Reihe: Telekommunikation @ Mediendienste · Band 6
Herausgegeben von Norbert Szyperski, Udo Winand, Dietrich Seibt, Rainer Kuhlen und Rudolf Pospischil

Martin Engelien/Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien

Workshop GeNeMe99
Gemeinschaften in Neuen Medien

TU Dresden, 28./29.10.1999

JOSEF EUL VERLAG
Lohmar · Köln

Reihe: Telekommunikation @ Mediendienste · Band 6

Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, Prof. Dr. Rainer Kuhlen, Konstanz, und Dr. Rudolf Pospischil, Brüssel

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inform. (FH) Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien

Workshop GeNeMe99
Gemeinschaften in Neuen Medien

TU Dresden, 28./29.10.1999

JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

GeNeMe <1999 Dresden> :
GeNeMe 99 : Gemeinschaften in neuen Medien ; Dresden, 28./29.10.1999, an der Fakultät Informatik der Technischen Universität Dresden / Technische Universität Dresden, Fakultät Informatik, Institut für Informationssysteme, Forschungsgruppe "Entwurfsmethoden und Werkzeuge für Anwendungssysteme". Martin Engelien ; Jens Homann (Hrsg.). – Lohmar ; Köln : Eul, 1999
(Reihe: Telekommunikation @ Mediendienste ; Bd. 6)
ISBN 3-89012-710-X

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 91 08 91
Fax: 0 22 05 / 91 08 92
http://www.eul-verlag.de
eul.verlag.gmbh@t-online.de

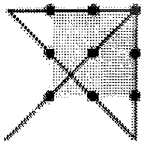
Printed in Germany
Druck: Rosch-Buch, Scheßlitz

Gedruckt auf säurefreiem, 100% chlorfrei gebleichtem, alterungsbeständigem Papier nach DIN 6738

Technische Universität Dresden
Fakultät Informatik • Institut für Informationssysteme
Forschungsgruppe „Entwurfsmethoden und Werkzeuge für Anwendungssysteme"

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inform. (FH) Jens Homann
(Hrsg.)

an der
Fakultät Informatik der Technischen Universität Dresden



Gefördert von der Klaus Tschira Stiftung,
gemeinnützige Gesellschaft mit beschränkter Haftung

sowie unter Mitwirkung der
GI-Regionalgruppe Dresden

am 28./29.10.1999
in Dresden

B.3. Context modeling of agile software and a context-based approach to support virtual enterprises

Dipl.-Inform. Duy-Tuan Nguyen
Dr. V. Do
TU Dresden

Abstract

In the practice of software development contexts have been only implicitly modeled and transformed in fixed part of software. The information about context is dispersed in objects across the application. Each context change leads to modification or new development of the software.

The context modeling helps developer to separate context objects from context-dependent objects. It allows better reuse of analysis, design and implementation models, if the context of certain objects is changed. The context modeling is interesting and necessary, when the software should be agile – i.e. when the environment and the condition of the software could be changed permanently, e.g. in case of platform for virtual enterprises. The paper introduces a novel approach to support processes within generic platforms for virtual enterprises: the context-based approach. The main advantage of the approach lies in its generic capacity, which allows the users to define processes flexibly to support their own enterprises.

In this paper, we discuss further the phenomenon of extra-context logic, its modeling and its application case. The information of extra-context logic provides not only the better understanding of application domain, but also can be used by a wizard to support the interaction of users working with multiple systems.

1 Motivation of context-modeling

The word "context" is derived from the Latin "con" + "texere" meaning to weave together. The Oxford English Dictionary gives as one of its meanings "ambient conditions, a set of circumstances; relation to circumstances". The importance of context in software engineering is now just beginning to be realized. Many important topics in computer science can benefit from the theoretical and practical use of context.

Most changes or new development of software can be traced back to context change. Till now, any context change will be understood as the change of a requirement, which leads to the new development of the software. The object-orientation provides several mechanisms for reuse but this does not seem to be sufficient. A context should be conceived as an object and can be changed as necessary. Computer applications traditionally expect a static execution environment. However, this prerequisite is

generally not possible for agile systems, such as platform for virtual enterprises (abbr. PVE), where the world around an application (the PVE) is constantly changing.
This paper explores how to support and also to exploit the dynamic configurations and social settings characteristic of PVE with the help of context modeling.

