Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2009 Proceedings

European Conference on Information Systems (ECIS)

2009

An evaluation of user acceptance of a corporate intranet

Ansger Jacob
Universität Hohenheim, ansger.jacob@uni-hohenheim.de

Marcus Muller
Universität Hohenheim, marcus.mueller@uni-hohenheim.de

Stefan Kirn
Universität Hohenheim, stefan.kirn@uni-hohenheim.de

AN EVENT-DRIVEN APPROACH TO DYNAMIC SITUATION DETECTION

Jacob, Ansger, Universität Hohenheim, Information Systems 2, Schwerzstraße 35, 70599 Stuttgart, Germany, ansger.jacob@uni-hohenheim.de

Müller, Marcus, Universität Hohenheim, Information Systems 2, Schwerzstraße 35, 70599 Stuttgart, Germany, marcus.mueller@uni-hohenheim.de

Kirn, Stefan, Universität Hohenheim, Information Systems 2, Schwerzstraße 35, 70599 Stuttgart, Germany, stefan.kirn@uni-hohenheim.de

Abstract

To cope with the emerging demand for individual products and services, new approaches for handling the individualization in dynamic supply chains are needed. Our object of investigation is to establish Situation-awareness (SA) in an assistant information system for supply chain partners to cope with individualization-driven demands. Our contribution is an event-driven architecture design to enable situational planning using the Situation Calculus upon distributed contextual information. Sensors raise events among supply chain partner's information systems, which are combined to context events in order to derive situations on partner level. Our works follows the Design Science in Information Systems Research to design and evaluate the architecture design. The preliminary descriptive evaluation uses life stock transports as an example.

Keywords: Event-driven architecture, context-awareness, Situation Calculus, product individualization, supply chain management

1 INTRODUCTION

After a period of mass production, the demand for individualized products with prices similar to mass products is growing (Piller 2003; Pine II 1993; Reichwald and Piller 2006). In order to produce individualized goods, supply chain partners with special capabilities get together in adaptive multilevel supply chains. They work together for a limited period of time or for fulfilling the demand of only few customers. Afterwards that they participate in other supply chains to bring in their special production capabilities there (Dietrich et al. 2007). New mechanisms are needed to help combining the right production capabilities quickly and to ensure a high level of product quality as well as supply chain efficiency. Individualization research is about these adaptive mechanisms for participating organisations, products and supporting information systems in dynamic multi-tier supply chains (Kirn et al. 2008). Fleisch et al. (2005) show the past and ongoing development of information technology usage in companies from using IT for single functions to *individualized 1:1 coordination with customers* across company borders. In logistics individualized goods lead to an *atomisation of delivery* with more frequent deliveries and smaller quantities (Freitag et al. 2004). To support these future needs, real-time-management based on ubiquitous computing technologies and identification of individual products will be needed. Our work focuses on an information systems concept supporting the product individualisation process while using context and situation interpretation. Stated by Mertens (2004) Situation-awareness of information systems evolves into a major field of interest for Business Information Systems (BIS) research.

Our research is motivated by a lack of information system architectures supporting multi-tier product individualization in an autonomic manner. Object of investigation is an event-driven architecture which implements situation-based planning. The situation planning is done for each partner on top of shared context information in a dynamic multi-tier value system. The perspective taken is the decoupling of context generation and its interpretation through situation logic. This allows to access varying context information based on partner specific situation logic. Partners can join the value network and collect available context information relevant for their value transformation from all other partners to adjust their planning based on their own needs. We argue that situation logic on top of distributed context information in BIS supports individualized monitoring of the production, storage and logistics in dynamic value networks. We will present an instantiation of the architecture in life stock transports to monitor the transport for different supply chain participants. Our work follows the Design Science in Information Systems Research proposed by Hevner et al. (2004).

Our work uses a livestock transport example for evaluation of the proposed framework. Long journey livestock transports need monitoring of different parameters during transport. Architectures today, as shown in the technical specification proposed by the European Union (2006), use static rules causing alarms based on sensor events recorded by the transport company. To show the individualization supporting potential of our approach, we propose a company border crossing approach enabling situation planning for all partners in the livestock supply chain. This allows the optimization of business processes for the involved partners according to the goods situation. The situation can be set to the individual customer needs of a product, for example meat from well treated animals.

The rest of the paper is organized as follows: In the second section related work in mass customization, context and situation information systems research is shown. In section three we describe our architecture and in section four present an evaluation of the theoretical approach in life cattle transports fulfilling the European decree for long journey transports (2004). Finally we give an outlook on future work in section five.

