An Extended Model for Real-Time Systems

Dominikus Herzberg
Ericsson Eurolab Deutschland GmbH
Ericsson Allee 1
52134 Herzogenrath, Germany
Dominikus.Herzberg@eed.ericsson.se

André Marburger
Department of Computer Science III
Technical University of Aachen
52074 Aachen, Germany
marand@i3.informatik.rwth-aachen.de

SCI/ISAS 2000

Abstract

Many systems are or include real-time systems, the range spans from embedded systems up to large, complex systems like nodes and devices in telecommunication networks. For the purpose of modeling and designing such systems, a more thorough understanding of the generic structure of real-time systems is required. Unfortunately, there is no standardized definition of real-time systems in the literature nor have any models of real-time systems been discussed in depth. This report tries to fill this gap and extends the generic model for real-time systems. Most notable is the distinction of a *real-time system* versus a *technical system*, the introduction of *reference points*, and the idea of a *process domain model*. This contributes to a more refined view on real-time systems and their surrounding environment. In addition, some consequences for the software architecture of real-time systems are discussed.

Keywords
real-time systems, embedded systems, real-time modeling

1 Introduction

The use and the understanding of the term “real-time system” is not consistent in the literature. It is a mixture of characterizing attributes and structural properties of a system. For example: On one hand it is said that a real-time system fulfills timing constraints, i.e. a real-time system has to react to a stimulus in a certain time frame; in this example the guaranteed response time is an attribute, which characterizes a real-time system. On the other hand, real-time systems are often classified as “embedded systems”. An embedded system is seen as a specific part of a larger system, which is a structural aspect. Besides this lack of clarity in terminology, there is not even common agreement on the word “real-time”. The following bullets summarize findings from studying the literature:

- Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time in which the results are produced [10]. After more than a decade this definition still seems to be the greatest common denominator. Here, “real-time” is an attribute to “system”. Because of their specific field of application, further attributes are usually associated with real-time systems. Included in this category are e.g. reliability, fault tolerance, adaptability, and speed [13].

- The most popular classification is the distinction in *hard* and *soft* real-time systems. Hard real-time systems are under deadline constraints. Passing a deadline is considered unacceptable. A soft real-time system retains some tasks, which are still valuable for execution even if they miss their deadlines [13]. Following [8] telephony systems belong to the class of soft real-time systems: passing of deadlines is accepted as long as the number of failures is below a defined threshold. While this might be true in general there are other components in telecommunication networks, which have to fulfill hard real-time constraints. For example, the time delay perceived as acceptable for voice transmission in a speech conversation places tough time limitations on a mobile phone for speech en- and decoding including cyphering and channel coding.

- A very rudimentary structure of the basic elements of a real-time system is given by [8]: it consists of hardware, namely *sensors* and *effectors*, the environment, and software. The sensors and effectors interact with the environment, the software controls the actions of the hardware via a hardware interface. A similar description using different terminology can be found in [11]: A real-time system consists of a controlling and a controlled system. The controlling system interacts with its environment using information about the environment available from various sensors and activating elements in the environment through “actuators”. The controlled system can be viewed as the environment with which the computer interacts.

- There is a loose reasoning as to why timing aspects and structural issues of a real-time system are related: Timing correctness requirements arise because of the physical impact of the controlling systems' activities upon its environment. That means that the environment needs to be monitored periodically as well as that sensed information needs to be processed in time [13]. This implies that we have to distinguish the environment from a controlling part and that detecting and acting devices are needed.

- The definition of an embedded system is vague; it mainly describes a structural aspect. In its most general form, an embedded system is simply a computer system hidden in a technical product [9]. A more concrete definition is, most embedded systems consist of a small microcontroller and limited software situated within e.g. an automobile or a video recorder [12]. Three issues seem to be important here: (1) size matters, (2) an embedded system is part of a technical system, and (3) it serves the purpose of the technical system and not vice versa. Issue (3) especially helps deliminate non-embedded systems from embedded systems. A PC (Personal Computer) for instance is a general purpose computing machine, the software and the CPU (Central Processing Unit) are an integral part of it. This eliminates a PC from being an embedded system. A counterexample might be a mobile phone. The DSP (Digital Signal Processing) chip and its software serve a single purpose: to offer phone functionality.