2 What is a context ?

The concept of 'context' has been the research object of natural language processing (NLP) and logical artificial intelligence (AI) for a long time. According to [Crystal 1991], 'context' is a general term in linguistics and phonetics to refer to specific parts of an utterence (or text) near or adjacent to a unit (e.g. sound, word) which is the focus of attention. Leech (1991) notes that the specification of context (whether linguistic or non-linguistic) has the effect of *narrowing down* the communicative possibilities of a message. The *particularization* of meaning can take place in assorted ways, including:

- elimination of certain ambiguities or multiple meanings in the message
- clarification of the referents of deictics and definite descriptions
- supplying of information which the writer has omitted through ellipsis
- interpretation of tense, and
- determination of the scope of quantifiers.

In the field of Artificial Intelligence, context is defined as the knowledge that is needed to reason about another system, for the purpose of solving a problem. When using a well-defined ontology, such as Cyc [Guha 1995], a well-defined partition (called Microtheory) of an ontology is assigned as a context.
In the field of database, context is defined as the meaning, context, organization and properties of data. It is model using meta-data associated with the data [Sciore et al 1992]. A context may be identified or represented using the following [Kashyap et al 1996]:

- by association with a database or a group of databases
- as the relationship in which an entity participates
- from a schema architecture
- at a very elementary level, as a named collection of domains of objects.

From the point of view of computer scientists a context is an execution environment encompassing a computing unit. A context is the circumstances surrounding an act or event. For some authors a context is only a complex software entity that encapsulates actions while an object is a basic software entity, which models actions [Buffo 1996]. The context in this sense is only a container, a community of objects, which coordinates the interactions of objects it contains. We use the concept of context only, if an entity enters or exits a context, its behavior may be changed. If we conceive a context as an

object (contextual object), this object must have some characteristics. If a contextual object is attached to some other objects, the behaviors of these objects may be changed if necessary. The attachment of a contextual object to an object is equivalent to the entry of the object to the context. The detachment of the contextual object from an object means the exit from the context.

3 Characteristics of context

1. Context is specific to a particular problem solving task and provides representation, assumptions and mechanisms tailored for the problem it was set up to solve.
2. Each context has its scope, the scope of the problem solving.
3. To solve a problem a group of objects enters the corresponding context solves the problem by applying mechanisms and exits the context.
4. The context is necessary for the application of the mechanisms and is not affected by such application.
5. A Context can be created to solve a certain problem. There would be an overlapping of contexts.
6. An object can be found in one or several contexts.
7. If a group of objects is located in two contexts, there may be conflicts between mechanisms to solve the problem. The decontextualization is the process which lifts (or chooses) the correct solving mechanisms among the conflicting mechanisms.

The context of a system differs from its state. The state of a system (or an object) represents the cumulative results of its behavior (see [Booch 1994]). The state of a system is its inherent property. While interacting (executing computation step) the object can change its state. The context of a system however does not belong to its inherent properties and represents its environment which is needed for an execution step to take place but which is not affected by such step.

4 Types of context-based systems

The context of an object of a system can be changed. According to the flexibility of the context definition we can classify four levels of context-based systems.

1. The new context can be discovered *dynamically* at run-time by the systems. The context-dependent objects react to changes in the context they are situated and change their behavior according to the newly created contextual object. We call such systems as *context-aware* system, since they adapt according to changes over time. Generally, such systems have four generic contextual capabilities: sensing, adaptation, resource discovery, and augmentation [Pascoe 1998]. Context-aware

systems are typically integrated in wearable computers and mobile computing systems ([Pascoe 1998], [Schilit 1995]).

2. The new context can be created at run-time by the users. The users define the new contextual object *by configuration,* which specifies how context-dependent objects are influenced by the newly created contextual objects. The context-based system contains a repository of primitives and allows the users to compose contextual objects from the primitives. At the executing time the system behaves according to the composed contextual object.
3. The new context can be defined *by programming* of new contextual objects. There is a separation of contextual information from context-dependent objects. The separation provides better reuse mechanisms in software development, because the context-dependent objects do not need to be modified and only the contextual objects have to be implemented. More details can be found in the section " context relation in design and implementation".
4. The new context can be defined *by adding* context information to context-dependent objects. The context information is not separated from context-dependent objects. In this kind of systems no contextual objects exist explicitly. The context information is rather scattered in context-dependent objects. The behavior of context-dependent objects will be modified, so that it can behave correctly in the new context. Normally, there will be one or more IF-THEN clauses inserted in the context-dependent object behaviors to define context information.

