A Meta-Service for Event Notification

Doris Jung and Annika Hinze

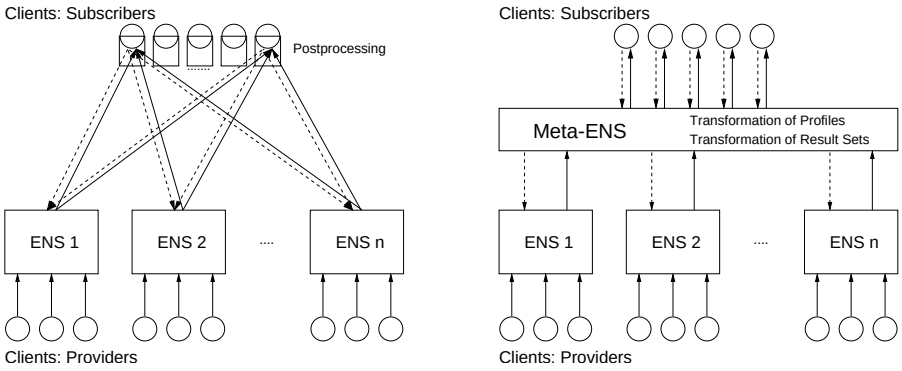
University of Waikato, New Zealand
{d.jung,hinze}@cs.waikato.ac.nz

Abstract. The integration of event information from diverse event notification sources is, as with meta-searching over heterogeneous search engines, a challenging task. Due to the complexity of event filter languages, known solutions for heterogeneous searching cannot be applied for event notification
In this paper, we propose the concept and design of a Meta Service for Event Notification. We define transformation rules for exchanging event filter definitions and event notifications between various event services and sources. We transform each filter defined at a meta-service into a filter expressed in the language of each event notification source. Due to unavoidable asymmetry in the semantics of different langues, some superfluous information may be delivered to the meta-service. These notifications are then post-processed to reduce the number of spurious messages. We present a survey and classification of filter languages for event notification, which serves as basis for the transformation rules. The proposed rules are implemented in a prototype transformation module for a Meta Service for Event Notification.

1 Introduction

Alerting Services or Event Notification Services (ENS) inform their users about changes that have occurred at information objects. These changes are called events. Information objects can be, e.g., documents in a digital library or temperature sensors in a facility management system; events can be caused, e.g., by new, changed or deleted objects. The service actively or passively observes the information objects at the providers sites (e.g., documents in digital libraries or sensors in buildings). Users describe their interest in form of personal profiles that define filter conditions for the information delivery. In a widely distributed application context, each of the considered applications may employ their own alerting services (e.g., as done for digital libraries provided by different publishing houses or as currently available for tourist information). Users on the other hand, are interested in combined information from diverse and heterogeneous sources. Similar to the problem of information querying over widely distributed information sources, here we encounter the problem of distributed filtering over heterogeneous event sources.

Unfortunately, the results known from research in meta-searching [12] and query rewriting for search over heterogeneous sources [3, 21] cannot simply be applied to the new context of event notification. Advanced filter conditions are more complex than search queries; in fact, they can be seen as extensions of search queries: A simple filter expression can be seen as a standing search query. Additionally, filter expressions can contain sophisticated event pattern descriptions referring to temporal succession of events, such as sequences and disjunction of events [11, 19, 20].



(a) independent ENS (b) Meta-ENS

Fig. 1. Communication of clients with several independent ENS vs with a Meta-ENS

1.1 Problem Statement and Contribution of the Paper

The existence of several independent event notification services causes a number of problems, see Figure 1(a) for illustration:

1. Subscribers are forced to subscribe the same profile to a number of services; these use different filter languages (i.e., the profiles have to be expressed differently) with differing expressiveness. In Figure 1(a), the large number of dashed arrows from each subscriber indicates the repeated subscriptions.
2. Composite events combining events from different providers that are handled by different services cannot be directly subscribed to. In consequence, the client has to subscribe to (several) separate services and implement post-filtering locally. In Figure 1(a), the arrows from each ENS indicate the notifications that have to be post-filtered at the subscribers' sides.
3. If providers serve several services, the duplicates have to be removed in a post-filter process at the client side. In Figure 1(a), the postfiltering is depicted as boxes at the subscribers' sides.