2 THEORETICAL FOUNDATION

After decades of unifying production and products in order to achieve Economies of Scale and Scope, approaches in Mass Customization aim to support individualization and to keep the cost advantages of mass production (Piller 2003; Pine II 1993). Differentiation from competitors becomes even more important in a global economy. In future there will be more options for individualizing goods which are more suitable for differing customer needs worldwide. An example in shoe industry brings up promising results for customized goods engineered in multitier supply chains (Dietrich et al. 2007; EwoMacs 2007). To achieve this, significant changes in information system design for supply chain information systems have to be applied (Dietrich et al. 2006). On top of the findings of Mass Customization, Kirn et al. (2008) use a value system centric approach to individualization of goods and services. According to Porter (1985) a value system couples single value adding activities across more than one company with certain roles like suppliers, OEM, retailer, service provider or customer, which can be described using the Porter Value Chain model. The hypothesis in individualization research is that individualization of goods and services itself is an exclusive value for the customer that he or she is willing to pay for. Context and situations obtain a major role in building BIS for supporting pervasively adaptive value systems. Goods can be produced and monitored individually to support individualization in mostly automated manufacturing processes in an economic manner. Value system partners can be involved according to the needs of a special good without high cost efforts and the costs for individualization will decrease enormously. To achieve this, adaption of the value system in time, space and economy is implied. We realize adaption of value systems using invoked actions based on detected context and situations in our architecture.

In 1999 Dey, Salber and Abowd (2001) present the context toolkit. It offers a solution for rapid prototyping of context-aware applications and shows an integrated approach to context-awareness. The six main features of the context toolkit are *encapsulation of sensors*, *access to context data through a network API*, *abstraction of context data through interpreters*, *sharing of context data* through a distributed infrastructure, *storage of context data* and *basic access control for privacy protection*. To provide these functionalities, the following objects are considered in the context toolkit. *Context widgets* give access to context information for applications using the context. Interpreters can raise the abstraction of a piece of context. *Aggregators* are used to collect multiple pieces of context information that are logically related into a common repository. *Services* execute actions on behalf of applications. In contrast to a context widget retrieving state information a service changes state information in the environment via an actuator. Finally the *Discoverer* maintains a registry of context capabilities (widgets, interpreters, aggregators and services) actually existing in the framework. The context toolkit is a framework for dealing with context elements on different levels of abstraction, but Dey and Abowd do not explicitly define situations. The term situation can be derived from their definition for context (Dey and Abowed 1999): "*Context is 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 applications themselves.*" Following this definition a situation is seen as a combination of context elements and reflects the detection of these elements. It does not cope with the requirements of a situation which may occur in future or try to work with special requirements a situation may bring up by its dynamic state in time. The Context Toolkit integrates important functionalities for dealing with context elements, and hence we adapt some elements of their approach to handle context information in our architecture.

Some approaches aim to individualize process execution according to a given context. Schicker et al. (2007) follow the approach of Case Based Reasoning (CBR) to find similarities between processes and electronic services within the processes to enforce the reuse of already existing knowledge. Once a process similar to the actually needed is found, its structure is changing using the adaptive CBR mechanisms. The possible operations are *add*, *delete*, *change order* and *change of structural relations* to adapt the dimensions within the CBR process. The proposed architecture implies high manual effort

in setup and maintaining. Unknown context elements are not considered in order to have a complete picture of the process. These shortcomings are addressed in our work. Instead of forming processes based on services, we change the process based on upcoming situations. A situation defines the context it needs to know.

Handling vague context information using situation mechanisms is made by using methods applied in artificial intelligence. Bachfischer et al. (2007) make use of Bayesian Networks in order to adapt user interfaces according to given context information. Their approach optimizes complex modern car interfaces for accessing entertainment, air conditioning, navigation system, telephone and more functions. The Bayesian Network is used to estimate the driver's situation using sensor input from the car. After the interface of the car is adapted the driver can reach functions in which he or she is most probably interested in the current situation on top level. A high temperature inside the car may lead to a high ranking for the buttons to control the air-conditioning. Currently, their implementation tries to anticipate the driver's situation and displays a set of POI[1] accordingly, e.g. the display shows gas stations when the gas level is low. This solution is possible in closed systems like a car, but cannot be prewired in dynamic supply chain networks. Anagnostopoulos et al. (2007) make use of the fuzzy logic theory to cope with imperfect observations of context. They use and define situation-awareness as "*the particular kind of context-awareness, where situations are viewed as logically aggregated pieces of context*" (Anagnostopoulos et al. 2007). Their definition is one of only few definitions differentiating between context and situations, but finally implies the aggregation of context only. For them a situation-aware application has "*to estimate the user's current situation(s) and react appropriately*" and to "*autonomously adapt to the user's current situational context*" (Anagnostopoulos et al. 2007). They propose a situation estimation architecture, which applies rules for situation detection and action determination by exploiting the user's reactions in order to become more pervasive. According to a given estimated situation of a user the system applies actions based on an ontology containing relations between situations, persons and contexts. The context is differentiated into personal, temporal, special and artefact context. Using fuzzy algorithms and learning capabilities is a considerable future enhancement for our work. But the approach Anagnostopoulos et al. take has its limits in changing sensor environments, which we will cope with.