Our context-based system, the PVE, supports *context definition by configuration*. The PVE system contains a repository of primitives. The users can define a new context by composing these primitives. The context-supporting mechanism of the system will be explained in the section 6.

5 Context modeling

5.1 Context modeling in analysis

There are many approaches proposed for modeling contexts in the analysis.

In [Pascoe 1998], the author finds out *that "context is a subjective concept that is defined by the entity that perceives it"*. It was proposed to model *context "as the subset of physical and conceptual states of interest to a particular entity"*.

The CIS (contextual information service) model in [Pascoe 1998] consists of a world that is comprised of artifacts. These artifacts in turn consist of a name, type and set of contextual states. A CIS client can access any state of any artifact in the world or can add its own artifacts. The CIS also contains a state catalog that lists the various generic

states. Each state has one or more formats and types, methods to translate between them and a set of operations. These elements are only visible when adding a state to the catalog or augmenting an existing one.

In [Buffo 1996] context is only a unit which is composed of components and connections. Components are either objects or contexts. Connections are linking components and their context. The following figure shows a system modeling a chocolate factory. The factory consists of two contexts, each of them modeling a sub-factory. The first context is the production unit, which produces both covers and pralines and merges them to chocolates. The second context is the packing unit, which packs the chocolates into suitable boxes. This example illustrates the hierarchy of the context; "factory" encompasses "production unit" and "packing unit", both of them encompass some objects. Moreover one can see that connections are respecting the hierarchy.

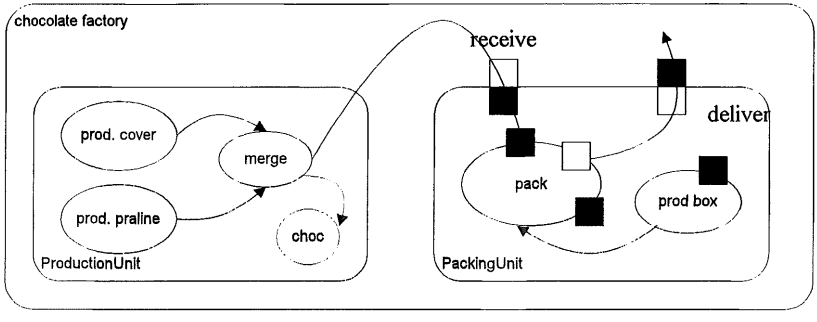


Figure 1: Context as simple container

The concept "context" in this work can be compared with "subject" of [Coad 1990]. Contexts are static. There is no way to define the context dynamically. Context in this sense is only a container of objects or further contexts. It is a concept, which is only of importance to the person viewing the system. The individual software entities are not able to react to changes regarding their contexts.

In [Berkem 1998] 'contextual objects' represent goals and contexts by object's contextual behaviors that express implicitly attributes, relationship and methods that depend on a goal and on a context. Modeling goals require that each activity inside a process step is conducted by a driver object. Contextual objects collaborate to realize the goal whenever a driver object enters into one of its lifecycle stages. For example an 'order' incites other objects such as a product, customer, delivery, invoice etc. to react depending on the context.

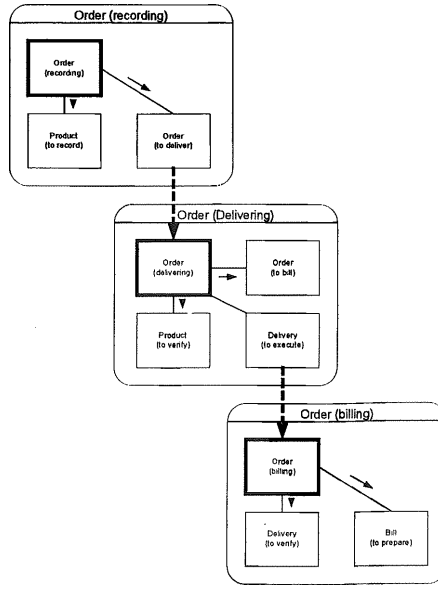


Figure 2: Contextual and context-dependent objects

This modeling approach emphasizes behavioral difference of objects, which pass through different processes. These processes are modeled as contextual objects. Contextual objects in this sense are container of driver objects, which behave differently in another container. Using this approach, context can only be modeled by collaboration of objects. The object behaviors contain context information, which lead to some difficulties by defining the new context and also leads to bad reusability (All objects which collaborate to define the new context must be modified again).