- Embedded systems may or may not have real-time constraints [11], but many real-time systems are embedded systems [12].

To conclude: The special character of systems, which have a physical impact on the "real" world by means of reactiveness is most significantly described by the requirement on the timing constraints to be met by the system. Such systems are called *real-time systems.* Additional properties, which reflect other aspects of the physical impact character, include reliability, fault tolerance, stability, safety and so on. As yet there is no commonly agreed list of properties, which constitute a real-time system. Moreover, the physical impact nature of such systems implies a rough structure: a controlling part interacting with the environment (the controlled part) through sensors and effectors. Many real-time systems are embedded systems, that means they serve a specific purpose in a technical system.

The model identified so far is not specific enough to allow any detailed insight into the nature of real-time systems, especially for structural properties. There is a need to refine this view. A system theory oriented approach should help gain further insight into structural properties. An extended model of real-time systems will help improve the understanding of such systems from an architectural point of view. In section 2, an extended model is comprehensively presented. In section 3, consequences on the software architecture of a real-time system are discussed.

2 An Extended Model for Real-Time Systems

2.1 A Discussion about Extensions

Figure 1 shows an extended model for real-time systems. Note that the introduction of a system called *technical system* and the intrinsic motivated association of the software of the controlling system with the controlled system are both novel aspects.[1] As the reader will notice, the following discussion of figure 1 is influenced by a system theory approach. To help with understanding, a real-life example has been taken: The technical system "car" has a real-time system measuring the speed of the car and displaying the calculated speed on a digital display. The sensor for the speedometer has been introduced to continuously control the functioning of the device as it is regarded as safety critical. In the example, "environment" has been reduced to "street" and "user" to "driver".

- The basic elements of a real-time model still remain valid: it consists of a controlled system and a controlling system. Sensors and effectors classify two types of devices, which reflect relations of impact. Sensors communicate events of the controlled system to the controlling system, whereas effectors do the opposite.

- The model tries to separate structural issues from system attributes. For example, the decomposition of a system in a controlled and a controlling system is a structural property. Reactiveness under specific time constraints is an attribute, which leads to the classification *real-time.* As previously mentioned, further properties might refine the attribute "real-time". This is indicated in figure 1 by some "attributing" text.

- The box labeled with "HW" condenses the hardware required to establish an interface between the sensors/effectors and the software; it could be, for example, an analogue/digital (A/D) converter. The computing platform as well as other hardware devices are neglected but can be subsumed under the "HW" box if needs be. Note that we cannot make any assumptions about the hardware structure. We do not know if a pair of sensors/effectors is associated with a single A/D converter or not.

[1] The model was very much inspired by material from Thomas Muth (UAB, Ericsson).