There are also situation- and individualization-related approaches adapted from artificial intelligence methods in supply-chain research. Freitag et al. (2004) describe the autonomous logistic concepts developed within the CRC 637 – Autonomous Cooperating Logistics Processes: A Paradigm Shift and its Limitations. They establish autonomy and a new control paradigm for logistic processes. Decisions about a process can be made without human intervention using autonomous software agents, sensor networks and mobile computing in a transport entity. According to sensor data input, autonomous agents representing each involved entity make decisions within these ubiquitous surroundings to optimize the transport. The approach is one of only few approaches not being user-centered in context aware system design. In contrast to our distributed situation planning approach for each partner, they shift the control paradigm in value networks to autonomous control without human intervention. Our approach relies on similar sensor approaches to integrate the system into adaptive business processes and to rely on decisions made by a system, but does not use autonomous agent technology.

In 1969 John McCarthy and Patrick J. Hayes (1969) pick up a situation description for machines in "Some philosophical Problems from the Standpoint of Artificial Intelligence". They propose the Situation Calculus capable of handling situations in information systems. They form situations based on fluents, causalities, actions, strategies and knowledge, and abilities. A situation is defined by McCarthy and Hayes as *"... the complete state of the universe at an instant of time*". All changes to the universe are result of an action, which are applied to a fluent. Changing fluents in a current Situation leads to a new Situation. Using a model of a situation, a machine would be capable of taking

[1] POI (Point of interest) is an overlay map containing spatial information about certain topics as the location of restaurants, gas stations, sights etc.

actions while only limited knowledge would be needed. Further work from Reiter (1991) helped solving the frame problem (sometimes) in the Situation Calculus. The extended Situation Calculus based on Reiter is a logic containing elements for describing a situation, but does not contain any reasoning or representation abilities. The programming language Golog presented by Levesque et al. (1997) bases on Reiter's findings to model complex operations based on the Situation Calculus. Meanwhile Golog emerged in many ways. There are approaches enhancing Golog to using concurrency (Giacomo et al. 2000), incremental program execution under incomplete knowledge with interleaved action, planning, sensing and exogenous events (Sardina et al. 2004), and decision theoretic planning for robot soccer (Ferrein et al. 2005). Li and Iijima (2007) stated in their analyses of the Situation Calculus use in IS, that "*despite of the existence of these problems, the Situation Calculus could be applied to business systems when some tradeoffs are made between expressiveness and reasoning*". Li and Iijima propose Web Service Composition, Context Awareness and Business Process Management as fields of application for the Situation Calculus. We use the Situation Calculus based on Reiter's approaches to build the situation logic for planning situations according to goals and based on context information. There are different extensions considered, as for example incremental program execution under incomplete knowledge proposed by Sardina et al. (2004).

Already researchers make use of the Situation Calculus in BIS. Dong et al. (2004) use it to apply autonomous computing in ubiquitous computing services. Their approach separates available fluents into a precondition fluent base reflecting the maximum of fluents affecting a situation, decisive fluents affecting a situation and non decisive fluent not affecting the situation. The Validity Theory ensures only valid components are Compared to approaches using Finite State Machines for situation detection, the Situation Calculus allows the detection of non-predefined situations during runtime. The Validity Theory formalizes application-specific validity requirements to validity-ensured policies, on which computers can autonomously choose to serve a service request. The approach is a promising way of reasoning using the Situation Calculus and Validation Theory in ubiquitous computing, but research is still at its beginning. The authors argue that commercial development methodologies are not available for the Situation Calculus, and that it is still difficult to deal with the Situation Calculus on engineering level. The works of Dong et al. are the main starting point for our situation-related work.

Concluding the related work there is no satisfying approach to use situation logic on top of real world information represented by sensor-based context information, but there are many promising approaches in context and situation research already. We include suitable research identified from context and situation research, especially the ideas from the Context Toolkit from Dey et al. (2001) in our context layer and the Situation Calculus with its extensions in our situation layer (McCarthy and Hayes 1969).

3 AN EVENT-DRIVEN APPROACH TO DYNAMIC SITUATION DETECTION

In this section we propose an event-driven architecture which is capable of situation handling on top of distributed real world information. It enables higher-level IS to use the paradigms of context- and situation-awareness based on sensor information. The novelty of our work compared to other approaches is the combination of a Situation Calculus approach for situational reasoning with an event-driven context abstraction logic based on sensors. This enables partners in collaborative environments like dynamic supply chains to maintain their own situation logic expressing their need, whilst product-related context information can be accessed from all partners in the collaborative environment. The following is a presentation of the three-layered design of this approach.