5.2 Context relation in design and implementation

In [Seiter et al 1996] the context relation in design and implementation was introduced. It directly supports behavior evolution. At the design level a new form of arrow between classes shows a context relation and at the implementation level a small extension of C++ was used. The paper demonstrates how the context relation can be used to easily program the Adapter, Bridge etc. patterns.

The context relation exists between classes supporting behavior evolution while maintaining the safety and performance benefits available with string typed, class-based languages. The context relation produces class hierarchies orthogonal to inheritance hierarchies, in many cases replacing inheritance. The basic idea is that if class C is context related to base class B, then B-objects can get their functionality dynamically

altered by C-objects. The language construct for doing this is a generalization of the method update in Abadi and Cardelli's imperative object calculus. A C-object may be explicitly attached to a B-object or it may be implicitly attached to a group of B-objects for the duration of a method invocation. This supports reuse of the class B because like inheritance, the class B does not know about class C. Unlike dynamic inheritance for B-objects, methods in the context class C override methods in the base class B. The relation supports dynamic behavior evolution because the C-object related to a B-object can vary at run-time.

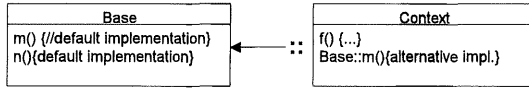


Figure 3: Separation of context-dependent and contextual objects

The figure above shows the relation between a base class and a context class. The context class does not inherit methods from the base class and is not considered a subclass of the base class. Rather the context class can define methods for its own class inheritances such as the f method as well as redefine method implementations for base class inheritances such as the m method. A context class may have multiple direct base classes. The context relation is drawn as an arrow between classes using the C++ scope operator :: to emphasize its role of altering the base class implementation.

The authors of the paper claimed that by adding context relations to the design and implementation vocabulary many known design patterns become unnecessary.

The strategy pattern allows an algorithm to be encapsulated as a construct and supports a mechanism for varying parts of the algorithm. Thus a strategy allows multiple implementations to be defined for a single interface. For this purpose the pattern uses the aggregation and inheritance relations. A Receiver class has a method m. A Strategy class hierarchy is defined to provide alternative implementations of the method m with each concrete strategy subclass representing a particular variation. An instance of the receiver class will reference a strategy object. The implementation of the m method for the receiver class simply delegates the request to its strategy, passing its self-reference along as an argument. The m method defined in the concrete strategy is executed to perform the correct variation of the algorithm for the receiver object.

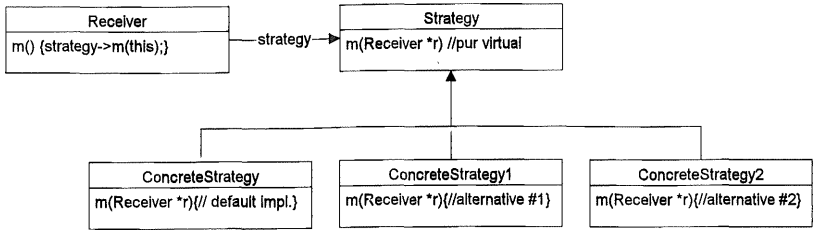


Figure 4: Strategy pattern

The following figure describes how to model the Strategy pattern using context relations. The context relation may be labeled with the name of the method being redefined to facilitate then diagram understanding. Notice there no longer exists the abstract class Strategy, as required in the pattern described by [Gamma 1994].

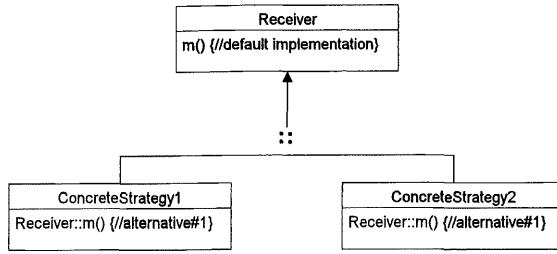


Figure 5: Context relation in strategy pattern

In [Seiter 1996] the mapping of context relations to run-time behaviors was proposed. In the traditional static class model each object conceptually has *one context* defining its rune-time behavior namely its static class implementation.