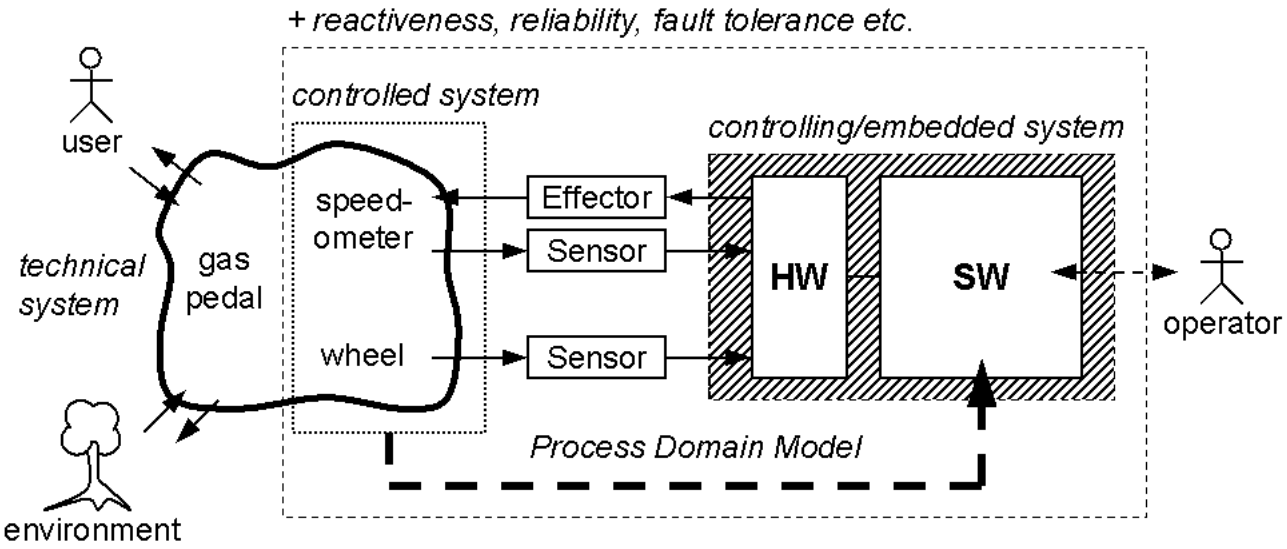

Figure 1: An extended model for real-time systems

- Sensors and effectors could be interpreted as exporting, respectively importing reference points in the controlled system. Consequently, sensors and effectors belong to the controlled system. This fact is not sufficiently reflected in figure 1.
- Reference points are associated with an entity in the controlled system. An *exporting* reference point is a defined exit point of matter, information, or energy from the technical system to the external world. The intention is to allow the state of the external world to be influenced in a specific way. For example, the wheels of a car on a lifting platform fail their intention of moving the car forward. An *importing reference* point is a defined entry point of matter, information, or energy from the external world to the technical system, which allows the state of the technical system to be changed.
- Reference points are an intellectual concept and denote only those entry/exit points, which are relevant to the technical system, others are neglected.
- Physically, a transfer of matter, information, or energy is always equivalent to a change of state; it is experienced as a loss or gain of matter, information, or energy. Even though every technical system is physical, the point is that a technical system is a construction that is sensitive to some physical changes in state at its reference points and while remaining relatively insensitive to others. A sensor, for example, ideally is a technically state insensitive reference point; it should transfer information about the technical system's internals without changing a technically sensitive state. In this sense, a sensor simply observes the controlled system and reports its observations to the controlling system. In contrast, an effector should have impact on a technically sensitive state.
- The controlled system is not equal to the environment, it covers just a part of it as there are only technical entities with associated reference points (sensors and effectors). This view is fundamentally different to the traditional one of real-time systems. What is new is the introduction of a *technical system.*
- The technical system is composed of technical entities. Many of these entities have an impact on or are in connection with the physical world, namely the environment. The relation between technical entities is defined by the technical system. These relations might be mechanical, electromagnetic, chemical, etc. Additionally, the technical system might interact with a user. A system, which has no user interface, is called *autonomous.*
- The interaction between the technical system and its environment or a user is also via reference points in the technical system.
- Note that the definition of a real-time system still includes the controlled system. The controlled system including reference points (sensors/effectors) is the intersection of the technical and the real-time system.
- In practice, a real-time system is usually seen as a component of a technical system. For example, it would be inconceivable for today's automobiles not to have computers controlling the fuel-injection engine. According to our view, the controlling system resides outside the technical system because reference points mark interfaces to the external world

and hide implementation details. Interfaces are some kind of ports to the outside and reflect “expectations” of the external world; it is insignificant as to how the external world looks. For example: The technical component, “engine”, demands specific synchronizing conditions on fuel-injection, which are formulated as requirements. Maybe the engineers come up with a mechanical solution or they decide to define suitable reference points for sensors and effectors and control fuel-injection by a computing device.[2] Once a specific solution is established, it is typically regarded as a technical component of the system. This ultimately blurs the system’s borders, which conceptionally still exist. Here, we will stick to a clearer conceptional view.

