interact directly with the physical world their actions are partially also dependent on the physical world. Hence it is not possible to reason about the precise behaviour (schedulability of functions, duty cycles, bandwidth allocation, etc) of a device before the device is installed and becomes part of the physical world. It is not possible to predict all the possible combinations of applications that a device is going to be part of, or the nature of the inputs from the physical world. We can consider a simple example from everyday life – when we design a wrench we have no way of predicting what bolts and nuts that wrench will interact with during the lifetime of the wrench and by whom and how often the wrench will be used. The same way we have no way of predicting what interactions an embedded computing device will be part of during its lifetime. Instead the device must be designed and implemented in a way that makes it possible to cope and interact with the changing environment.

4.1.6 Smart space project

The Research Laboratory for Proactive Technologies has participated in a smart space project, implementing a ubiquitous computing environment prototype. The Iris smart dust motes from Crossbow Technology are embedded into the environment at fixed positions. The smart dust motes are equipped with sensors, which allow them to collect information on the physical properties of the environment. The information is converted to situational parameter values to which any device connected to the network can subscribe. The situational information (which includes the environmental conditions) collected by the nodes is also stored in the network in a distributed manner. The network of motes also functions as a positioning system utilizing a modified distributed version of the CABP positioning algorithm presented in the *Positioning* section of the current chapter. The motes in the environment provide human users with information in a context sensitive manner, as specified by the user. Human users that are equipped with a portable computer are provided situational information of interest by the mote network based on the situation of the user, which is in turn determined by the mote network. A use case scenario contains a human user that via the graphical user interface on a computer subscribes to information from the mote network in a context sensitive manner, which means that the information delivered to the user depends (among other things) on the situation of the user.

4.2 Sensor signal interpretation

The sensor signal interpretation case study deals with the interpretation of a sensor signal by a single processor. In the context of the situation awareness levels this case study deals with level 1 SA or perception – the process of computing a situation parameter value from the signal of a sensor that is interfaced to the physical world. As outlined in the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter the sensor signal interpretation may comprise more than just an algorithmic transformation of the output of the sensor signal to a value used in higher level computations. The output of the perception step (sensor signal processing) may also depend on other factors which may be expressed in values of other situation parameters. Depending on the algorithm used in the perception step the output may also depend on the past situation parameter values, which in some cases are called context histories (Preden, et al., 2006). The history of the computation may be stored internally in the situation parameter computation function, i.e. it may not be accessible outside of the computation. The case study originated in the work done by the author of the thesis on wireless sensor networks. This work is described in greater detail in (Preden, et al., 2007).

As the name implies, the sensor network nodes must process sensory input. Whilst digital systems that process data acquired from the physical world via sensor have been developed for decades already the approach based on situation awareness based presents a different abstract view on how the incoming sensor data could be processed. This approach also offers a way for solving dependencies in the data processing between data originating from various sensors and other data sources.

The interpretation of an individual sensor input (for example the resistance of a thermistor) can be viewed as the processing of an input stream of data samples. The stream is characterized by a time-set, which defines the time instances when the samples in the stream arrive for processing. This view of data item processing in a stream is motivated by the Q model (Motus, et al., 1994). The time instances when the samples in the stream arrive for processing depend on the time instance when the sensor value is sampled but in addition there are some deterministic and some non-deterministic factors that delay the processing of the data (starting from the signal input to the analogue-digital converter and ending in the retrieval of the value from the memory). The individual data samples of the stream (that arrive at different time instances) are not necessarily processed the same way since the processing of the items in the stream may depend on previous values of data samples in the stream. In addition the processing of individual data samples in the stream may depend on the external situation (values of situation parameters reflecting the state of the external world), which may change over time. For instance, the processing of the thermistor input may depend on the supply voltage when the resistance of the thermistor is measured with a voltage divider circuit and the reference voltage is not compensated for the voltage drop of the supply voltage. In this above case the voltage can be viewed as being part of the virtual (computing) situation for situation parameter computation function that processes the thermistor signal input.

4.2.1 Sensor signals as a source of situational information

For thermistor the input stream is the ADC (Analogue to Digital Converter) output from the ADC channel that is connected to the thermistor circuit and the output is the temperature. A notation similar to the Q-model (Motus, et al., 1994) can be used for denoting the interpretation of the data samples from a specific sensor:

$$temperature_t = P_t\left(T_S \times ADC\_Output_{temp\_sensor}\right)$$

where $temperature_t$ is the situation parameter value (the temperature estimation at time instance *t*), P_t is the stream function used to process the stream, T_S is the stream timeset and $ADC\_Output_{temp\_sensor}$ is the value of the data item (the output of the analogue-digital converter) corresponding to a specific time instance.

As discussed in the beginning of the section the time it takes to process each data sample is also situation dependent – due to the fact that the interpreting functions may contain internal memory, the computation time may depend on the values of current and previous data samples (in some cases the intermediate results of a previous computation can be reused).

The example can be expanded upon by considering a quite good, yet trivial example of processing the stream of data acquired from a resistive humidity sensor, where the resistance of the sensor depends (non-linearly) on humidity and temperature. The signal acquired from the sensor (typically in the form voltage which can be converted to sensor resistance) must be converted to engineering units, which in case of a relative humidity sensor is relative humidity in percentages. Since the relation between the resistance of the sensor and the engineering units (relative humidity) in non-linear the conversion may be quite resource consuming. For devices with low processing power it is common to use a table-based approach when a non-linear conversion is required. The values in the table closest to the actual sensor value are looked up and interpolation is performed between these values (linearity of the function is assumed between the table values, which provides sufficient accuracy for most applications). Table 1 is a section of the sensor resistance table of a resistive humidity sensor H25K5A. The table rows contain the resistance corresponding to a specific humidity and the columns contain the resistance corresponding to a specific temperature at a specific humidity.

Humidity	Temperature	
	20	25
30	3300	2500
35	1800	1300
40	840	630
45	216	166

Table 1 Humidity sensor resistance

In order to convert the resistance of a humidity sensor into relative humidity four look-ups must be made from the table based on the measured resistance and the current ambient temperature. When the four values have been acquired interpolations are performed to compute the relative humidity corresponding to the temperature and the resistance of the humidity sensor. To obtain the resistance of the sensor the ADC reading must be interpreted first - if a voltage divider is used this interpretation depends on the supply voltage (if the voltage drop of the supply voltage is not compensated).

Thus it can be concluded that the conversion of the elements in the humidity sensor resistance input stream to a situation parameter value (which in this case may be the engineering units – the relative humidity value) depends on temporally and spatially matching elements in two other situation parameter streams - the temperature situation parameter and the supply voltage situation parameter. The voltage and temperature situation parameter values form the computing situation for computing the humidity situation parameter value.

$$voltage_t = P_v\left(T_S \times ADC\_Output_{\sup ply\_voltage}\right)$$

$$temperature_t = P_t\left(T_S \times ADC\_Output_{temp\_sensor}\right)$$

$$humidity = P_h\left(T_h \times ADC\_Output_{humidity\_sensor}\right)\begin{Bmatrix} P_h \xrightarrow{dependent} P_t \\ P_h \xrightarrow{dependent} P_v \end{Bmatrix}$$

It may seem that there is a direct functional dependency between the relative humidity, the resistance of the humidity sensor, the resistance of the thermistor, and the supply voltage at a specific time instance. It is so when we neglect the temporal and spatial aspects of the data or we can guarantee that all the data that is used in the computation satisfies the temporal and spatial constraints. It is clear that we cannot neglect the temporal and spatial aspects since in this case the result is that the computed values do not reflect the real state of the physical world (a temperature reading obtained yesterday outdoors has little value when we need to evaluate the relative humidity in a room right now). Guaranteeing that the data satisfies the temporal and spatial constraints without attaching the temporal and spatial validity intervals explicitly to the data items turns out to be more difficult than it seems even in case of single processor devices. When the data arrives from different (physically dislocated) sources (agents) the problem becomes more acute since in this case guaranteeing temporal and spatial consistency is even more difficult if not impossible when no validity information is associated with the data.

So the solution is to use the approach based on exchange of situation parameters, which makes it (with the help of appropriate middleware as outlined in section *3.4 Middleware dedicated to exchanging situational* information) easier to provide the situation parameter computation function with data that satisfies the temporal and spatial constraints of the function.

Each situation parameter computation function contains the requirements for the input data, i.e. what situation parameters are required as input data for computing the given parameter value. Clearly the situation parameter computation functions are autonomous by themselves, i.e. the fact that another function uses the output of a given function should not affect the behaviour of a given function. As described in the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter the constraints of situation parameter computation functions are propagated from the higher level functions down to the lower level functions.

Sensor signal processing using the approach based on the exchange of situation parameter values is further explained via a diagram in Figure 13.

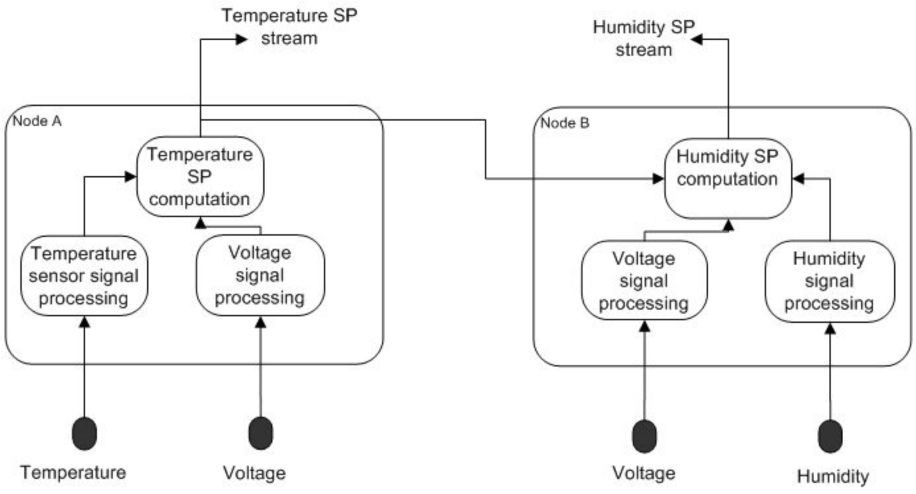

Temperature SP stream
Humidity SP stream
Node A
Node B
Temperature SP computation
Humidity SP computation
Temperature sensor signal processing
Voltage signal processing
Voltage signal processing
Humidity signal processing
Temperature
Voltage
Voltage
Humidity

Figure 13 Sensor signal interpretation

Figure 13 visualizes the sensor signal interpretation example. The temperature sensor is attached to *Node A* and the humidity sensor is attached to *Node B*. Both sensors are resistive, which means that the resistance of the sensors is measured via the change in the voltage of the circuit that contains the sensor. The supply (reference) voltage is also measured in order to compensate for the drop of the voltage when the supply voltage of the battery of the node drops. In the lower level the logical processing units (*temperature signal processing, voltage signal processing* and *humidity signal processing*) the ADC output is converted to the corresponding engineering units (voltage) and supplemented with metadata (timestamp). In the situation parameter (SP) computation block (the *temperature SP computation* block in *Node A* and the *humidity SP computation* block in *Node B*) the corresponding SP value is computed based on the situation parameter values. Since *Node B* is not equipped with a temperature sensor the temperature SP value is received from *Node A*. In case of SP values that are communicated between nodes the spatial validity information must accompany the parameter value.

In addition to the situation parameter computation being situation dependent the time-set for the input data streams can also be situation dependent. From a real-time system developer's viewpoint the approach is quite natural - if the dynamics of the system is known then sampling can be done at a lower rate when the parameter value is not close to critical, but a higher rate is required when the parameter value is closer to the critical value. This approach is especially useful if the available power is limited and sensing is power intensive. The issues of power consumption are discussed in detail in the *Applying smart dust motes in monitoring applications* section.

For ensuring the temporal consistency of the different input data either the approach outlined in the *Realizing situation aware spatially distributed mobile computing systems* section can be used or alternatively one could execute the situation parameter computation at the arrival of items in one of the data streams. The approach outlined in the *Realizing situation aware spatially distributed mobile computing systems* section assumes that the situation parameter computation function execution times set the temporal constraints (hence define the time-set) for the sensor signal acquisition. The middleware concept introduced in section *Middleware dedicated to exchanging situational* information enables the use of either concept assuming the middleware is capable of performing the required conditioning of data.

Still it is preferred that the timesets of the streams of the different situation parameters match, which means that the temporal constraints of a high level situation parameter must be propagated down to the lower-level situation parameter computation functions. The *matching* of the parameter stream timesets means that the temporal validity intervals of the parameters should overlap, i.e. it should not be required to compute valid parameter values based on past values in the same stream (which are not valid any more). The temporal constraints are defined by the execution times of the high level situation parameter function and the temporal validity intervals of the source data. The lower level situation parameter values need not be computed every time a higher level situation parameter computation function is executed. The requirement is just that the lower level situation parameter values (the source data) must be valid during the computation of the higher-level situation parameter computation. Naturally the temporal validity interval of the higher-level situation parameter value depends on the validity of the source data.

4.2.2 Situation parameter computation – a concrete example

Below the sensor signal interpretation example is outlined using the semantics introduced in the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter. Two high-level situation parameters are defined to illustrate the example better: relative comfort level for a human being and safe operating conditions for a piece of equipment. In both cases the situation parameter value is computed using the relative humidity and ambient temperature situation parameters as input. The algorithms for computing the higher-level situation parameter values can be changed without any obligations to change the lower levels. For example the human comfort is very individual so a person could presumably define the setpoints for computing the comfort situation parameter himself. Typically in case of relative comfort for human in relation to humidity and temperature the following reasoning can be applied: most people find high temperature (e.g. above 25 degrees Celsius) combined with high relative humidity (e.g. above 80%) rather uncomfortable while low temperature (e.g. below -5 degrees Celsius) combined with high relative humidity is usually also considered uncomfortable. Anything in between can be considered comfortable at various levels.

For a piece of equipment the safe operating conditions are usually quite well fixed: when the temperature and the relative humidity are in the allowed range the operating conditions are safe. Similarly to humans also for each piece of equipment the set-points for computing the safe operating conditions situation parameter values are different and depend on the specifications of the piece of equipment.

The situation parameter tuples for human comfort and safe operating conditions for a piece of equipment can be expressed as follows. $S_{Joe\_Comfort} = \left(SP_{Joe\_comfort}, S_t, S_l \right)$ expresses the relative comfort for the human named *Joe* and $S_{safe\_deviceA} = \left(SP_{safe\_deviceA}, S_t, S_l \right)$ expresses the safe operating conditions for *deviceA*. The function specifications for computing both of these situation parameter values are quite similar: $S_{Joe\_comfort} = f\left(SP_T, SP_H \right)$ and $S_{Safe\_deviceA} = f\left(SP_T, SP_H \right)$.

In order to compute these higher level situation parameter values the lower level situation parameters must be computed first. For temperature the situation parameter tuple is the following: $S_T = \left(SP_p, S_t, S_l \right)$where SP_p is the temperature situation parameter value (which may be temperature value in Celsius), S_t is the time period for which the temperature value holds and S_l is the area for which the temperature value holds. The time period for which the temperature value holds is computed based on the properties of the environment. The area for which the temperature value holds is derived from the location of the temperature sensor. If the sensor is located outdoors the temperature value may hold for an area with a rather large radius while if the sensor is located in a room the temperature value may only hold for that specific room. The dynamics of temperature depend on very many factors, For example in a room the rate of change of temperature depends on the volume of the room, the power of the heating system, the number of people in the room, the mode and power of the ventilation system and so on. Following the example outlined in the previous sections the temperature situation parameter value computation function uses the temperature sensor reading and the voltage as an input, which can be expressed as follows: $SP_T = f\left(s_T, s_V\right)$. The SP_T stands for the situation parameter value while the s_T and s_V stand respectively for the temperature and voltage sensor readings.

A similar tuple is constructed for the humidity situation parameter $S_H = \left(SP_p, S_t, S_l\right)$. In case of the humidity situation parameter the computation of the situation parameter value is more complex since in order to compute the humidity situation parameter value the temperature situation parameter value, the reading of the humidity sensor and the voltage reading are used – $S_H = f\left(SP_T, s_H, s_V\right)$.

Both the relative human comfort and the safe operating condition for a piece of equipment situation parameter take as input the humidity and temperature situation parameters. In addition the temporal and spatial constraints for both situation parameters must be specified before the parameter value can be computed as otherwise the constraints for the source data are not known.

4.3 Applying smart dust motes in monitoring applications

Smart dust motes can be applied in various monitoring scenarios, one of them being industrial monitoring. Within the scope of this case study a monitoring system for industrial equipment was developed (Preden, et al., 2007). The case study is an exploratory one which provided good source information for further research in the direction of situation aware embedded systems.

A monitoring system for industrial machinery was implemented based on Berkeley motes available from Crossbow, Inc. The motes used in the monitoring system were MPR2400CA MICAz motes which were chosen because of the ZigBee communication interface operating in the 2.4 GHz frequency band – this bans when used on these motes provides relatively high bandwidth and a higher resistance to EMC noise (when compared to, for example, Mica2 motes from the same provider) due to the availability of several (16) channels and an active channel selection scheme (CSMA/CA) used by the IEEE802.15.4 protocol. The higher bandwidth (250 kbps) compared to some other wireless communication technologies allows for shorter transmission times, thereby possibly reducing the power consumption.

The sensor boards used on the motes were MTS310CA and MDA100 available from Crossbow, Inc. The MDA100 was populated (in addition to the onboard thermistor YSI44006 and light sensor Clairex CL94L and) with a Sencera humidity sensor H25K5A and four Rhopoint ACW-006 temperature sensors connected to the sensor board via a 2m cable, which allows placement of the sensors at desired locations on the monitored machine. The MTS310CA is equipped (among other sensors) with a MEMS acceleration sensor from Analog Devices – the ADXL202JE – and a microphone, which were used in the application.

Sampling of sensor data should be done with the smallest feasible interval (naturally considering the characteristics of the system and the signal) to obtain as much information as possible. At the same time the smaller the sampling interval the more power is consumed – node cannot enter the sleep state and the appropriate sensors must be powered. Therefore it makes sense to increase the sampling interval to the greatest value acceptable by the application. Surprisingly test results show as justified by the data in the tables below, the reduction in power consumption from increasing the sampling interval is not linear and from a certain level there is no point in increasing the interval. As it can be seen from Table 2 the power consumption differs greatly between the different modes of the device. The current drawn by all the sensors connected to MDA100 is relatively small (<1 mA) but still substantial when compared to the current draw in sleep mode.

Mode	Current draw (mA)
Sleep mode, with timer on	~0,02
Processor running, radio off	12
Processor running, all MDA100 sensors powered	13
Processor running, radio in idle listening mode	32
Processor running, radio transmit mode	30

Table 2 Power consumption in different modes

Motes interpret the sensor readings and transmit the interpretation results to the sink node, where the data is written to a database. The sensors are connected to the

ADC channels via a voltage divider circuit, which means that the interpretation of the ADC reading also depends of the supply voltage (which decreases over time since the battery voltage drops). The temperature values are transmitted to the sink node if the change was more than 10% of the current reading, or if the temperature reading is above a given threshold.

Operation	Execution time (ms)
Data acquisition from a single ADC channel	0,26
ADC output conversion to resistance	0,2
Temperature sensor ADC input value conversion to engineering units using table	0,4 – 0,9
Computing the logarithm function	0,8
Temperature sensor ADC output conversion to engineering units using the suggested function	1,1
One data packet send time (between calling the send function and send done)	2 – 7

Table 3 Operation execution times as measured on MICAz mote

The tables Table 2 and Table 3 provide the source information for computing the average power consumption of a node at different sampling intervals.

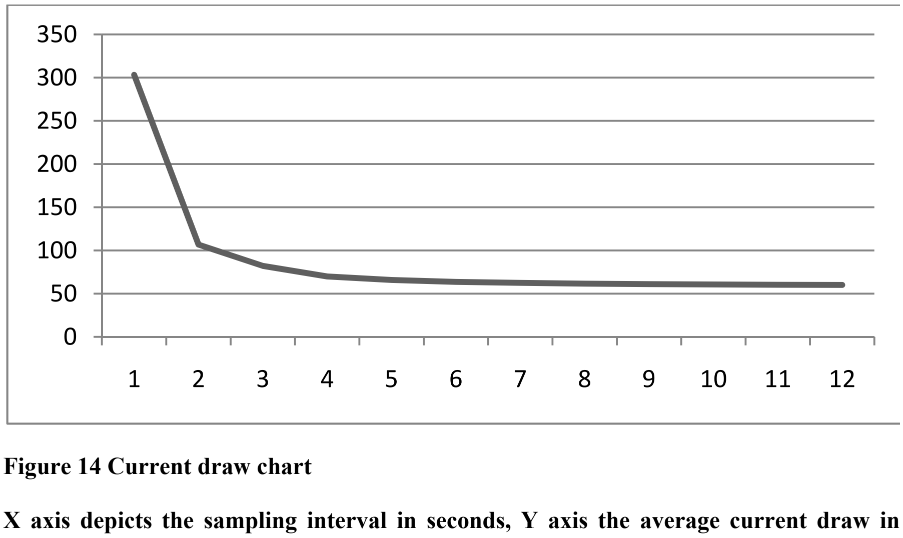


Figure 14 Current draw chart

X axis depicts the sampling interval in seconds, Y axis the average current draw in milliamps

The chart in Figure 14 illustrates average current draw with different sampling intervals when data is sampled from five ADC channels. It is assumed that between data sampling the node is at sleep. The chart shows that increasing the sampling interval to greater than 4 seconds has very little benefits in terms of power conservation as the average current drop after that is not very significant. The current consumption curve depends on the processor and computation times. The power consumption of the communication tasks is not considered on the chart as this was outside the scope of the case study.

4.4 Data representation for situational information propagation

A large amount of the practical work presented in this section was performed by Raido Pahtma in the context of the VõVõ project. Raido did the implementations of the concepts for the smart dust motes and also for the PC. The principles presented in this section allow propagation of situation parameter values together with validity information in sensor networks. This work is presented in greater detail in (Preden, et al., 2009).

4.4.1 Distributed data processing

Historically wireless sensor networks have been built around a single data collection and processing entity (a sink node) to which the sensor nodes delivered the collected data using a MANET type of network, which means that the sensor nodes were just used for data acquisition and forwarding (Intanagonwiwat, et al., 2003). In the beginning of section *Distributed applications based on smart dust* the prevailing sensor network architectures were described in a bit more detail and some of the shortcomings of these architectures were explained.

As in case of many applications the information required from the network is not the sensor data but instead the current state of the world (which may be reflected in the situation parameter values) in the monitored area (which of course is reflected in the sensor data). So the data could be interpreted locally to quite a high level of abstraction locally, fusing data from nearby nodes which would result in only high-level concepts being transmitted over the network.

In the following a short overview of related work in distributed data processing solutions in the area of wireless systems is given.

In (Huebscher, et al., 2008) the authors come to the understanding that wireless embedded nodes must evolve from homogeneous devices running unsophisticated programs in environmental monitoring applications to complicated heterogeneous devices in ubiquitous computing (Weiser, 1991) realm. In-network data processing and inter-node communication allows the reduction of bandwidth requirements, thereby prolonging the network lifetime, in addition the data generated by the network could be directly accessed by the network clients, eliminating the need for a sink node. Several approaches in that direction have been proposed. The Milan middleware (Heinzelman, et al., 2004) allows for data fusion while using the quality of service values of sensors with respect to a specific property. As with other data aggregation solutions the temporal and spatial properties of data are not addressed explicitly and if these properties are relevant on the application layer (which is usually the case, support for these properties must be added separately).

The concept of semantic information fusion, which is an in-network inference process where raw sensor data is processed, resulting in exchange of semantic interpretation of data between nodes, has been also explored in the context of sensor networks (Friedlander, et al., 2002), (Friedlander, 2005), (Whitehouse, 2006). In (Whitehouse, 2006) a framework is presented that allows users to pose declarative queries over semantic interpretations of sensor data, however the presented framework assumes a centrally managed architecture and does not cater for in-network data processing, deviating from the objectives in ubiquitous computing solutions. The in-network data aggregation method which the authors call the directed diffusion paradigm (Intanagonwiwat, et al., 2003) is a simplistic data subscription method for sensor networks where the subscriber can specify the type of data and the spatial properties of data, however due to its rigidness it does not work well for highly dynamic networks or networks where there is more than one data consumer.

The configuration requirements for the sensor networks pose also difficult problems as in most applications it is assumed that the placement of the sensors is well known (e.g. the Sustainable Bridges application (Marron, 2005)) and the application relies on the fixed placement of nodes.

Architecture for distributed applications in sensor networks which minimizes configuration requirements and allows for in-network data aggregation is proposed in (Preden, et al., 2006). However a full implementation of the architecture was not completed and it is clear that the research ideas presented in that paper need further development before usable implementations are possible.

4.4.2 Exchange of situational data

In order to make use of the situational data generated by network nodes the situation parameters must be exchanged between the nodes. In addition to the situation parameters that are exchanged also the metadata (such as validity area and temporal validity period) of the situation parameters must be communicated.

It is clear that fixed packet formats are not suitable for this task – the set of situation parameters is quite large, the metadata of the situation parameters is not always fixed and there may be various representations for the metadata.

In addition to the exchange of situational data also the data interpretation rules must be exchanged in cases when a situation parameter type is unknown but can be translated to a type known by an agent.

In order to exchange the situation parameter values between agents a new data encoding format was developed for use in sensor networks. The objective of the development work was to come up with an encoding that allows for seamless transition of data from the lowest level up to the high level more powerful nodes. In case of the high-power nodes (e.g. PCs) the protocol of choice would be XML, so a direct transition of data from the low-level encoding to XML would be desirable without any need for converting the data itself.