Today context- and situation information is used mainly in optimizing the human computer interface (HCI) design. An example for this field of application is the use of information about the position of a device which minimizes the data presented to the user to the information relevant for the current

location. This context service is called location-based service as described by Wehrmann (2004) in his work about situation-dependent mobile services. Beyond location nearly any context information can be used in a way to simplify interaction with a device. By traditional means an application becomes context-aware if it causes actions according to changing context information. The definition of context-awareness presented by Dey (2001) emphasizes the need for a user: "*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 contrast to this user-centred view we follow a more system-centred one. Distributed sensors in collaborative environments offer their information in an event-driven manner. This is matched with situation logic and its reasoning capabilities. Context sources become exchangeable and situations are detected and processed upon their availability. Our architecture integrates situation reasoning based on assumptions and targets on top of dynamic context knowledge. A sketch of the architecture including two partners sharing their context information is shown in figure 1 and described in the following sections.

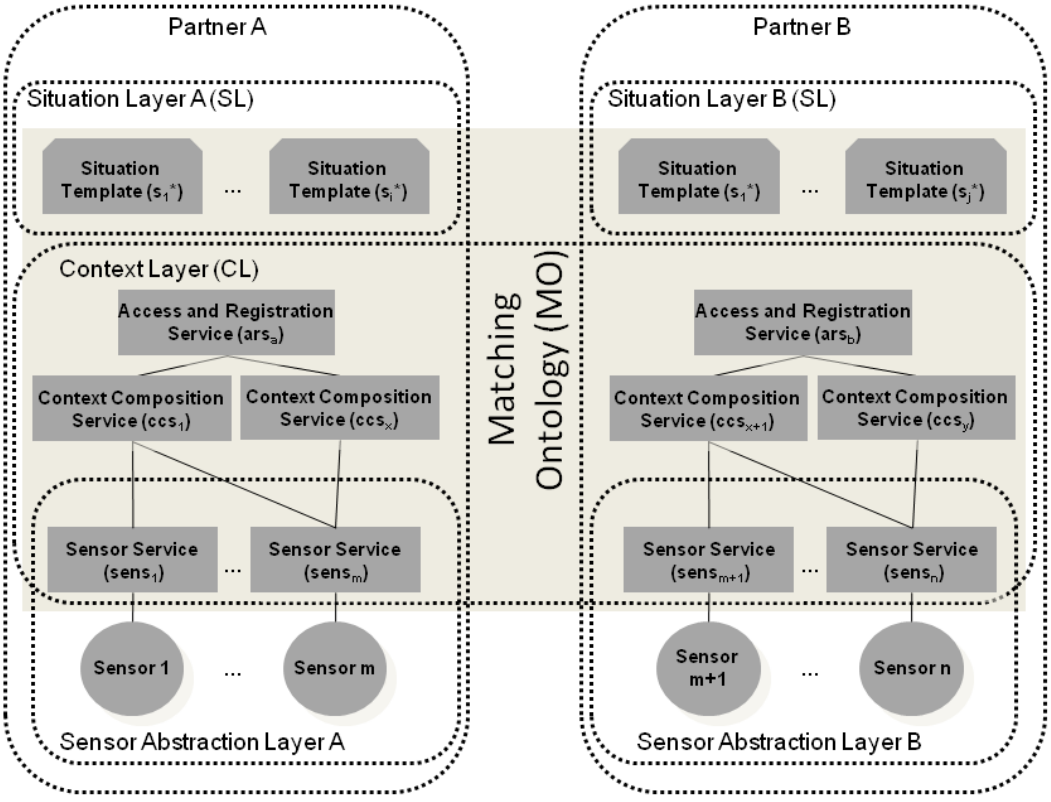

Figure 1. An architecture model for dynamic situation detection

At first we introduce the elements of the Sensor Abstraction Layer (*SAL*) operating on event level. The Sensor Services (*sens*) hide the specifics of a certain sensor. All communications with the sensor are leveraged to a unified XML-based format. A short history of sensor values, stored in a database of the sensor hosting node, is implemented to allow queries involving the past. Information about the sensor range and the position are stored in the database as well. It is developed similar to the concept of Widgets from the Context Toolkit and extends them in standardization using Web Service technology (Dey et al. 2001). For the description of *sens* we use a simplified Sensor Model Language (SensorML) as proposed by the Open Geospacial Consortium (2007). SensorML itself is a very powerful sensor description language. Beyond the sensor description itself SensorML stores metadata about the sensor, especially the sensors position, its availability or its range in space. We used SensorML to develop our corresponding database.

Besides physical sensors, already available data can also be accessed through the sensor abstraction layer. The data can be encapsulated like conventional sensors in SensorML using a standard *sens* services. Examples for explicit information interesting for encapsulation could be weather information, traffic information or data from a company's database. All sensor services are part of a partners system and are accessible by other partners for the purpose of a specific supply chain need.

On the level above the *SAL,* the Context Layer (CL) contains Context Composition Services (*ccs*). The *ccs* contain a composition logic and combine elementary sensor information or other *ccs* information with higher aggregated context information. These are similar to the concept of Aggregators Dey et al. used in the Context Toolkit expect the fact a *ccs* can integrate other *ccs* (Dey et al. 2001). The *ccs* raise the level of context in a cascading manner. A possible combination, for example, is having one sensor detecting the current outside temperature and another sensor detecting the illumination level in the same local area. These can be combined to a *ccs* delivering information about day and night time more accurately. The *ccs* are valid depending on their relationship in space and time (for example "in the same room at the same time") and can expire according to the availability of underlying *sens* or other *ccs*.