- To summarize: (1) The controlling system, users, and the environment are external to the technical system; the distinction made is a classification of the technical system’s external world. (2) Reference points are associated with technical entities and point out interfaces to the external world. (3) Reference points are either importing or exporting. Exporting reference points can be sensitive or insensitive to the technical state of the technical system.[3]

- Every transactional system that aims to influence its environment has to be aware of the current state of the environment [1].[4] However, a pure representation of states does not help unless one knows how the states are related and influence each other. In other words, a domain model is needed, or – according to [1] – a so-called Process Domain Model (PDM). A PDM does not only define the static structure of a domain (a so-called Structural Domain Model, SDM), it additionally takes into consideration the processing dimension.

- It is important to see that one only needs to have a PDM of the controlled system and not of the whole technical system. This is also new and in contrast to traditional views on real-time systems. For the purpose of the real-time system, there is no need to completely understand the technical system, its entities and their relations; however, it is essential to understand the function of the controlled system. In the example in figure 1, it is of no interest in regard to the real-time system that pressing the gas pedal has an effect on the engine, which in turn has an effect on the wheels. The only thing that must be understood by the real-time system is, how the entities of the controlled system i.e. the wheel, the speedometer, and the sensors and effectors function. In other words, the software in the controlling system must have a built in accurate representation (model) of the controlled system. Formally speaking, there exists an association between the software of the controlling system and the controlled system by means of a PDM.

- The interesting consequence is that we have an indication of a structural property of the software: it must somehow reflect the domain it is controlling. The question is *how* and to which extend the PDM structures the software. In some cases, formulas might model the controlled system, in others a set of context rules might provide an alternative. More complicated cases require data structures and models, which reflect properties and relations of the controlled domain. We can state the hypothesis that the more complex the controlled system and the task of controlling it is, the more evident the relation of the software internals (data structures, software architecture) and the controlled system structure (the PDM) must be. In an extreme scenario the software mirrors the controlled system structure in a one-to-one manner. In any case, the software specifies a model of the controlled system!

- To finalize the discussion note that another type of user, who has nothing to do with the technical system at all, might interact with the controlling system. This user is called “operator”. Operation and maintenance are two examples of activities performed by an operator.

- The classification of the controlling system as an embedded system is of no value for this discussion.

2.2 A More Formal View

Following this discussion, figure 2 now shows the same facts in a more formal way, and gives an idea of how a notation for the concepts introduced may look like; only the operator is removed. Entities of the technical system are symbolized by boxes. Reference points are attached as triangles. Importing reference points are inside the box, exporting outside. Solid lines denote associations. Reference points denote also the direction of how to read an association if a name is attached. For other associations, a small arrow might indicate the direction of reading. For example, the speedometer *displays* information to the driver, the driver *presses* the gas pedal, the PDM *describes* the controlled system. The association between the controlled and the controlling system is designated by the PDM box. Closed dashed lines group entities and symbolize system borders. The question mark in the middle of the technical system indicates that the model does not describe the relation of the technical

[2] In fact, the topic is a bit tricky: a mechanical solution requires different kinds of reference points compared to a computer controlled solution. Ergo, the definition of reference points marks a decision that is a constraint on the solution space. The point is that reference points for a real-time system shift domain! Sensors and effectors mediate between the states of e.g. a mechanical or chemical domain and the software domain of a controlling system. We will come back to this point later on.

[3] A state insensitive importing reference point is of no value; according to the definition, a reference point is of relevance to the technical system. Ergo, state insensitive importing reference points do not exist.

[4] A transactional computation depends on the state of the environment in which it is carried out [1].