Due to the limitations of the more constrained network nodes (typically 8 bit processors with limited memory and computational power) and communication channels (maximum packet size is in the order of about a hundred bytes) the conventional methods for data encoding are not very well suited.

4.4.3 Structuring data

Structuring of situational data is based on object-subject-value expressions. For simplification the predicates in the expressions are omitted, the predicate "has" is applied to all subject-object relationships. The object field describes the situation parameter type while the value field contains the value of the situation parameter. In some respects the described approach is similar to the solution described in (Tammet, et al., 2008). However due to the different nature of data communicated for achieving situation awareness the approach described in (Tammet, et al., 2008) cannot be directly applied. Similarly to RDF expressions there is a possibility to build complex structures of expressions (via the subject field). Both the subject and value elements can be also omitted in case there is no need to associate the expression with another expression (via the subject field) or when the expression carries no value, e.g. when the expression is the top level of a complex expression (the value field is omitted). For instance when a query for a situation parameter with no constraints is made only the object type need to be transmitted.

The expressions are 3-tuples $E = (object, subject, value)$ containing an *object,* a *subject* and a *value.* The *subject* can refer to another expression in a set of expressions. The *object* types can be situation parameter types, metadata types or other object types (e.g. in case of a mathematical equation the mathematical operations are also objects). The metadata types are used for describing the properties of the situation parameter. Obviously the *object* type constants must be pre-specified at design time.

The simplest expression is an *object-value* pair (the *subject* is omitted); an example of this in the natural language would be "the average temperature in Celsius has the value 5". If the constant *0x0F* is used with the situation parameter (*object* type) "average temperature in Celsius" the resulting expression will be $E = (0x0F, 0, 5)$. The *subject* field contains a *0* as the expression does not have a subject.

It is clear that such an expression by itself has little value, but by associating some other expressions with the described expression we get a set of expressions that can convey information about a specific region. As the *subject* field in the expression can refer to any expression in a set of expressions it is possible to create complex expressions consisting of several atomic expressions. An example with four expressions is given below.

$$E_A = (O_A, 0, V_A)$$

$$E_B = (O_B, E_A, 0)$$

$$E_C = (O_C, E_B, V_C)$$

$$E_D = (O_D, E_B, V_D)$$

In expression E_A a value for a situation parameter is given, there is no *subject* field as the situation parameter is not associated with any other expressions. Expression E_B associates (via the *subject* field) a complex *object* (O_B) with the expression E_A but no value is given as the data type is complex and the values are contained in the following expressions. Expressions E_C and E_D are both associated (via the *subject* field) with the expression (and thereby also the complex *object* O_B) E_B and also values are included in both expressions.

4.4.4 Validity intervals

In order to encode metadata that is associated with the situation parameters the encodings for both the spatial and temporal validity intervals should be specified. The temporal validity interval specifies the validity of a situation parameter value in the time domain and the spatial validity interval specifies the area where the situation parameter value holds.

Temporal validity intervals can be expressed using any time representation model and format. Keeping in mind that several absolute or relative metric times as well as several time concepts (e.g. strictly increasing, reversible) can be used in a distributed system a set of temporal interval representation formats must be selected for a given system with an agreement on the expression formats. The absolute temporal interval can be for example expressed using the following set of expressions:

$$E_{TS} = (O_T, 0, T_s)$$

$$E_{TI} = (O_{TI}, E_{TS}, T_I)$$

where in E_{TS} the *object* field specifies the object of type O_T (absolute time) and the *value* contains T_S which is the start time of the interval in absolute time (e.g. UNIX time). The expression E_{TI} is associated with the expression E_{TS} via the *subject* field, the *object* type O_{TI} specifies a time interval and the *value* field T_i contains the length of the temporal interval.

The simplest form of expressing a spatial validity interval is via a circle as only the coordinates of the centre and the radius of the circle are required. This can be expressed as follows:

$$E_{SR} = (O_{SR}, 0, R)$$

$$E_{coordX} = (O_{coordX}, E_{SR}, X)$$

$$E_{coordY} = (O_{coordY}, E_{SR}, Y)$$

In the first expression E_{SR} the identifier in the *object* field specifies the circle spatial validity area type O_{SR} and the radius of the circle is contained in the *value* field. The centre coordinates of the circle are in the expressions E_{coordX} and E_{coordY} both of which are associated with the circular validity area via the *subject*.

The spatial validity area can be also expressed using more complex shapes, such as polygons, in which case the number of points for which the coordinates must be specified is greater. As the same principles that are used for describing the circular validity area can be also used for the more complex area types, an example is not deemed necessary here.

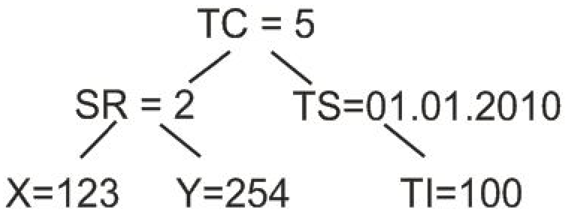


Figure 15 Expression hierarchy for temperature example

The hierarchy of expressions depicted in Figure 15 shows how the temperature situation parameter value can be associated with temporal and spatial validity intervals. The expression *TC = 5* on the figure should be interpreted as *the temperature object has the value 5*. The expressions containing the temporal (*TS*) and spatial (*SR*) validity intervals are associated with the temperature expression. The temporal interval is of type absolute temporal interval which means that the absolute time when the validity interval starts (which is 01.01.2010 in this case) is specified. The expression containing the length of the interval (which is 100 time units in this case) is associated with the interval start time via the subject field. The spatial validity interval (or area) is specified by using three expressions – the first one specifies the type of interval used – a circle and the radius of the circle (SR = 2) and the two other expressions that are associated with first one specify the x and y coordinates (which are 123 and 254 correspondingly). This means that the temperature value was 5 in the circle with a radius of 2 units and the centre coordinates of x=123 and y=254 starting at 01.01.2010 for 100 time units.

In order to express the above example using expressions a few constant values must be fixed.

Type	Constant	Description
T_C	0xD1	Temperature in Celsius
S_R	0xB0	Circular area
$Coord_X$	0xB1	X coordinate
$Coord_Y$	0xB2	Y coordinate
T_S	0xB3	Start time of absolute temporal validity interval
T_I	0xB4	Length of the temporal validity interval

Table 4 Value representation example constants

Using the constant values from Table 4 the following expressions can be formed to represent the temperature value and its validity intervals.

$$E_A = (0xD1, 0, 5)$$

$$E_B = (0xB0, E_A, 0)$$

$$E_C = (0xB1, E_B, 123)$$

$$E_D = (0xB2, E_B, 254)$$

$$E_E = (0xB3, E_A, 01.01.2010)$$

$$E_F = (0xB4, E_E, 100)$$

The above expressions represent the same information that was encoded on the graph depicted on Figure 15 using the constants from Table 4.

4.4.5 Situation parameter type conversion

In case of situation parameter types that are unknown to a computing agent, conversion rules that allow for converting unknown situation parameter types to known situation parameter types need to be used. These conversions can be performed by a dedicated entity in the network or the conversion rules can be transmitted over the network to an agent that needs to perform the given conversion.

The rules for the conversions can be expressed using the same types of expressions as described in the previous sections.

The concept is best explained via a trivial example. For converting temperature in Celsius to temperature in Fahrenheit the following equation can be used $T_F = T_C \times 1.8 + 32$, where T_F is temperature in Fahrenheit and T_C is temperature in Celsius. The graph of the hierarchy of expressions is depicted in Figure 16.

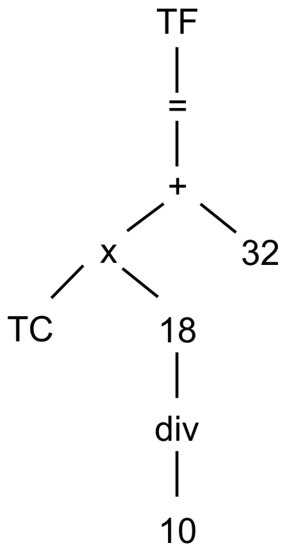


Figure 16 Expression hierarchy for data conversion example

In order to express the equation in the expression format outlined above a few constants must be fixed. The constants are listed in Table 5. The constants in the table are arbitrarily chosen for illustration purposes only. It is clear that the constants for mathematical symbols should be the same for all the equations. It is also beneficial to specify the constants for the situation parameter types in a separate range of constants in order to simplify the processing of data.

Type	Constant	Description
T_F	0xD0	Temperature in Fahrenheit
T_C	0xD1	Temperature in Celsius
=	0xD2	Equal
+	0xD3	Addition
*	0xD4	Multiplication
Const	0xD5	Constant, value field contains a constant
Divider	0xD6	Divider, value field contains a divider which is used to divide the value of the expression pointed to by the subject field.

Table 5 Data conversion example constants

Utilizing the constants from Table 5 the following expressions can be formed.

$$E_A = (0xD0, 0, 0)$$

$$E_B = (0xD2, E_A, 0)$$

$$E_C = (0xD3, E_B, 0)$$

$$E_D = (0xD5, E_C, 0)$$

$$E_E = (0xD4, E_C, 0)$$

$$E_F = (0xD1, E_E, 0)$$

$$E_G = (0xD5, E_E, 18)$$

$$E_H = (0xD6, E_G, 10)$$

As one can see the number of expressions is by one greater than the number of elements in the equation (including constants and mathematical operators). The reason for that lays on the fact that the data encoding format outlined in the following section only supports integer data types and for communicating real types a divider must be used. It is evident that the described expression format can be used to describe almost any kind of expression, provided the constants representing the required operators are described.

The conversion rules can be propagated only for higher-level situation parameter values. The relationships between low-level situation parameter types and hardware capabilities of a specific node are hard coded into the node as it can be assumed that its configuration is fixed during the node development and does not change at run time. An example of this is the conversion of a temperature sensor reading to a situation parameter value reflecting the temperature – this conversion is fixed as it is dependent on the hardware of the node (sensor type, ADC type, etc).

4.4.6 Data Encoding

In the previous sections the principles for structuring the data were outlined. In order to communicate the data from one agent to another, the data must be encoded in a format which is suitable for digital communication, expressive and compact in order to spare communication bandwidth and at the same time the encoding format should not be too demanding in terms of computational resources.

In the conventional computing world the protocol of choice would be XML, which satisfies some of these requirements. Because of its good properties it also makes sense to use XML for communication between computers with sufficient computation power and communication bandwidth. However in the world of wireless sensor networks the use of XML is not desirable if the network nodes need to process and utilize the data themselves.

XML is quite verbose, making it not suitable for use in case of nodes with very limited communication bandwidth and limited computational resources. For example in case of TinyOS, a popular operating system in the WSN world, the default packet payload length is 27 bytes which can be increased if required but communicating XML data over such links is not sensible. The fact that Crossbow's MicaZ smart dust mote has a RAM size of 4 KBytes and ROM size of 128 Kbytes (Crossbow, 2010) also limits the applications that can be run on the hardware. It must be kept in mind that in addition to the application software and data, the operating system must also fit in the memory. Other encoding formats, such as Abstract Syntax Notation 1 (ASN.1) were also considered but deemed too complex and not suitable for the task.

Instead an encoding format was developed that would include provisions for *object-subject-value* data encoding at the byte level. At the same time the encoding format should allow for direct conversion to XML (with not data interpretation requirements) as there is no sense in using non-standard formats in case of more conventional computing devices.

The developed encoding format allows for a very compact data representation which at the same time creates minimum overhead in terms of parsing as each expression can be parsed separately, and there are no complex predefined structures at the byte level. Data is encoded using pairs of tags and their values with an *object* tag followed by an *object* identifier (a pre-specified constant), *subject* tag followed by a *subject* reference, and *value* tag followed by a value. The *object, subject* and *value* tags also specify the length of the data following the tag. An example list of tags used for data encoding is listed in Table 6.

Data Type	Object	Subject	Value
int8	0xA0	0xB0	0xC0
int16	0xA1	0xB1	0xC1
int32	0xA2	0xB2	0xC2
uint32	0xA3	0xB3	0xC3

Table 6 Sample Object, Subject, Value tags

The temperature example "the average temperature in Celsius has the value 5" brought above would be encoded as follows:

0xA0	*object tag*
0x0f	*object constant*
0xC0	*value tag*
0x05	*value*

As one can observe the chosen data structuring and encoding format is compact and straightforward to parse, thus suitable for use in the context of devices with very limited computational resources. The same compact data format is also suitable for implementing a data store.

4.4.7 Distributed data store

Because the proposed universal data representation format can be used to express and encode any data generated and exchanged in a network, a distributed data store for storing the exchanged data can be easily created. The advantage lying here in the fact that any node storing the data need not to be able to interpret the data, i.e. the object types used need not to be known to the node, the node can just store the 3-tuples forming the expressions. No knowledge of the object types is required for responding to queries either – a node can just make a lookup in its data store for the queried object. If a query for an object type (a situation parameter) is made any node that has data with the queried object type in its data store (that also satisfies the constraints of the query if the query contained any constraints) can respond.

4.4.8 Applications

The outlined data exchange and encoding principles can be applied in the context of a distributed application. The application description needs to include the situation parameter types (object types) and constraints (temporal and spatial validity intervals) of the situation parameters that are required for running the application. A framework for describing applications in such a way is also described in (Preden, et al., 2006), although the work described there does not consider the exchange of situation parameters but instead describes generic distributed application architecture. The approach described in section *3.4 Middleware dedicated to exchanging situational* information which was motivated by the previous work has provisions for the exchange of situation parameters. The middleware takes care of the issues of providing the required situation parameters to the higher processing level, so at the higher level just the situation parameter values can be immediately processed.

In case the provider of a situation parameter (that would also satisfy the required constraints) can't be found on the network, the node that is trying to initiate the application will have to look for conversion rules or conversion providers that would allow to convert an available situation parameter value into the unavailable but required situation parameter value.

4.4.9 Prototype implementation

In order to validate the outlined approach a prototype implementation was done at the Research Laboratory for Proactive Technologies using Iris WSN nodes from Crossbow Technology Inc. In the prototype application the nodes monitor environmental conditions (such as ambient temperature or relative humidity), compute certain situation parameter values (such as average or current temperature) and store the data in the local data store. The data can be either queried from the nodes or a subscription can be made to receive regular updates of some situation parameter value. Both data value queries and subscriptions are realized using the approach described above with special packet types for queries and subscriptions. In queries and subscriptions the situation parameter type constant is used for specifying the type of situation parameter that the client is interested in and the validity interval is used for specifying the constraints upon the situation parameter that the client is interested in. Agents involved in the prototype are location-aware, so queries can be made with constraints on the area for which the situation parameter should be valid. An example of this is a subscription to average temperature value from a specific room for a specific period of time. The nodes also implement a distributed data store, storing both the situation parameter values (together with validity intervals) computed locally and also transmitted by the peers.

4.5 Positioning

Since location is an important aspect of situational information, several case studies have been performed in the Research Laboratory for Proactive Technologies on that topic. In this section some of the aspects of mobile agent positioning are described, with focus on indoor positioning methods. The case studies focused on resource constrained nodes (smart dust motes), with wireless communication interfaces that are networked into an ad-hoc network. This work is presented in greater detail in (Preden, 2006) and (Pahtma, et al., 2009) and in a paper to be published in 2010.

The physical position of an agent determines the spatial validity area of the data (and the situation parameters computed based on that data) collected by the agent. Certainly, the position of an agent should be determined only in systems where position is relevant (Meriste, et al., 2005). The position information can be used by several logical layers in such an agent – from low-level functions such as routing (in case of location aware routing protocols) to assigning spatial validity information to higher level situation parameters. Location awareness (knowledge of the current position of an agent) is easily achievable for well predefined systems in a context of limited deployment. However if the system configuration is not predefined, deployment occurs on an ad-hoc basis, the agents are mobile, the location of nodes must be determined dynamically.

Any agent deployment (be it smart dust motes, RF ID tags or any other type of devices that could be subject to automated deployment over a target area) raises the problem of positioning the deployed agents. In case of mobile agents (or multi-agents) the position of a sensor or an agent within a node may also be relevant – a single agent may be equipped with several sensors, or several agents may be located on a single mobile platform, which at a higher level of abstraction can be viewed as a single agent.

Agents can be positioned using the following techniques:

1) agents are deployed at predefined known locations (depending on the application the location may be stored in the node or remotely) and the positions of agents are known based on that information;

2) agents are positioned using some positioning hardware and an existing infrastructure. Examples of such infrastructure for (mainly) outdoors are satellite-based positioning systems: GPS (McKee, 2003), Galileo and GLONASS. For indoor applications the following systems have been developed: Cricket (Nissanka, et al., 2000), RADAR (Bahl, et al., 2000) and Active Bat (Ward, et al., 1997), to mention a few. There are also numerous systems based on wireless LAN (WiFi) technology, some of which are also offered commercially. In wireless LAN based systems the access points are used as positioning infrastructure and each deployment needs to be calibrated to adapt it to the properties of the environment;

3) some network nodes - anchor nodes - are location aware and other nodes are positioned utilizing the anchor nodes (Niculescu, et al., 2004).

Deployment of network nodes in large quantities makes one assumption – the cost of the deployed nodes is relatively low, which means that the hardware of the nodes should be as simple and cheap as possible. Adding positioning hardware or complex positioning software features contradicts with the low cost principle. In case of nodes that are not mobile after deployment (in a scenario where the nodes have to be positioned only once after deployment) the built-in positioning capabilities of the node are used only once after deployment which is not very economical.

In case of devices that cannot be positioned using merely device resources – such as very thin sensor network nodes or RF ID tags – some means of external positioning must be used. Such external positioning mechanism can be implemented by using anchor nodes that are able to communicate with each other, detect the network nodes and in a collaborative effort position the nodes. Same positioning principles apply when the positioning is realized by a mobile location aware node which determines its own location and the relative locations of the non-mobile nodes. After the data has been collected from several locations the mobile node is able to compute the positions of the non-mobile nodes using the same principles that are used by the anchor nodes to determine the nodes' positions (Meriste, et al., 2005).

Existing geometric positioning methods which yield a position estimate in a coordinate system based on location aware anchor nodes can be classified into four categories:

- algorithms that employ network connectivity information,
- algorithms that rely on the distance estimations,
- algorithms that use bearing information and
- algorithms that combine two or more or the above methods.

The algorithms that make use of the connectivity information give a quite rough estimate for the node position, which can be refined using other methods. The accuracy of the position estimate of the other approaches depends of the precision of the distance or bearing estimations. The accuracy of these estimates depends of the hardware that is used, the environmental conditions (reflections, interferences, etc) and the algorithms used for determining the position using the distance or bearing estimates. The algorithms that rely on distance estimation utilize mainly various multilateration methods while the algorithms that rely on bearing information utilize triangulation methods.

In case of network connectivity based approaches the position of a node is estimates based on the node's connectivity to anchor nodes. Several variations exist even on this method.

The approach described in (Bulusu, et al., 2000) allows estimating the node position locally based on locations of the anchor nodes. The anchor nodes transmit beacon signals and the node position is assumed to be the centroid of the coordinates of the anchor nodes to which a specific node is connected. The approach assumes that the anchor nodes are regularly spaced over the entire target area. The algorithm makes no provisions for irregular anchor placement (which would seem a more realistic assumption in ad-hoc deployments) and gives clearly false results if there are obstructing objects which disable communication with some of the anchors in the target area. The algorithm returns a single point – the estimated node position – as the result, which even if it is accompanied by an error figure gives only a rough position estimate, considering the positioning method itself.

The localization algorithm based on multidimensional scaling (MDS) presented in (Shang, et al., 2004) relies also on connectivity information between the nodes but in order to give a position estimate to a node the algorithm involves the construction of the logical map of the entire network. The task of construction of such a map requires a substantial amount of communication which consumes a considerable amount of energy. In case of large networks and thin nodes the construction of the logical map of the network by the network nodes may be very difficult if not impossible due to computational and storage constraints (although the task is linearly complex in terms of the number of nodes in the network). The approach also assumes relatively high connectivity of the network – 12 for random networks and 6 for grid networks, which may not be achievable in case of all networks.

In case of the multilateration method distance estimates to at least three known positions (anchor nodes) are used to compute the position of a fourth node. GPS is the most well known example of a system that uses multilateration (Parkinson, et al., 1996). There exist several possibilities for solving the equations involved in multilateration such as Taylor expansion (Niculescu, et al., 2004) or variations on the Least-Mean-Square method as the one presented in (Chintalapudi, et al., 2004).

The triangulation algorithms use bearing information in relation to at least two anchor nodes to compute the position of a third node. The basic idea of triangulation can be expressed as follows: assuming that we know the positions of the vertices of a triangle and the angles between the lines connecting a point in the triangle and the triangle vertices we can calculate the coordinates of the point in the triangle.

Naturally there exist variations and combinations of multilateration and triangulation based approaches but all of these algorithms rely on either distance or bearing estimations in relation to anchor nodes to compute a position estimate of a node. Using the distance or bearing estimates, nodes can use any of several distributed or local positioning techniques to determine their positions in the given coordinate system. It must be noted that the accuracy of the computed position estimate depends on the accuracy of the distance estimation which is the reason why methods based on multilateration and triangulation yield good results in principle, but problems exist in real applications with estimating node positions in a reliable and simple manner as the distance and bearing estimates are not very accurate in real systems.

Any reflections of the signal, no matter what type of signal is used will create substantial errors in the bearing information. In addition the detection of bearing may be quite cumbersome and complicated in case of thin sensor network nodes used in real applications.

The errors created by reflections present a problem for all types of ranging based on RF and acoustic signals described below. RF-based ranging can be based on measuring the received signal strength or time of flight of the signal. Although time of flight is theoretically a precise method for measuring distance, using only time of flight information is not realistic because it requires the clocks of the nodes to be very precisely synchronized, which is very hard if not impossible to achieve in real deployments.

Measuring the received signal strength at the receiver for the purpose of estimating the distance to the transmitter presents also several challenges. The RF propagation in an environment must be accurately characterized by a model that is simple enough to be used by the sensor network nodes. The environment must be quite homogeneous for a RF propagation model to be used successfully. Since the same RF propagation model cannot be used in all environments there is a need to create different models and load the appropriate models to the nodes before deployment or at runtime which is quite cumbersome. The applicability of received signal strength indication is evaluated in some of the case studies presented later in this chapter.

There exists a group of ranging techniques that employ the time difference of arrival of simultaneously transmitted RF and acoustic signals for estimating the distance between the transmitter and receiver (Nissanka, et al., 2000), (Whitehouse, et al., 2002), (Savvides, et al., 2002). Taking into account the difference in the speed of light (RF) and the speed of sound and the measured interval, the distance to the anchor node can be computed. As all other ranging techniques this solution assumes a direct line of sight between the transmitter and the receiver which makes these approaches sensitive to reflections and unusable in environments with large interfering objects.

It can be said that the common problem with multilateration and triangulation based solutions is acquiring accurate bearing and / or distance estimates in real applications. Problems with current connectivity based approaches are different – such algorithms provide either quite inaccurate position estimates or assume a grid-like layout of the positioning beacons, which limit the usage of these algorithms in real applications.

4.5.1 Evaluation of Received Radio Signal Strength Indication

A study on evaluating the use of the received signal strength indication (RSSI) value was conducted in the Research Laboratory for Proactive Technologies to evaluate the applicability of the method for distance estimations. The practical work involved in the case study was performed by Raido Pahtma, Priit Pikk and Romi Agar as part of the work on their master theses.

Received (radio) signal strength indication (RSSI) is one tool that many researchers in the field deem to be applicable for distance estimations. Some even claim that the RSSI value can be used to determine the distance between two communicating nodes accurately (Halder, et al., 2008), (Shin, et al., 2008). It seems natural that since radio signal strength attenuates as a function of distance – the greater the distance the smaller the perceived RSSI value (assuming the transmitter power is constant) – the RSSI value could be also used to estimate the distance between two communicating nodes.

However things turn out to be more complex in practice – RSSI is affected by the distance between the nodes but there are quite a few other factors that affect the radio signal propagation and hence the RSSI value that is perceived by a node. These factors include (but are not limited to) interference from other signal sources, reflection, diffraction, and presence and nature of objects in the path of the radio wave. In an ideal case the RSSI value perceived by receivers at an equal distance from a transmitter is equal, which in practice, as experiment results presented below is not always the case.

Radio irregularity of a device can be divided into two main factors: 1) the heterogeneous properties of devices - antenna type (directional or omnidirectional), transmission power, antenna gains, receiver sensitivity, receiver threshold and the Signal-Noise Ratio (SNR) and 2) the non-isotropic properties of propagation - media properties include the media type, background noise and various other environmental factors (Zhou, 2004).

The RSSI evaluation case study set out to evaluate the RSSI property of smart dust motes. It is clear that in a (near) perfect environment (anechoic chamber) the properties of a device can be determined quite well, a model of the antenna can be created and the function between the RSSI value and the distance can be determined. However such models have a limited use in real applications since the physical world ads a level of uncertainty to the RSSI value. Several case studies were performed in realistic deployment environments where the RSSI value perceived by motes was recorded in a fixed environment in order to determine how the signal strength is actually perceived by the nodes themselves and if the RSSI values are consistent.

A fair amount of work exists in the area, however no promising applicable results can be found from the literature that was surveyed before and during the case study. In (Halder, et al., 2008) the authors attempt to make use of the RSSI value for ranging by applying an adaptive filter to the acquired RSSI values. The proposed method improves the distance estimation based on RSSI values when compared to distance estimations based on unfiltered RSSI values but even when the proposed method is applied the error exceeds the estimated value in some cases. In (Shin, et al., 2008) another filtering method for acquired RSSI values is proposed. However, the obtained results are similar to the work presented in (Halder, et al., 2008) as the error is in the order of the evaluated value (distance) itself, which makes the distance estimation attempt rather futile.