The Access and Registry Service (*ars*) gives access to one instantiated context framework on top of a cascade of *ccs* or even only a single *sens*. It contains a registry of underlying *sens* and *ccs* and is the entry point for other partners accessing one partners services. The *ars* can be combined with other partner's *ars*. In this case all lower services become available in both *ars*. The combination of different *ars* is applied to connect different partner domains for a limited time or to reduce handling complexity in case of larger networks.

All element names from *SAL*, *CL* and *SL* use the concepts of a common Matching Ontology (*MO*). Thus we can estimate the closeness of two elements and even match similar elements of every layer.

Formally the *CL* is a multidimensional undirected graph, in which R are all available edges, $\alpha : R \rightarrow (SENS, CCS, ARS)$ being the starting points and $\omega : R \rightarrow (SENS, CCS, ARS)$ being the endpoints of an edge. Within this configuration, each $ccs \in CCS$ can be connected to any $sens \in SENS$ or other $ccs \in CCS$ to generate higher levels of context:

$$CL = ((SENS, CCS, ARS), R, \alpha, \omega)$$

Each Sensor is connected to one *sens* exclusively. A valid minimum *CL* is one *sens* and one *ars* to access the *sens*:

$$CL_{\min} = ((sens, ars), R, \alpha(sens), \omega(ars)) \text{ and } | R | = 1$$

One level above the shared and common *CL,* the partner-specific Situation Layer (*SL*) is located, which is using the Situation Calculus from McCarthy and Hayes (1969) as well as further developments from Reiter (1991). The *SL*'s main task is to detect a situation *s* upon situation template axioms *s** taken from a situation template repository *S**.

The Situation Calculus is a first-order language with some second-order features for representing dynamic domains. According to its definition, all changes to the world are result of an action and the situation "*represents the complete state of the world at an instant of time*" (McCarthy and Hayes 1969). Thus a world history is a sequence of *actions* and is represented by the first order term s*ituation*. The constant S_0 denotes the initial situation in which no actions have yet occurred. Objects catch everything else relevant depending on the situations domain. All $a \in ACTION$ are specified by an action precondition axiom and a successor state. The action precondition axiom is highly important for our work, because we need to find relevant preconditions fulfilled for a certain situation according context detected:

$$Poss(A(x_1, ..., x_n), s) \equiv \Pi_A(x_1, ..., x_n, s)$$

Π_A is a formula uniform in s with free variables x_i. To define the closeness of a *situation* to a success, Dong et al. (2004) use special fluents on top of Reiter's approach to estimate the closeness to a successor state. Their concept of *incremental successor states* containing the distinguished binary function *do(a, s)* denotes the situation resulting from performing action *a* in situation *s*. $\Psi_{F,a}$ helps to find the way coming closer to a better situation *s* performing an action *a* instead of having a binary function like in traditional Situation Calculus:

$$Poss(a(x), s) \supset F(do(a(x), s) = \Psi_{F,a}(x, F(s))$$

Dong et al. (2004) argue that the definitions for a fluent in standard Situation Calculus would be too general, and therefore propose an extension to make it capable of comparing situations. They classify into functional fluents ($\overrightarrow{FF}$) meaning all available fluents, decisive fluents ($\overrightarrow{DF}$) meaning a minimum set of fluents for a situation, non decisive fluents ($\overrightarrow{NF}$) meaning possible fluents for a situation and a precondition fluent base ($\overrightarrow{PFB}$) containing all fluents which may affect a situation:

$$\overrightarrow{FF} \subset \overrightarrow{PFB} = \overrightarrow{NF} \bigcup \overrightarrow{DF} \text{ and } \overrightarrow{DF} \bigcap \overrightarrow{NF} = \varnothing$$

Decisive fluents are all atomic fluents or ones that cannot be derived by other fluents for a specific situation. Based on the differentiation from Dong et al. (2004), a comparison of situations according to fluent classification is done in our approach.

Using situation validity based on Validity Theory, Dong et al. (2004) define a minimum set of relational fluents to ensure the situation validity. For one situation a possible validity fluent is defined as:

$$VA(\vec{y}, s) : Object^n \times Situation \rightarrow boolean$$

All *VA* in the vector *y* contain special terms in which a situation *s* is valid. The situation *s* is valid only if all *VA(s)* in *y* hold. A situation can have none, one or more validity arguments to define the state in which a situation is valid. This is used in our approach to ensure that certain situations only come true when valid configurations of events occur.

