Temporal and Modal Logic Based Event Languages for the Development of Reactive Application Systems

Erich Ortner[1]
ortner@winf.tu-darmstadt.de
Tobias Schneider[1]
schneider@winf.tu-darmstadt.de

[1] Technische Universität Darmstadt, Development of Application Systems,
Hochschulstraße 1,
64287 Darmstadt, Germany

Abstract. The Future Internet is one of the key techniques to support the organizational processes in a reactive way. The Internet itself has developed from a mere data-centric organization into a platform for applications (services). To orchestrate the multiplicity of services in the Internet an event-driven and language-critical architecture is needed. Moreover a simple boolean logic just differentiating the states true and wrong is not sufficient to cope with the diversity of events. States of applications in the Future Internet are time dependent and logic dealing with these problems needs to be extended in the directions of temporal logic and modal logic. Moreover existing architectures have to be extended to be able to handle events effectively. Hence, the extension of service-oriented architectures to event-driven, reactive architectures is a necessary step. To perform this task IT-experts with an interdisciplinary background are needed.

Keywords: Event Modeling, Web 2.0, Temporal Logic, Modal Logic, Event-driven Architectures. Interdisciplinarity, Language-Critical Development of Application Systems

1 Introduction

The development from the *Internet of things* to the *Internet of events* proceeds with an amazing speed, not known before in the business world and also not in our everyday life. The Future Internet (FI) is one of the key technologies to support the organizational processes in every business. The Internet itself has developed from a mere data-centric organization into a platform for process-centric applications. This reflects the paradigm shift from data to organization in Computer Science [1].
Just about ten years ago the introduction of the email as the standard way of communication between business partners revolutionized the processes of making contracts and internal communication. For the first time ever, cheap communication technique in written form was available for the whole business world. Soon afterwards business went to the Internet and the first retail and business concepts with

Ebay and Amazon became available. During the recent years human beings and their role in the Internet and their relationship to it were in the centre of interest, driven by the possibilities to participate in developing content of the Internet via blogs, social networks (such as Facebook) or video portals (such as YouTube). This development became famous under the keyword Web 2.0.
The most recent development is to see the Internet itself as a platform for reactive applications (services) and not as an application itself anymore. The launch of Google Chrome, with the goal of replacing the operating System as standard platform for applications with the browser more than just a sign of this process.
To achieve successes in this very demanding new stage in the development of the Internet, collaboration between the following key factors is needed: technology, organization and human beings. This can only succeed in an enterprise if they are understood in a holistic way. The catch phrase "Total Application System Science" is already going the circuit internationally. But: Not the *Internet of things*, but the *Internet of events* (in the sense of – as far as possible – schematically organized, controlled processes such as important events) represents a central challenge to all enterprises, administrations and even our private lives (Figure 1) [2].

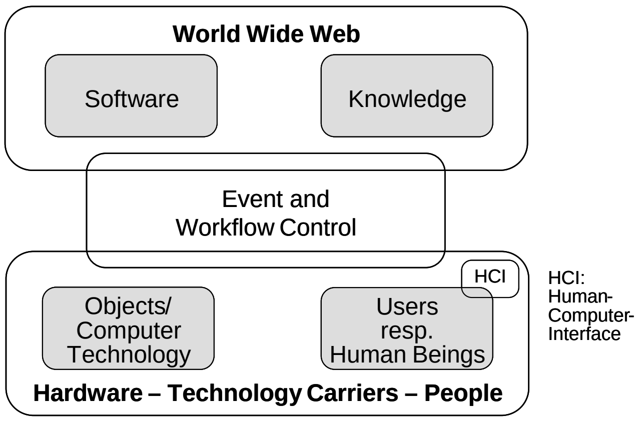

World Wide Web
Software
Knowledge
Event and Workflow Control
HCI
HCI: Human-Computer-Interface
Objects/ Computer Technology
Users resp. Human Beings
Hardware – Technology Carriers – People

Figure 1: From an "Internet of things" to an "Internet of events" following Ortner [2]

To master these challenges the nature of events has to be investigated with the approach of Wittgenstein who sees the world as the totality of events and not of things [3]. We propose in this paper an approach to characterize events and to deal with the complex events. Moreover this approach is taken to build an event-driven architecture for reactive application systems on a language-critical basis.