The study presented in (Yang, et al., 2007) points out that the radiation pattern of a typical 802.15.4 antenna is not perfectly omnidirectional, which explains some of the inconsistencies of the RSSI values.

4.5.1.1 Hardware

MicaZ wireless sensor network nodes from Crossbow Inc were used in the studies. The MicaZ motes were used as transmitters and receivers and the RSSI values recorded and evaluated during the study were obtained from the devices.

MicaZ motes are equipped with a CC2420 radio from Texas Instruments which is IEEE 802.15.4/ZigBee compliant and it operates on the 2.4GHz unlicensed ISM (industrial, scientific and medical) frequency band (2400-2483.5 MHz). In the IEEE 802.15.4 standard (IEEE, 2006) the utilized frequency band is divided into 16 channels each occupying 2 MHz with 5 MHz separation between channels. The channels are labelled 11 to 26. As the same ISM band is also used by a variety of commercial equipment – Bluetooth, WiFi, some cordless phones and car alarms, the probability of radio interference is quite high.

The MicaZ motes are equipped with an external 1/4 wavelength monopole antenna as standard. In some studies the standard antenna was replaced with an external TP-LINK antenna with better characteristics.

The software used on the motes in all the studies ran on the TinyOS 2.1 operating system and the operating system drivers and networking layer were used for achieving radio connectivity between nodes.

4.5.1.2 Obtaining RSSI

The CC2420 radio used on the MicaZ motes features a built-in RSSI indicator. The RSSI value is always averaged over 8 symbol periods (128 µs) and has an offset of -45dBm. The latest RSSI value is obtained from the radio with the rest of the packet data and stored in the TinyOS message_t struct. The RSSI value has an accuracy of +/- 6 dBm.

The CC2420 offers the possibility to adjust the power level used during transmission (output power) in 31 steps (in a logarithmic scale) with 31 corresponding to 0dBm and 1 to approximately -35dBm.

The experiments were carried out in the Tallinn University of Technology building that has a steel reinforced concrete floor (which is quite a realistic operating environment for wireless sensor networks).

In the studies the specific motes were assigned with roles of transmitter and receiver and appropriate application software was loaded to the motes. The receiving motes obtained the RSSI value from the radio and transmitted the results to a PC where the data was stored for later analysis.

4.5.1.3 RSSI Case Study 1

All experiments in this study used the configuration of one transmitting mote and four receiver motes arranged in a linear configuration facing the transmitter. The transmitter was always stationary and the receivers were moved from 10cm to 800cm from the transmitter in 25cm steps (15cm for the first step). Tests were carried out with two sets of motes – one set equipped with MicaZ standard antennas and another set equipped with TP-LINK 5dBi antennas (TL-ANT2405C) soldered permanently to the PCB. Two sets of experiments were performed – with motes placed on the floor and with motes placed on wooden frames 56cm above the floor. At each step (location) the transmitter transmitted 10 packets with all possible transmission strengths and the receivers reported the perceived RSSI values to a PC with a 100ms interval between the packets. The average of the received RSSI values is used on the graphs. In the interpretation of the RSSI values the -45 dBm offset of the CC2420 RSSI value is taken into account.

Figure 17 shows the received signal strength indication on a distance of 3m – 6m from the transmitter as perceived by the MicaZ motes equipped with TP-LINK antennas. The horizontal axis represents the distance between the transmitter and receiver in centimetres and the vertical axis represents the received signal strength as perceived by the receiver in dBm. The transmitter used maximum transmission power for transmitting the messages. Each line depicts the received signal strength as perceived by an individual mote – four motes were used as receivers, hence there are four lines on the graph.

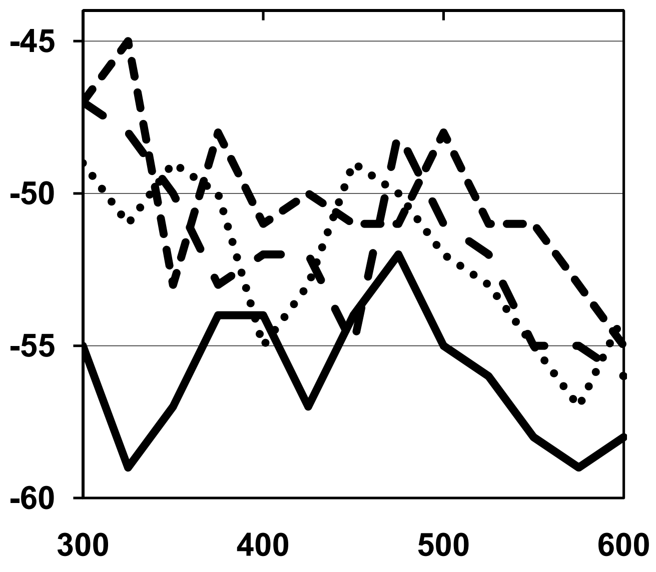


Figure 17 RSSI measured by MicaZ motes equipped with TP-LINK antennas

As it can be observed from Figure 17 the RSSI values perceived by individual motes are not identical although the distance between the transmitter and the four receivers was the same. It is possible that the variance in RSSI values is caused by the allowed error level of the RSSI indication (the specification promises accuracy in the range of +/- 6dBm) however since the RSSI values are averaged over several readings and since the individual readings were quite consistent this is not likely the case. The RSSI values fluctuate very little when measured by one mote for different messages from one transmitter, but differ greatly between different receivers in the same location. The specification of the radio chip (Texas Instruments, 2007) says that the RSSI value of the chip is linear with respect to distance which would allow the RSSI values to be used with confidence.

The variance in the RSSI values between motes cannot be explained by only physical differences in the devices either, since values obtained from different devices cannot be correlated well. No correlation between measured RSSI values and battery voltage was recorded either, assuming that the supply voltage is sufficient for normal mote operation.

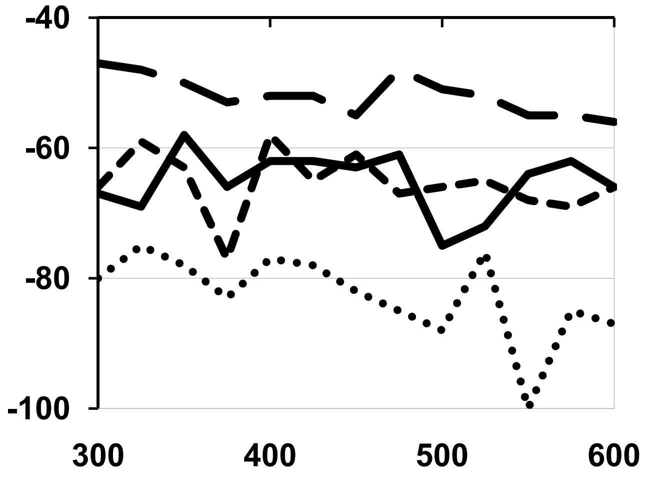


Figure 18 Received signal strength measured by MicaZ motes

Figure 18 depicts tests with various antennas on the floor and 56 cm from the floor on a wooden frame. The dotted line in the bottom depicts the signal strength recorded by a MicaZ with standard antenna placed on the floor; the dashed line in the middle depicts the signal strength measured by a MicaZ with standard antenna and placed 56 cm from the floor. The solid line in the middle depicts the received signal strength measured by a MicaZ with TP-Link antenna while placed on the floor, and the top dashed line depicts the signal strength recorded by a MicaZ equipped with a TP-Link antenna positioned 56 cm from the floor.

It is clear that the proximity of the floor and the antenna type affect the RSSI reading but again it is difficult to find a correlation between the RSSI graphs.

The RSSI values are similarly affected by environmental conditions at all output power levels. It can be observed that as output power increases, so does the RSSI value, but very little new information can be obtained about the distance between the transmitter and the receiver. It is clear that higher power levels increase the range of the transmission, but for indoor environments with a clear line-of-sight, the messages reach the received when the transmit power level is above 5. The occasional drops in RSSI levels are most likely caused by the metal in the reinforced concrete floor. The fluctuations are much less evident with larger antennas and when the motes are higher above ground. Over 130 000 packets were processed by the PC during this study.

4.5.1.4 RSSI Case Study 2

The second RSSI case study was concerned with obtaining RSSI values at different frequencies with the configuration of the rest of the test setup being constant.

Since the 2.4 GHz ISM band has many potential sources of interference, finding a good channel is essential for establishing a good communication link. Monitoring the RSSI value may provide a good means for selecting the optimal channel for communication.

The tests were carried out in a typical office environment with two MicaZ motes equipped with standard antennas. The motes were placed on two separate office desks (80 cm from the ground) 2 meters apart with line of sight visibility between the motes and antennas in a vertical position. Three different test scenarios were studied, each with two different measuring strategies: 1) the transmission channel is changed after 100 packets 2) the transmission channel is changed after each packet. In both cases the originating node transmitted 100 packets on each channel (altogether 1600 packets).

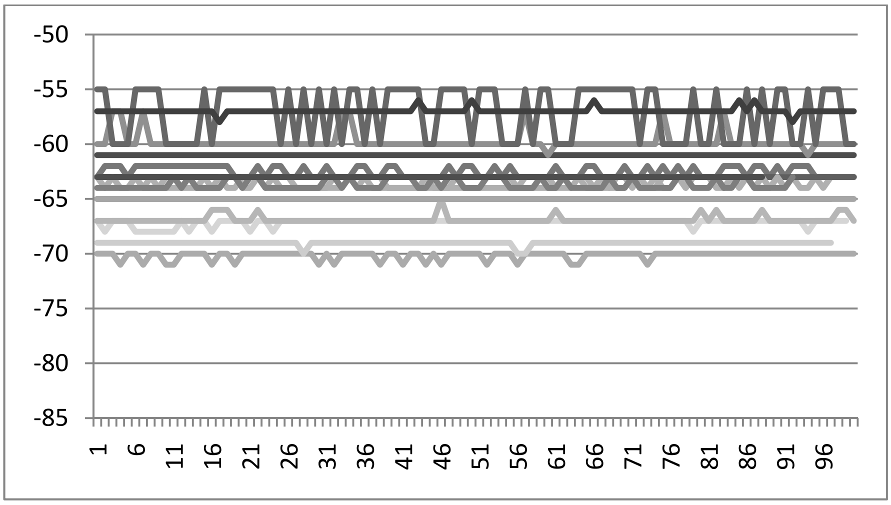


Figure 19 RSSI values at different channels

Figure 19 shows the averaged RSSI values over time on different channels (every line shows the signal strength on an individual channel). The y-axis depicts the signal strength in decibels and the x-axis shows time from the start of the experiment. As it can be seen from the figure the signal strength on different channels differs quite much and the signal strength on individual channels also fluctuates over time.

The RSSI readings were taken in an environment with no physical movement in the room. In this case both measuring strategies indicated similar results with minor signal strength fluctuations on the channels, with the average RSSI value for each channel changing not more than 1dBm in subsequent tests. It must be kept in mind that the CC2420 radio chip specification (Texas Instruments, 2007) states that the RSSI value has an accuracy of ±6 dBm. The observed RSSI values were between -55 and -69 dBm, which means that the perceived signal strength on different channels is not equal.

4.5.1.5 Conclusions from the RSSI case studies

As both case studies showed the RSSI values are related to the actual received signal strength (as can be expected), which in turn is effected by very many factors, among which are the transmission power, distance between the transmitter and receiver, angle of the devices, the channel in use and many other factors. As the RSSI value is affected by so many factors it cannot be expected that there is a good correlation between the distance to the transmitter and the RSSI value so using the RSSI value for obtaining distance estimations has limited applicability in real deployments.

4.5.2 Communication area based positioning (CABP)

Due to the difficulties associated with correlating the RSSI value to distance estimations and therefore of the applicability of positioning methods that rely on distance estimations, an alternative positioning method was developed at the Research Laboratory for Proactive Technologies by Jürgo Preden (Preden, 2006). The communication area based positioning method allows to position nodes using the estimated communication ranges of the nodes in a given environment, no distance or bearing estimations between anchor nodes and nodes being positioned are required.

The algorithm allows to define the area where a node is located using the intersections of communication areas of the anchor nodes that are able to communicate with the given node. The algorithm also allows for iterative refinement of the area definition of a node, based on new anchor nodes that are able to communicate with the specific node. The involved computations are quite simple – at each step the intersection points of communication area borders are computed and added to the set of points that define the area where the node is located. The algorithm can also utilize information from anchor nodes that are not able to communicate with a specific node to further refine the node's position estimate.

Although the algorithm was developed for positioning sensor network nodes and RF ID tags the algorithm can be used for localizing any kind of phenomena that is detectable by sensors and which detection area can be described by a geometric shape that can be formally presented such as a circle or an ellipse.

The nodes are assumed to be equipped with omnidirectional antennas, which means that the communication radii of the nodes (in a homogeneous environment) can be represented by a circle. The radii itself could have different values, depending of the current transmission power. However the algorithm can be applied if the communication areas of the nodes can be described using fixed geometric shapes. In case of more complex shapes (as compared to circles or ellipses) the formulae involved in the computations will be also more complex and the criteria for creating the sets used in the algorithm become more complex.

The proposed algorithm uses only communication ranges of location aware nodes (anchor nodes) at given locations and information if communication with a node, which position needs to be determined, is possible at a given position. It is assumed that the maximum communication range at a given transmission power is fairly consistent in the given physical environment.

When the environmental conditions change to the extent that the communication range changes appropriate adjustments to the communication range value used in computations can be made at runtime by the anchor nodes based on the collected information. Coarse-grained adjustments to the estimated communication range of an anchor node can be made based on the coordinates of other anchor nodes that a specific anchor node is able to communicate with. If an anchor node is able to communicate with another anchor node, the distance between the anchor nodes can be computed and the communication range at the specific transmission power can be assumed to be at least that computed distance. If an anchor node is alive but no direct communication from a specific anchor node is possible with that node then the maximum communication radius can be assumed to be less than the distance to that anchor node (assuming of course that the communication range is consistent).

When the algorithm is applied the result returned is the description of the area where the node, which position is being determined, is located. The algorithm allows for incremental refinement of that area definition as new anchor nodes which are able or are not able to communicate with a node are added to the set of nodes that are used in the computations. The minimum number of anchor nodes required to compute area of a node using the proposed algorithm is two.

Before a description of the algorithm can be given the following terms must be defined:

- LA node – location aware node – a node which knows its position in a coordinate system and its communication range in the current environment. The LA node is able to communicate with other nodes in the environment.

- NLA node – not a location aware node – a node, which is not aware of its location and which location needs to be determined. Any node for which there is a position estimate but which position estimation is being refined is also called a NLA node. The NLA node is able to communicate with other nodes in the environment.

Node's area – is used to denote the description of the area, where the node is located.

4.5.2.1 Description of the algorithm

When an LA node *A* is able to communicate with a NLA node, we can assume that the NLA node is located within the communication range of the LA node. This gives us an estimate on the location of the NLA node. If there is another LA node *B* which is also able to communicate with the NLA node the area description of the NLA node location can be created.

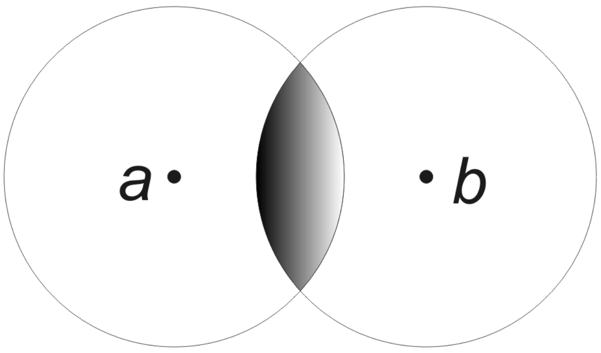


Figure 20 Defining node area with two location aware nodes

In Figure 20 LA nodes *A* and *B* are both able to communicate with a NLA node *C* (not depicted on the figure), which means that NLA node *C* must be located in the greyed-out region. The intersection of the communication radii of node *A* and node *B* describes the area where the node must be located. The centre of the area can be used as the estimated position of NLA node *C* and distance between the centre and the intersection point with the greatest distance to the centre of the area defines the radius of the area.

If there are LA nodes which are not able to communicate with node *C* but which communication areas intersect with the current node *C* area, then the intersection of the two areas (the communication area of the LA node which is not able to communicate with *C* and the current node *C* area) must be subtracted from the current node *C* area. The result of the subtraction yields a new area description, which becomes the new description of the area where node *C* is located. The concept is explained below in more detail.

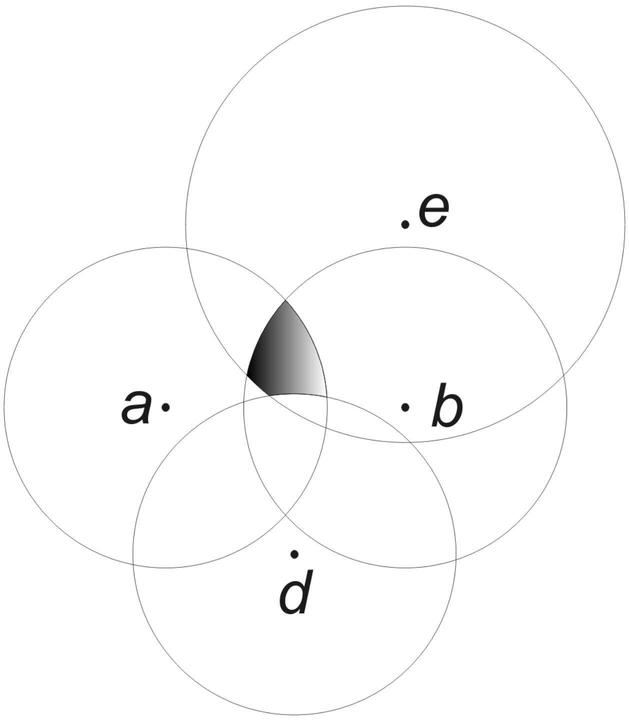


Figure 21 Defining node area with four location aware nodes

In Figure 21 more LA nodes are added. When communication with node *C* is not possible by LA node *D* the intersection of the current *C*'s area and node *D* communication area must be subtracted from *C*'s current area. If node *e* is able to communicate with node *C* then *C*'s area is redefined again, this time the intersection of the *C*'s area and *E*'s communication radius becomes the new *C*'s area.

4.5.2.2 Simulation results

Simulations were carried out using a custom application that implemented the proposed algorithm. The simulation data was generated by the same application in an additional step, separate from the simulation itself.

All simulations were run on a test set of NLA nodes, the locations of the NLA nodes were generated by a pseudo random number generation algorithm, the average NLA node density was 0.01 nodes per one area unit (100 nodes distributed over an area of 100 x 100 units).

The LA nodes were randomly distributed over the same simulation area. Simulations were run with different number of LA nodes, the number of randomly distributed LA nodes varying from 5 to 70. Simulations were also run with different communication radii of LA nodes, the communication radii ranging from 5 to 30.

The node degree (i.e. the number of LA nodes in a ranging neighbourhood, or communication range of a NLA node) that is achieved as a result of such a configuration may seem small – the average node degree in the simulations was about six – compared to other examples from literature (Chintalapudi, et al., 2004) but any algorithms should be evaluated under conditions which at least try to be close to realistic. As lower densities are deemed to be realistic, the author used lower densities in the simulations.

The results of simulations were quite predictable in some respects – as it can be expected the average positioning error is greater when the communication radius of the LA nodes is greater. Naturally the positioning error is not always greater with greater communication radii but assuming nearly even distribution of LA and NLA nodes, and a smaller number of LA nodes in case of larger communication radii the positioning errors tend to be greater with greater communication radii.

The positioning error was estimated by comparing the centre of the estimated NLA node position area to the actual position of the NLA node. The average position estimation error can be reduced if the number of LA nodes is increased. During the simulations a lower bound of the positioning error was encountered at which increasing the number of LA nodes (and placing the added nodes randomly into the area) had very little effect on the reduction of the position estimation error. Of course the positioning error could be reduced by selecting the positions for the newly added nodes but this was not of interest in the context of the experiment. Figure 22 illustrates how increasing LA node density can have very little effect on decreasing the minimum NLA node position estimation error, the error was 6 for the given simulation.

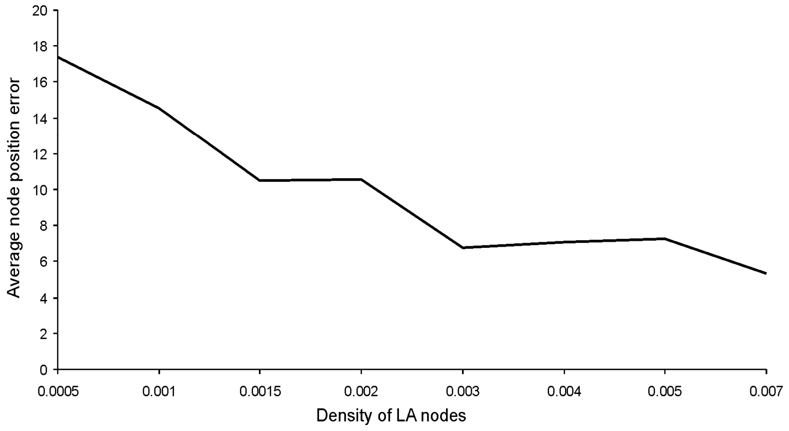


Figure 22 Relation between density of LA nodes and average error of node positioning

The minimum number of LA nodes required for all NLA nodes to receive a position estimate and the position estimation error with the given number of LA nodes with the given communication radius gives good reference points for comparison. The minimum required LA node densities are shown in Table 7. It must be noted that the average node degree in all the three cases in Table 7 is about 6.6 LA nodes per each NLA node. The density of NLA nodes in the simulations was 0.01, the density of LA nodes varied as shown in column two. Column 3 shows the average number of members in set P, i.e. the average number of points required to describe node areas.

Communication radius	Density of LA nodes	Number of members in *P*	Average position estimation error
10	0.025	3.82	2.23
20	0.007	3.63	5.32
30	0.003	3.77	6.9

Table 7 Communication Radius Relation to Position Estimation Error

As it can be seen from Table 7 in case of larger communication radii the position estimates for all the nodes can be computed with a relatively small number of nodes taking into account the low density of the NLA nodes. In case of tests with smaller radii the number of required nodes was larger. The reason for that were the "outliers" in the NLA set which were near the border of the simulation area.

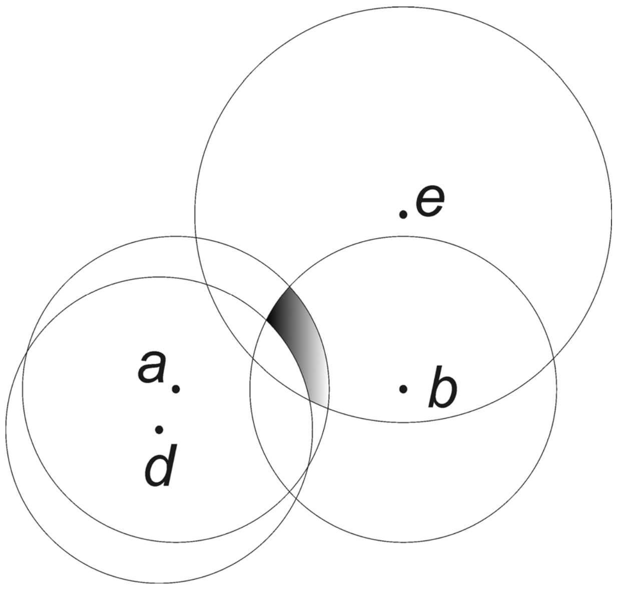


Figure 23 Member of set K may decrease the total area but not the area radius

Figure 22 illustrates how increasing LA node density has a greater affect on decreasing the number of NLA nodes with no position estimate when the density of LA nodes is lower. After LA node density has reached 0.003 and 10 % of the NLA nodes are without a position estimate the effect of adding more LA nodes is very small. This phenomenon can be contributed to the fact that the last 10% of the NLA nodes lie near the borders of the area where the LA node concentration is smaller.

The simulations also showed that using members from set K in the area definition did not reduce the radius of the NLA area much, while the number of points used for the area definition was increased substantially It is clear that when the simplest area representation (the circle) is used for analysis, the steps in the algorithm involving members from set K can be skipped. However as including members from set K does reduce the total area of the NLA node these computations should be performed if more complex area representations are used. It can be seen from

Figure 23 how including node *D*, which belongs to set K (all other LA nodes A, B and E belong to set L) does not decrease the area radius much but it does decrease the total area substantially.

4.5.2.3 Test results

An experimental test bed was set up in the Department of Computer Control in the Tallinn University of Technology. An indoor positioning system consisting of Cricket beacons was placed in the environment with the beacons set up to give references for a Cartesian coordinate system defined in the test area. MICA2 network nodes and RF ID tags were placed at known coordinates and a mobile node was moved around in the environment collecting data on positions where detection of a specific device was possible. The mobile node was equipped with a Cricket receiver, a PC, a MICA2 mote with a serial adapter and a RF ID tag receiver. The Cricket beacons developed at MIT and manufactured by Crossbow Technology, Inc. operated at 433 MHz. The MICA2 motes developed at UC Berkeley and also manufactured by Crossbow Technology, Inc. operated at 433 MHz. RF ID receiver was a WJ Communications' MPR6000 module in PC card format which operated in the 902-988 MHz range. The mobile node positioned itself using Cricket beacons. The PC on the mobile platform was able to communicate with the MICA2 motes placed in the environment using the MICA2 mote with the serial adapter attached to it.

Before any positioning tests were run, initial experiments to determine the communication radius of the motes, and the detection radius of RF ID tags were conducted in the test environment. These radii were used when the test results were analyzed and the areas of the nodes and tags were determined.