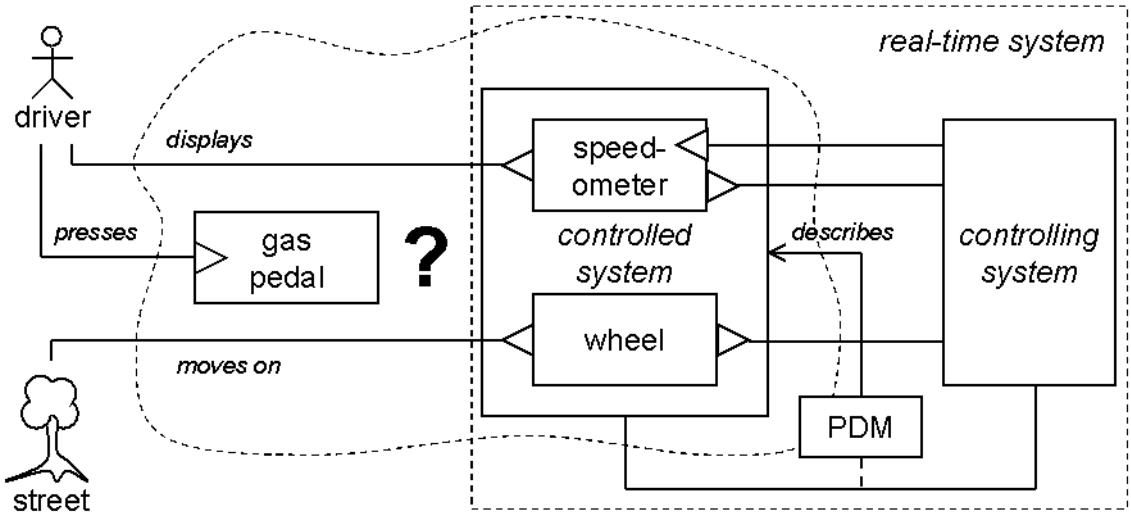

Figure 2: A formal view on an extended model for real-time systems

entities nor that the identification of entities outside the controlled system is complete. Note that the model of the controlling system is incomplete as well but it is described by the PDM.

2.3 Collaborating Distributed Real-Time Systems

In particular, telecommunication systems are characterized by a network of distributed collaborating units. Most of these units fall under the class of real-time systems. Therefore it is of special interest to discuss some consequences if two or more real-time systems modeled according to figure 1 collaborate and communicate to each other. In figure 3 some elements of a real-time system model are removed for reasons of simplicity.

- Two technical systems can *physically* only interact via two or more technical entities. In principle, this can be any entity in the upper or lower technical system in figure 3.
- Communication is established via an external connecting entity. This entity is usually part of the environment. Two phones could be, for example, either connected by a fiber optics cable or, in the case of mobile phones, by the “air” (electromagnetic waves). Instead of an external entity, a shared technical entity might also be an option, but this blurs the system’s borders. To be conceptionally clear, such a shared resource should be moved out and be defined as external, while replacment entities in the technical systems have to be introduced if required.
- Once more we can define reference points associated with technical entities, which adapt a technical entity to an external entity. The “space” between reference points in one technical system and their corresponding reference points in another technical system is called *interface*. Every interface is associated with an external entity. The number of external entities addressed by a set of corresponding reference points determines the number of interfaces.
- Besides this physical communication relation, one can abstract the communication between technical entities as a communication flow between two (or more) technical systems, indicated by the dashed arrow next to the solid arrow (in figure 3). This abstraction technique is well known in the telecommunication domain; the most prominent example is the OSI (Open Systems Interconnection) reference model [5]. One usually calls the set of messages and the rules of communication a *protocol.*
- On a user or application layer, one can abstract the communication relation even further. On this level, users can experience virtually direct communication (via a protocol) even though several technical systems might be involved. This fact does not adequately come across in figure 3.

To conclude: The title of this subsection is a bit misleading: just on a physical level, technical systems communicate and collaborate with each other. The real-time system is involved insofar as technical entities of the controlled system are involved. Currently, it is unclear if all communicating entities of a telecommunication system belong to the controlled system.