An umbrella service could combine all providers but would force a flat homogenization of the providers, while ignoring the existing heterogeneity of the providers and services. Moreover, there are the issues of trust, downwards compatibility, company strategy, and required integration of legacy systems.

As a solution to the three problems we propose the equivalent of a Meta-Search Engine: a *Meta Event Notification Service* (Meta-ENS), see Figure 1(b). Our solution allows for and supports the heterogeneity of services and providers. It integrates services while accepting their differences and diversity. The advantages are evident: Subscribers can have a uniform access for profile definition, having access to several event sources. Users are not repeatedly notified about the same event, i.e., duplicate recognition can be implemented on the meta-service level. In addition, security and privacy issues are easier to address. A number of research questions emerge as a result of the analysis given above, which have to be answered for the design of the meta-service:

1. Which event patterns for composite events are typically supported in profile and filter languages for event notification services? Are there categories of languages?
2. How to translate the event patterns in one language into the patterns of a target language such that profile definitions can be converted between languages? How are the result sets influenced by the transformation? What postprocessing is necessary for re-transforming the result sets to match the initial profile query?
3. How to detect duplicates of event messages that refer to the same event? How to detect messages referring to the same event?

1.2 Focus and Organization of the paper

In this paper, we will address the first two questions, which we believe to be essential for the implementation of a Meta-ENS. For the elimination of duplicates, existing techniques from information retrieval and information dissemination may be employed (see, e.g., [23]). Note that we do not make assumptions abut the nature of the services, e.g., distributed or centralized services. We abstract from the problems of event detection and ordering in a distributed environment.

In the remainder of the paper, we propose the detailed design of a *Meta Service for Event Notification* that translates filter expressions for heterogeneous event notification services. After a brief introduction of the concepts of filter languages (Section 2), we first analyze the filter languages of existing alerting services in order to identify typical event patterns (Section 3). In Section 3.2, the services will be ordered into groups based on the expressiveness of their filter languages. Based on this classification, we propose a set of transformation rules for the translation of filter expressions between these groups (Section 4). We conclude the paper by a summary and an outlook towards further research and challenges to be addressed.

2 Concepts

In this section, we introduce the basic concepts of event notification services. A more detailed discussion of models and terms can be found in [9]. Event notification services inform its users about events that occurred on a given set of objects. Events are reported to the service by means of event messages. Objects have certain states, defined by their properties at a certain time, e.g., the state of a database, the content of a web-page.

Definition 1 (Event). *An event is the occurrence of a state transition of an object of interest at a certain point in time. Events are reported by means of event messages (or notifications), which contain a timestamp referring to the event's occurrence time.*

Events have no duration. Events may be state changes in databases, signals in message systems. We consider *primitive events* and *composite events*, which are formed by combining primitive and composite events. The set of composite events $\mathbb{E}_C$ detectable by a certain system is defined by its system event algebra, i.e, by its filter and profile semantics. Composite events are created based on an event algebra. Event composition defines new event instances. The new (composite) event instances inherit the characteristics of

all contributing events; the event occurrence time is defined by the composition operator. We denote the fact that a set of event instances contributes to a composite event by the $\succ$ operator:

Definition 2 (Composition Contribution $\succ$). *Let $e_1, ..., e_n \in \mathbb{E}$ be event instances that contribute to the composite event $e \in \mathbb{E}_C$. This relation is expressed as $\{e_1, ..., e_n\} \succ e$. The $e_1, ..., e_n$ can be primitive or composite event instances.*

One of the central terms of an event notification service is the user profile:

Definition 3 (Profile). *A profile is a query q_{exp} that is periodically evaluated by the Event Notification Service against incoming events, i.e., a query that is evaluated against the trace of events reported to the service.*