2 Complex Event Modeling

The idea to integrate events from the environment of an IT-system has been discussed a lot since their introduction [4]. Recently, an interesting approach to describe the interaction in an event-driven system using process algebras in the context of protocol modeling was published [5] .In the business world the modeling technique of event-driven process chains is still dominant [6]. So long, no scientific founded notation to describe the nature of the Future Internet exists so far. This is a result of the fact, that the reactive nature of the Internet requires new models, methods (founded on temporal and modal logic) and tools for the efficient, reactive event processing. To cope with the subject of events a classification and a definition for the nature of events is needed. An event is the incidence of a state because of the change of the values of certain attributes [7]. The alert reader will notice that this definition is in conformity with Mittelstraß who defines an event as a 2-dimensional relation on facts which exists if an empiric change occurs when the first fact – the origin of the event – is carried over in the second fact – the effect of the event [8].

To handle events in a language critical way it is important to define a language of events. An event in this context is something what happens or occurs, i.e. statements which contain some occurrence in the sense of "to happen" [2]. To reflect the nature of an event let us analyze the following statement:

"The grandfather drinks a glass of wine."

In a language-critical statement it would be even more correct, to specify the statement in the following way:

"It happens that the grandfather drinks a glass of wine."

When modelling events, it needs to be distinguished between *elementary* and *complex* events. An *elementary* event is an event which cannot be broken into other events in the context of the model. A *complex* event is an event which contains several elementary or complex events as components. When we regard our example, the question arises, if we observe an elementary or a complex event. Here, several views can be applied. If we regard the statement as the description of a complex event, we may split this event into the following simple events:

1. "It happens that the grandfather takes the glass to his mouth."
2. "It happens that the grandfather tilts the glass into his mouth."
3. "It happens that the grandfather swallows the wine."

When we inspect our example, we discover that even the components 1-3 of our complex event may be split into several subcomponents e.g. "It happens that the grandfather wraps his fingers around the glass." So when does the splitting into sub-events stop? The answer lies in the approach, how we model our subject. In a bottom-up approach we have to define elementary events and build complex events out of them as components. In a top down approach we start with complex events and split them into sub-complex events until we reach a level which we define as the level of elementary events. The level in which the elementary events are built depends on the level of granularity which is needed to describe the examined system together with the users on the hand and the IT-experts on the other hand.

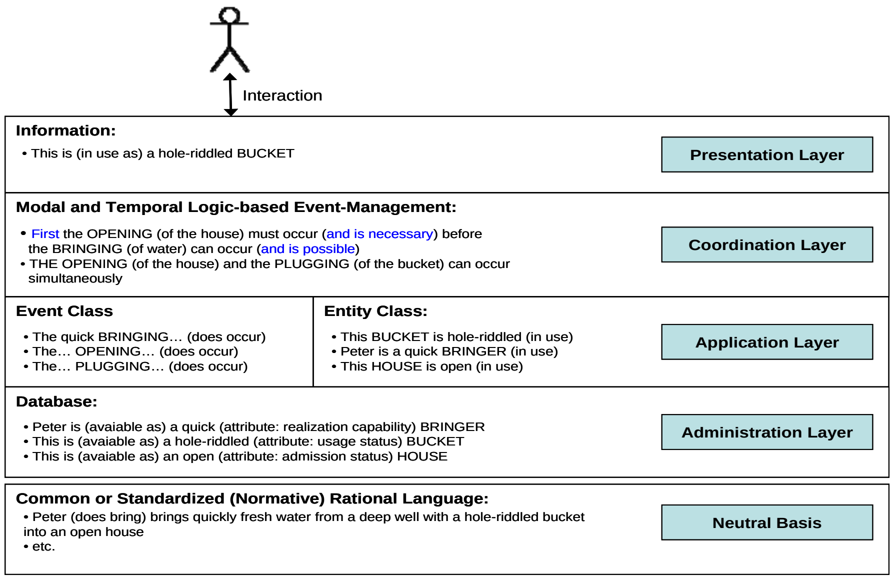
Moreover an event may be the origin or the result of a process [7] or sub-processes. For classification purposes events can be distinguished between origin, complexity and type. Chandy differentiates here between normal and abnormal types of events depending of the fact, if they appear as a regular event in an IT-System or not [9]. Normal events are part of the regular operating of an IT-System can be handled without any problems. To handle abnormal events a further differentiation needs to be made: anticipated and non-anticipated abnormal events. Anticipated abnormal (such as a delay of shipment) events aren't part of the regular operating of a system, but the system has to be prepared to deal with such kinds of events. Moreover, patterns to identify and to handle such events exist. Non-anticipated abnormal (such as an attack on the IT-System) events on the other hand, aren't expected by the IT-System and there is a lack of patterns to identify these events. The existence of such an event can only be identified, if an analyst recognizes the irregularities in the pattern of behaviour [7]. In our approach we demonstrate how to deal with these events in the following chapters.