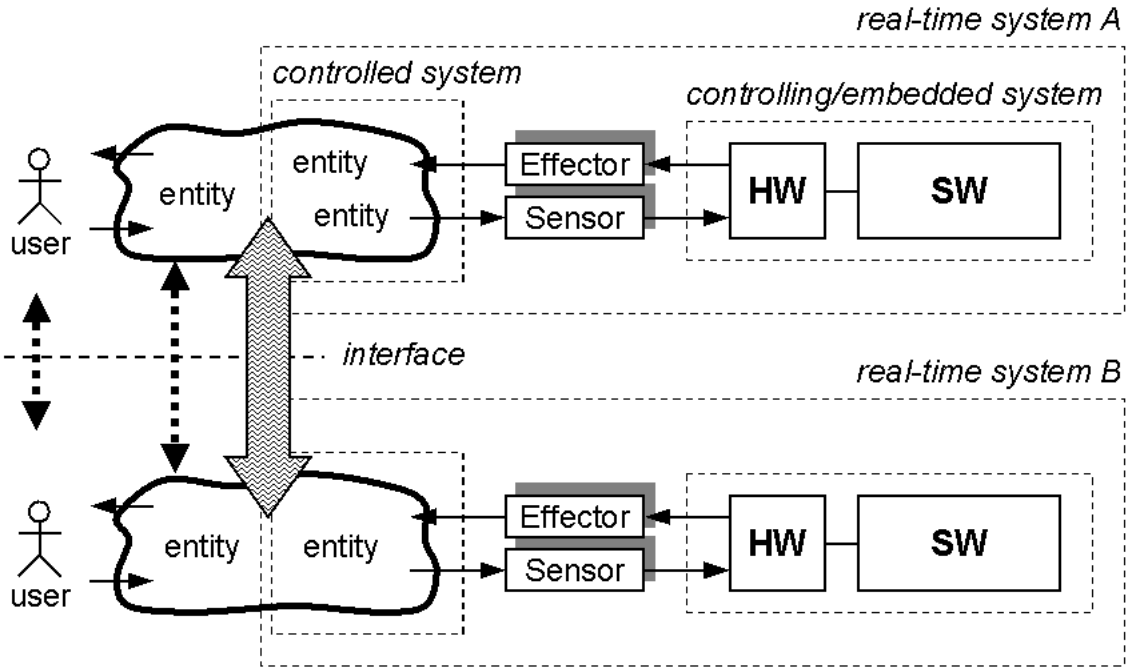

Figure 3: The collaboration of distributed real-time systems

3 The Software Architecture of Real-Time Systems

The discussion showed that the PDM has an impact on the software architecture of a real-time system. The question is open as to how and to which extend the PDM has impact on the software. The hypothesis, that the complexity of the controlled system and the difficulty in controlling it most probably lead to a close relation of the PDM and the software structure, is based on empirical insights in the telecommunication domain. In telecommunications the technical system and its architecture are usually specified in so-called standards. Studying the software of Ericsson's MSC (Mobile Switching Centre), the AXE10, which is specified by the GSM standard, one finds close mapping of the technical architecture and the software structure. This mapping also seems to be required to intellectually master the challenge to continuously adapt and modify the system according to changing and evolving standards. Consequently, the PDM seems to be an important part in the process of modeling real-time systems. Further studies in this area are needed.

One could also gain some insights about the software structure by investigating the topology of real-time systems. The idea is to model the software as a graph, which decomposes any system in a set of vertices and connecting edges, and measures some structural properties, which are dependened on some parameters. In general, a vertex represents an atomic computational unit, which is self-contained in the sense that it has its own local memory and a finite set of states. Communication relationships between such computational units are represented by edges. Note that an edge is bidirectional, it allows two-way communication. The point is that any edge must not exceed a fixed time limit and that this time limit is significantly below the time required to execute a specific functionality. This introduces hard real-time constraints in to the system and enforces communication and cooperation in the software "network". In view of WATTS' work [14], two extremes of networks can be considered: regular networks and random networks. Regular networks are highly clustered whereas random networks have a short characteristic path length. Rewiring only a few edges in a regular graph in a random fashion results in a dramatic cut in the characteristic path length but retains the clustering: exactly these properties characterize a so-called "small-world" graph. In other words, we can expect that the software structure of real-time systems is a sort of "small-world" structure: Clusters enable fast execution of a specific functionality (no time is wasted by going via non-contributing edges) whereas a short characteristic path length enables to quickly change functional context to another cluster. The empirical proof of this hypothesis is outstanding.