We are using the Situation Calculus in the *SL* to match with context information available from *CL* for a current situation s_i* from all available situations $s^* \in S^*$. The syntax of actions, objects and fluents in *SL* is taken from the *MO*. This ensures a unique syntax in *CL* and *SL*. The matching process starts within the *SL* matching *precondition axioms* to *ccs* or *sens* of the *CL*. The search starts at the highest level in *CL*, the most upper *ccs*, going down the graph along the *ccs* to the *sens* using graph search algorithms in order to find a best match to a situation template $s^* \in S^*$. Using the *MO* a possible combination of *sens* and *ccs* can lead to new *ccs* matching a specific precondition axiom. Detectable situations are all $s_i^* \in S^*$ with a matching between precondition axioms and *ccs* or *sens*:

$$s_i^* = \max \left\{ \Pi_A(x_1, ..., x_n, s_i^*) \bigcap (CCS \cup SENS) \right\}$$

Once all detectable $s_i^* \in S^*$ have been found, the plans can be applied to a detected situation. A plan can lead to new detectable situations within the *SL* with new matchings to the *CL*. The *MO* is used as taxonomy in order to have a unique vocabulary to find similarities between the elements of the *CL* and *SL*. In order to prevent a time-consuming search for a best match when applied to the available *SenS* and *CCS* in *CL*, prebuilt graphs for all elements of the *CL* are generated every time there is an update in the *CL*. Additional functional validity fluents can be used to ensure validity of a *ccs* or *sens* in space and time in more complex environments.

In this section we presented the theoretical approach of our event-oriented architecture. It allows new sensors and services to be added and removed from the *CL* during runtime (Michelson 2006). If new sensors including their corresponding *sens* are integrated, they register with their correspondent *ars* which may spread the information to other related *ars*. The *ars* inform current subscribers in *SL* about

the new service, which may book the service then. The architecture ensures the possibility to compose dynamic value systems containing many sensors from changing value system partners. Also mobile sensors which are changing their context in space and time can be handled dynamically, for example the sensors of a life stock transport can be added logically to the system of the butcher.

4 EVALUATION

The empirical evaluation of the implemented framework in livestock transports will be determined in the future. Instead we use a descriptive evaluation to detect the situation *heat shock for animals in trailer* for the animal transport company and *heat stress* for the butchery now. Both situations are based on sensors in the same *CL* monitoring the drinking water level ($sens_1$) and temperature ($sens_2$) inside the trailer, and a third sensor monitoring the amount of water the animals drank on the farm in the stable ($sens_3$) before the transport. This data is accessible via a short history of all sensor values.

At first we model the *CL* as a graph starting with the *ars* on top of the sensors and containing direct links from *ars* to every *sens* in the trailer and is aware the *ars* from the farm:

$$CL = ((sens_1, sens_2, sens_3, ars), R, \alpha(ars), \omega(sens_1, sens_2, sens_3))$$

The object to be manipulated by the situation *heat shock for animals in trailer* for the transport company in $SL_{transport}$ is the route the transporter will take. To alter an existing route ($res(route_o)$) in situation *s* we need the situational preconditions *low water consumption before transport* (*con_low*), *high temperature in trailer* (*temp_hi*) and *no drinking water in trailer* (¬*wat*) fulfilled:

$$Poss(res(route_o), s) \equiv con\_low(s, route_o) \wedge temp\_hi(s, route_o) \wedge \neg wat(s, route_o)$$

Altering a $route_o$ in a situation *s* should lead to a new $route_n$ which either has enough water consumption before the transport detected, or has enough drinking water for the resulting driving time, or has no high temperature for the resulting route. A successor state after altering the $route_o$ in situation s to a new $route_n$ is:

$$route_n(do(res(route_o), s) \equiv \neg con\_low(s, route_n) \vee \neg temp\_hi(s, route_n) \vee wat(s, route_n)$$

The situation validity is ensured checking against the maximum driving time t_{max} allowed for long journey livestock transports. In European law this is set to a maximum of 8 hours without a break for the animals (European Union 2004):

$$VA(s) \overset{def}{\equiv} (t(route_n) \leq t_{\max})$$

Using an *MO* with values for livestock transports the elements *con_low* can be matched to $sens_3$ with a minimum amount each animal has drunk before, *temp_hi* is matched to $sens_2$ with a certain minimum level of temperature detected and *wat* is matched to a low-level-indication of $sens_1$. The maximum journey time is set in the *MO* as well. Each possible $route_n$ considered does fulfil the successor state above. In case there is no route alternative, the driver is informed according to the given situation.

With the same sensor information we can conclude to a different situation for a different supply chain partner. The object to be manipulated by the situation *heat stress* in the butchery's $SL_{butcher}$ is the heat stress for the animals transported, which can be detected using temperature and water consumption sensors. Water consumption uses the history of water levels in order to calculate the water consumption. The heat stress condition is important in order to estimate the quality of the meat. The preconditions needed only depend on a high temperature exposure *high temperature (temp_hi)* and a *high level of water consumption (Δwat)*:

$$Poss(de(heatstress), s) \equiv temp\_hi(s, heatstress) \wedge \Delta wat(s, heatstress)$$

Depending on the precondition fulfilment, the rest time of the animals before being butchered can be determined by the butchery.