We distinguish *event instances* from *event classes*. An event class is a set of events specified by a profile while an event instance relates to the actual occurrence of an event. In the following, we simply use the term *event* whenever the distinction is clear from the context. Events (instances) are denoted by lower Latin e with indices, i.e., $e_1, e_2, \ldots$, while event classes are denoted by upper Latin E with indices, i.e., $E_1, E_2, \ldots$. The fact that an event e_i is an instance of an event class E_j is denoted *membership*, i.e., $e_i \in E_j$. This relationship is non-exclusive, i.e., $e_i \in E_j$ and $e_i \in E_k$ is possible even with $E_j \neq E_k$. Event classes may also have subclasses, so that $e_i \in E_j \subset E_k$. The timestamp of an event $e \in E_1$ is denoted $t(e)$.

Definition 4 (Duplicate). *Duplicates of events are event instances that belong to the same event class.*

Note that duplicate events refer to separate event instances – in contrast, the same event instance might be reported twice to the service, leading to duplicate event messages. Duplicate events could be subsequent changes of the same document in a digital library, but also all events referring to a certain document collection. Note that duplicates need not necessarily have identical event types or identical timestamps.

In a ENS, query profiles are evaluated against the history of all observed events.

Composite Event Pattern Operators This section informally describes the concepts of the most common operators for composite events. Event composition defines new event instances that inherit the characteristics of all contributing events. The occurrence time of the composite event is defined by the composition operator. The events e_1 and e_2 used in the definitions below can be any primitive or composite event; E_1 and E_2 refer to event classes with $E_1 \neq E_2$. $t(.)$ refers to occurrence times defined based on a reference time system, T denotes time spans in reference time units. We use the contribution operator $\succ$ (cf. Definition 2) to identify the events that contribute to a composite event. Note that temporal operators are defined on event instances as well as on event classes, resulting in event instances and event classes, respectively.

Disjunction: The *disjunction* $(E_1|E_2)$ of events occurs if either $e_1 \in E_1$ or $e_2 \in E_2$ occurs. The occurrence time of the composite event $e_3 \in (E_1|E_2)$ is defined as the time of the occurrence of either e_1 or e_2 respectively: $t(e_3) := t(e_1)$ with $\{e_1\} \succ e_3$ or $t(e_3) := t(e_2)$ with $\{e_2\} \succ e_3$.

Conjunction: The *conjunction* $(E_1, E_2)_T$ occurs if both $e_1 \in E_1$ and $e_2 \in E_2$ occur, regardless of the order. The conjunction constructor has a temporal parameter that describes the maximal length of the interval between e_1 and e_2.[1] The time of the composite event $e_3 \in (E_1, E_2)_T$ with $\{e_1, e_2\} \succ e_3$ is the time of the last event: $t(e_3) := max\{t(e_1), t(e_2)\}$.

Sequence: The *sequence* $(E_1; E_2)_T$ occurs when first $e_1 \in E_1$ and afterwards $e_2 \in E_2$ occurs. T defines the maximal temporal distance of the events. The time of the event $e_3 \in (E_1; E_2)_T$ with $\{e_1, e_2\} \succ e_3$ is equal to the time of e_2: $t(e_3) := t(e_2)$.

Negation: The *negation* $\overline{E}_T$ defines a "passive" event; it means that no $e \in E$ occurs for an interval $[t_{start}, t_{end}]$ with $t_{end} = t_{start} + T$ of time. The occurrence time of $\overline{e}_T \in \overline{E}_T$ is the point of time at the end of the period, $t(\overline{e}_T) := t_{end}$ When clear from the context, we write $\overline{e}_T$ when referring to a passive event.

Simultaneity: The *simultaneity* $(E_1 : E_2)_T$ occurs when both events $e_1 \in E_1$ and afterwards $e_2 \in E_2$ happen it the same time: $t(e_1) = t(e_2)$.

Selection: The *selection* $E^{[i]}$ defines the occurrence of the i^{th} event $e \in E$ of a sequence of events of class E, $i \in \mathbb{N}$.