The MICA2 motes positioned in the target area transmitted packets at predefined intervals with variable transmission power, each packet containing the transmission power at which the packet was transmitted. The receiver received the transmitted packets and recorded the coordinates at which the packets were received together with the transmission power of the packets. The recorded data was analyzed using the same program which was used for the simulations.

With RF ID tags the known directional sensitivity of RF ID tag reading became quite apparent during the tests – the tags could be read only if the orientation of the antenna was optimal for the tag. This means that the K set in the algorithm can't be used for the RF ID tag positioning as even if a receiver is within the detection radius of the tag, the tag can't be detected in case of incorrect antenna positioning.

The maximum reading range of RF ID tags was 3 meters and with selecting reading points closer to the edge of the communication area, the average positioning error using the circle representation of the node area was 80 cm. With random reading points the average positioning error was 1.8 meters. As the presented algorithm is insensitive to antenna positioning and reflections of the signal do not have a big effect on the precision of the algorithm (when the normal detection range of a tag is 3 meters and a tag is detected then it can be presumed quite safely that the tag is in the given radius) it can be used quite well for positioning RF ID tags by a location aware mobile RF ID reader.

For motes the communication range was between 2 – 15 meters (depending on the transmission power used) and the average positioning error was 0.36 meters. The K set was not used in case of the motes either as there we enough readings in the L set.

4.5.3 Comparison of multilateration and CABP method

In order to evaluate the performance of the CABP positioning method, a case study was performed that allowed comparing the CABP method with established multilateration methods. The positioning system used in the case study is completely made up of smart dust nodes. The network consists of location aware (LA) nodes and not location aware (NLA) nodes that are worn by users. The LA nodes provide a positioning service to the NLA nodes. The practical part of the case study was conducted by Raido Pahtma.

The case study used service based software architecture to cater for the dynamic nature of the positioning application. The service requires that the node interested in obtaining a position estimate transmits broadcast messages to the network (a service request), containing information about the transmission power of the messages. All the LA nodes estimate the distance to the NLA node based on the information provided in the message and the RSSI to distance function embedded in the node. Messages transmitted with higher power also travel to a greater distance and therefore the transmission power of the message must be taken into account when converting the RSSI value to distance. The RSSI to distance function could also be refined by monitoring normal network traffic, but regular messages usually do not contain information about the node and transmission properties (power, antenna/radio type). The service request messages also contain a unique identifier associating a tag with the physical location of the node at the time of the transmission. Essentially one service request message is enough, but if the node is relatively stationary, then several can be transmitted to take advantage of averaging and additional information provided by using different transmission power levels.

The location aware nodes store the information for a limited time and perform the distance estimation task, complementing it if additional information is obtained. The distance estimates are available to other nodes. The position estimate can be calculated by the NLA node being positioned by retrieving the distance information from LA nodes or by LA nodes in the network if the transmitter asks them to provide a position estimate. In the latter case a computation node selection process is initiated which results in the selection of the LA node that collects the distance estimations from other LA nodes and applies a positioning algorithm.

The selection is performed using a simple election scheme where each node expresses its cost to fulfil the request as a comparable value and announces that value to the others, delaying the announcement if the cost is high (which may be the case when the node has fulfilled several requests recently). If a node hears that another has announced a lower cost, then it drops its own attempt, decreasing its cost for subsequent requests. A node that completed the calculation increases its cost value for future operations. This scheme could be extended to also take into account workloads from other services executed by the node and energy reserves available at the node.

Querying the distance estimates can be done with several constraints to alleviate the potentially large number of responses to a broadcast query. For example, smaller estimates are queried first and larger ones are taken into account only if too few were initially obtained. This arrangement actually mostly benefits the multilateration algorithm, which is more severely affected by inaccurately long distance estimates; the CABP (Preden, 2006) algorithm naturally filters such estimations. The packets containing the distance estimates also contain the coordinates of the LA node that estimated the distance.

The positioning algorithm component is started once a sufficient number of distance estimates have been acquired. Several algorithms can be used for computing the estimated position of the NLA node from the distance estimates. The current system uses two methods: CABP and multilateration (Lee, 1975). The test system runs both algorithms on the same dataset to evaluate the performance of the algorithms.

As a first step in the CABP algorithm, the distance estimates are sorted in an increasing order to favour smaller areas, which define the location more precisely. The algorithm then works by incrementally refining the common area of the communication areas. This is done by examining the intersection of each new communication area with areas already used in defining the common area. New information is used to decrease the common area and areas that were made redundant are discarded.

It is normal for distance estimates to be greater than the actual distance, because the radio signal is weakened by any obstacles in its path. However, it is also entirely possible for the RSSI measurement to indicate a smaller distance. This can create a situation where the communication areas of some location aware nodes do not intersect, essentially indicating that the NLA node is in two places at once. In such cases the algorithm simply increases the conflicting areas proportionally until they do intersect. It is reasonable to expect the node to be somewhere between the areas, although it is not always the case either.

The final common area description is formed once information from all the location aware nodes has been processed. At this step, the area is a set of intersection points and segments of circles connecting the intersection points. A representation like this is too detailed for most cases, so it is condensed down to a circle, described by the centre of the area and the distance to the furthest point from the centre. The more detailed description can certainly be used as well, but it is currently not practical due to the low accuracy of the input data.

The CABP algorithm provides a result if it is given at least two input areas (actually it provides a result with one area, but then the result is the same area).

The hyperbolic multilateration algorithm requires a minimum of three distance estimates to work, but will not provide a result if the reference points are on the same line.

The software of the nodes also includes several optional components, which are used to gather data about RSSI measurements, estimated distances and the data that reached the positioning node. This data is retrieved by a PC after each positioning attempt for later analysis.

The positioning software can be easily deployed on any smart dust node running the TinyOS 2.1 operating system that can provide a distance estimate to other nodes.

4.5.3.1 Study 1

The first study used standard Crossbow MicaZ motes. MicaZ motes have a CC2420 radio which has 31 different output power levels and enables the measurement of RSSI with a granularity of 1 dBm. The placement of nodes for the study was idealized as all the nodes were approximately 0.5 meters above the floor on wooden stands, including the NLA node. Such a setup was used in order to minimize the possible interferences from the steel reinforced concrete floor.

The first configuration used 9 LA nodes in a grid with the distance between nodes 1.5 meters in one direction and 2 meters in the other direction. The NLA node was moved to 54 different locations within the area. Later the centre nodes of each edge of the grid were removed and the measurements repeated.

With all 9 nodes in place, the average positioning error for the centre of the estimated area was 88 cm using the CABP algorithm and the average positioning error was 152 cm using the multilateration algorithm. The multilateration result was generally just indicating that the node is somewhere near the centre of the area.

With 5 nodes, the average positioning error for the centre of the CABP area increased to 132 cm and slightly decreased for multilateration, to 149 cm. The impact of the removal of some of the nodes on the performance of CABP is considerable, especially taking into account the small distances between the nodes.

Finally, 14 nodes were distributed over three rooms and a portion of the corridor connecting the rooms. The NLA node was placed in 5 different locations in each room and in some additional locations between the rooms. The system worked reasonably well for CABP, with an average positioning error of 115 cm. The multilateration algorithm showed however much poorer results, having an average error of 700 cm.

While the system was able to perform reasonably well, the placement of nodes on special stands is clearly not very practical.

4.5.3.2 Study 2

The second study used custom Crossbow Iris OEM based motes with Antennafactor ANT-2.4-WRT dome antennas. The Iris modules are equipped with AT86RF230 radio which has 16 output power levels and can measure RSSI with a granularity of 3 dBm. The AT86RF230 has better range than the CC2420 radio of the MicaZ motes. The study used 27 LA nodes, which were attached to the ceiling in the Department of Computer Control in Tallinn University of Technology, 4 in each of the two smaller rooms, 6 in the larger room and 13 in the long corridor connecting the rooms (Figure

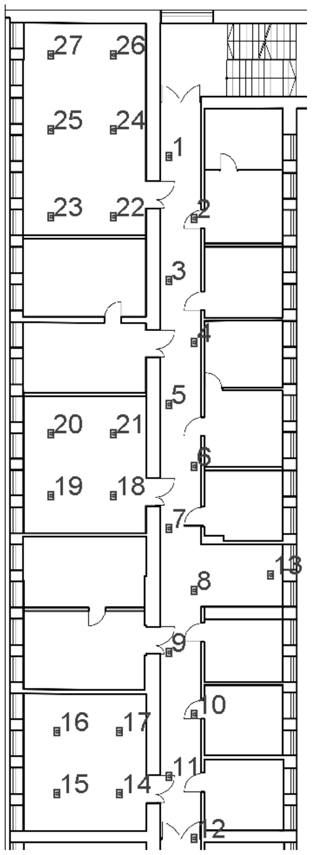


Figure 24 Anchor node placement in the case study

24). This placement is much more practical, but makes distance estimation less accurate. The NLA nodes in this setup are planned as devices that can be stored in the users pocket and provide a link to the sensor network for a PDA or tablet PC, connected to the NLA node with a serial cable or a Bluetooth cable replacement module. For these experiments, the NLA node was attached to a stand, approximating the height that an average user might carry the device at.

The system is able to take into account the z coordinates of the nodes if the NLA node provides its own z coordinate (height from the floor, which is the base for the z-axis). The multilateration algorithm can be expanded to three dimensions without a substantial increase in computational complexity, but CABP would become computationally far too complex to be practical on current smart dust motes. In practice though, the accuracy of the distance estimates is so low, that we can safely ignore the height difference altogether.

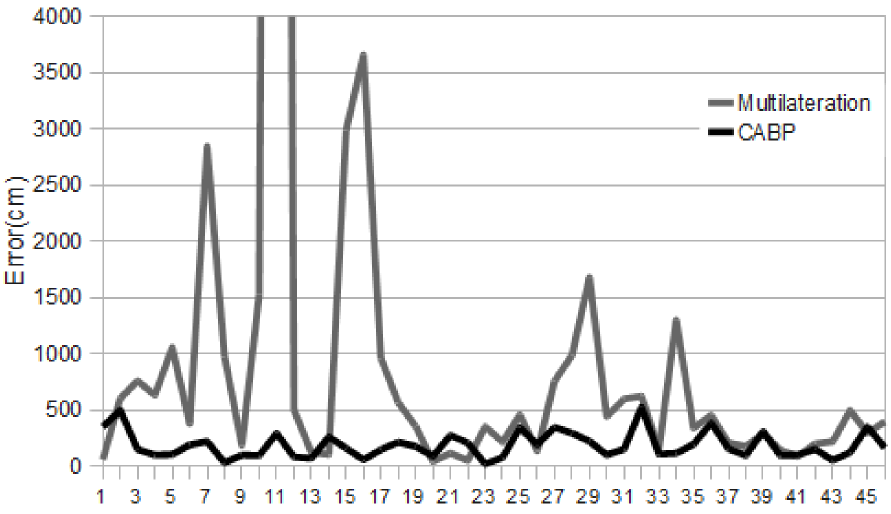


Figure 25 Positioning error for the Multilateration and CABP methods

In the first part of the study, 180 positioning attempts were analyzed from 54 different locations. This covered several locations that were near walls or otherwise difficult for positioning. CABP gave an average error of 292 cm, multilateration however was unable to provide reasonable results. The multilateration result was usually corrupted by distance estimates created based on signals that had travelled though walls and therefore it was decided to use the shortest available distance estimates in case of the multilateration algorithm for positioning the node. The system was complemented to find three of the shortest estimates from nodes that form a triangle and use those for positioning with multilateration.

In the second part of the study, with the new multilateration configuration, the NLA node was moved to 46 different locations, trying to approximate a trajectory a human user might take. Multilateration now performed much better in some locations, but was still unable to provide good results in others. The average error for the CABP algorithm was 188 cm and around 500 cm for the multilateration algorithm, if some of the more unsuccessful positioning attempts are excluded.

The results of the study show that a positioning system based on smart dust motes using received signal strength indication is usable (with limitations) for location aware applications that do not require very accurate positioning information.

In the performed experiments the CABP algorithm showed better results when compared to the multilateration algorithm. The better performance of the CABP algorithm is due to the filtering capability that is inherent to the algorithm.

The positioning accuracy of both algorithms can be improved by using better filtering and data conditioning algorithms for the distance estimations but this is subject to further study.

4.6 Positioning & navigation by reference

A novel positioning and navigation method developed within the scope of the Roboswarm project (EU project supported by Sixth Framework Programme for Research and Technological Development, contract number 045255) is discussed in the current section.

The reason for development of a new positioning and navigation method lies in the fact that the target application does not have a fixed pre-deployed positioning infrastructure, i.e. there are no location aware beacons and there is no universal (in terms of the application) coordinate system that could be used. However the application specification foresaw the use of RFID tags, so the developed method relies on RFID tags placed in the environment for positioning and navigation.

A factor that complicates the navigation and positioning system design lies in the fact that the distance and angle of a tag cannot be determined exactly by a mobile agent. However there was a need for a distributed positioning scheme, which would aid mobile agents in positioning and navigation. The agents in the application have to fulfil cleaning tasks and the positioning and navigation scheme was required in order for the robots to be able to navigate from one cleaning area to the next and also navigate within the cleaning area.

The agents can leave information in the environment by writing it to the RFID tags. The information written to the RFID tags can be again read by the other agents that move around in the environment. That way the agents are able to update (among other things) the positioning and navigation information on the RFID tags. Other types of situational information in the environment was related to the application specific tasks of the agents, e.g. the required cleaning frequency of areas, the last cleaning operation performed in an area (including the time when the operation was performed and the type of operation).

The type of communication used in this application – indirect exchange of situational information (including both application specific and navigation related information) falls in the category of indirect interaction, where the provider of the information has no knowledge of by whom and when will consume the information. Neither can a consumer be sure that the information it requires is available when the information is needed.

An interesting aspect of the application lies in the fact that the situational information is written to the RFID tags in the environment and the information on the RFID tags can be only read if the agent is in the proximity of the tag. That means that the agents must be able to operate with limited situational information and when better situational information is required the agent must move around to discover more information providers, i.e. RFID tags. So the functionality of the mediator as a provider of situational information is quite limited in this application.

The developed scheme does not rely on a global map or any kind of global knowledge of the operation area. The following terms are used:

- **Base vector** – a vector that is used as a base vector for describing the angles in a virtual space
- **Virtual space** – a space that contains one base vector; one physical space (one room) may contain one or more virtual spaces, several physical spaces may be contained in one virtual space
- **Position pointer** – information on an RFID tag on the location of another RFID tag, some object or another virtual space. The information is in the form of polar coordinates where the polar axis is identical to the base vector of the virtual space and the radial coordinate expresses the distance

between the RFID tag where the information is written to and the object that the pointer points to. (e.g. “virtual space A, 10m, bearing 10°”).

- **POI** – point of interest, an entity relevant to the agents, identified by an RFID tag, e.g. a cleaning area, a virtual space, an entity in a virtual area (table, bed, etc)

- **Tag** – RFID tag that may or may not have rewriteable memory.

The relevant objects and waypoints of the area are stored in a distributed manner on the RFID tags in the area. Each RFID tag contains information on its surroundings, enabling positioning (an agent can determine where it is) and navigation (a tag may contain information on what is the physical relation between the tag, POIs and virtual spaces).

When a mobile agent enters a virtual space it establishes the direction of the base vector of the virtual space (and the relation of the mobile agent to the base vector) using a set of RFID tags that contain enough information to establish orientation.

The proposed positioning scheme allows for iterative refinement of the navigation information. In addition the agents navigating in the environment need no information from outside – all the information required for positioning and navigation can be obtained from the environment.

The information on the RFID tags could be extended to also contain some trajectory information in addition to line-of-sight information – e.g. pointers in a polar coordinate system.

4.6.1.1 Virtual spaces

A virtual space is a physical area identified during the deployment of the system as logically non-separable (e.g. a single room is usually a single virtual space, whereas a corridor and a room are separate virtual spaces). A virtual space is identified by RFID tags containing information on the virtual space.

Within a virtual space there exists a set of tags that allows for the establishment of the base vector of the virtual space. There may be also RFID tags in a virtual space that contain information which allows positioning & navigation within the virtual space – the tags can be used to identify objects in a virtual space and information on the tags can be used to navigate from one object or tag to another.

4.6.1.2 Virtual Space Weight

The static weight of each virtual space is specified during deployment. Dynamic weight parameters are computed based on situation parameter values that are written to RFID tags in the virtual space by the mobile agents and updated by the agents on a regular basis. This means that the agents compute the virtual space weight incrementally based on the information they obtain from the RFID tags they can communicate with. The virtual space weight situation parameter value computed by an agent is then compared to the weight that is written on a tag. If the situation parameter values are not equal the situation parameter value on the RFID tag is overwritten. The following aspects are considered in the computation of a virtual space weight:

- The number of virtual spaces that are accessible from the given virtual space (the higher the number of accessible spaces the higher the weight)
- The time interval from the current time to the time instance when the current virtual space was cleaned the last time (the longer the interval, the higher the weight)
- The time interval from the current time to the time instance when the virtual spaces accessible from the current virtual space were last cleaned (the longer the interval that higher the weight)

So the weight of a virtual space can be computed using the following formula:

$$weight_\alpha = static\_w_\alpha \times \left(next\_clean\_time - last\_clean\_time\right)_\alpha + \sum_{i=1}^{n} static\_w_i \times \left(next\_clean\_time - last\_clean\_time\right)_i$$

where

$weight_\alpha$ is the weight of virtual space α
$static\_w_\alpha$ is the static weight of virtual space α assigned at deployment
$next\_clean\_time$ is the next cleaning time of the virtual space
$last\_clean\_time$ is the last cleaning time of the virtual space.

4.6.1.3 Position Pointers

To make navigation between virtual spaces and also within a virtual space possible the RFID tags in a virtual space also contain pointers to neighbouring virtual spaces and to POIs within the current virtual space.

If mobile agent needs to move to another virtual space it will select it based on the pointers available in the current virtual space that point to other virtual spaces. The next virtual space is selected based on the weight of the virtual space.

4.6.1.4 Dynamic updating of navigational information

The setup phase consists of the following steps:

- RFID tags are placed in the environment. In each virtual space at least two tags must be placed for specifying the base vector. The reason why several tags are required for the base vector lies in the fact that there is no practical way to obtain universal orientation information in an artificial environment – compass cannot be used as due to the large metal objects in artificial environments the earth's magnetic field is distorted. All the POIs must be also marked with an RFID tag.

- Information on virtual locations and/or relations between tags is written to the RFID tags. The locations of the RFID tags and the relations of the tags in physical space can be determined either by either a location aware mobile agent (which determined its location using a map or other means) or by a human. The virtual space identifiers must be assigned by humans anyhow.

- During the setup phase a location aware mobile agent must navigate through all the rooms and write information on the virtual spaces (and possibly navigational information on POIs) to the RFID tags.

Agents moving in the environment can write navigational information on the environment to the RFID tags as they build routes and move past tags in the environment.

Examples of the usage scenarios can be following:

- An agent coming from one virtual space and arriving at another virtual space can write to an RFID tag the direction (and distance) of the virtual space it came from, once it has determined the base vector of the virtual space it arrived at.

- An agent moving around in a room, moving from one tag to another can write the direction (and distance) of the tag it came from to the tag it arrived at. That way the navigational information can be incrementally improved with agents moving around quite randomly initially and when the navigation information has been improved the agents can move around in a more deterministic manner.

4.6.1.5 Sample use case

A mobile agent enters a room, reads a tag in the room, which identifies the virtual space. The situational information on the virtual space is updated. The agent moves on, trying to locate the base vector tags in the room. Once the RFID tags that enable determining the base vector direction of the current virtual space the agent determines the base vector angle and updates the base vector situation parameter. Once the base vector situation parameter is valid (it contains the base vector angle for the room that the agent is in) the agent locates the tag, which points to the point where the cleaning in the room should be started. The agent moves to the point where the cleaning should be started by employing the situational information obtained from the RFID tags in the virtual space and starts cleaning the room.

Once the cleaning has been finished, the agent re-establishes the base vector situation parameter value (during cleaning the base vector information may have become corrupt). Once the base vector has been re-established the agent locates the situational information on the virtual spaces accessible from the current virtual space. Based on the information acquired on the virtual spaces the agent selects a new virtual space, locates the navigational information to that virtual space and navigates to the next virtual space.

The positioning algorithm

The positioning and navigation algorithm is described in the form of Moore state machines, i.e. actions performed in each state depend only on the current state and state transitions are triggered by inputs. Some states have no actions and these states are introduced to simplify the understating of the algorithm.

Acquiring base vector

The base vector information is required to perform any navigation in a virtual space. Once an agent enters a new virtual space the first task of the agent is to update the agent’s base vector situation parameter value.

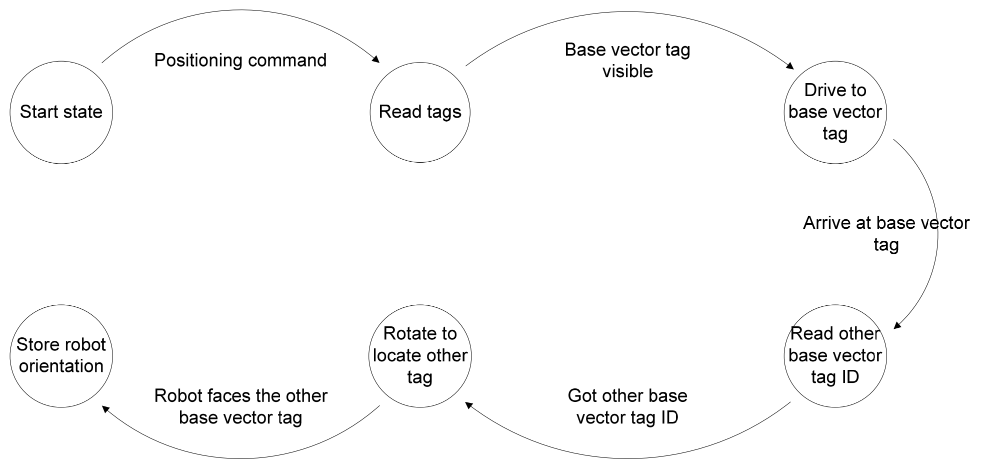

Positioning command
Base vector tag visible
Start state
Read tags
Drive to base vector tag
Arrive at base vector tag
Store robot orientation
Rotate to locate other tag
Read other base vector tag ID
Robot faces the other base vector tag
Got other base vector tag ID

Figure 26 Acquiring base vector

The base vector is a situation parameter that is acquired using the base vector tags. The mobile agent must constantly update the base vector situation parameter values as it rotates, as the angle of the base vector in relation to the orientation of the mobile agent changes. The base vector situation parameter value becomes invalid when the mobile agent leaves a virtual space.

Moving to POI

In order to move to POI the situational information on POIs must be acquired first, and naturally the base vector situation parameter value must be valid.

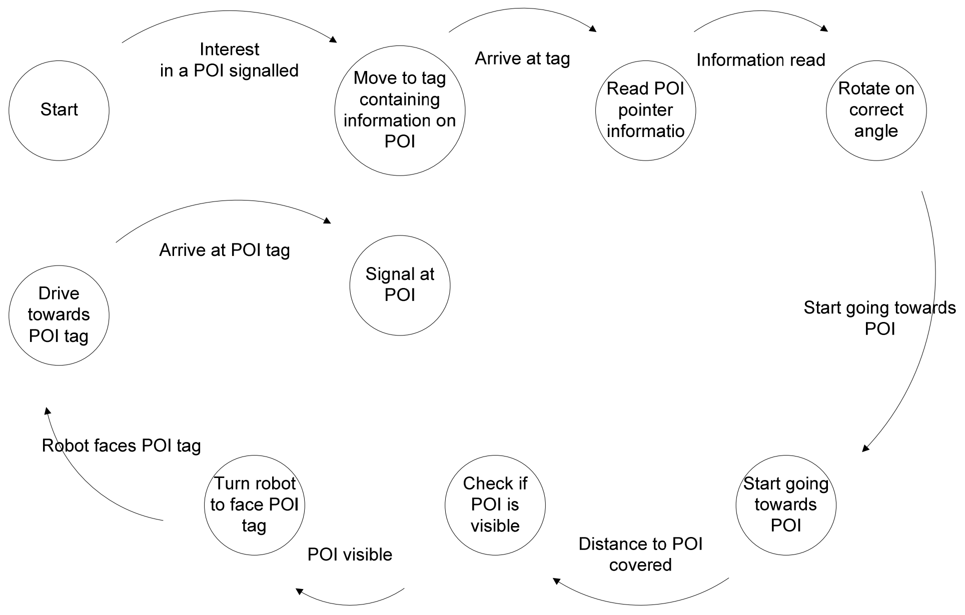


Figure 27 Moving to POI

4.7 Vehicle guidance systems in NEC context

The main purpose of a Network Enables Capabilities system is to provide situational information to its components so the principles of situational information representation and propagation outlined in the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter can be well applied in these systems. The work done by in this area that the author has been involved in is described in greater detail in (Meriste, et al., 2005), (Preden, et al., 2009) and (Motus, et al., 2009).

4.7.1 Network enabled capabilities – NEC

The term networked enabled capabilities (NEC) (Jonas, 2005) has become quite common but mostly it is used in a narrow sense – as a system enabling human users to gain greater situation awareness. However NEC can also be viewed in a wider sense as a source of information for both manned and unmanned collaborating entities (agents). Especially unmanned platforms can substantially benefit from the shared information available in the NEC system. In this context we can view a NEC system as a heterogeneous distributed computing system, the humans being part of the system, acting both as consumers and providers of data. In order to benefit from the information available in a NEC system the design of an autonomous system must facilitate the use of that information. A NEC system can be essentially viewed as a multi-agent system so the task at hand is how to make use of situation parameters in a multi-agent system.