Besides these structural insights into real-time systems other structural impacts have to be considered as well, which might even dominate the organization of the software. Large scale software systems especially tend to structurally degrade over time. Properties like maintainability, extendability, adaptability, flexibility, openness etc. aim at decreasing the software entropy; a proper software structure has to be established. Today's solution to this problem are component-oriented frameworks and request broker architectures [4]. The problem concerning real-time systems is such that an architectural "superstructure" might decrease real-time characteristics. This is not a trivial challenge to software engineers, specially when re-engineering legacy real-time systems.

4 Summary

Traditionally, the controlled part of a real-time system is seen as the environment of the real-time system. The model presented refines this view and introduces the system borders of a technical system consisting of technical entities. The following insights are therefore essential: (a) the controlled system is only part of the technical system and (b) the controlled system has an structural impact on the software of the controlling system, which is characterized by a Process Domain Model (PDM). Reference points as introduced, fulfill the function of connecting entities of different contexts (or domains, to use another word). They represent a first-class concept, meaning that they do not solve the problem of how this connection, the transfer of matter, information, or energy, is achieved. A general discussion about shared phenomena and connection domains can be found in [6, 7]. The discussion about the impacts on the software architecture of real-time systems showed that requirements outside the scope of real-time properties also have an influence, especially for large scale software systems.

The results presented might not only contribute to the research of real-time systems [12] but also to research areas in conceptual modeling, especially the relationships between the real world and the software world [3]. Advances in modeling the PDM are of particular interest here. Therefore, some investigations on real-life examples and how they correspond to the model presented are required.

References

[1] Alfs Berztiss. Contexts, Domains, and Software. In Bouquet et al. [2], pages 443–446.

[2] P. Bouquet, L. Serafini, P. Breézillon, M. Beuerecetti, and F. Castellani, editors. *Modeling and Using Context; Second International and Interdisciplinary Conference, CONTEXT'99, Trento, Italy, September, 1999*, number 1688 in Lecture Notes in Artificial Intelligence. Springer, 1999.

[3] Peter P. Chen, Bernhard Thalheim, and Leah Y. Wong. Future Directions of Conceptual Modeling. In Bouquet et al. [2], pages 287–301.

[4] Hanns-Helmuth Deubler. A Viable System Structure for Large-scale Software Systems. *Software – Practice and Experience*, 29(12):1025–1047, 1999.

[5] Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. ITU-T Recommendation X.200, International Telecommunication Union, July 1994.

[6] Michael Jackson. *Software Requirements and Specifications – a lexicon of practice, principles, and prejudices*. Addison-Wesley, 1995.

[7] Benjamin L. Kovitz. *Practical Software Requirements – A Manual of Content and Style*. Manning, Greenwich, 1999.

[8] Bran Selic, Garth Gullekson, and Paul T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, Inc., 1994.

[9] David E. Simon. *An Embedded Software Primer*. Addison-Wesley, 1999.

[10] John Stankovic. Misconceptions about real-time computing: A serious problem for next generation systems. *IEEE Computer*, 21(10):10–19, October 1988.

[11] John A. Stankovic. Real-time and embedded systems. *ACM Computing Surveys*, 28(1):205–208, March 1996.

[12] John A. Stankovic et al. Strategic directions in real-time and embedded systems. *ACM Computing Surveys*, 28(4):751–763, December 1996.

[13] Allen B. Tucker. *The Computer Science and Engineering Handbook*, chapter 78: Real-Time and Embedded Systems, pages 1709–1724. CRC Press, 1997.

[14] Duncan J. Watts. *Small Worlds – The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.