In the proposed run-time behavior model if class C is context related to base class B it defines alternative implementations for the base class methods. Instantiating class C produces a context object which may *be* used to dynamically alter class B. Attaching the C context object to a method invocation causes the static B class definition to be dynamically composed with the method updates given in class C, thus producing a dynamic B class definition. The basic approach of the implementation relies on context tables which are a dynamic version of static virtual function tables.

C++ extension	Purpose
o::=c	Context object c attached to object o, dynamic class of o ~ static class of o composed with context c
o::= +c	Context object c attached to object o, dynamic class of o ~ dynamic class of o composed with context c

m() ::+ {c}	Context object c attached to method m invocation
{+c}	incremental composition of methods (versus overriding}
context	meta-variable referencing current context object
Class C {B::m(){...}}	method update, C redefines B's m method

The context modeling of [Seiter 1996] is very useful in the design and implementation. Unfortunately the application of context in the analysis was not mentioned. The approach allows the definition of contextual object at the coding time. The context can be coded as needed. The base class does not know about the context class. This property of the approach allows the flexible definition and use of the context. This approach does not allow the definition of context by configuring but only by coding.

6 Context support within the PVE

One of the primary goals of the PVE is to support the cooperation processes within the virtual enterprises. The process "Virtual Enterprise Formation" is a typical cooperation process and can be used to demonstrate the context support within the PVE. This process is started by a member A of the community who has identified a market opportunity. However A's competencies are not sufficient to satisfy the market opportunity. Hence partners must be searched and their competencies need to be integrated into a single value chain by himself/herself. After A has selected partner candidates using a profile databases he/she sends a "Request for Participation in a VE" to potential partners (B, C,...). Following messages (i.e. "Request for more information" or "More information") may be exchanged between the partners. This process may result in the foundation of a virtual enterprise with the participant of A, B, C etc. Figure 6 depicts the process "Virtual Enterprise Foundation".

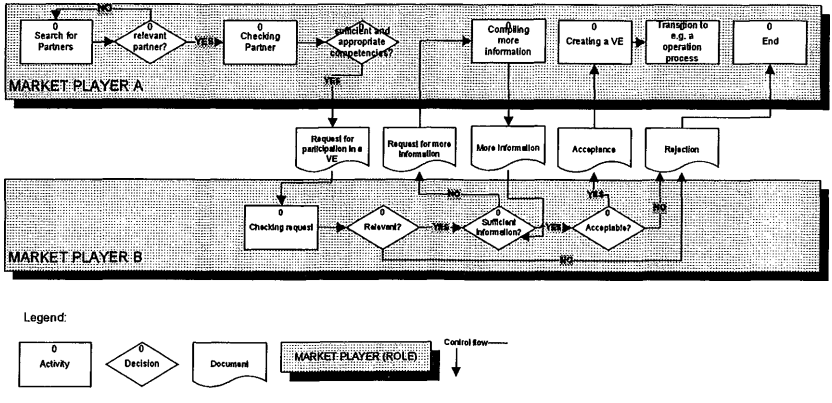


Figure 6: Process "Virtual Enterprise Foundation"

This process consists of activities performed by the one (A) or the other partner (B). A sequence of these activities is associated with a communication act by which a message is sent from one to another partner. The sequences of those communication acts have to be supported. Typically within a communication act a message is transported from one partner to another using a communication service. We support these communication acts (and their sequences) by associating contextual information to this message and providing a tool to handle this message. This contextual information provides the PVE with additional data on the basis of which it can perform certain action. These actions need also to be defined within the process description. Hence we denote this coupling of messages and the communication tools handling it as *context-based communication.*

In the context-based system of the PVE processes (e.g. "VE Formation", "Contacting") can be specified in sub-processes (e.g. "Request for Participation in a VE", "Acceptance"). Processes are presented by contextual objects. Hence we define the context "VE-Formation" which in turn contains several sub-contexts e.g. "Request for Participation in a VE", "Request for more Information". Each sub-context represents an individual type of communication act within the process.