It must be noted that there is no situation that would be interpreted by all computing (or human) agents which are part of a NEC system in the same way, and that would cause the same behaviour in all agents. Vehicle guidance systems as a type of agents that are part of a NEC system can benefit from the situation awareness concepts. Vehicles can act as situational information providers to other agents (e.g. providing situational information derived from sensors attached to, or communicating with the vehicle), situational information consumers and vehicles can be also used for propagation of situational information.

There are various types of situational information relevant to vehicles, examples of this are terrain type (affects the speed of movement), obstacles (obstacle avoidance), tactical and strategic situation, information on peers and adversaries. There can be also many sources of situational information, such as other vehicles (air & ground), stationary agents and remote sources. One source of situational information to agents (as part of a NEC system) can be smart dust motes. These devices, although very constrained in their computational power are capable of computing lower level situation parameter values as illustrated in the previous case studies. Using appropriate software architecture these devices can be situation aware themselves, utilizing situation parameter values computed locally or by other agents. Vehicles are one source of situational information to such devices, providing for example location information to deployed motes. So smart dust motes can be consumers of situational information and they can be a source of situational information to other devices. If a smart dust mote is location-aware it is able to associate spatial validity information with the situation parameter values computed locally.

A multi-layer architecture would be preferable for the design and implementation of a situation aware mobile agent. Combining all aspects of the mobile agent into one software layer would result in too complex design which is difficult to maintain. In order for the mobile agent to be situation aware they need to have at least two layers dealing with the situational information. The first one interprets the situational information available to the agent, and the second one selects a behaviour matching with the situation as perceived by the mobile agent and with the properties of the agent.

4.8 Case study of situation aware mobile agents

4.8.1 Mobile agent

The first mobile agent case study was developed by Jürgo Preden utilizing the Mathworks Matlab models created by Andri Riid for researching the low level control aspects. The case study was developed within the scope of the ITEA project "Gene-Auto – Automatic Software Code generation for Real-time Embedded Systems". The implementation of the physical agent (a mobile robot) was done by Raido Pahtma and Risto Serg. The results of the case study (including the navigation of the physical robot) were presented at the ITEA fair in Rotterdam in 2008.

The first case study is concerned with the control of a mobile agent. The mobile agent is equipped with proximity sensors for sensing obstacles and differential drive for controlling the direction and speed of the robot. The case study was performed in several steps, in each step the complexity of the software model of the mobile agent was increased, which also resulted in improved accuracy of control. The case study was initially implemented and simulated in Mathworks Matlab as a combined Simulink/Stateflow model. The model of the mobile agent, the low-level fuzzy logic based controller of the mobile agent and the simulation environment was implemented by Andri Riid (Riid, et al., 2004) while the high level control of the mobile agent (including sensor signal interpretation and mobile agent path planning) was implemented by Jürgo Preden. The hardware abstraction layer of the physical robot was implemented by Risto Serg and the agent’s software environment including the integration of the code generated from high-level models was done by Raido Pahtma.

As noted in the beginning of the section a multi-layered architecture would be preferable for the situation aware mobile agent as there are several aspects of the mobile agent that need to be addressed. The agent’s software consists of four layers, the two lower ones are responsible for control of the vehicle while the higher two layers are responsible for 1) interpreting the situational information and 2) selecting appropriate course of action depending on the situation. The layered approach has the advantage of decoupling the control logic from the situational information processing which allows changing separately either one or the other part as desired. One can even replace the lower level control logic completely (for example with the control logic from another mobile platform) and still utilize the situational information handling layers (with appropriate modifications if required) from the original design.

The low-level control is clearly best handled with more or less conventional control methods utilizing for example PID (proportional–integral–derivative) controller or fuzzy logic. However, in order for utilize the situational data the agent needs have at least two layers above the control algorithms. The higher levels acquire and interpret the situational information available to the agent and select a behaviour based on the situation as perceived by the mobile agent and the properties of the agent.

The layered architecture was achieved by extending the control architecture presented in (Riid, et al., 2004) by adding a situation evaluation and a control layers on top of the existing two control layers. The situation evaluation layer interprets the situational information and is implemented as a set of state machines. In the current mobile agent case study the issue of acquiring and exchanging the situational information is set aside, this topic is covered in the *Middleware dedicated to exchanging situational* information section of the *Enhancing the Situation Awareness Concept for Cyber-Physical Systems* chapter. The situational information acquired from sources external to the vehicle is exchanged via shared memory store in the simulation environment. The issue of data exchange is a separate research topic which is not addressed in the current case study.

After the behaviour of the agent had been validated in Matlab simulations, code could be generated directly from the Matlab models and transferred directly to the mobile agent. Such an approach is preferable as indicated in (Weigert, et al., 2008) – automatic code generation potentially introduces major productivity improvements in software production due to the ability to have industrial quality software code available on the target processor with a reduced amount of development time. Any change in system requirements (situation parameter or vehicle behaviour specification) can be first integrated to the high-level model where also the desired effect can be verified, after which the code can be generated from the model and integrated to the target platform. Code generated from a high-level model has several advantages over hand-written code, for example code maintainability is improved - when enhancements or defect fixes are made, the code is regenerated (and re-optimized) from scratch (Weigert, et al., 2008). Also code reuse is improved as the lack of platform detail embedded in a design means that cross-platform reuse of designs and tests are enabled (Weigert, et al., 2008). Within the scope of the case study code was generated from Matlab models using the Gene-Auto code generator (Toom A., 2008), developed during the Gene-Auto project.

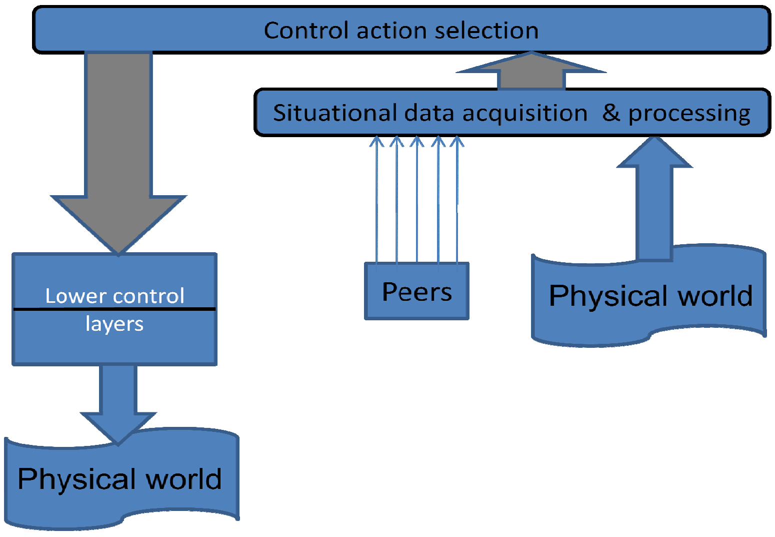


Figure 28 Multi-layer vehicle software architecture

4.8.2 Mobile agent task and physical implementation

The physical implementation of the mobile platform was a four-wheeled vehicle with differential steering equipped with four servo motors and proximity sensors in the front of the vehicle as depicted in Figure 29 Vehicle. The vehicle is capable of positioning itself in an indoor environment using an indoor positioning system based on Cricket positioning hardware from Crossbow Technology. The Cricket positioning system is based on special Cricket devices that are equipped with both ultrasonic and RF interfaces, enabling distance estimations based on time difference of arrival of RF and ultrasonic signals. The robot is equipped with several embedded computers and the task of the robot is to move from an arbitrary start position to a desired end position while avoiding obstacles.

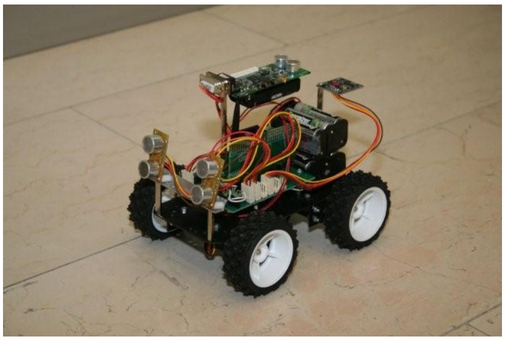


Figure 29 Vehicle

The front of the mobile platform is equipped with 2 SRF08 ultrasonic rangers for detecting obstacles in the range of 3 – 600cm. In the centre of the robot there is an electronic (MEMS) compass CMP2X from Mindsensors.com for determining the orientation of the robot. A Cricket MCS41CA positioning system receiver from Crossbow Technology enables positioning of the robot.

The complexity of sensor interfaces and servomotor control is contained in the Hardware Abstraction Layer (HAL), which allows to apply generalized control algorithms on the mobile platform with minimum or no modifications when the hardware of the robot changes. HAL consists of Atmel AVR based Robostix board from Gumstix, Inc. and corresponding firmware that interfaces with actuators and sensors. The Robostix board provides PWM (pulse width modulation) signal to servo motors and I2C bus master service for ultrasonic rangers and compass. The HAL allows for 4 speeds forward and 4 speeds backwards both for left and right side motors. This configuration gives flexible manoeuvring abilities the mobile platform. The HAL is interfaced to higher level hardware via a serial interface.

All the layers of the agent software run on a Gumstix Connex 400 board based on a 32bit Intel XScale processor running custom Linux OS. This board is interfaced to HAL and the positioning system node by serial interfaces. The agent software code generated from the Mathworks Matlab models is loaded to this sub-system and the messaging interface to the HAL layer is sufficiently high-level, leaving the lower-level control issues to the HAL layer.

4.8.3 Lower control layers

On the lower control layers the control actions are computed based on the current state and the objectives of the vehicle. Vehicle position is determined by three state variables x, y and Φ = [-90°, 270°], where the latter is the angle between onward direction of the vehicle and the x-axis Figure 30 in an established coordinate system. The width and length of the vehicle are denoted by *w* and *l*, respectively.

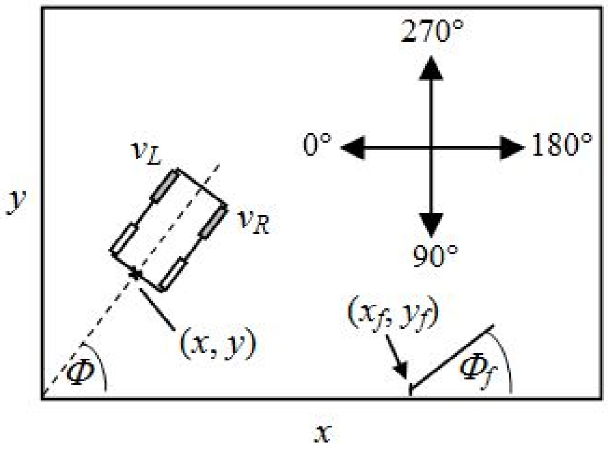


Figure 30 Vehicle and its main variables at the low level control layer

The problem for the low-level control is formulated as follows: the vehicle must arrive from the arbitrary initial position (xi, yi, Φi) to the predefined destination (xf, yf, Φf). The vehicle moves forward with the speed that is determined by speeds (tangential velocities) of front wheels, vL and vR. These two parameters also provide the means for controlling the vehicle and our task is to define appropriate profiles of vL and vR throughout the control cycle.

For vehicle movement simulation, the following set of equations is implemented in MATLAB.

$$\begin{cases} \Phi_{t+1} = \Phi_t + \dfrac{(v_R - v_L)}{w}\Delta t, \\ x_{t+1} = x_t + \dfrac{(v_R + v_L)}{2}\Delta t \cos\Phi_{t+1}, \\ y_{t+1} = y_t + \dfrac{(v_R + v_L)}{2}\Delta t \sin\Phi_{t+1}. \end{cases}$$

Implementation of the control task is based on the ideas of (Riid, et al., 2004), and is distributed between two units (Figure 31). The main component of the system is the fuzzy logic based trajectory mapping unit (TMU) that specifies the optimal vehicle angle (Φr) for the given point in input space determined by its current coordinates x and y. Computation of speeds vL and vR that would force the vehicle to take desired orientation is carried out by a separate steering block SB.

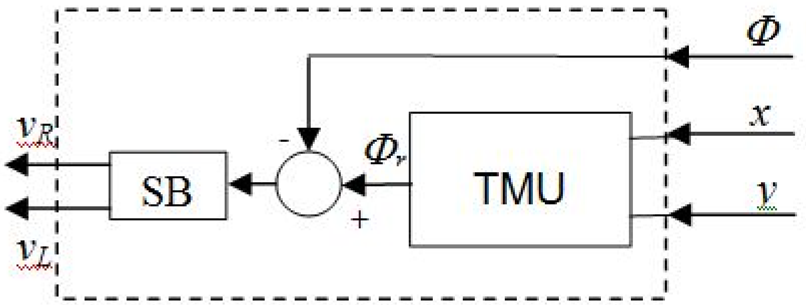


Figure 31 Hierarchical low-level control system

While the TMU used in (Riid, et al., 2004) is used without modification in the current application, the steering block in (Riid, et al., 2004), however, originally consisted only of a PD controller because in this application the vehicle was controlled by a steering angle θ of front wheels. For differential steering control scheme in current application this is complemented by a calibration block that computes appropriate v_R and v_L based on θ from the PD controller output (Figure 32) using the following equations

$$\begin{cases} v_R = \dfrac{v_{\max}}{2} + \dfrac{v_{\max} \cdot \theta}{2 \cdot \theta_{\max}}, \\ v_L = \dfrac{v_{\max}}{2} - \dfrac{v_{\max} \cdot \theta}{2 \cdot \theta_{\max}}. \end{cases} \tag{2}$$

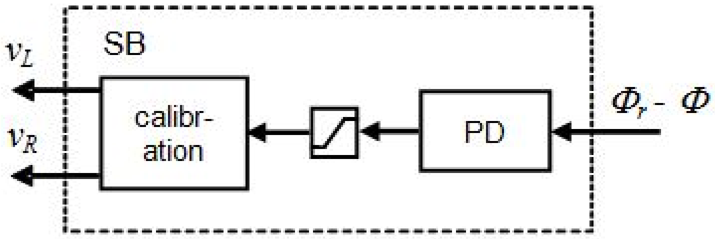


Figure 32 Contents of the steering block

It must be noted that the maximum forward speed of the vehicle is determined by vmax/2.

The performance of the lower-layer control algorithms was validated via stand-alone simulations in Mathworks Matlab without the higher situation-aware layer.

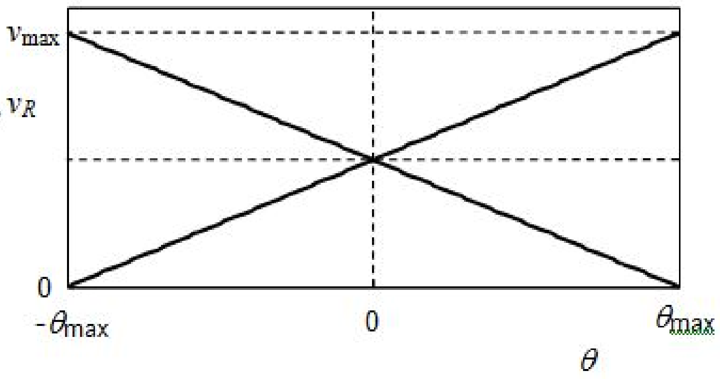
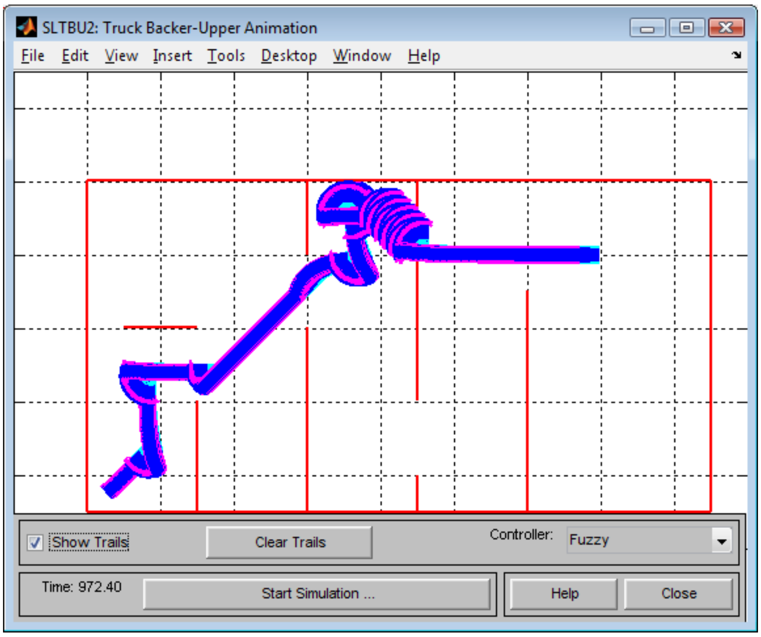


Figure 33 Functions vR = f1(0) and vL = f2(0)

4.8.4 Situational information processing and control layers

The task of the mobile agent is to navigate through a maze of obstacles. The situational information available to the agent locally is limited to the information on obstacles from the proximity sensors and the last actions that the agent had taken (including the results of these actions). The situational information processing and control layer are implemented as a set of state machines in Stateflow.

The situational information available to the situational information processing layer (to the Stateflow state machines) can be acquired from the sensors attached to the agent (local situational information), or from the other agents that are part of the simulation (either mobile or non-mobile) in which case a shared memory store was used as interaction medium. Depending on the current situation and the situational information received, the appropriate course of action is chosen by the control layer and target set points are communicated to the lower level. The control layer that utilizes the situational information changes the destination for the mobile agent when an obstacle is encountered and once the agent has passed the obstacle the original destination is restored. The information on the current destination is propagated to the lower control layers that handle the control of the agent.


SLTBU2: Truck Backer-Upper Animation
File Edit View Insert Tools Desktop Window Help
Show Trails
Clear Trails
Start Simulation ...
Help
Close
Controller: Fuzzy

Time: 972.40

Figure 34 Obstacle avoidance visualization

A separate state machine is used for processing the value of each type of situation parameter so each situation parameter relevant to the agent can potentially change the internal state of the agent. From the states of these individual state machines the higher-level state of the agent is determined. The control action (e.g. the speed or desired destination) of the agent is selected based on the higher-level situation derived from the active states of the low-level state machines and the goal of the agent. The situation parameter interpretation and control layers were designed empirically based on system requirements.

4.8.5 Case study results

The case study showed the validity of the design approach as after the design of the agent had been validated in simulations the code was generated for the target platform from the Matlab model containing Simulink and Stateflow components, eliminating the manual coding step. The agent navigated to the destination set by the human user while avoiding the previously unknown obstacles on its way. The behaviour of the real vehicle was similar to the behaviour of the simulated vehicle after the problems arising from the peculiarities of the hardware of the vehicle were solved.

4.9 Collaborating mobile agent case study

The second mobile agent case study builds upon the architecture presented in the first mobile agent case study. The simulation was initially created by Jürgo Preden but later enhanced by an intern of the Research Laboratory for Proactive Technologies – Domenico Bianco in the summer of 2009 following to the instructions of Jürgo Preden.

The exchange and utilization of situational information by mobile entities is enhanced in this case study (when compared to the first mobile agent case study) by explicitly separating local and global information. Instead of situational information being directly exchanged by the mobile agents, each agent has a situational information mediator that provides the relevant situational information to the control parts of the mobile agent when the mobile agent requires it.

The architecture for exchanging situational information is similar to the architecture presented in section *3.4 Middleware dedicated to exchanging situational information*. However, instead of the approach suggested in the referenced section, the mediators do not interact with each other directly but instead there is a central data store to which all the mediators provide information and from where the mediators fetch the information required by the mobile agent they represent.

This simplifies the design of the mediators as some of the features of mediators need not be implemented. For example, the discovery and negotiation phase between the mediators is not required – this however does not change the basic concept of situational information mediation. All the situational data generated by a mobile agent is written to the *data out* data store of the agent. The *mediator* selects the situational data relevant to other mobile agents (since the situational information subscription mechanism was not implemented the information relevant to other agents was hard-coded into the simulation), and writes it to the central situational data store from where the other *mediators* can fetch the data. In the same manner the *mediator* fetches only the data relevant to the mobile agent it represents, and writes data to the *data in* data store from where the agent is able to read the data, perform the reasoning process and select the optimum output action in the current situation. The relation between the agent and the mediator is depicted in Figure 35.

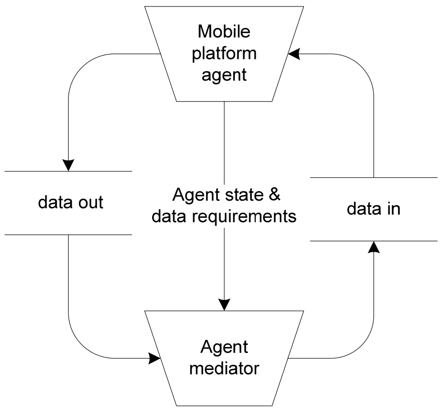


Figure 35 Agent and mediator structure

The constraints on the information required by a mobile agent depend on the type of the mobile vehicle, the state of the vehicle and the current location of the vehicle so the mediator fetches only the information relevant to the mobile agent at the current time. For example depending on the location (the current coordinates) and the direction of the vehicle only the information valid in its proximity and in the path of the vehicle is fetched from the central situational data store.

4.9.1 Simulation scenario

The approach outlined above was validated in a case study involving autonomous collaborating vehicles. In the case study several vehicles had to follow a certain path to reach their objective – a destination specified by the human user running the simulation. The terrain type on the path changes and if the terrain type is unfavourable (e.g. loose sand or mud) the vehicle is slowed down by the environment after the vehicle has entered the unfavourable terrain at medium speed.

If the vehicle enters the unfavourable terrain at high speed it is not slowed down but of course there are penalties introduced by the high speed in terms of vehicle covertness. For the reason of covertness the normal speed of movement for the vehicles is medium. In the scenario the information on the unfavourable terrain is not known to vehicles before the simulation is started so a vehicle that enters the unfavourable terrain at the normal speed of movement (medium) is slowed down. However when a vehicle detects the unfavourable terrain, it can share that situational information with other vehicles. If the vehicle obtains the information about the terrain type from another vehicle it is able to speed up before entering the unfavourable terrain and thereby avoid the slowdown.

The simulation is further complicated by adversaries in the area, which may be mobile or non-mobile. If a vehicle is in the proximity of an adversary, and the speed of the mobile agent is above a certain threshold (above medium), the adversary "notices" the mobile agent and "kills it". The situational information regarding the adversaries can be shared between the vehicles, giving the vehicles a chance to slow down before approaching the adversary.

4.9.2 Simulation components

Every vehicle in the simulation is implemented as an agent interfaced to a mediator as depicted in Figure 35. The agent is responsible for processing the situational information, and selection of agent actions, the agent mediator is responsible for providing situational information acquired from sensors and from other agents to its host agent and mediating the information generated by the host agent to the other agents. The agent utilizes the multilevel architecture introduced in the beginning of the section. This means that while the higher levels deal with processing of situational information and action selection the lower-levels are concerned with the control of the vehicle, and setting the control parameters for the vehicle, such as immediate destination and speed.

As explained above, in order to reduce the complexity of the implementation the situational information generated by the agents is propagated to a central situational data store by the agent mediators. The layout of the agent mediators and the central data store is depicted on Figure 36. Each agent mediator can then select the data that is of interest to the agent it represents based on the type and validity information of the situational data. The central situational data store is essentially a database which the mediators can add information to and make queries from.

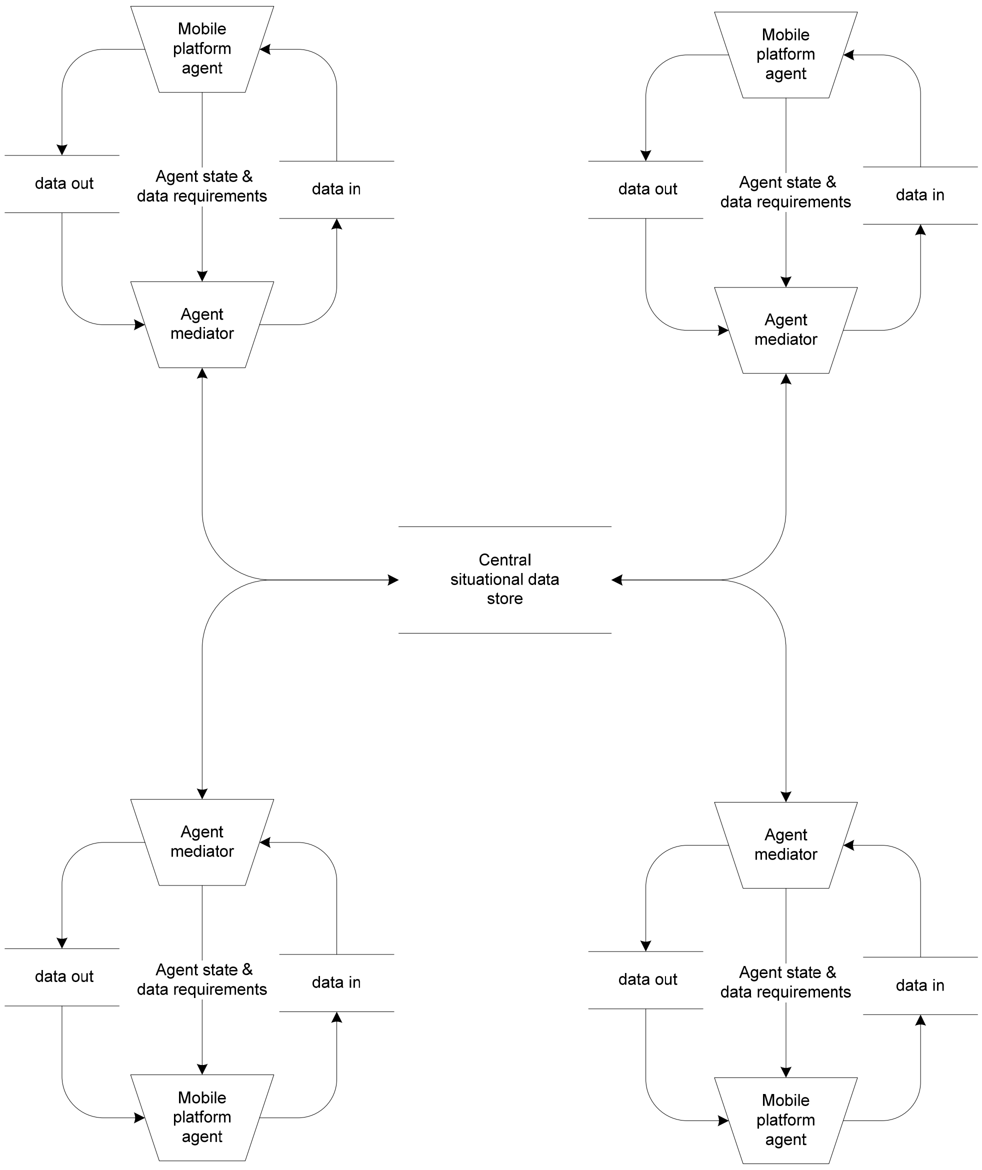


Figure 36 Exchange of situational information via a central data store

The design foresees that the agent propagates the information on situation parameters and constraints imposed upon these parameters to the mediator and the mediator is responsible for acquiring and supplying the situational data that satisfies the constraints, to the agent. An interesting aspect of the case study lies in the fact that while the situational parameters relevant to the agent are fixed the constraints on some of these parameters change dynamically as the agent moves.