The model of composite events consists of (primitive or composite) events combined through event constructors. Note that temporal operators are defined on event instances as well as on event classes, resulting in instances and classes, respectively. This means that operators on event classes form profiles, i.e. queries, whereas operators on event instances describe certain composite event instances.

Composite Event Pattern Parameters In addition to the event operators, we define the two parameters of consumption mode and duplicate handling. Consumption mode is a concept concerning the strategy of evaluation in respect to the event history. When specifying a profile it is necessary to define whether event instances should be disposed of after matching or whether they should be considered again for a new filtering process. if disposed, there are two possibilities to do so: 'delete' and 'delete and reapply'. For 'delete', all event instances which occurred before the matched event instance are deleted. The other option is to delete only those event which have taken part in the matched event instance. If no event instances are deleted this is called 'keep'.

Duplicate handling describes which event instances out of a list of identical duplicates are regarded for the filtering process. The following possibilities are relevant for our analysis: first, last, all, n^{th}, and n to m. The values refer to the ordering number of the duplicate events.

3 Survey of Profile Definition Languages in ENS

This section addresses the first problem that we identified in the introduction (Problem 1): Which event patterns for composite events are typically supported in profile and filter languages for ENS and are there categories of languages? We have analyzed filter languages of several event-based systems. This section presents the initial results of our analysis, which has been carried out in three steps.

[1] $(E_1, E_2)_\infty$ refers to an event composition without temporal restrictions.

1. *The overview*: For each system, we list the supported operators, support of time frames, consumption mode, and duplicate handling. These analyses are based on the available literature, i.e., we refer to the operators and their parameters the way the initial publication does. Consequently, there are differences in the semantics and symbols compared to the ones introduced in Section 2. This overview is given subsequently in Section 3.1.
2. *Comparative Study*: For each filter language, the filter operators are translated into the terminology used here. Based on this, we perform a uniform comparison of the approaches. This comparative study is presented in Section 3.2.
3. *Language Groups*: Based on the comparative study, we identified five groups (types) of filter languages for event-based services. Section 3.3 presents the definition of the language groups. These language groups form the basis for the design of the meta-service for event notification and event-based communication.

3.1 Overview of systems and supported event patterns

Our overview of filter languages is presented ordered by system type; we analyzed the following types[2]: Event Notification Services (see Table 1), Event-based Infrastructures (see Table 2), Event-based Infrastructures (see Table 2) and Hybrid systems (see Table 3). An extended analysis that also covers active database systems and event actin systems can be found in [13]. For each of the systems, we analyzed the following characteristics of the profile languages: operators for building event patterns, support of time frames in the patterns, the consumption modes and the supported duplicate handling.

Event Notification Services We analyzed a selection of eight typical event notification services: A-MediAS [9], an adaptive and integrating event notification service; the Corba Notification Service [8], Elvin [22]; Hermes [4], an event notification service for digital libraries; Keryx [1], which is designed to distribute notifications in the internet; READY [7], the sequel of the event-action system YEAST; and Siena [2]. The results are shown in Table 1. Most ENS still only support primitive events, with research focussing on efficient filter algorithms.

Event-based Infrastructures In the category of event-based Infrastructures, we analyzed Cobea [16], which is used e.g. for the management of networks; Rebeca [5], an event-based architecture for electronic commerce; Regis [17], a development environment for distributed systems that has been extended by the pattern language GEM [19]; and Salamander [18], a system for the distribution of web-applications. The results are shown in Table 2.

Hybrid Systems Hybrid systems are able to handle a variety of event sources: web-documents, databases and files. We examined the systems Conquer [15] and OpenCQ [14] from the Continual Queries project, and Eve [6]. Eve combines characteristics of active databases and event-based architectures to execute event driven workflows. The result of the analysis is shown in Table 3. Hybrid systems combine events from different sources, supporting a variety of event patterns.

As illustrated in this section, the analyzed systems support a variety of event patterns, using various operators and auxiliary parameters. Note, that the list of analyzed

[2] Note that the exact distinction between the types may be arguable.