All fluents described in the examples are within the $\overrightarrow{DF}$, because they affect the current situation. But we could also detect the situation if we would have an additional parameter like a considered traffic jam, but no sensor to match to. Then the route would also be altered ignoring traffic jams and the traffic information is part of $\overrightarrow{NF}$. This becomes important when dealing with large amounts of fluents.

Using the proposed architecture for monitoring the transports leads to an adaption of the transport process during runtime in contrast to single alarm events like high temperature or low water in other installations of a monitoring environments for long journey livestock transports (Joint Research Centre 2008). This helps not only notifying in case something goes wrong, but preventing something might go wrong and assist finding solutions. Also other solutions don't provide individual situations to be detected by different partners. We presented the detection of two simple situations *heat shock for animals in trailer* for the animal transport company and *heatstress* for the butcher in order to optimize the company's business processes based on sensors of the truck and the farm. The sensors are interchangeable enabling new applications in environments with changing sensor sources.

5 CONCLUSION

We presented the concept of an architecture which uses information about the situation on top of context information for complex context-based decisions in pervasive computing. The framework is currently under development and the approach in this work is of theoretical nature. There are still open issues to solve within the design, especially developing an efficient matching of the elements in the *CL* and the situation templates *S** based on the *MO* and an efficient structure of the MO.

Our approach is limited to static situation execution yet. As long as parts of the situation are detected and no argument is invalid, the situation is considered true. In our future work we need to add the ability to handle uncertainty in situation detection, for example when two of three preconditions are fulfilled. Vassos and Levesque (2007) have an interesting approach using the Situation Calculus with incomplete knowledge of an agent.

A next major step is the implementation of the livestock sensor scenario using IndiGolog proposed by Sardina et al. (2004). We will use RFID for animal identification, and sensors detecting conditions in the trailer as well as external parameters. Other supply chain partners are represented by their own situation client. Within the implemented scenario we will be able to test scalability issues which will be a challenge to deal with.

In our future work a connection between the framework and a higher level transport information system will be implemented to demonstrate the advantage of situation handling in logistics. Whereas new situations based on a changing context are determined, an automated adaption of the planning process is achievable. Furthermore interesting topics cover human feedback to enhance the system making more precise situation detections or continuous sensor values substituting other values at a certain level. An integration of digital world models in our architecture could enhance the capabilities of the framework by using spatial relations. Nicklas et al. (2001) explore using spatial aware applications in digital world models. This could enhance our situation reasoning by physical distance of objects, for example causing a situation "Possible infection transfer" by two animal transports being next to each other on a rest area.

Our architecture is not limited to animal transports. It can be used whenever distributed systems share a common data in an event based manner. Thus it can be used in many setting in supply chain BIS. We are still at the beginning of discovering situation potential in supply chain BIS, and we expect an increasing level of automation and lower coordination costs.

6 ACKNOWLEDGEMENTS

This work was supported by the German Ministry for Research and Technology (BMBF) within the research project IT FoodTrace – IT supported Food Traceability (2007).

REFERENCES

Anagnostopoulos, C. B., Ntarladimas, Y. and Hadjiefthymiades, S. (2007). Situational computing: An innovative architecture with imprecise reasoning. J. Syst. Softw. 80 (12), 1993-2014.

Bachfischer, K., Bohnenberger, T., Hofmann, M., Wäller, C. and Wu, Y. (2007). Kontext-adaptive Fahrerinformationssystem am Beispiel eines Navigationssystems. Künstliche Intelligenz 03/2007 (3), 57-63.

Dey, A. K. (2001). Understanding and Using Context. Personal and Ubiquitous Computing 5 (1), 4-7.

Dey, A. K. and Abowed, G. D. (1999). Towards a better understanding of context and context-awareness. GVU Technical Report GIT-GVU-99-22. GVU Center, Atlanta, USA.

Dey, A. K., Salber, D. and Abowd, G. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal 16 (2-4), 97-166.

Dietrich, A. J., Kirn, S. and Sugumaran, V. (2007). A Service-oriented Architecture for Mass Customization – A Shoe Industry Case Study. IEEE Transactions on Engineering Management 54 (1), 190-204.

Dietrich, A. J., Kirn, S. and Timm, I. J. (2006). Implications of mass customisation on business information systems. Int. J. of Mass Customization 1 (2), 218-236.

Dong, W., Xu, K. and Lin, M. (2004). A Situation Calculus-based Approach To Model Ubiquitous Information Services. 2004-4-2. State Key Laboratory of Software Development Environment, Computer Science & Engineering Department, Beijing University of Aeronautics & Astronautics, Beijing, China.

European Union (2006). Technical Specifications for Navigation Systems in Long Journey Animal Transports, Technical Report GO7-TRVA/ (2006). Centre, J. R., 1-90.