3 Building an Event-Driven Architecture on a Language-Critical Fundament

As an extension of service-oriented architecture (SOA), the construction of so-called event-driven architectures (EDA) is promoted [7]. The main goal of a SOA is to eliminate the lack of resilience and inefficiency of system by disassembling the system into components which are easier to handle [10]. These systems still lack the transition from a passive system in which the user initiates all activities into an event-driven, reactive system where the call of web services may be triggered by events. Of course these architectures need a well founded methodology as a background. A similar approach is known from the database world with the concept of active databases. An active database management system is one which automatically executes specified actions when specified conditions arise [11]. This is solved with Event-Condition-Action (ECA) rules as formalism for active database capabilities [11]. We propose here a temporal and modal logic-based model of an event-driven architecture (Figure 2) to build a logical base for event languages. Our proposed EDA consists out of an administration layer, an application layer a coordination layer and a presentation layer. The task of the presentation layer is to interact with human beings and should be the standard way of interaction of the system with the outside world. On the ground level there needs to be a normative and rational language. As in every system at first it is necessary to eliminate all language defects:

- Checking synonyms
- Eliminating homonyms
- Identifying equipollences
- Clarifying vagueness
- Replacing wrong designators

A detailed description of these language defects can be found in [12]. In the construction of such a language, the grammar of a normative and relational language should be neutral against modelling languages [13], e.g. the object-oriented languages and the UML-dialects.



Interaction
Information:
• This is (in use as) a hole-riddled BUCKET
Presentation Layer
Modal and Temporal Logic-based Event-Management:
• First the OPENING (of the house) must occur (and is necessary) before the BRINGING (of water) can occur (and is possible)
• THE OPENING (of the house) and the PLUGGING (of the bucket) can occur simultaneously
Coordination Layer
Event Class
• The quick BRINGING… (does occur)
• The… OPENING… (does occur)
• The… PLUGGING… (does occur)
Entity Class:
• This BUCKET is hole-riddled (in use)
• Peter is a quick BRINGER (in use)
• This HOUSE is open (in use)
Application Layer
Database:
• Peter is (avaiable as) a quick (attribute: realization capability) BRINGER
• This is (avaiable as) a hole-riddled (attribute: usage status) BUCKET
• This is (avaiable as) an open (attribute: admission status) HOUSE
Administration Layer
Common or Standardized (Normative) Rational Language:
• Peter (does bring) brings quickly fresh water from a deep well with a hole-riddled bucket into an open house
• etc.
Neutral Basis

Figure 2: Structure of a language logic based EDA

In the changeover of a thing-language to an event-language approach for application system development two main directions can be identified (Figure 3). The first direction focuses on things which can exist. Here, the evolution of the language goes from the thing over processes to events. The second direction has the dialog itself as a background and develops from the thing over the communication to interaction. Communications and interactions can also be seen as events. Following the way in this direction the rules of dialogical logic have to be applied [14]. As it can be seen in Figure 2, we direct the later way to the presentation layer and the user, whereas the first is located in the coordination layer. The reasons for this decision are going to be explained in the following:
Due to the nature of events in a reactive system, the communication of the system with events is not possible in the presentation layer. Hence, the entry point for handling events in the system needs to be in the coordination layer. Here the stability of the system against abnormal events has to be proved. At this point the ability of language-logical system has also to deal with non-anticipated events.