System	Operators	Time frame	Consumption Mode	Duplicate handling
A-mediAS	Conjunction: $(E_1 \& E_2)$ Disjunction: $(E_1 \| E_2)$ Sequence: $(E_1; E_2)$ Negation: $(E_1 - E_2)$ Selection: $First(E_1)$	yes	keep, delete, delete and reapply	first, last, all, n^{th}, n to m
CORBA notification service	only primitive events	–	–	–
Elvin	only primitive events	–	–	–
Hermes	only primitive events	–	–	–
Keryx	only primitive events	–	–	–
Ready	Conjunction: $(E_1 \&\& E_2)$ Disjunction: $(E_1 \| E_2)$ Sequence: $(E_1; E_2)$ Negation: $(not\ E_1)$			first, last, all, n^{th}, n to m
Siena	Sequence: $(E_1.E_2)$	–	delete	first

Table 1. Composite event operators in Event Notification Services

systems cannot be exhaustive, but considers a representative set of selected systems and languages. In the next section, we introduce our classification of filter languages, which allows to identify language groups that support typical subsets of event patterns.

3.2 Classification of Filter Languages - a Comparative Study

This section presents our *comparative study* of filter languages: This is the second step in our approach to answer the question for common patterns and groups of filter languages (Problem 1). We translate each system's operators into the terminology used here, in order to allow for a uniform comparison of the approaches. We first introduce our classification methodology and then present the actual classification. This classification shall be the basis for identifying typical language groups in the next section.

Extending the survey presented in the last section, we have classified the profile languages of selected event systems. We developed a set of classification criteria, which are a combination of the semantic language characteristics defined by Hinze/Voisard [10] and Zimmer/Unland [24]. Both works describe the semantics of filter languages. Both use operators for event patterns and additional parameters.

Composite Event Pattern Operators As shown in the previous section, the systems use different operators for event patterns. Additionally, equally named operators do not necessarily have the same semantics while similar semantics might be expressed using different operators. Moreover, the exact semantic description of these operators is rarely given in literature. Here, we will translate all systems' operators into the following schema: conjunction, disjunction, negation, selection, sequence and simultaneity (see Table 4).

System	Operators	Time frame	Consumption Mode	Duplicate handling
Cobea	Conjunction: $(E_1 \& E_2)$ Disjunction: $(E_1 \| E_2)$ Sequence: $(E_1; E_2)$ Whenever: $(\$E_1)$ Without: $(E_1 - E_2)$	Duration	Keep events	all
Rebeca	Conjunction: Disjunction: Sequence: Negation:	yes	Delete and reapply, (recent, chronicle)	–
Regis	Conjunction: $(E_1 \& E_2)$ Disjunction: $(E_1 \| \| E_2)$ Sequence: $(E_1; E_2)$ Negation: $(\{E_1; E_2\}!E_3)$ Time: $(E_1 + timeperiod)$	Duration-window	Delete all events	first
Salamander	only primitive events	–	–	–

Table 2. Composite event operators in Event-based Infrastructures

Event Pattern Parameters Considering the analyzed systems, it becomes clear that to simply consider the operators is not sufficient in order to convey the full semantic meaning. Each system offers parameters, which further define/change the operators semantics. We shall briefly describe the different parameters proposed in the two schemas.

Hinze/Voisard define two parameters: event instance selection and event instance consumption (see left column in Table 4). 'Event instance selection' describes which events qualify for a composite event and how duplicated events handled. Examples are to select the first event in a list of duplicates, the last one or a particular n^{th} one. 'Event instance consumption' defines which events are consumed by composite events, i.e., removed from the matching trace. Options are to keep the selected event instances, to remove them, or to remove them and reapply the event pattern (similar to our definition in Section 2). In [10], only two of the three options are formally defined. Both event pattern parameters can be combined freely.