European Union (2004). Verordnung (EG) Nr. 1/2005 des Rates vom 22. Dezember 2004 zum Schutz von Tieren beim Transport und damit zusammenhängenden Vorgängen. 1-44.

EwoMacs (2007). EwoMacs Homepage. http://www.wi2.uni-hohenheim.de/de/p_ewomacs.html, access date 2008-03-11.

Ferrein, A., Fritz, C. and Lakemeyer, G. (2005). Using Golog for Deliberation and Team Coordination in Robotic Soccer. Künstliche Intelligenz 05 (01), 24-30.

Fleisch, E., Christ, O. and Dierkes, M. (2005). Die betriebswirtschaftliche Vision des Internets der Dinge. In: Fleisch, E. (Hrsg.): Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis. Visionen, Technologien, Anwendungen, Handlungsanleitungen. Springer, Berlin, 3-38.

Freitag, M., Herzog, O. and Scholz-Reiter, B. (2004). Selbststeuerung logistischer Prozesse – Ein Paradigmenwechsel und seine Grenzen. Industrie Management - Zeitschrift für industrielle Geschäftsprozesse 20 (1), 23-27.

Giacomo, G. d., Lespérance, Y. and Levesque, H. J. (2000). ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence 121 (1-2), 109-169.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly 28 (1), 75-105.

IT FoodTrace (2007). IT FoodTrace Homepage. http://www.itfoodtrace.de, access date 2007-04-17.

Joint Research Centre (2008). Animal Welfare in Transportation. http://awt.jrc.it/, access date 2008-01-21.

Kirn, S., Anhalt, C., Bieser, T., Jacob, A., Klein, A. and Leukel, J. (2008). Individualisierung von Sachgütern und Dienstleistungen durch Adaptivität von Wertschöpfungssystemen in Raum, Zeit und Ökonomie. In: Kirn, S. (Hrsg.): Individualization Engineering. Cuvillier-Verlag, Göttingen, 3-60.

Levesque, H. J., Reiter, R., Lespérance, Y., Lin, F. and Scherl, R. B. (1997). GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming 31 59-84.
Li, B. and Iijima, J. (2007). A Survey on Application of Situation Calculus in Business Information Systems. International Conference on Convergence Information Technology.
McCarthy, J. and Hayes, P. J. (1969). Some philosophical Problems from the Standpoint of Artificial Intelligence. In: Meltzer, B.; Michie, D. (Hrsg.): Machine Intelligence 4. Edinburgh University Press, 463-502.
Mertens, P. (2004). Zufriedenheit ist die Feindin des Fortschritts – ein Blick auf das Fach Wirtschaftsinformatik. Universität Erlangen-Nürnberg, Bereich Wirtschaftsinformatik I. access date 2008-03-27.
Michelson, B. M. (2006). Event-Driven Architecture Overview. Patricia Seybold Group, Boston.
Nicklas, D., Großmann, M., Schwarz, T., Volz, S. and Mitschang, B. (2001). A Model-Based, Open Architecture for Mobile, Spatially Aware Applications. 7th International Symposium on Advances in Spatial and Temporal Databases.
Open Geospacial Consortium Inc. (2007). Sensor Model Language (SensorML). http://www.opengeospatial.org/standards/sensorml, access date 2008-07-05.
Piller, F. (2003). Mass Customization - Ein wettbewerbsstrategisches Konzept im Informationszeitalter. Deutscher Universitätsverlag, Wiesbaden.
Pine II, B. J. (1993). Mass Customization - The New Frontier in Business Competition. Harvard Business School Press, Boston, Massachusetts.
Porter, M. E. (1985). Competitive Advantage. Free Press, New York.
Reichwald, R. and Piller, F. (2006). Interaktive Wertschöpfung - Open Innovation, Individualisierung und neue Formen der Arbeitsteilung. Gabler, Wiesbaden.
Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (Hrsg.): Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, California, 359 –380.
Sardina, S., Giacomo, G. D., Lespéance, Y. and Levesque, H. J. (2004). On the semantics of deliberation in IndiGolog - from the theory to implementation. Annals of Mathematics and Artificial Intelligence 2004 (41), 259-299.
Schicker, G., Kaiser, C. and Bodendorf, F. (2007). Individualisierung von Prozessen und E-Services mithilfe von Case Based Reasoning. In: Oberweis, A.; Weinhardt, C.; Gimpel, H.; Koschmider, A.; Pankratius, V.; Schnizler, B. (Hrsg.): eOrganisation: Service-, Prozess-, Market-Engineering, 8. Internationale Tagung Wirtschaftsinformatik. Universitätsverlag Karlsruhe, Karlsruhe, 713-730.
Vassos, S. and Levesque, H. (2007). Progression of Situation Calculus Action Theories with Incomplete Information. Twentieth International Conference on Artificial Intelligence. Hyderabad, India.
Wehrmann, J. (2004). Situationsabhängige mobile Dienste. PhD Thesis, Lehrstuhl für Wirtschaftsinformatik III, Friedrich-Alexander-Universität.