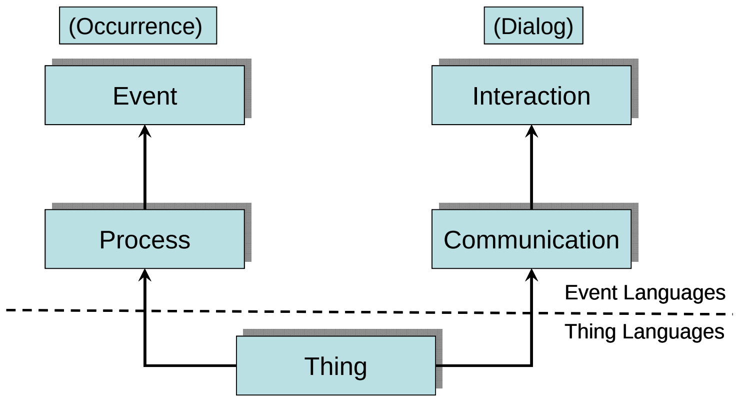


Figure 3: Two directions of development henceforward a thing-language

As non-anticipated, abnormal events are highly unspecified, the system has to be prepared to deal with such kinds of events. We propose here two approaches (Chapter 3.1, 3.2, 3.3). The dialog and interaction aspects will be published in a later paper.

3.1 The Integration of Temporal Logic in the EDA to Deal with the Time Dimension of Abnormal Non-Anticipated Events

Time itself may be disassembled in the components past, present and future. Moreover it is important to distinguish statements using these components into *temporally definite* and *temporally indefinite* statements. We shall say that a statement is *temporally definite* if its truth or falsity is independent of the time at which it is asserted [15] e.g.:

1. “It sometimes rains in London.”
2. “It always rains in London.”

Regarding these statements it is clear that their truth or falsity is unaffected, no matter what answer is given to the question “When was the statement made?”[15].
On the other hand, *temporally indefinite* statements are not independent of their time of assertion e.g. [15]:

1. “It is now raining in London.”
2. “It will rain in London sometime next week.”

In analogy to this classification of statements we are also able to classify the nature of events. Hence, we call an event *temporally definite* if it is dependent on its time of occurrence and *temporally indefinite* if it’s independent of its time of occurrence [15].

In the context of event modelling temporally definite and temporally indefinite events have a complete different character. To describe the temporally definite dimension of an event, we use the modalities *sometimes* and *always*. If we choose to observe an event in its temporally definite dimension following statements are possible:

1. "It happens *sometimes*, that the grandfather drinks a glass of wine."
2. "It happens *always*, that the grandfather drinks a glass of wine."

The usage of the temporally indefinite dimension leads to the following statements:

1. "It happens *now*, that the grandfather drinks a glass of wine."
2. "*Before* it happens, that the grandfather drinks a glass of wine, it happens that the grandfather opens the bottle."
3. "*After* it happened that the grandfather drank the wine, it happens that he goes to bed.

As we can see from our example the modalities *now*, *before* and *after* characterize the temporally indefinite dimension of an event.

3.2 The Integration of Modal Logic in the EDA to Deal with the Uncertainty of the Occurrence of an Abnormal Event

Modal logic is the part of logic which describes the logic of necessary and possible statements [16]. It extends boolean logic which only describes statements as true or false with the modalities necessary and possible. Statements which are false can actually be possible, whereas also true statements haven't to be necessary. We say a statement is *possible*, if a world may exist in which this statement is true, e.g. "All human beings are green." (This is wrong in our world, but we can think of a world in which this statement might be true.). We call a statement necessary, if it has to be true in all thinkable worlds, e.g. "All circles are round." (This is true and it needs to be true in all thinkable worlds, because a circle is defined as round) [16].