The context-dependent objects (in [Berkem 1998] driver objects) are objects which represent the communication tool and its surrounding objects e.g. "Mail-Service", "Virtual Enterprise" and "Virtual Enterprise Space". Their behavior in the context "Information Request" is different in the context "Acceptance" and is specified in the context definition. A context will be defined by composing primitives from the repository in the context definition phase. In the execution phase if the objects "Mail-Service", "Virtual Enterprise" etc. belong to the context "Acceptance" and a message arriving the Mail-Service object confirms that a Market player wants to join a virtual enterprise, then the Mail-Service object creates the Virtual Enterprise object. The Mail-Service object would not behave so if it were dependent of the context "Information Request".

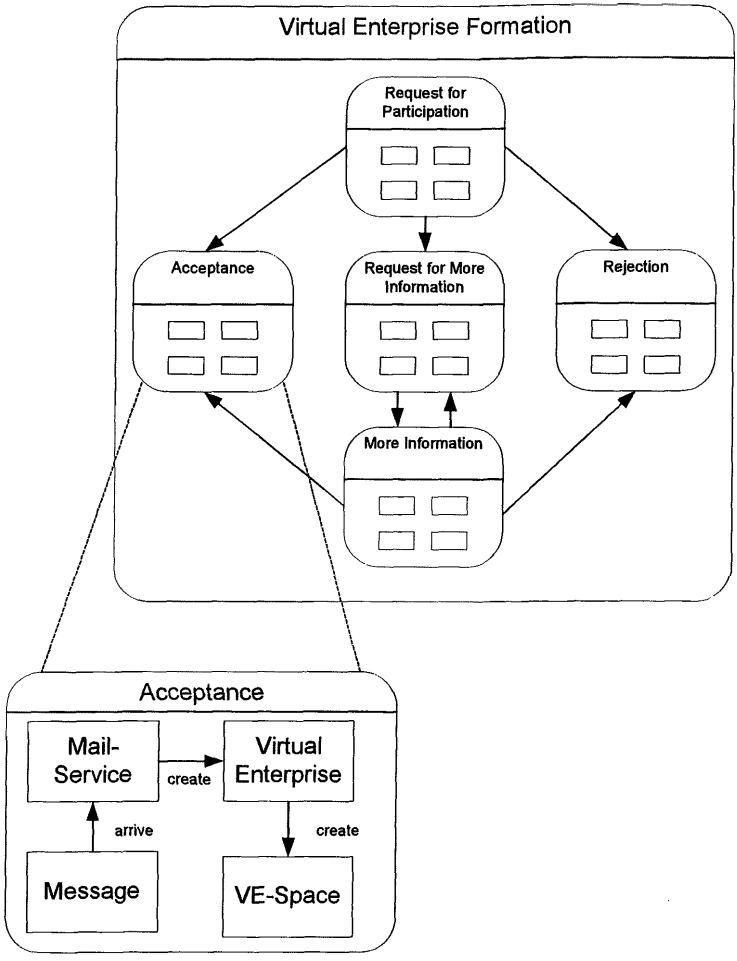


Figure 7: Context and Context-dependent Objects in VE Formation

$Process_{VE\text{-}Formation}$= {(RPVe, RmoreI), (RPVe, Reject), (RPVe, Accept), (RMoreI, MoreI), (MoreI, RmoreI), (MoreI, Reject), (MoreI, Accept)}

With

RPVe = "Request for Participation in a VE"

Rmore = "Request for More Information"

MoreI = "More Information"

Accept= "Acceptance"

Reject = "Rejection"

Individual sub-contexts can be provided with the signature of an action, which will be performed upon receiving or sending a message associated with this sub-context. For example, if a message is received within the sub-context "Acceptance" the action "form a VE" will be performed. Generally a sub-context is provided with an action which is performed upon receiving a message associated with it. This process description is transformed into a format which can be understood by a repository (e.g. a relational database).

In order to support context-based communication the PVE forms a message by attaching the contextual object with the user-generated content and send it to the receiver. Once the message has arrived the receiver can react according to the sub-context attached to the message. This sub-context determines which sub-contexts are available to send them back to the sender. For example if a "Request for Participation in a VE" arrives this message can only be answered by "Request for More Information", "Acceptance" or "Rejection" (see Figure 7).

While contexts and sub-contexts support the communication within the PVE, extra-context information can be used to enhance the convenience of user interaction. The following section will introduce this phenomenon and suggest how to use it in platforms for virtual enterprises.