For example the coordinates of an agent (which determine constraints to the situational data) are propagated to the agent mediator which then makes queries to the central situational data store, based on these changing constraints. Once the required data has been acquired it is placed in the *data in* data store from where the agent can fetch the data for processing. In the same manner the situational data generated by the agent is placed into the *data out* data store from where the agent mediator can fetch the data and provide it to mediators of other agents via the central situational data store if such data has been requested.

Clearly the use of a central situational data store is a simplification but it does not affect the behaviour of the individual mediators much. If no central data store would be used, a mediator would have to discover the other mediators, identify if the discovered mediators are able to provide information of interest and then subscribe to that information.

4.9.3 Simulation implementation and results

The simulation is a combined Mathworks Matlab Simulink / Stateflow simulation with a visual user interface for validating the simulation results. The animation on the visual user interface of the simulation is depicted on Figure 37. A common coordinate system is used in the simulation which allows identifying objects by coordinates (among other parameters) and also enables to reconstruct the path of the vehicles from the coordinate’s log.

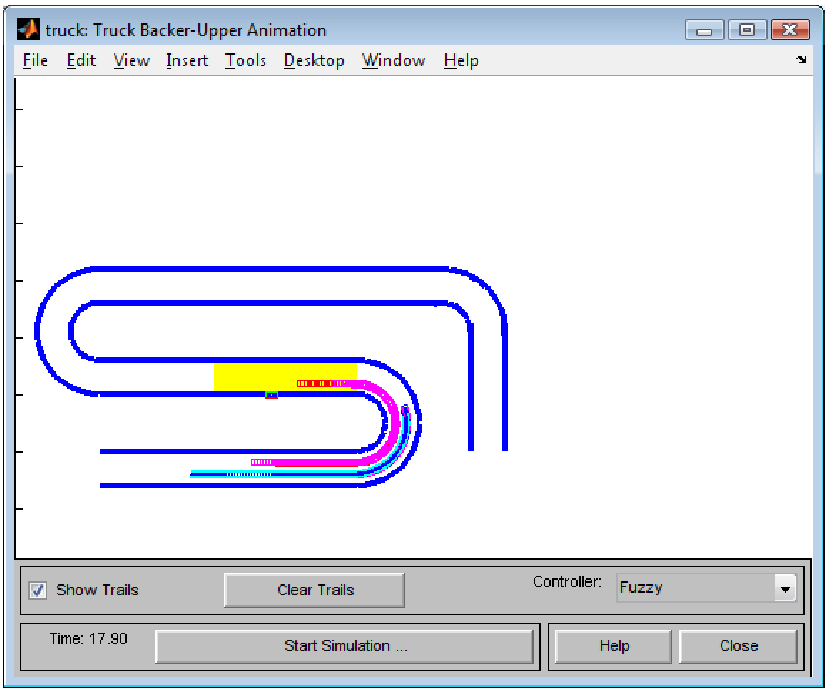

truck: Truck Backer-Upper Animation
File Edit View Insert Tools Desktop Window Help
Show Trails
Clear Trails
Controller:
Fuzzy
Time: 17.90
Start Simulation ...
Help
Close

Figure 37 Mobile platform simulation animation

The following situation parameters are used in the simulation i.e. the mobile agents, detected, stored and used the following situation parameter values:

- unfavourable terrain start and end
- static enemy
- mobile enemy

The spatial and temporal validity information (coordinates for the situational data and timestamps) are stored for all the situational information items. Each mediator selects the situational data based on the spatial and temporal properties (where and when the parameter value is valid) of the data. The validity period of the data is evaluated by the consumer of the data, e.g. the agent mediator discards the situational data record if the timestamp of the record is too old.

The information stored for different types of situational data varies depending on the data type. For example for the unfavourable terrain type both the start and end coordinates of the unfavourable terrain must be recorded, in addition to the timestamp. For an adversary the coordinates of the adversary, the type of the adversary and the timestamp should be recorded. Since the coordinates of the adversary cannot be determined by a vehicle, the coordinates of the vehicle that detected the adversary are recorded at the time instant when the adversary was detected (which is often the time instant when the friendly vehicle was "killed"). Of course the information must be interpreted accordingly as the indirect information on the location of the adversary adds uncertainty.

The simulations showed that the exchange of situational information can improve the performance of the autonomous vehicles. The concept of situational information mediator demonstrated a potential to reduce the amount of information presented to (and processed by) the vehicle agent. Vehicle agents were able to make decisions based on situational information collected by the other vehicle agents and thereby improve the probability of mission success – e.g. to reach the desired destination.

5 Conclusions and open problems

The major trend in modern computer applications is a move towards pervasive computing systems, which are inevitably interacting with the physical world. Consequently the concept of Cyber-Physical Systems will become increasingly influential in the future. While such systems are, to a great extent a reality already, we as system designers and implementers, still face great difficulties in demonstrating reliability and trustworthiness of the design of Cyber-Physical Systems. This thesis suggests that the introduction of the concept of situation awareness (and team situation awareness) will help to describe and analyse the interactions and the behaviour of future Cyber-Physical Systems, starting from the specification and design phase. Situation awareness is achieved via the use of situation parameters which are measured, or computed, and interpreted by the individual components of a system. Situation parameters are associated with validity (including temporal and spatial) information as some properties of the system are not known at design nor at deployment time, therefore the system must be able to dynamically cater for changes of these properties. This enables the system as a whole to maintain a coherent view on the world in order to harmonize the actions and responses of the components of the system and the system as a whole, as appropriate in the current situation.

The thesis extends the concept of a mediator that operates as an arbiter of situational information. The mediator can consider other types of validity information besides temporal and in addition the mediator is an active arbitrator of information, acquiring and providing relevant information.

The thesis also presents a distributed system of systems concept that builds upon the notion of situation awareness. The information exchanged in a distributed application is in the form of situation parameter values, which are computed and exchanged by and between the system components.

5.1 Open problems

The current thesis is just a starting point in the development of the concept of situation awareness in network of computing systems. Many aspects of those systems have been only mentioned in the thesis and many problems only referenced. Some of the higher-level problems that are still open are:

- constraint propagation, calculation of situation parameter values, and their validity areas

- description, validation and verification of systems' behaviour using situational information.

Situation parameter constraints propagation entails the calculation of (spatial and temporal) validity areas for situation parameter values based on application requirements. When a situation parameter value with specific validity intervals is required at the application level the source data required for computing that situation parameter values must also satisfy some specific constraints. These constraint values must be computed and propagated all the way down to the lowest level situation parameters that are used as source data.

System behaviour description based on situational information involves the specification of behaviour of system components (from which the system behaviour emerges) based on the situation parameter values. This task is non-trivial since one must consider the effects of the individual component behaviour to the behaviour to the system as a whole, i.e. the actions of the system components must be harmonized to be not orthogonal in the same high-level situation. One must also consider appropriate impact of system's behaviour on a specific state of the world and how the state of the world is reflected in the situation parameter values.

5.2 Future research challenges

While the thesis opens the door for employing situational information in computing systems it also presents some new and exiting research challenges that the author hopes to tackle in the future.

Some of these challenges are

- On-line validation and verification of Cyber-Physical Systems by applying the situation awareness concept
- Automated identification of new situations
- Level 3 of situation awareness – projection, i.e. anticipating future trends, situations, etc and learning to influence the future by proactive interactions with the environment, and/or applying possible self-X features of the system

The verification and validation of SA in interactive systems can be performed at several levels. At each level we are able to say how an entity behaves in a certain situation. We are also able to say how a situation is reached, or created by the computing system (what events must occur in the environment in order to enable the computing system to reach a particular situation). However we should be able to detect the behaviour that is not allowed and be able to assure that such behaviour does not occur.

Automated identification of new situations is a promising concept that could, when applied appropriately, ease the task of the system designer, as not all the situations (and actions in those situations) would have to be pre-specified by the system designer. Instead the system could, over time, infer the situations of interest and appropriate actions in these situations, by observing the world and the response of the world to specific actions of the system.

A topic that is very challenging but at the same also very promising is the development of Level 3 situation awareness in computing systems. This entails projecting the future states of the world based on the situational history that a computing system possesses.

6 Bibliography

Abowd Gregory and Mynatt Elizabeth Charting past, present, and future research in ubiquitous computing; ACM Transactions on Computer Human Interaction (TOCHI). - New York : ACM, 2000. - Vol. 7. - ISSN: 1073-0516.

Adams M. J., Tenney Y. J. and Pew R. W. Situation Awareness and the Cognitive Management of Complex Systems [Journal]; Human Factors. - [s.l.] : Human Factors and Ergonomics Society, March 1995. - 1 : Vol. 37.

Anderson P. W. More is Different [Journal]; Science, New Series. - [s.l.] : American Association for the Advancement of Science, 1972. - 4047 : Vol. 177. - pp. 393-396.

Artman H. and Garbis C. Situation Awareness as Distributed Cognition [Conference]; Proceedings of ECCE (European Conference on Cognitive Ergonomics). - Limerick : [s.n.], 1998.

Bahl P. and Padmanabhan V. N. RADAR: An In-Building RF-based User Location and Tracking System [Report]. - Redmond : Microsoft research, 2000.

Barsalou LW Situated simulation in the human conceptual system; Language and Cognitive Processes. - [s.l.] : Psychology Press, 2003. - Vol. 18. - pp. 513 - 562.

Barwise Jon Scenes and Other Situations [Journal]; The Journal of Philosophy. - [s.l.] : Journal of Philosophy, Inc., July 1981. - 7 : Vol. 78. - pp. 369 - 397.

Bass B. and Gurevich Y. Algorithms: A quest for absolute definitions [Journal]; Bulletin of European Association for Theoretical Computer Science. - 2003. - Vol. 81. - pp. 1 - 30.

Bulusu N., Heidemann J. and Estrin D. GPS-less low-cost outdoor localization for very small devices [Journal]; IEEE Personal Communications. - [s.l.] : IEEE, October 2000. - 5 : Vol. 7. - pp. 28-34.

Burks A. W. and Burks A. R. The ENIAC: The First General Purpose Electronic Computer [Journal]; IEEE Annals of the History of Computing. - [s.l.] : IEEE Computer Society, 1981. - 4 : Vol. 3. - pp. 310 - 389.

Carreta T.R., Perry, J.D.C., Ree, M.J. Prediction of Situational Awareness in F-15 Pilots [Journal]; The International Journal of Aviation Psychology. - 1996. - Vol. 6. - pp. 21 - 41.

Caserta Ryan J. and Abrams Lise The relevance of situation awareness in older adults' cognitive functioning: a review. - Berlin : Springer Berlin / Heidelberg, April 2007. - pp. 3 - 13.

Chintalapudi K.; Govindan, R.; Sukhatme, G.; Dhariwal, A. Ad-hoc localization using ranging and sectoring [Conference]; Proceedings of IEEE INFOCOM'04. - Hong Kong : IEEE, 2004.

Coffman E. G. and Denning P. J. Operating Systems Theory [Book]. - New Jersey : Prentice-Hall, 1973.

Copeland B. J. Colossus: The Secrets of Bletchley Park's Codebraking Computers [Book]. - Oxford : Oxford University Press, 2006. - ISBN 0-19-284055-X.

Crossbow MICAz 2.4GHz - Wireless Module [Online]; Crossbow Technology. - 5 April 2010. - 5 April 2010. - http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

Dey, A.; Mankoff, J.; Abowd, G.; Carter, S. Distributed Mediation of Ambiguous Context in Aware Environments [Report]. - [s.l.] : ACM Press, 2002. - pp. 121-130.

Dey Anind K. Providing Architectural Support for Building Context-Aware Applications. - [s.l.] : Georgia Institute of Technology, 2000.

Dosch W., Meriste M. and Motus L. Enriching Interactive Components with Again Commands [Conference]. - Los Alamitos : IEEE Computer Soc, 2007. - pp. 200 - 205.

Dretske Fred Seeing and Knowing [Book]. - Chicago : The University of Chicago Press, 1969. - ISBN 0-7100-6213-3.

Drews F.A. and Westenskow D.R. The right picture is worth a thousand numbers: data displays in anesthesia [Journal]; The Journal of the Human Factors and Ergonomics Society. - Santa Monica : Human Factors and Ergonomics Society, 2006. - 1 : Vol. 48. - pp. 59–71.

Endsley M. R. Design and evaluation for situation awareness enhacement [Conference]; Proceedings of the Human Factors Society 32nd Annual Meeting Vol 1. - Santa Monica, CA : Human Factors Society, 1988. - pp. 97 - 101.

Endsley M., R. Towards a theory of situation awareness in dynamic systems [Journal]; Human Factors. - [s.l.] : Human Factors and Ergonomics Society, 1995. - Vol. 37 (1). - pp. 32-64.

Endsley Mica R and Garland Daniel J Situation awareness analysis and measurement [Book]. - London : Lawrence Erlbum Associates, Publishers, 2000.

Flach J. M. Situation Awaraness: Proceed with caution [Journal]; Human Factors, 37(1). - 1995. - pp. 149 - 157.

Friedlander D. S. and Phoha S. Semantic information fusion for coordinated signal processing in mobile sensor networks [Journal]; International Journal of High Performance Computing Applications. - 2002. - 3 : Vol. 16. - pp. 235–241.

Friedlander D. S. Semantic information extraction. In Distributed Sensor Networks [Book Section] / ed. Brooks S. S. Iyengar and R. R.. - [s.l.] : CRC Press, 2005.

Friedman D. P. and Wise D. S. The Impact of Applicative Programming on Multiprocessing; Proc.1976 International Conference on Parallel Processing. - 1976. - pp. 263 - 272.

Gaba D.M., Howard S.K. and Small S.D. Situation awareness in anesthesiology [Journal]; The Journal of the Human Factors and Ergonomics Society. - Santa Monica : Human Factors and Ergonomics Society, 1995. - 1 : Vol. 37. - pp. 20 - 31.

Gellersen Hans. W., Schmidt A. and Beigl M. Multi-sensor context-awareness in mobile devices and smart artifacts [Journal]; Mobile Networks and Applications. - [s.l.] : Kluwer Academic Publishers, 2002. - 5 : Vol. 7. - pp. 341 - 351.

Goldin D. and Wegner M. Persistent Turing Machines [Report]. - [s.l.] : Brown University Technical Report, 1998.

Goldin D. and Wegner P. Behavior and Expressiveness of Persistent Turing Machines [Report]. - [s.l.] : CS Technical Report, Brown University, 1999.

Goldin D. Persistent Turing Machines as a Model of Interactive Computation [Journal]; Lecture Notes In Computer Science; Vol. 1762 . - 2000. - pp. 116 - 135 .

Goldin D., Keil D. and Wegner P. An interactive viewpoint on the role of UML [Book Section]; Unified Modeling Language: Systems Analysis, Design, and Development Issues / ed. Siau K. and Halpin K.. - Hershey : Idea Group Publishing, 2001.

Goldin Dina, Smolka, Scott A., Wegner, Peter Interactive Computation The New Paradigm [Book]. - Berlin Heideberg New York : Springer-Verlag , 2006.

Güdemann M, Ortmeier F and Reif W. Formal Modeling and Verification of Systems with Self-x Properties [Book Section]; Autonomic and Trusted Computing. - Berlin : Springer Berlin / Heidelberg, 2006. - Vol. 4158/2006. - ISSN 1611-3349.

Halbwachs N. Synchronous programming of reactive systems [Book]. - [s.l.] : Kluwer Academic Publishers, 1993. - p. 174.

Halder, S. J.; Choi, T. Y.; Park, J. H.; Kang, S. H.; Park, S.W.; Park, J. G. Enhanced ranging using adaptive filter of ZIGBEE RSSI and LQI measurement [Conference]; Proceeedings of the 10th International Conference of Information Integration and Web-based Applications & Services. - Linz : ACM, 2008.

Hampton J., Ross H. Concepts and meaning: introduction to the special issue on conceptual representation; Language and Cognitive Processes. - [s.l.] : Psychology Press, December 2003. - Vol. 18 (5/6). - pp. 505 - 512.

Hampton James Ross Helen Concepts and meaning: introduction to the special issue on conceptual representation; Language and Cognitive Processes. - [s.l.] : Psychology Press, December 2003. - Vol. 18 (5/6). - pp. 505 - 512.

Harel D. and Pnueli A. On the Development of Reactive Systems [Journal]; Logics and Models of Concurrent Systems / ed. Apt K. R.. - New York : Springer-Verlag, 1985. - Vols. F-13. - pp. 477-498.

Harel D. Statecharts: A Visual Approach to Complex Systems [Report] / Dept. of Applied Mathematics ; The Weizmann Institute of Science. - 1984.

Harel D. Statecharts: A Visual Formalism for Complex Systems [Journal]; Science of Computer Programming. - June 1987. - pp. 231-274.

Heinzelman, W.B; Murphy, A.,L.; Carvalho, H., S.; Perillo, M.,A. Middleware to support sensor network applications [Journal]; IEEE Network. - [s.l.] : IEEE, 2004. - 1 : Vol. 18. - pp. 6 - 14.

Heisenberg Werner Physics and Philosophy: The Revolution in Modern Science; Lectures delivered at University of St. Andrews, Scotland, Winter 1955-1956. - 1958.

Helander J. and Sigurdsson S. Self-Tuning Planned Actions: Time to Make Real-Time SOAP Real [Conference]; Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05) . - 2005.

Hintikka Jaakko Information, Causality and the Logic of Perception [Book Section]; Intentions of Intentionality. - Boston : D. Reidel Pub. Co, 1975.

Hoare C.A.R. An axiomatic basis for computer programming [Article]; Communications of the ACM. - October 1969. - pp. 576 - 585.

Hopcroft J. and Ullman J. Introduction to Automata Theory, Languages, and Computation [Book]. - [s.l.] : Addison-Wesley, 1979.

Huebscher M. C and McCann J. A. A survey of autonomic computing—degrees, models, and applications [Journal]; ACM Computing Surveys (CSUR). - [s.l.] : ACM, August 2008. - 3 : Vol. 40.

IEEE Computer Society IEEE Std 802.15.4. - New York : [s.n.], 2006.

Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Heideman, J.; Silva, F. Directed diffusion for wireless sensor networking [Journal]; Transactions on Networking. - [s.l.] : IEEE Press, February 2003. - 1 : Vol. 11. - pp. 2 - 16.

Intille, S.; Bao, L.; Tapia, E.; Rondoni, J. Acquiring In Situ Training Data for Context-Aware Ubiquitous Computing Applications; Proceedings of the SIGCHI conference on Human factors in computing systems. - Vienna : ACM, 2004. - pp. 1 - 8. - ISBN 1-58113-702-8.

Jonas K.W. Network-centric operations: European capabilities [Report] / Report of the Defence Committee ; Inter-parliamentary European Security and Defence Assembly. - Brussels : [s.n.], 2005.

Jones D. g. and Endsley M. R. Sorces of situation awareness errors n aviation [Journal]; Aviation, Space and Environmental Medicine, 67 (6). - 1996. - pp. 507 - 512.

Lee E. A. Computing Foundations and Practice for Cyber-Physical Systems: A Preliminary Report [Report]. - [s.l.] : University of California, Berkeley, 2007. - UCB/EECS-2007-72.

Lee H. Accuracy Limitations of Hyperbolic Multilateration Systems [Journal]; IEEE Transactions on Aerospace and Electronic Systems. - [s.l.] : IEEE, 1975. - 1 : Vols. AES-11. - pp. 16-29.

Lewis L., Buford J. and Jakobson G. Inferring threats in urban environments with uncertain and approximate data: an agent-based approach [Journal]; Applied Intelligence. - [s.l.] : Kluwer Academic Publishers, June 2009. - 3 : Vol. 30. - pp. 220 - 232.

Mamaysky H. Lo A. W. and Wang J. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. [Journal]; Journal of Finance. - 2000. - 4 : Vol. 40.

Marron P.J. Lachenmann, A. Minder, D. Hahner, J. Sauter, R. Rothermel, K. TinyCubus: a flexible and adaptive framework sensor networks [Conference]; Proceeedings of the Second European Workshop on Wireless Sensor Networks. - 2005. - pp. 278- 289.

Matheus, C.; Kokar, M.; Baclawski, K.; Letkowski, J. An Application of Semantic Web Technologies to Situation Awarenes [Conference]; Proceedings of the 4th International Semantic Web Conference, ISWC 2005. - Galway : [s.n.], 2005.

Matheus C., Baclawski K. and Kokar M. Derivation of ontological relations using formal methods in a situation awareness scenario [Conference]; Proc of SPIE Conference on Multisensor, Multisource Information Fusion,. - 2003. - pp. 298-309.

McKee L. The Spatial Web [Online]; www.opengis.org. - Open Geospatial Consortium, 2003.

Meriste, M.; Motus, L.; Kelder, T.; Helekivi, J.; Marandi, A.; Preden, J. Location awareness of information agents [Conference]; Advances in Databases and Information Systems, LNCS. - [s.l.] : Springer Verlag, 2005. - Vol. 3631. - pp. 199-208.

Milner R. The Polyadic π-calculus [Report] / Lab. for Foundations of Computer Science, Dept. of Computer Science ; University of Edinburgh. - Edinburgh : Technical Report ECS-LFCS-91-180, 1991.

Mirriam Webster online dictionary. - 25 05 2009. - http://www.merriam-webster.com/dictionary.

Motus L. Modeling metric time [Book Section]; UML for Real: Design of Embedded Real-time Systems / book auth. Selic B., Lavagno L. and Martin G. (Eds),. - Norwell, : Kluwer Academic Publ., 2003.

Motus L., Meriste M. and Dosch W. Time-awareness and Proactivity in Models of Interactive Computation [Journal]; Electronic Notes in Theoretical Computer Science. - [s.l.] : Elsevier, 2005. - Vol. 141. - pp. 69 - 95.

Motus L., Meriste M. and Dosch W. Time-awareness and Proactivity in Models of Interactive Computation [Journal]; Electronic Notes in Theoretical Computer Science. - [s.l.] : Springer, 2005. - Vol. 141. - pp. 69-95.

Motus L., Meriste M. and Preden J. Towards Middleware Based Situation Awareness [Conference]; Proceedings of the Situation Management Workshop at MILCOM 2009. - Boston : IEEE Operations Center, 2009. - pp. 1 - 7. - ISBN 978-1-4244-5238-5.

Motus Leo and Rodd Michael G Timing Analysis of Real-Time Software [Book]. - Oxford : Elsevier, 1994. - ISBN 0080420257.

Motus Leo Timing problems and their handling at system integration [Book Section]; Artificial Intelligence in Industrial Decision Making, Control and Automation / ed. Tzafestas S.G. and Verbruggen H. B.. - [s.l.] : Kluwer Academic Publishers, 1995.

Niculescu D. and Nath B. Position and orientation in ad hoc networks [Conference]; Ad Hoc Networks. - 2004. - Vol. 2. - pp. 133-151.

Niculescu D. and Nath B. Position and orientation in ad hoc networks [Journal]; Ad Hoc Networks. - April 2004. - 5 : Vol. 2. - pp. 133-151.

Nissanka B., Chakraborty A. and Balakrishnan H. The Cricket location-support system [Conference]; Proceedings of the 6th annual international conference on Mobile computing and networking. - 2000. - pp. 32-43.

Nuclear Regulatory Commission U.S. Three Mile Island Accident [Report] : http://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html / Public Document Room ; United States Nuclear Regulatory Commission. - Washington, D.C. : Public Document Room, United States Nuclear Regulatory Commission, 2009.

P. Salmon N.A. Stanton, D.P. Jenkins, G.H. Walker, M.M.S. Young, A. Aujla What is really going on? Review, Critique and Extension of Situation Awareness Theory [Journal]; Engineering Psychology and Cognitive Ergonomics / ed. Harris D.. - 2007. - pp. 407-416.

Pahtma, R.; Preden, J.; Agar, R.; Pikk, P. Utilization of Received Signal Strength Indication by Embedded Nodes [Journal]; Electronics and Electrical Engineering. - Vilnius : Kaunas University of Technology, 2009. - 93 : Vol. 5. - pp. 39 - 43.