Unland/Zimmer defined separate parameters for concurrency, consumption, selection, traversion, and coupling (see middle column in Table 4). Concurrency can be 'overlapping' or 'non-overlapping', allowing for components of different event-instances to overlap each other or not. Consumption may be 'shared', 'exclusive parameter', or 'exclusive': Either no event-instance is deleted, or all event-instances which have taken part in the matching of a composite event are deleted, or all event-instances before the terminating event-instance of a composite event are deleted. These parameters are similar to the Event instance consumption, but not identical. The selection parameter is similar to and can be expressed via Hinze/Voisard's Event Instance selection.

The parameters are partially interdependent: Shared consumption mode requires overlapping concurrency and an exclusive consumption mode is only logical in combination with a non-overlapping concurrency. The concurrency mode for the exclusive

System	Operators	Time frame	Consumption Mode	Duplicate handling
CQ: Conquer and OpenCQ	Conjunction Disjunction Sequence: $(E_1; E_2)$ Negation: Simultaneity: $(E_1 \| E_2)$	yes	–	–
Eve	Conjunction: $(CON(E_1, E_2, sw))$ Disjunction: $(DEX(E_1, E_2))$ Sequence: $(SEQ(E_1, E_2, sw))$ Simultaneity: $(CCR(E_1, E_2, sw))$ Negation: $(NEG(E_1, (E_2, E_3, sw), sw))$ Repetition: $REP(E_1, times, sw)$	yes	delete and reapply (chronicle)	first, n^{th}

Table 3. Composite event operators in Hybrid Systems

consumption is undefined. This interdependence of parameters is our main reason for primarily following Hinze/Voisard's classification.

Not all parameters are applicable for the systems we are interested in, e.g., the concurrency mode and the traversion mode. The traversion mode, which describes the direction of traversing composite events is irrelevant here, since systems filter their events in the timely order and not backwards. This parameter will not be included in our classification. The coupling mode defines whether the components of different event-instances may be interleaving. These modes are expressed by Hinze/Voisard using negation and wildcards. We therefore exclude this parameter from our classifications.

We followed a hybrid approach and use a combination of both schemas, which is shown in the right column of Table 4. Our *combined classification schema* combines the operators proposed in the two schemas and also uses a combination of the proposed event pattern parameters. We use the characteristics from Unland/Zimmer's consumption mode; the duplicate handling is based on Hinze/Voisard's Event Instance Selection.

We now use the combined classification schema for our comparative study of filter languages in event-based systems. The results of the study are presented as a language classification in Table 5. This table serves three purposes: It gives an overview of event-based filter languages, provides a uniform analysis of the languages (i.e., translated into a common schema), and gives a first impression of the operators and parameters typically supported in event-based systems.

Based on the language classification, we make the following observations in the comparative study: If composite events are supported, all[3] of these systems implement conjunction and disjunction (i.e., operators without ordering). Some implement the sequence operators (requires ordering), fewer the negation (required observation). Selection and simultaneity are rarely supported: Selection is a special case of duplicate handling and simultaneity is difficult to determine for distributed systems - it can be expressed by conjunction with a small ϵ–time frame. Time frames are not always sup-

[3] With the exception of Siena that only supports a single operator for research purposes.

Hinze/Voisard	Unland/Zimmer	Combined Classification
Composite Event Pattern Operators		
conjunction	conjunction	conjunction
disjunction	disjunction	disjunction
sequence	sequence	sequence
negation	negation	negation
selection	-	selection
-	simultaneity	simultaneity
Time frames		
Event Instance Consumption	Consumption Mode	Consumption Mode
Event Instance Selection	Parameter Selection	Duplicate Handling
-	Concurrency Mode	-
(above parameter/operators combination)	Coupling Mode	-
(above parameter combination)	Traversion Mode	-

Table 4. Comparison of the two schemas for semantic classification and our combined schema.

ported, requiring a time handling strategy for distributed systems. Consumption mode and duplicate handling are rarely made explicit. If they are explicit, several options are supported, otherwise they are hard coded in the system and difficult to determine.