7 Extra-context logic

In the requirement documentation software engineers get the description of the user's work practice. This description contains usually not only context but also extra-context information. The concept "extra-context" was introduced in [Bender 1998] and was used to describe interactions between actors, which happen out of the system boundary. [Bender 1998] discusses the relation of strategies and organizational structure to extra-context. Unfortunately the paper does not show clearly why and how we should model the extra-context logic.

The challenge of development of a system is to keep it coherent to anther systems, so that it supports the users and fits their expectations while extending and transforming their work practice as prescribed by the vision. Coherence isn't just about the consistency of the user interface – a coherent system keeps the user's work orderly and natural. Keeping a system coherent is hard enough when it's one user doing a single task. It's even worse in real systems which support multiple people playing multiple roles across the whole business while using several systems. Typically the users work with multiple systems each designed to solve a single problem. These systems don't address the user's whole job and don't attempt to make the work fit together across the different tools. When work practice is too large and complex to see or it's too hard to

address all at once, it's easier to write simple systems that address single problems. But then the systems chop up the work and leave it up to the user to put it back together by taking extra steps or doing additional work on the side. The logic behind these steps is the extra-context logic. It shows the coherence between different tools or applications.
We use the VE Formation process to demonstrate the phenomenon extra-context. After receiving a message with the sub-context "Acceptance" from market player B it is likely that market player A starts a budget planning tool to check financial issues or a Web browser to check the local tax law. Moreover one could imagine that a meeting has to be summoned using a communication system internal to A after a "Rejection" is received". However these steps *do not need* to be performed. The execution of these steps is optional and depends on "soft" factors such as the mood of the users, their knowledge base, their cultural background or "harder" factors like organizational constraints. This assumption is an example of extra-context information, i.e. information about concepts and their relationships, which are outside the actual system but nevertheless influence the execution of processes within the system. Figure shows the extra-context relation between PVE and other external applications such as a budget planning tool or an internal communication tool.

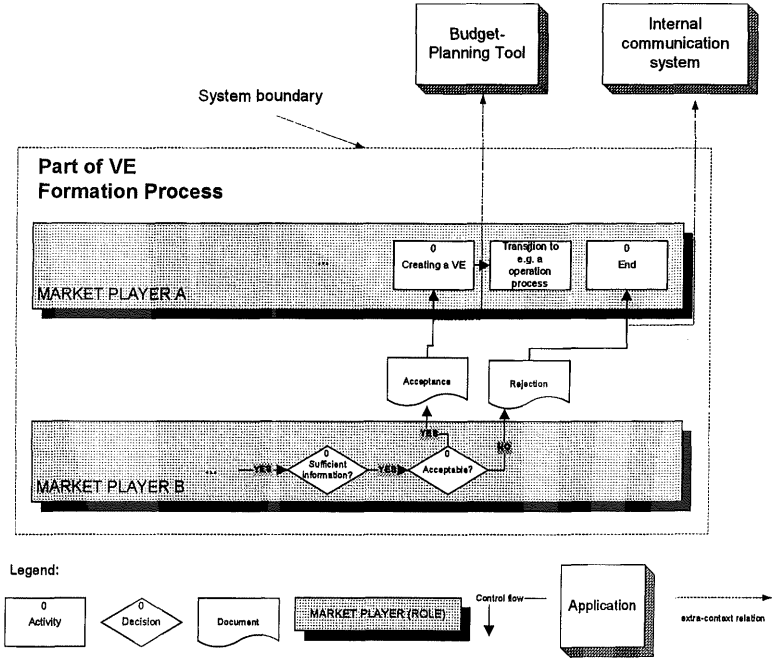


Figure 8: Extra-context logic between PVE and external applications

The difference between context and extra-context is that contextual information is needed to support processes which are executed within PVE. Supporting extra-context allows the combination of both processes internal to PVE and procedures external to this application.

We defined extra-context as the environment of application actors (users who execute the application, equipment, which is connected to the application etc.) while the context of the application represents the environment of the application itself and consists of the interaction between the application and its actors. The extra-context has the following characteristics:

- It depends on "soft factors" (emotion of users, knowledge base or cultural background).
- It is based on assumptions.
- It affects indirectly the user interactions.
- It affects the usability of a system, i.e. if at al and how the system is used.

The extra-context information of the system requirement has been ignored until now. It is correct that such information relies only on assumption and could not be transformed in application logic, because such transformation restricts the user interaction. (You can image how upset the user would get when he doesn't want to execute some steps but got nevertheless the step executed, e.g. starting the browser). We claim that such information is very useful and can be used for instance by a system wizard (or a system assistant), which suggests the users what to do next or allows the integration of cooperation processes to their internal processes.