Parkinson B. and Spilker J. Global Positioning System: Theory and Application [Report]. - [s.l.] : American Institute of Astronautics and Aeronautics, 1996.

Penjam J. and Meriste M. Attributed Finite Automata [Report]. - Tallinn : Res.Rep. CS23/91, Inst of Cybernetics, Estonian Academy of Sciences, Estonia and Dep. of Comp. Sci., University of Turku, Finland, 1991.

Perry M. Distributed cognition [Book Section]; HCI models, theories and frameworks / ed. JM Carroll. - San Francisco : Morgan-Kaufman, 2003.

Preden, J; Serg, R; Riid, A; Mõtus, L; Pahtma, R. Vehicle Guidance Systems in NEC context [Conference]; Proceedings of NATO RTO/IST Symposium on "Intelligent Uninhabited Vehicle Guidance Systems". - Munchen : North Atlantic Treaty Organization, 2009.

Preden J and Helander J, Situation aware computing in distributed computing systems [Conference]; Proceedings of the 10th Symposium on Programming Languages and Software Tools. - Budabest : Eötvös Lorand University, Faculty of Informatics: Eotvos University Press, 2007. - pp. 280 - 292.

Preden J. and Helander J. Auto-adaptation Driven by Observed Context Histories [Conference]; UbiComp workshop ECHISE 2006: Exploiting Context Histories in Smart Environments. - 2006.

Preden J. and Helander J. H. Auto-adaptation Driven by Observed Context Histories [Conference]; Proceedings of ECHISE 2006: Exploiting Context Histories in Smart Environments workshop at UbiComp 2006. - Irvine, Ca : [s.n.], 2006.

Preden J. and Pahtma R. Exchanging situational information in embedded networks [Conference]; Proceedings of the IEEE 2nd International Conference on Adaptive Science & Technology. - Accra : IEEE, 2009. - pp. 265-272.

Preden J. Communication Area Based Positioning [Conference]; Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Systems. - Vancouver : IEEE, 2006. - pp. 336 - 347.

Preden J., Sarkans M. and Otto T. Diagnistocs of Machining and Assembly Systems by Networked Motes [Journal]; Machine Engineering. - 2007. - 1-2 : Vol. 7. - pp. 68 - 77.

Riid A., Pahhomov D. and Rüstern E. Car Navigation and Collision Avoidance System with Fuzzy Logic [Conference]; Proc. IEEE International Conference on Fuzzy Systems. - Budapest : IEEE Operations Centre, 2004. - Vol. 3. - pp. 1443-1448.

Rodgers M. D., Mogford R.H. and B. Strauch Post hoc assessment of situation awareness in air traffic control incidents and major aircraft accidents [Book Section]; Situation awareness analysis and measurement / ed. Endsley M.R. and Garland D.J.. - London : Lawrence Erlbaum, 2000.

S. Yau Y. Wang and F. Karim Developing Situation-Awareness in Middleware for Ubicomp Environments [Conference]; Proc. 26th Int'l Computer Software and Applications Conference. - 2002. - pp. 233 - 238.

Salmon, P.; Stanton, N.A.; Jenkins, D.P.; Walker, G.H.; Young, M.M.S.; Aujla, A. What is really going on? Review, Critique and Extension of Situation Awareness Theory [Journal]; Engineering Psychology and Cognitive Ergonomics, / ed. Harris D.. - 2007. - Vol. LNAI 4562. - pp. 407-416.

Savvides A., Park H. and Srivastava M. The Bits and Flops of the N-hop Multilateration Primitive for Node Localization Problems [Conference]; proceedings of the First International Workshop for Wireless Sensor Networks and Applications (WSNA). - 2002.

Schilit B., Adams, N., Want, R. Context-Aware Computing Applications [Conference]; Proceedings of the Workshop on Mobile Computing Systems and Applications. - [s.l.] : IEEE Computer Society, 1994. - pp. 85-90.

Schmidt A. Ubiquitous computing - computing in context; A thesis submitted to Lancaster University for the degree of Ph.D. in Computer Science,. - Lancaster : [s.n.], 2002.

Schyns P.G., Goldstone R.L. and Thibaut J.P. The development of features in object concepts; Behavioral and Brain Sciences. - [s.l.] : Cambridge University Press, 1998. - Vol. 21. - pp. 1 - 54.

Shang Y. and Ruml W. Improved MDS-based localization [Conference]; Proceedings of IEEE Infocom '04. - Hong Kong : IEEE, 2004. - pp. 2640-2651.

Sheng B., Li, Q. and Mao W. Data storage placement in sensor networks [Conference]; Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'06).. - Florence, Italy : ACM, 2006. - pp. 344–355.

Shin, S; Son, B.; Kim, W.; Kim2, J. ERFS: Enhanced RSSI value Filtering Schema for Localization in Wireless Sensor Networks [Book Section]; Wireless Sensor and Actor Networks II. - [s.l.] : Springer Boston, 2008.

Stanton, NA.; Stewart, R.; Harris, D.; Houghton, RJ.; Baber, C.; McMaster, R.; Salmon, P.; Hoyle, G.; Walker, G.; Young, MS.; Linsell, M.; Dymott, R.; Green, D. Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology [Journal]; Ergonomics . - [s.l.] : Taylor & Francis, London, 2006. - Vol. 49. - pp. 1288–1311. - ISSN 0014-0139.

Stepney, S.; Clark, J.A.; Tyrrell, V; Johnson, V; Timmis, J.; Partridge, D.; Adamatzky, A.; Smith, R.E. Journeys in Non-Classical Computation. A Grand Challenge for Computing Research [Online]. - March 2004. - 22 07 2008. - http://www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/stepney.pdf.

Stepney, S.; Braunstein, S.; Clark, J.A.; Tyrrell, A.; Adamatzky, A.; Smith, R. E.; Addis, T.; Johnson, C.; Timmis, J.; Welch, P.; Milner, R.; Partidge, D. Journeys in non-classical computation I: A grand challenge for computing research [Book]. - [s.l.] : The International Journal of Parallel, Emergent and Distributed Systems, 2005. - Vol. 20 : pp. 5 - 19. - ISSN 1744-5760.

Sterling L., S. and Kuldar T. The Art of Agent-Oriennted Modelling [Book]. - Cambridge : MIT Press, 2009.

Stewart, R.; Stanton, N., A.; Harris, D.; Baber, C.; Salmon, P.; Mock, M.; Tatlock, K.; Wells, L.; Kay, K. Distributed situation awareness in an Airborne Warning and Control System: application of novel ergonomics methodology [Journal]; Cognition, Technology and Work . - [s.l.] : Springer-Verlag, 2008. - 3 : Vol. 10. - pp. 221 - 229. - ISSN:1435-5558 .

Stout T. M. and Williams T. J. Pioneering Work in the Field of Computer Process Control [Journal]; IEEE Annals of the History of Computing. - [s.l.] : IEEE Computer Society. - 1 : Vol. 17. - pp. 6 - 18.

Stümpel A. Stream Based Design of Distributed Systems through Refinement [Report] : Doctoral thesis. - Berlin : Logos Verlag Berlin, 2003. - p. xii+218. - ISBN 3-8325-0462-1.

Sulistyawati, K.; Chui, Y.P.; Tham, Y.M.; Wee, Y.K. Evaluation of Process Tracing Technique to Assess Pilot; Engineering Psychology and Cognitive Ergonomics / ed. Harris D.. - Berlin : Springer-Verlag, 2007. - pp. 824 - 833.

Tammet, T.; Vain, J.; Puusepp, A.; Reilent, E.; Kuusik, A. RFID-based Communications for a Self-Organising Robot Swarm [Conference]; Proceedings of the Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '08). - 2008. - pp. 45 – 54.

Tennenhouse D. Proactive computing [Journal]; Communications of the ACM. - [s.l.] : Association for Computing Machinery, May 2000. - 5 : Vol. 43. - pp. 43 - 50. - ISSN:0001-0782 .

Texas Instruments 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. - 2007.

Toom A. Naks T., Pantel M., Gandriau M., Wati I. GeneAuto: An Automatic Code Generator for a safe subset of SimuLink/StateFlow [Conference]; European

Congress on Embedded Real-Time Software (ERTS 2008). - Toulouse : [s.n.], 2008.

Turing A On computable numbers, with an application to the Entscheidungsproblem [Journal]; Proceedings of the London Mathematical Society. - [s.l.] : Proceedings of the London Mathematical Society, 1936. - 2 : Vol. 42. - pp. 230 - 265.

Turley J. Embedded processors by the numbers [Journal]; Embedded Systems Programming. - May 1999. - 5 : Vol. 12.

von Neumann J. First Draft of a Report on EDVAC [Journal]; IEEE Annals of the History of Computing. - [s.l.] : IEEE Computer Society, 1993. - 4 : Vol. 15. - pp. 27 - 75. - ISN 1058-6180.

Walker, G. H.; H., Gibson; Stanton, N. A.; Baber, C; Salmon, P.; Green, D. Event analysis of systemic teamwork (EAST): a novel integration of ergonomics methods to analyse C4i activity [Journal]; Ergonomics. - October 2006. - 12 & 13 : Vol. 49. - pp. 1345 - 1369.

Wang Yu An FSM model for situation-aware mobile application software systems [Conference]; Proc. 42nd ACM Southeast Regional Conference. - Huntsville, Alabama : [s.n.], 2004. - pp. 52 - 57.

Ward A., Jones A. and Hopper A. A New Location Technique for the Active Office [Journal]; IEEE Personal Communications. - [s.l.] : IEEE, October 1997. - 5 : Vol. 4. - pp. 42 - 47.

Wegner P Why Interaction is More Powerful than Algorithms [Journal]; Communications of the ACM. - 1997.

Wegner P. and Eberbach E. New models of Computation [Journal]; The Computer Journal. - 2004. - 1 : Vol. 47. - pp. 4 - 9.

Weigert, T.; Weil, F.; van den Berg, A.; Dietz, P., Marth, K. Automated Code Generation for Industrial-Strength Systems [Conference]; Proc. of 32nd Annual IEEE International Conference on Computer Software and Applications COMPSAC '08. - Turku : IEEE Operations Centre, 2008. - pp. 464–472.

Weiser Mark The computer of the 21st century; Scientific American. - [s.l.] : Scientific American Inc, Sept. 1991. - Vol. 265. - pp. 66 - 75.

Whitehouse K. and Culler D. Calibration as a Parameter Estimation Problem in Sensor Network [Conference]; ACM Workshop on Sensor Networks and Applications. - Atlanta : ACM, 2002.

Whitehouse K., Liu, J., Zhao, F. Semantic streams: A framework for composable inference over sensor data [Journal]; Lecture Notes in Computer Science / ed. Römer K., Karl H. and Mattern F.. - Zurich, Switzerland : Springer Verlag, 2006. - Vol. 3868. - pp. 5–20.

Yang S. and Cha H. An Empirical Study of Antenna Characteristics Toward RF-Based Localization for IEEE 802.15.4 Sensor Nodes [Conference]; Proceedings of the 4th European Conference on Wireless Sensor Networks. - Delf : [s.n.], 2007.

Yau, S; Wang, Y; Huang, D.; P., Hoh Situation-Aware Contract Specification Language for Middleware for Ubiquitous Computing [Conference]; The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems. - 2003.

Yau S., Wang Y. and Karim F. Developing Situation-Awareness in Middleware for Ubicomp Environments [Conference]; Proc. 26th Int'l Computer Software and Applications Conference. - Oxford, UK : [s.n.], 2002.

Zhou G., He, T., Krishnamurthy, S., and Stankovic, J. A. Impact of Radio Irregularity on Wireless Sensor Networks [Conference]; Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys '04) . - 2004.

List of publications

Meriste, M.; Motus, L.; Kelder, T.; Helekivi, J.; Marandi, A.; Preden, J., (2005) „Location-awareness of information agents“ Advances in Databases and Information Systems, Springer Verlag, LNCS, vol. 3631, 199-208

Preden, J, "Communication area based positioning", Proc. The 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems, ISBN 1-4244-0507-6, October 2006, pp 336 - 347

Preden, J., "Context Aware Ad-Hoc Network Nodes", Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK esimese aastakonverentsi artiklite kogumik: Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK esimene aastakonverents, 2006, (47 - 49). Eesti: OÜ Infotrükk

Preden,J., "Tark tolm", Akadeemia, (7), 2006, 1577 - 1588

Preden, J; Helander, J; "Auto-adaptation Driven by Observed Context Histories", UbiComp workshop ECHISE 2006: Exploiting Context Histories in Smart Environments

Motus, L.; Meriste, M.; Preden, J., "Network Enabled Capabilities – Grassroots Perspectives", NATO RTO/IST Symposium on "Dynamic Communication Management" (16-1 - 16-13), 2006, North Atlantic Treaty Organization

Helander, J.; Preden, J., "Adapting the Auto to a New Tune", 1st Workshop on Models and Analysis for Automotive Systems / RTSS 2006 : The 27th IEEE Real-Time Systems Symposium: 1st Workshop on Models and Analysis for Automotive Systems : Held in conjunction with RTSS 2006, 5-8 December 2006, Rio de Janeiro, Brazil. , 2006, 21 - 24., December 2006, pp 21 - 24

Preden, J.; Otto, T., "Tolmuga tööstust jälgimas", Eesti Ekspress lisa Homme, 07.02.2007

Preden, J.; Sarkans, M.; Otto, T. Diagnostics of Machining and Assembly Systems by Networked Motes. Machine Engineering, 2007, 7(1-2), 68 - 77

Preden, J, "Computing in context", Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK teise aastakonverentsi artiklite kogumik: Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK teine aastakonverents, 2007, Eesti: OÜ Infotrükk

Preden, J; Helander, J, "Situation aware computing in distributed computing systems", Proceedings of the 10th Symposium on Programming Languages and Software Tools 2007, Budapest, 14th-16th of June, 2007. (Toim.) Horvath, Z.; Kozma, L.; Zsok, V.. Eötvös Lorand University, Faculty of Informatics: Eotvos University Press, 2007, 280 - 292

Preden, J; Sarkans, M; Otto, T, "Smart dust based modular laboratory kit for monitoring workshop machinery", 8th International Workshop on Research and Education in Mechatronics, 2007

Preden, J. (2008). Situation awareness of computing agents. Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK kolmanda aastakonverentsi artiklite kogumik : 25.-26. aprill 2008, Voore külalistemaja (11 - 13). Tallinn: Tallinna Tehnikaülikooli Kirjastus

Mõtus, L.; Meriste, M.; Preden, J.-S. (2008). NNEC Technologies Focused on use by Semi-Autonomous Groups. In: Proceedings, SCI-187 Symposium on Agility, Resilience and Control in NEC : SCI-187 Symposium on Agility, Resilience and Control in NEC; Amsterdam; 27.-29. May 2008. North Atlantic Treaty Organization, 2008, 1 - 16

Preden, J. (2008). Smart dust. Dudziak, R.; Köhn, C.; Sell, R. (Toim.). Integrated Systems and Design (47 - 53).Kaunas University of Technology Press

Pahtma, R.; Preden, J.; Agar, R.; Pikk, P. (2009). Utilization of Received Signal Strength Indication by Embedded Nodes. Electronics and Electrical Engineering, 5, 2009, 39 - 43

Preden, J.; Pahtma (2009). Smart Dust Motes in Ubiquitous Computing Scenarios. Electronics and Electrical Engineering, 6, 2009, 19 - 23

Riid, A; Preden, J; Pahtma, R.;Serg, R.;Lints, T. Automatic Code Generation for Embedded Systems from High-Level Models, Electronics and Electrical Engineering, 7, 2009, pp 33 - 36

Preden, J; Serg, R; Riid, A; Mõtus, L; Pahtma, R (2009), Vehicle Guidance Systems in NEC context, NATO RTO/IST Symposium on "Intelligent Uninhabited Vehicle Guidance Systems", 2009, North Atlantic Treaty Organization

Motus, L; Meriste, M; Preden, J (2009), Towards Middleware Based Situation Awareness, Military Communications Conference, MILCOM 2009, IEEE Operations Center, ISBN 978-1-4244-5238-5, Boston, MA, USA, Oct 2009, pp 1 - 7

Preden, J.; Pahtma (2009). Exchanging situational information in embedded networks, IEEE 2nd International Conference on Adaptive Science & Technology, pp 265-272

Preden, J.; Helander, J. (2009) Context Awareness in Distributed Computing Systems, Annales Universitatis Scientarium Budapestinensis de Rolando Eötvös Nominatae., Sectio Comp. 31 (2009) 57-73

7 Appendix A: Formalisms for describing computation

This appendix gives a superficial overview of some existing formalisms that have been, or can be used for describing computing systems. In no way is this appendix a complete review of existing formalisms, the aim of the appendix is more to review formalisms suitable to describing parts of interactive computing systems. Descriptions of formalisms include the properties that are of interest in the context of interactive computation.

It must be noted here that the described formalisms are not directly comparable, their expressive power is not equivalent, and many of them are not meant for describing the same aspects of a computing system.

7.1.1 Turing Machine

A Turing Machine (TM) is an abstract device that transforms an input string into an output string via a sequence of operations applied in a prefixed order to an input string and interim results of computation until the output string is obtained as the result of the last operation, for short it can be said that a computation is expressed as a sequence of state transitions, starting from an initial state and terminating in an end state. A TM is a state transition machine $M = (S, \sum, \delta)$, with a finite set of states *S*, tape symbols $\sum$ and a state transition relation $\delta : S \times \sum \rightarrow S \times (\sum, L, R)$, where *L* is a shift to the left and *R* is a shift to the right. TMs transform finite input strings $x \in \sum^*$ to output string $y = M(x)$ by a finite sequence of steps, starting in a unique initial state and ending when a halting state has been reached. At each step *M* reads a tape symbol *i*, performs a state transition $(s, i) \rightarrow (s', o)$, writes a symbol *o* and/or moves the reading head one position left or right (Hopcroft, et al., 1979).

The class of functions computable by a TM are called Turing computable functions or computable functions since the concept of computability was in the middle of the 20th century mistakenly also coupled with Turing computability, i.e. anything that is computable can be expressed using a Turing machine. TM computes functions $y = f(x)$ from integers to integers (from strings to string). The computations of a TM are history independent, i.e. a given input always yields the same output. The reason for reproducibility is the fact that TMs always start at an initial state and obtains all their input prior to the start of the computation.

The fact that early computation which occurred in the computers could be fully described using the Turing computable functions lead many computer scientists to the incorrect conclusion that any computation that can be carried out in computers can be described using Turing computable functions. Turing machines are suitable for describing the computation that occurs in transformational systems and also parts of computation that occurs in reactive systems.

The Turing machine concept presumes that the input is presented to the machine, which then processes the data and generates an output without any interference during the computation, which is the nature of the computation in a transformational system. This presumption of non- interference also holds for parts of computation in reactive systems. However TMs has very limited applicability in the context of proactive systems. TMs can be used to only describe computation of rather primitive functions required in proactive systems while the critical parts of computation occurring in a proactive system cannot be described using TMs. It could be envisioned that when applying Turing machine paradigm in this context of proactive systems, we would face a large number of primitive functions interacting via non-stationary mediated interactions. The result might be operational, but its behaviour cannot be verified, not even approximately predicted; especially if the application is situation-aware (e.g. time-dependable, location dependable, etc

7.1.2 Persistent Turing Machine (PTM)

Persistent Turing Machine is a minimal extension of Turing Machine that expresses sequential interactive behaviour (Goldin, 2000). Goldin and Wegner introduce in (Goldin, et al., 1998) Persistent Turing Machines (PTMs) concept that extends the Turing Machine (TM) with a persistent work tape whose content is preserved between successive TM computations, known as PTM computation macro-steps. The persistent work tape is the memory of the PTM and the contents of the memory before and after a single computation step express the state of the PTM. The set of PTM states is infinite as it is represented by strings of unbounded length (Goldin, 2000). The PTM is in essence a multi-tape TM, which behaviour is characterized by input-output streams instead of being characterized by input-output strings as is the case with TMs. So PTMs are interactive systems where the output does not only depend of the current state and input but also on past interactions cannot be described with TMs.

The work-tape contents of PTM are is not directly observable, unlike inputs or outputs. Although the work-tape affects the output, it does not participate directly in the notion of PTM behaviour, as the behaviour of a PTM is observation based. Any attempt to model PTM computations with TMs, by making the work-tape contents an explicit part of the input, would thus fail. PTMs are sequential interaction machines (SIMs) inheriting from TMs the restriction that input tokens must be discrete and that any model of behaviour must ignore time-dependent aspects. They are therefore inappropriate for modelling real-time and embedded devices or physical control processes, whose expressiveness requires the full generality of SIMs. There are multi-stream interactive behaviours, such as distributed database systems or airline reservation systems, which cannot be modelled by SIMs at all, requiring a model with even more expressiveness – multi-stream interaction machines MIMs (Goldin, 2000).

Goldin and Wegner suggest in (Goldin, et al., 1999) that PTMs allow describing interaction machines since the work tape is not necessarily blank at the start of a computational step. The contents of the work tape have been generated during previous computation steps, the input to which is unknown at the current computation step (we assume that the input is generated by the outside world, which we cannot control).

While the TM models algorithmic behaviour where output can be directly derived from the current input, the PTM models interactive behaviour where the output is a function of the current input and also of the past inputs. A PTM models *sequential interactive behaviour* (Goldin, 2000), where the computing device or agent evolves as it processes the inputs. The evolution is expressed in the change of the PTM work-tape contents, so the PTM output tokens are a function of both the input and the contents of the work-tape.

While the PTM allows describing the change of the machine behaviour it does not allow to describe the aspects of the interactions that may be also relevant in terms of the behaviour of the machine – e.g. the temporal aspects of the interactions may influence the behaviour of the machine. Also the properties of data received via interactions (which cannot be described using PTMs) may affect the behaviour of the machine (i.e. the output).

So PTMs can be used to describe computation in transformational systems and sections of computations in reactive and proactive systems.

7.1.3 The π-calculus

The π-calculus was introduced by Robin Milner (Milner, 1991) as a model of computation for concurrent systems. It was conceived for describing and analyzing systems that consist of computing agents interacting with each other in a dynamic setting. The π calculus allows representation of processes, parallel composition of processes and communication of processes through channels. The π calculus uses the following denotation:

P and Q denote a process,

$P|Q$ denotes a process composed of processes P and Q running in parallel

$a(x).P$ denotes a process waiting to read a value x from channel a and after receiving the value, continues as P

$\bar{a}\langle x\rangle.P$ denotes a process waiting to send value x to channel a and after x has been consumed by some input process, continues as P

$(va)P$ ensures that a is a fresh channel in P.

$!P$ denotes an infinite number of copies of P running in parallel

$P+Q$ denotes a process that behaves like P or Q

0 denotes an inert process that does nothing

The π-calculus can be used to describe a concurrent system in a top-down manner – it allows describing the structure of the system in terms of processes and their interactions. The π-calculus cannot be used to describe the function that a process implements so essentially the formalism can be used for describing the interactions between processes. However more advanced properties of interactions, such as temporal and spatial selectiveness of processes cannot be described using π-calculus either. The formalism has applications in all types of computing systems on an abstract level, however it lacks the required detail to be applicable for analyzing the properties of interactions in proactive systems.

7.1.4 Hoare logic

Hoare logic, introduced in (Hoare, 1969) is a formal system for reasoning about the correctness of computer programs with the aid of mathematical logic. The central entities in Hoare logic are the Hoare triples $\{P\}C\{Q\}$ where $\{P\}$ and $\{Q\}$ are assertions and C is a command. The $\{P\}$ is the preconditions and $\{Q\}$ is the post-condition, when the precondition is met and the command is executed the post-condition should be established. Hoare logic does not exceed the power of first order predicate logic. Since Hoare logic deals with the correctness of transformations it can be applied to compositional verification of transformational systems. It can be applied for describing and analyzing of Turing computable functions and the mathematical superposition of these functions. Hence Hoare logic has limited applicability in the context or reactive and proactive systems (similar to Turing machine), namely in the transformational parts of these systems.

7.1.5 Statecharts

The Statecharts concept was originally introduced by David Harel in (Harel, 1984). The concept was put forward by Harel and Pnueli as formalism for describing the behaviour of complex reactive systems in a paper (Harel, et al., 1985). Statecharts extend conventional state-transition diagrams with notions of hierarchy, concurrency and communication (Harel, 1987). These properties make Statecharts more expressive when compared to Turing machine and help to keep the complexity of diagrams manageable. The reason for introducing Statecharts was that common state machines, which were believed to be the natural way for describing dynamic behaviour of complex systems, could not be used for describing reactive systems of realistic size. Harel claimed that the reason why complex systems cannot be described with common state machines is the unmanageable, exponentially growing multitude of states, all of which have to be arranged in a 'flat' not stratified fashion, resulting in an unstructured, unrealistic, and chaotic state diagram (Harel, 1987). Harel argues that to be useful a state/event approach must be modular, hierarchical and well-structured.

States in a Statechart can be repeatedly combined into higher-level states using either disjunctive or conjunctive modes of clustering. The clustering of states creates levels of states where the state that contains a cluster of states is a higher-level state. This in turn makes the states that are contained within higher-level state lower-level states. The hierarchy can have several levels and transitions in Statecharts are not level-restricted but can lead from a state on any level to a state on any other level. The graphical notation of statecharts has also been adapted by Mathworks in their tool called Stateflow, which allows describing the behaviour of complex systems using the statecharts semantics.