3.3 Language Groups

Based on the observations from our comparative study of languages in the previous section, we identify *language groups* (types) of filter languages for event-based systems. This is the third and final step in answering the question for typical event patterns and groups of filter languages for ENS (Problem 1).

These language groups form the basis for the design of the meta-service for event notification and event-based communication. In the next section, we address the second problem (as identified in the introduction) and define rules for profile transformations between these language groups. Parameters for Consumption Mode and Duplicate Handling (cf. Section 2) are very rarely explicitly described in the literature. For this reason, we did not include the parameters in the definition of groups – they will be consider separately. Thus, the languages are classified into groups based on their support for time frames and by their support for pattern operators.

We define five groups as shown in Table 6. There are two groups without time frame support: CEs support conjunction, disjunction and negation; a group member is PLAN. SCEs support conjunction, disjunction, negation, and sequences. Members are READY, Rebeca, and Active House (CEA).

There are three groups with time frame support: TCE offer conjunction, disjunction and sequence. Members of this group are Yeast and Sentinel (language Snoop). The OTCEs support conjunction, disjunction, sequence and negation; members of this group are Samos, Cobea, and GEM. STCE offer conjunction, disjunction, sequence, negation, and simultaneity. Members of this group are Eve, Conquer, and OpenCQ. The

Systems	Composite Events	Operators						Time frame	Consumption Mode			Duplicate handling			
		Conjunction	Disjunction	Sequence	Negation	Simultan.	Selection		keep	delete	delete & reapply	first	last	all	detailed
Active House	×	×	×	×	×							×			
A-mediAS	×	×	×	×	×		×	×	×	×	×	×	×	×	×
Cobea	×	×	×	×	×			×	×					×	
Corba NS	–														
Elvin	–														
Eve	×	×	×	×	×	×		×			×	×			×
GEM (Regis, Darwin)	×	×	×	×	×			×		×		×			
Hermes	–														
Keryx	–														
OpenCQ	×	×	×	×	×	×		×							
PLAN	×	×	×		×			×				×			
Ready	×	×	×	×	×							×	×	×	×
Rebeca	×	×	×	×	×						×				
Salamander	–														
Samos	×	×	×	×	×			×			×	×			×
Siena	×			×						×		×			
Snoop (Sentinel)	×	×	×	×				×		×	×		×	×	
Yeast	×	×	×	×				×				×		×	

Table 5. Comparison of Profile Definition Languages = Filter Languages, alphabetically ordered by system/language name. Characteristics are derived directly from the literature (whenever possible) or inferred from given examples

Time-frame-less Composite Events	Time-framed Composite Events
CE: Simple Composite Events *(conjunction, disjunction and negation)*	**TCE**: Simple Time-framed Composite Events *(conjunction, disjunction and sequence)*
	OTCE: Ordinary Time-framed Composite Events *(TCE and negation)*
SCE: Sophisticated Composite Events *(CE and sequence)*	**STCE**: Sophisticated Time-framed Composite Events *(OTCE and simultaneity)*

Table 6. Groups of Filter Languages

disequilibrium of the group assignment of negation and sequence is due to the different effect of time-frames on the operators.

3.4 Summary of findings regarding a classification of filter languages

The three steps of analyzing profile languages presented in this section are our answer to the first research question stated in the introduction of this paper (identification of typical event patterns and language groups). Firstly, we analyzed typical patterns for composite events in filter languages. Secondly, we compared the filter languages based on a classification schema. Thirdly, we used the classification to identify typical groups of filter languages. The findings of this section shall serve as a foundation for answering the second problem of finding transformation rules between languages from different groups. The second problem is addressed in the next section.

4 Profile and Result Transformations

We now address the problem of translating filter expressions between languages that use different operators and semantics (Problem 2). The answer to this problem shall provide a set of transformation rules that form the core of the proposed Meta-ENS for integrating heterogeneous event notification services. Here, we therefore especially consider the challenge of translating a filter expression of the meta-service in to the target language of other systems. The meta-service is assumed to support all of the composite event/profile concepts (operators and parameters) introduced in Section 2.