If we transcribe these perceptions on an EDA we are able to inspect the possibility and necessity of an event within a system. We define an event as possible, if there exists a state in the EDA in which the event can be processed. So an important task in the construction of an EDA is to build this architecture in a way that abnormal events become possible. Especially anticipated abnormal events have to be foreseen and to be respected in the architecture of the system. If we apply this modality on our example we can build statements like:

"It is *possible* that it happens that the grandfather drinks a glass of wine."

Unfortunately, necessity cannot be defined analogously in the context of events, because this would mean, that an event can or must happen in every state of the system. This would lead to deadlock situations. Hence, we define an event within a system as *necessary*, if an event *needs to happen* so that another event *may happen*. In contrary to possibility, necessity is dependent on the state of the system. Hence, we can build statements like:

"The event that the grandfather opens the bottle happens *necessarily*, before it happens that grandfather drinks the wine."

Regarding this example we can make an important observation. As the modality of necessity is dependent on the state of the system, the point of time when the event occurs is critical (Which we can see by the necessity of the use of the word "before"). Hence, we cannot describe necessity within our system without the use of temporal logics.

3.3 The Combination of Temporal and Modal Logic in an EDA with an Reactive Event Management or Reasoning Component

As we have seen in the last chapter the modal logic of events is strongly connected with the temporally indefinite dimension of the temporal logic of events. Both dimensions are compatible and as we have seen sometimes even dependent of each other (when necessity is regarded). Of course, the combination with the modality "it's possible" and "it's necessary" can be executed – that means a reasoning component is implemented – without any problems:

1. "It is possible that it happens now that the grandfather drinks a glass of wine."
2. "It is possible that it happens that the grandfather talks to his wife before he drinks a glass of wine."
3. "It is necessary that it happens that the grandfather talks to his wife before he drinks a glass of wine."

The relationship between modal logics and the temporally definite dimension is different. Here the use of the temporal modality "sometimes" forces the use of "it's possible" in the modal dimension, whereas the use of "always" implies "it's necessary". Hence, sentences with a structure like:

"It is possible that it happens that it always happens …."

are not a useful extension of our system.

4 Reactive Application Systems

The scientific basis to handle events in Computer Science and therefore also to handle complex events goes back to the theory of the development of reactive systems [4]. Here, the distinction between transformational and reactive systems is fundamental. A transformational system accepts inputs, performs transformations on them and produces outputs (Figure 4). In this context a system which may ask for additional inputs and/or produce some of their outputs as they go along is also covered by the definition of a transformational system [4]. The point, however, is that these systems perform input/output operations and perhaps prompting a user from time to time to provide extra information.

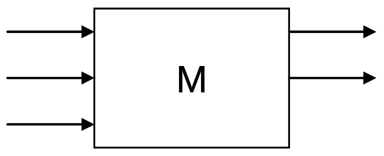


Figure 4: "Black box", a transformational system [4]

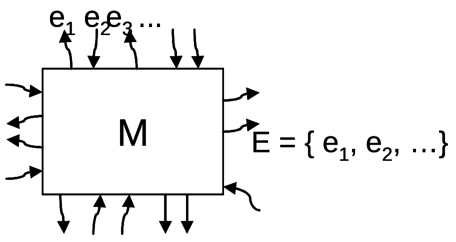


Figure 5: "Black cactus", a reactive system [4]

Reactive systems, on the other hand, are repeatedly prompted by the outside world and their role is to continuously respond to external inputs, e.g. user-interaction or events. A reactive system does not compute or perform a function, but is supposed to maintain a certain ongoing relationship with its environment [4] or its state. To reflect the totally different behaviour on can think of a reactive system as a "black cactus" (Figure 5) in which the thorns of the cactus represent the interface elements [4]. Of course an EDA is necessarily a reactive system. Hence, notations for temporal and modal logics have to be integrated in the system. Here modality can e.g. be expressed with the colour of the elements "e" of the set "E". The time dimension is located in the arrows of the system. A before-after relationship can be expressed with the direction of an arrow, whereas the temporal modalities "always" and "sometimes" can be expressed analogue to modalities with colours.