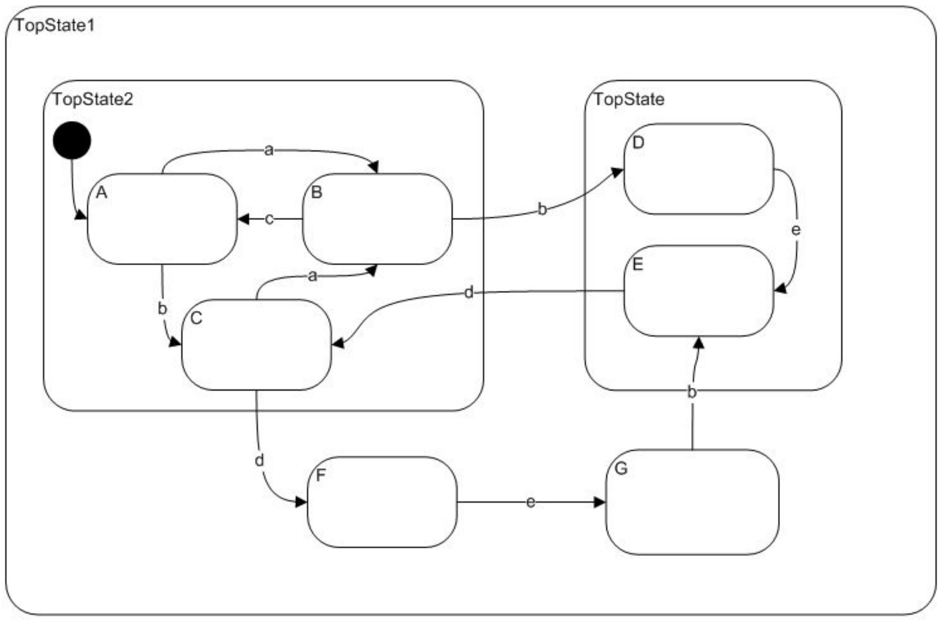


Figure 38 A sample statechart

Figure 38 depicts a sample Statechart with one top level state containing four sub-states. Two of the sub-states (*TopState2* and *TopState*) contain sub-states themselves. The states that are contained in *TopState1* are disjunctive, i.e. the system can either be in state *TopState2, TopState, F* or *G*.

The Statechart concept is close to interactive computation in its expressive power, so statecharts can be used for describing transformational, reactive and also proactive systems to some extent. However due to the fact that the formalism can not be used to express the temporal and spatial aspects of data it cannot be used to describe all aspects of a proactive computing system. Also statecharts only allow describing communication in a non-directed manner as it is only possible to model broadcast communication. The fact that only broadcast communication is possible limits the applicability of the formalism in proactive systems where selective and directed interaction is a critical system feature.

7.1.6 UML

The Unified Modelling Language (UML) allows to model interactive systems, whose behaviours emerge from the interaction of their components with each other and the environment (Goldin, et al., 2001). While UML allows the modelling of such systems it does not allow detailed and complete specification of such systems.

UML allows for different view of an interactive system to be modelled but the associations between the different views are not synchronized. The UML uses the notion of a computing entity which is a finitely specifiable software system, component or object. Computing entities may contain sub-entities, or be part of a larger computing entity. The viewable behaviour of the computing entity is determined by the outputs of the entity it produces and the inputs it consumes – the service that it provides to the environment, i.e. the actors in the environment (which may be users or other systems). Staring from the viewable behaviour it can be said that the internal behaviour of the system emerges out of the internal interactions among its components.

UML allows describing the viewable behaviour of the system and also specifying the internal structure and behaviour of the system (Goldin, et al., 2001). UML offers three views to the user: the static view, the dynamic view and the functional view. The static view describes the relations between the interactive components; it allows static description of objects, operations performed by the objects and the relations among objects. The diagrams that express the static view are *class diagrams, object diagrams and component diagrams*. The dynamic view allows describing the interactions within the system (between system components) and with the system (between the system and the environment) – the inter-object dynamics present in an interactive computing system. The diagrams that allow expressing the dynamic view are *use case, sequence, collaboration and state transition diagrams*. The functional view deals with the description of the behaviour of specific functions or methods, i.e. modelling transformational behaviour in an object. The functional view can be expressed using *activity diagrams, state diagrams and narratives*.

So UML allows describing the internal behaviour of a system and also the external behaviour by describing the interactions with entities external to the system via for example *use cases*. A *use case* is an abstraction that can represent the *input sources* of a system. A use case is described as an interaction with an *actor* who is constrained to a specific role, limiting the messages that the *actor* can generate or consume. However since an actor is an outside entity in the use case view its internal structure is not specified but only the external view is used. The actors that are associated with a system behave as a non-deterministic source of inputs as only the types of messages are specified but not the temporal relation between the messages, nor the (possible) relations between the actors.

UML supports modelling the system behaviour emerging from responding to (and interacting with) a stream of events with the next event of the stream not available before the previous has been processed. Since the event stream is not limited but open ended instead the expressive power of UML is clearly beyond that of Turing machines. As explained above modelling interactions with multiple actors is possible which means that concurrent interaction streams (each representing interaction with a separate autonomous actor) can be modelled using UML.

One of the main limiting factors of UML lies in the fact that linking the several descriptions of the system in a non ambiguous way is non-trivial, so one is able to describe the various aspects of an interactive computing system but it is not possible to create a complete and coherent description of such a system.

7.1.7 Petri nets

A Petri net is a formal, graphical, executable technique for the specification and analysis of concurrent, discrete event systems used for describing discrete distributed systems by focusing on visual representation. The visual representation of a Petri net is a bipartite graph.

A Petri net is a 5-tuple (S, T, F, M_0, W), where *S* is a finite set of places, *T* is a finite set of transitions, *F* is a finite set of directed arcs known as *flow relation*. *S* and *T* are disjoint and *F* is subject to a constraint, which says that no arc may connect two places or transitions - $F \subseteq (S \times T) \cup (T \times S)$. $M_0 : S \to N$ is an initial marking, where for each place $s \in S$ there are $n_s \in N$ tokens. Places are conditions which on the graphs are denoted by circles, transitions are discrete events which on the graphs, directed arcs express the flow which on the graphs are denoted by arrows. No arc may connect two places or transitions. The directed arcs, which express the flow relation in a Petri net, describe which places are pre- and post-conditions for which transitions. The places in the Petri net may contain a natural number of tokens, which control the firing of transitions in a net. A transition in a net may be executed whenever there is a token at the end of all input arcs to the transition. Whenever a transition is executed, all the tokens on the input arcs are consumed and tokens are placed on all the output arcs of the transition. When multiple transitions are enabled at the same time, the execution of a Petri net is nondeterministic, i.e. any of the enabled transitions may fire.

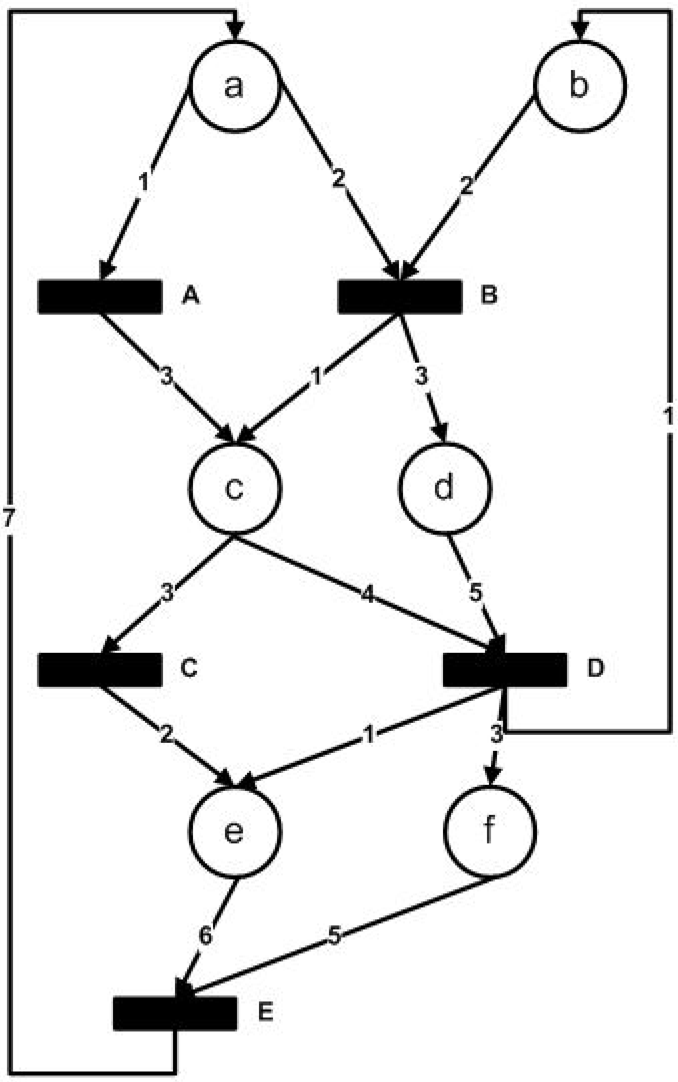


Figure 39 A sample Petri net

In the sample Petri net depicted in Figure 39 the set of places, *S* contains the elements *a, b, c, d, e, f.* The set of transitions, *T* contains the elements *A, B, C, D, E*. The set of arcs, *F* contains the elements *1, 2, 3, 4, 5, 6*.

Petri machines have the same expressive power as Turing machines, it can be said that a Petri net is a conjunction of Turing machines with the number of Turing machines that the Petri net models being equal to the number of tokens in the net.

7.1.8 Attribute automata

A finite attributed automaton augments finite automaton with a finite memory called attributes of states of the automaton. At any move of the automaton in attribute automata the information can be fetched from the attributes of the current state and stored into the attributes of the next state. The next state can be selected depending on the current state and the values of its attributes. Any move of the attribute automaton the contents of the memory can be finitely described.

An attributed automaton is defined in (Penjam, et al., 1991) as tuple

$AA = (I, S, A, v, f, s_0, m_0)$ where

I is a terminal alphabet,
S is an alphabet of states,
$A = \{A_s | s \in S\}$represents domains of attributes of states, each A_S is the domain of the attribute of state *s*,
$v : S \longrightarrow A$ *v* is an attribute mapping,
$f \subseteq \left(Sx\left(I \cup \{\varepsilon\}\right) \times \left(S \longrightarrow A\right)\right) \times \left(S \times \left(S \longrightarrow A\right)\right)$ is a transition relation,
$s_0 \in S$ is the initial state and
m_0 is the initial value of the attribute of the initial state.

A *configuration* of the finite attributed automaton *AA* is a triple
$(s, w, m, l) \in S \times I^* \times A_S \times A_x$,where $s \in S$ is the current state of the automaton,
w is the unused portion of the input string,
m is the value of the attribute of the current state and
l is the value of the attribute of the next symbol in the input string.

An attributed automaton allows describing an automaton where both the states and the input strings can have attributes; therefore it allows describing the behaviour of a reactive system, except for the temporal aspects. The finite memory of the attribute automaton reflects the current situation which is preserved between the computation steps. Also some aspects of proactive systems can be described.

7.1.9 The Q model

The description of the Q model can be found in (Motus, 1995) and (Motus, et al., 1994). The Q model was designed for describing and analysing timing properties of real-time embedded software, it comprises two types of components – processes and channels (that describe unidirectional interaction of processes). It neglects deliberately the algorithmic details of software, concentrating instead on describing and analyzing the structural attributes of components – e.g. timing constraints, properties of channels, validity of information, and consistency of constraints imposed on the components. A process is defined as: “A process is an atomic unit of computation, whose terminal behaviour is specified but whose internal operation is unspecified and of no interest to a discussion. Depending on the application, a process can be a computer instruction, a procedure or even an entire (unitary or distributed) system.” The term process can be used even in a wider sense – it may denote a computational process or a process occurring in the physical world, whether performed by natural or artificial entities. The formal specification of a process in the context of the Q model is the following:

A process (p) is considered to be a mapping (i.e. transformation) from its domain of definition (dom p) onto its value range (val p), the process time-set T(p) contains the activation instants of the mapping:

$$p : T(p) \times dom\ p \rightarrow val\ p$$

A process time-set is usually determined by the dynamic requirements of the environment, in some cases a process time-set may be in some cases also determined by the requirements of the computing system itself. Each process may have its own independent time-set, or may share it with the some of the other processes.

Process interaction in the Q model is realized by channels, which takes care of the time-selective data transfer between processes – selects the desired outputs of the producer process and presents them to the consumer process. The usage of channel makes it possible to design and implement both the producer and the consumer process without any considerations for the other party. The producer process simply produces the outputs and the channel accepts the outputs and forms the input to the consumer. A channel implements point-to-point one-way communication between two processes. The formal definition of the channel is the following: a channel is a mapping of the producer process (p_i) value range (output) to the consumer process (p_j) domain of definition (input), which can be expressed formally as follows:

$$\sigma_{ij} : val\ p_i \times T(p_i) \times T(p_j) \rightarrow proj_{val\ p_i} dom\ p_j$$

where $proj_{val\ p_i} dom\ p_j$ denotes the projection of the domain of p_j on the value range of p_i. Projection is important because one producer need not define the entire domain of the consumer process; it is more likely that the producer defines only part of it. Time-sets are required to allow description of the time-selectivity of the channel, which is described by the channel function $K(\sigma_{ij}, t)$, which defines for each $t \in T(p_j)$ a subset of $T(p_i)$ so that $K(\sigma_{ij}, t) \subset T(p_i), t \in T(p_j)$. A consumer process p_j, activated at $t \in T(p_j)$ only has access to the outputs of producer process p_i, resulting from activation at time instants $t \in K(\sigma_{ij}, t)$. In practice the channel function is expressed as an interval in the relative time (with respect to the consumer process's activation instant) for the associated pair of interacting processes.

Three types of channels have been introduced in the Q model: synchronous, semi-synchronous and asynchronous channels. In a synchronous channel $T(p_i) = T(p_j)$, which means that consumer's and producer's time-sets coincide.. In a semi-synchronous channel the producer time-set is given and it generates the time-set for the consumer process – $T(p_i) \rightarrow T(p_j)$. In case of an asynchronous channel the time-sets for the producer and consumer processes are independent (at least from the system designer point of view). Only this channel performs in a truly asynchronous mode as the name implies. The Q model also introduces the concept of *selector* processes. A *selector* process is a mapping whose execution is influenced by explicitly defined conditions: it can select only some of the variables as input from its domain or it may have more than one value range. Selection between the value ranges during a particular execution can depend on input data and/or interim results.

So the Q model allows describing process interaction where processes can select what they consume (including the data generation, i.e. the age of data). However the interactions between processes are predefined at system design-time.

Q models differs from the Turing machine as the input data to a process is not fixed – process start times are not fixed and therefore we do not know which outputs of a producer process are used as inputs by another process. A process may accept input data during its execution, which directly relaxes the most restricting constraint of Turing computable functions.

7.1.10 Stream processing

Stream processing is based on the concept of streams, stream functions and operations on them which model communication history in a computing system. A stream can be depicted as an ordered time series of data – such as readings from sensors, signals, messages, events, commands, etc.

Streams model the temporal succession of messages. Stream functions enable the processing of streams of undefined finite length, essentially sections of interaction streams. Example of stream processing when applied to interactive computation is described in (Dosch, et al., 2007). Stream processing as a tool for designing distributed computing systems and analysing the input / output behaviour of its components has been discussed in a PhD thesis by A. Stümpel (Stümpel, 2003).

A transformational is activated, takes in its input variable values (strings), processes the input variable values and terminates. A (stream processing) interactive computing system repeats this procedure (except for the termination) for each element of the input stream, with the difference that previously processed stream elements influence processing of the following stream elements. A stream processing system, after starting processing the first element of the input stream never stops before completing processing of the last element of the input stream.

Such behaviour is typical in pervasive computing systems which typically exhibit proactive behaviour and/or requirements for situation-aware behaviour. Hence the stream processing seems to be a natural model for studying the behaviour of interactive computing systems, besides the expressive power of stream processing being greater than that of the Turing Machine.

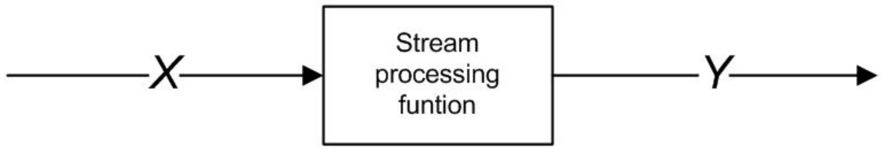


Figure 40 Stream Processing

In Figure 40 a sample stream processing function is depicted. The input stream *X* is converted to an output stream *Y* by the stream processing function. An important practical aspect of stream processing functions lays in the fact that a set of state machines that realize the stream processing functions can be generated automatically from a system of stream functions (Stümpel, 2003).

Curriculum Vitae

Personal Data

Name	Jürgo-Sören Preden
Date of birth	03.02.1978
Citizenship	Estonia

Contact Data

Address	Department of Computer Control, TUT, Ehitajate tee 5, 12618, Tallinn
Phone	+3726202117
E-mail	jurgo.preden@ttu.ee

Education

2005 - 2010	Tallinn University of Technology, Phd studies
2003 - 2005	Tallinn University of Technology, MSc
1996 - 2003	Tallinn University of Technology, BSc

Professional employment

2006 - ...	TUT, Department of Computer Control; researcher
2006 - ...	Smartdust Solutions, CTO
2006 - 2006	Internship at Microsoft Research, Redmond (06.2006 – 08.2006)

Current research topics

Computation and communication in ad-hoc networks, situation aware models of interactive computation.

Supervised theses

Rait Kapp, MSc, 2008, (supervisors) Leo Mõtus, Jürgo Preden, Comparison of code generated with the Gene-Auto code generator with hand written code, TUT, Department of Computer Control

Priit Pikk, MSc, 2009, (supervisor) Jürgo Preden, Positioning stationary motes with a moving platform, TUT, Department of Computer Control

Raido Pahtma, MSc, 2009, (supervisor) Jürgo Preden, Smart Dust Location Awareness, TUT, Department of Computer Control

Kaupo Vana, MSc, 2008, (supervisor) Jürgo Preden, Study of the ZigBee communication protocol, TUT, Department of Computer Control

Languages

Estonian – mother tongue
English – fluent
German – average
Finnish – average
Russian – poor

Elulookirjeldus

Isikuandmed

Nimi	Jürgo-Sören Preden
Sünniaeg	03.02.1978
Kodakondsus	Eesti

Kontaktandmed

Aadress	TTÜ Automaatikainstituut, Ehitajate tee 5, 12618, Tallinn
Telefon	6202117
E-post	jurgo.preden@ttu.ee

Hariduskäik

2005 - 2010	Tallinna Tehnikaülikool, doktoriõpe
2003 - 2005	Tallinna Tehnikaülikool, magistrikraad
1996 - 2003	Tallinna Tehnikaülikool, bakalaureusekraad

Teenistuskäik

2006 - ...	TTÜ, Infotehnoloogia teaduskond, Automaatikainstituut; teadur
2006 - ...	Smartdust Solutions, CTO
2006 - 2006	Intern Microsoft Researchis Redmondis (06.2006 – 08.2006)

Teadustöö põhisuunad

Arvutused ja kommunikatsioon spontaanvõrkudes, situatsiooniteadlikud interaktiivse arvutuse mudelid

Juhendatud väitekirjad

Rait Kapp, magistrikraad (teaduskraad), 2008, (juh) Leo Mõtus, Jürgo Preden, Gene-Auto koodigeneraatoriga genereeritud koodi võrdlemine käsitsi kirjutatud koodiga, TTÜ, Infotehnoloogia teaduskond, Automaatikainstituut,

Priit Pikk, magistrikraad, 2009, (juh) Jürgo Preden, Positioning stationary motes with a moving platform (Statsionaarsete kübemete positsioneerimine liikuva platvormiga), TTÜ, Infotehnoloogia teaduskond, Automaatikainstituut

Raido Pahtma, magistrikraad, 2009, (juh) Jürgo Preden, Smart Dust Location Awareness (Arupuru asukohateadlikkus), TTÜ, Infotehnoloogia teaduskond, Automaatikainstituut

Kaupo Vana, magistrikraad, 2008, (juh) Jürgo Preden, ZigBee kommunikatsiooniprotokolli uurimine, TTÜ, Infotehnoloogia teaduskond, Automaatikainstituut

Keelteoskus

Eesti – emakeel
Inglise – kõrgtase
Saksa – kesktase
Soome – kesktase
Vene – algtase

DISSERTATIONS DEFENDED AT TALLINN UNIVERSITY OF TECHNOLOGY ON *INFORMATICS AND SYSTEM ENGINEERING*

1. **Lea Elmik**. Informational modelling of a communication office. 1992.
2. **Kalle Tammemäe**. Control intensive digital system synthesis. 1997.
3. **Eerik Lossmann**. Complex signal classification algorithms, based on the third-order statistical models. 1999.
4. **Kaido Kikkas**. Using the Internet in rehabilitation of people with mobility impairments – case studies and views from Estonia. 1999.
5. **Nazmun Nahar**. Global electronic commerce process: business-to-business. 1999.
6. **Jevgeni Riipulk**. Microwave radiometry for medical applications. 2000.
7. **Alar Kuusik**. Compact smart home systems: design and verification of cost effective hardware solutions. 2001.
8. **Jaan Raik**. Hierarchical test generation for digital circuits represented by decision diagrams. 2001.
9. **Andri Riid**. Transparent fuzzy systems: model and control. 2002.
10. **Marina Brik**. Investigation and development of test generation methods for control part of digital systems. 2002.
11. **Raul Land**. Synchronous approximation and processing of sampled data signals. 2002.
12. **Ants Ronk**. An extended block-adaptive Fourier analyser for analysis and reproduction of periodic components of band-limited discrete-time signals. 2002.
13. **Toivo Paavle**. System level modeling of the phase locked loops: behavioral analysis and parameterization. 2003.
14. **Irina Astrova**. On integration of object-oriented applications with relational databases. 2003.
15. **Kuldar Taveter**. A multi-perspective methodology for agent-oriented business modelling and simulation. 2004.
16. **Taivo Kangilaski**. Eesti Energia käiduhaldussüsteem. 2004.
17. **Artur Jutman**. Selected issues of modeling, verification and testing of digital systems. 2004.
18. **Ander Tenno**. Simulation and estimation of electro-chemical processes in maintenance-free batteries with fixed electrolyte. 2004.

19. **Oleg Korolkov**. Formation of diffusion welded Al contacts to semiconductor silicon. 2004.

20. **Risto Vaarandi**. Tools and techniques for event log analysis. 2005.

21. **Marko Koort**. Transmitter power control in wireless communication systems. 2005.

22. **Raul Savimaa**. Modelling emergent behaviour of organizations. Time-aware, UML and agent based approach. 2005.

23. **Raido Kurel**. Investigation of electrical characteristics of SiC based complementary JBS structures. 2005.

24. **Rainer Taniloo**. Ökonoomsete negatiivse diferentsiaaltakistusega astmete ja elementide disainimine ja optimeerimine. 2005.

25. **Pauli Lallo.** Adaptive secure data transmission method for OSI level I. 2005.

26. **Deniss Kumlander**. Some practical algorithms to solve the maximum clique problem. 2005.

27. **Tarmo Veskioja**. Stable marriage problem and college admission. 2005.

28. **Elena Fomina**. Low power finite state machine synthesis. 2005.

29. **Eero Ivask**. Digital test in WEB-based environment 2006.

30. **Виктор Войтович**. Разработка технологий выращивания из жидкой фазы эпитаксиальных структур арсенида галлия с высоковольтным p-n переходом и изготовления диодов на их основе. 2006.

31. **Tanel Alumäe**. Methods for Estonian large vocabulary speech recognition. 2006.

32. **Erki Eessaar**. Relational and object-relational database management systems as platforms for managing softwareengineering artefacts. 2006.

33. **Rauno Gordon**. Modelling of cardiac dynamics and intracardiac bio-impedance. 2007.

34. **Madis Listak**. A task-oriented design of a biologically inspired underwater robot. 2007.

35. **Elmet Orasson**. Hybrid built-in self-test. Methods and tools for analysis and optimization of BIST. 2007.

36. **Eduard Petlenkov**. Neural networks based identification and control of nonlinear systems: ANARX model based approach. 2007.

37. **Toomas Kirt**. Concept formation in exploratory data analysis: case studies of linguistic and banking data. 2007.

38. **Juhan-Peep Ernits**. Two state space reduction techniques for explicit state model checking. 2007.

39. **Innar Liiv**. Pattern discovery using seriation and matrix reordering: A unified view, extensions and an application to inventory management. 2008.

40. **Andrei Pokatilov**. Development of national standard for voltage unit based on solid-state references. 2008.

41. **Karin Lindroos**. Mapping social structures by formal non-linear information processing methods: case studies of Estonian islands environments. 2008.

42. **Maksim Jenihhin**. Simulation-based hardware verification with high-level decision diagrams. 2008.

43. **Ando Saabas**. Logics for low-level code and proof-preserving program transformations. 2008.

44. **Ilja Tšahhirov**. Security protocols analysis in the computational model – dependency flow graphs-based approach. 2008.

45. **Toomas Ruuben**. Wideband digital beamforming in sonar systems. 2009.

46. **Sergei Devadze**. Fault Simulation of Digital Systems. 2009.

47. **Andrei Krivošei**. Model based method for adaptive decomposition of the thoracic bio-impedance variations into cardiac and respiratory components.

48. **Vineeth Govind**. DfT-based external test and diagnosis of mesh-like networks on chips. 2009.

49. **Andres Kull**. Model-based testing of reactive systems. 2009.

50. **Ants Torim**. Formal concepts in the theory of monotone systems. 2009.

51. **Erika Matsak**. Discovering logical constructs from Estonian children language. 2009.

52. **Paul Annus**. Multichannel bioimpedance spectroscopy: instrumentation methods and design principles. 2009.

53. **Maris Tõnso**. Computer algebra tools for modelling, analysis and synthesis for nonlinear control systems. 2010.