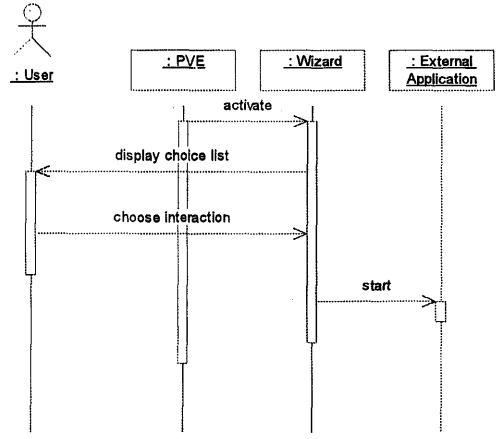


Figure 9: User interaction support according to extra-context information

The idea is to associate a list of possible external applications, which may be performed once a certain context, sub-context combination occurs. A wizard (system assistant) displays this list and the user having received the respective message can decide whether or not to start the application suggested.
We can derive extra-context logic from requirement documents and define the extra-context relation as tuples of contexts, sub-contexts, conditions and external applications, which are likely to be started. Such relations can be saved in an extra-context repository. At run-time the PVE activates the wizard if a received message is associated to a certain context with the condition being met.

8 Conclusion

This paper has made an overview of context modeling in software engineering. It has introduced a context-based approach to support processes within virtual enterprises. The context-based approach was incorporated into a prototype of a generic platform for virtual enterprises developed at the Dresden University of Technology. It remains to see if the context-based features of the prototype are flexible enough to be effectively used in the practice.
The phenomenon of extra-context logic has been discussed. We have proposed to use extra-context information to integrate external applications in a PVE and also to enhance the convenience for user interaction with the help of a wizard which tells the users what he/she should do next.

References

Bender, K.; Homann, J.(1998): Extra-kontext und Applikationslogik in Anwendungssystemen zur Unterstüzung virtueller Gemeinschaften. In Engelien, M. and Bender, K., eds., GeNeMe98: Gemeinschaften in den Neuen Medien, pages 23-36. TU Dresden, Josef Eul Verlag.

Berkem, B.(1998): 'Contextual Objects' or Goal Orientation for Bussiness Process Modeling. In ECOOP98-Object Oriented Technology, LNCS 1543.

Booch, G.(1994): Object-Oriented Analysis and Design with Applications. Benjamin Cummings, Redwood City, 2nd edition.

Buffo, M.; Buchs, D.(1996): Contextual Coordination between Objects. In Maldonado, J. and Masiero, P.C., eds, X SBES Brazilian Symposium on Software Engineering, Brazil, pages 341-356

Coad, P.; Yourdon, E.(1990): Object-Oriented Analysis. Computing Series. Yourdon Press, Englewood Cliffs, N.J.

Crystal, D.(1991): A dictionary of linguistics and phonetics. 3rd edition. Blackwell, Oxford UK

Guha, R.(1995): A formalization and Some Application. Ph.D. Thesis, Stanford University

Kashyap, V.; Sheth, A. (1996): Semantic and schemantic similarities between database objects: a context-based approach. The VLDB Journal 5, pages 276-304

Leech, G.(1981): Semantics: the study of meaning. 2nd edition. Penguin, Harmondsworth UK

Pascoe, J.(1998): Adding generic contextual capabilities to wearable computers. In The Second International Symposium on Wearable Computers, pages 92-99, Pittsburgh, October 1998. IEEE Computer Society

Schilit,B.N; Adams, N.; Want, R.(1994): Context-Aware Computing Applications. IEEE Workshop on Mobile Computing Systems and Applications, December 1994.

Schilit,B.N.(1995): A System Architecture for Context-Aware Mobile Computing. PhD thesis. http://sanbox.parc.xerox.com/parctab/

Sciore, E; Siegel, M; Rosentahl, A.(1992): Context interchange using meta-attributes. In: Int. Conf. On Information and Knowledge Management, Proceedings of the CIKM, Baltimore.

Seiter, L; Palsberg, J; Lieberherr, K.(1998): Evolution of Object Behavior using Context Relations. IIIE Transactions on Software Engineering, 24(1):pages 79-92