5 Conclusion

Finally we won't forget the people who work and participate in an EDA. One of the most important aspects is to integrate the people into the system. Here, the expertise of the system architects is the key factor. And we will talk about this part scientifically in our paper "Dialog-logic based event languages for the development of interactive application systems" which is in progress.

To cope with this vast and enormous progress people educated in a traditional way just studying one topic and not able to understand and research in associated fields are not sufficient anymore. With good reason, Jürgen Mittelstraß reminds us: "Who (even in a disciplinary framework) has not learned in an interdisciplinary way, will not be able to do research in an interdisciplinary way" [17]. The software industry has also identified this need and as can be clearly seen in the latest IBM Academic initiative claiming the demand for so-called T-shaped people [18]. The label T-shaped in this context refers to the type oft education people have acquired. The horizontal bar in the "T" stands for broad understanding of a field. On the other hand, the vertical bar of the "T" stands for deep technical skill. The most important aspect of interdisciplinarity is the integration of different competences e.g. methods and ways of thinking, which enables the holistic understanding of an object (field). This is a significant difference to multidisciplinarity, which only refers to the mere result of different disciplines working together [19]. So let's look forward how these "T-shaped" people form the FI on architectures based on the language-critical management of events.

6 References

1. Ortner, E.; Heinemann, E.: Memorandum zum Verhältnis von System- und Anwendungsinformatik: Diskussionspapier sieht Organisationsprozesse als das Fundament einer Neuausrichtung der Informatik, in: Computerzeitung Online vom 16. Juli 2007.
2. Ortner, E.: Sprachbasierte Informatik – Wie man mit Wörtern die Cyber-Welt bewegt. Edition am Gutenbergplatz, Leipzig (2005)
3. Wittgenstein, L.: Tractatus logico-philosophicus, Suhrkamp, Frankfurt a. Main (2003)
4. Harel, D., Pnuelli, A.: On the Development of Reactive Systems. In: NATO ASI Series, Vol. F13 Logics and Models of Concurrent Systems, Springer, Heidelberg (1985)
5. McNeile, A., Roubtsova, E.: Programming in Protocols: A Paradigm of Behavioural Programming. In: 3th International Conference on Evaluation of Novel Approaches to Software Engineering, pp.23-30. Springer, Heidelberg (2008)
6. Lehmann, R.: Integrierte Prozessmodelle mit Aris. dpunkt, Heidelberg (2008)
7. Rommelspacher, J.: Ereignisgetriebene Architekturen in: Wirtschafsinformatik 4, pp. 314-317 (2008)
8. Mittelstraß, J.: Enzyklopädie Philosophie und Wissenschaftstheorie, Band 2. Metzler, Stuttgart (2005)
9. Chandy, K.: Event-Driven Applications: Costs, Benefits and Design Approaches. Presentation at Gartner Application Integration and Web Service Summit 2006, California Institute of Technology (2006)
10. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA, Prentice Hall, New Jersey (2005)
11. McCarthy, R., Dayal, U.: The architecture of an active database management system. In 1th Conference on Management of Data, ACM SIGMOD, pp 215–224. Portland (1989)
12. Ortner, E.: Language-critical Enterprise and Software Engineering. In: Proceedings of the Fourteenth Americans Conference of Information Systems, Toronto (2008)
13. Lorenzen, P.: Lehrbuch der konstruktiven Wissenschaftstheorie, BI-Wissenschaftsverlag, Mannheim (1987)
14. Kamlah, W., Lorenzen, P.: Logische Propädeutik, Metzler, Stuttgart (1992)
15. Rescher N., Urquhart A., Temporal Logic, Springer, New York (1971)
16. Hughes George E., Cresswell Max J.: A new introduction to modal logic, Routledge, London (1996)
17. Mittelstraß, J.: Der Flug der Eule: Von der Vernunft der Wissenschaft und der Aufgabe der Philosophie, Frankfurt a.M. (1997)
18. IBM Academic initiative, *http://www-03.ibm.com/press/us/en/pressrelease/21581.wss*
19. Ortner/Heinemann: Organizational Computer Science: An Interdisciplinary Approach to Understand Organizational Engineering