4.1 Transformation Methodology

For each language group, we introduce transformation rules for translating filter expressions defined at the Meta-ENS into equivalent filter expressions using a language of the group. As can be derived from the group definitions, a simple translation of filter expressions between groups is not possible. Instead, for different semantic concepts in two distinct groups, we have to find expressions that are semantically close. Additionally, auxiliary profiles and post-filtering may be required.

Profile Transformations If a certain operator does not exist in one language a transcription expression has to be used. These transcriptions may be more or less expressive than the source expression. We define therefore four types of transformations: equivalent, positive, negative, and transferring transformation. We denote these transformations with the arrow-notation that is shown in Table 7. It is an extension of the notation used for Boolean transformations [3]. Equivalent transformations lead to expressions that have identical result sets. Positive transformations result in expressions that are less selective than the original - potentially creating larger result sets; negative transformations result in more selective expressions compared to the original filter expression (creating smaller result sets). Larger result sets without subsequent postfiltering lead to false positives in client notifications. Smaller result sets lead to missed event notifications. Transferring transformations (when omitting event patterns) use postfiltering and auxiliary profiles.

Post-filtering and Auxiliary profiles For the considered transformations between language groups, not all of the original operations can be expressed in the languages of less powerful groups. In order to use weaker systems in cooperation with stronger ones, auxiliary profiles (i.e., additional filter expressions) have to be defined at the services. The filter results are delivered to the stronger system which then needs to perform additional simple filter operations (post-filtering).

Notification Transformation Differing from query transformation, the result set obtained in an event notification service is not simply a set of tuples or documents. For ENS, the result reflects the filter expression, i.e., the temporal connection between the events is reported. If for two communicating systems, the less expressive system receives a message from a more expressive one, the notification might not be comprehensible to the less expressive filter language. Lets consider the following example: Consider two systems A and B, where the filter language of A supports only sequences and disjunction, the filter language of B supports only conjunction. The systems cannot cooperate directly, since their set of filter operators are disjunct. In order to cooperate, system A defines a profile p_A at the Meta-ENS ($p_A = ((E_1; E_2)|(E_2; E_1))$). Meta-ENS transforms this expression into a profile p_B that is defined at system B: $p_B = (E1, E_2)$ with $p_A \longleftrightarrow p_B$. When system B sends a notification $n_B = (e1, e_2)$ to the Meta-ENS, the system A is notified by the transformed message $n_A = ((e_1; e_2)|(e_2; e_1))$. Thus, not only the filter expressions have to be transformed for the cooperation but also the notifications.

Transformation	Notation
Equivalent Transformation	$\longleftrightarrow$
Positive Transformation	$\xrightarrow{+}$
Negative Transformation	$\xrightarrow{-}$
Transferring Transformation	$\xleftarrow{\#}$

Table 7. Types of Transformations

Operators	Target Group CE — Meta-ENS timeless	timed
Conjunction	$(E_1, E_2) \longleftrightarrow (E_1, E_2)_\infty$	$(E_1, E_2) \xleftarrow{+} (E_1, E_2)_T$
Disjunction	$(E_1 \vert E_2) \longleftrightarrow (E_1 \vert E_2)$	—
Sequence	$(E_1, E_2) \xleftarrow{\#} (E_1; E_2)_\infty, t(N(E_1)) < t(N(e_2))$	$(E_1, E_2) \xleftarrow{+} (E_1; E_2)_T$
Negation	—	$\overline{(E_1)} \xleftarrow{-} \overline{(E_1)}_T$
Simultaneity	$(E_1, E_2) \xleftarrow{\#} (E_1 : E_2), t(N(E_1)) = t(N(E_2))$	—
Selection	$(E_1) \xleftarrow{+} (E_1)^{[1]}, (E_1, E_1) \xleftarrow{+} (E_1)^{[2]}, \ldots$	—

Table 8. Target system without time frames: CE - Meta-ENS: E refers to event classes, $N(E)$ to notifications regarding an event in class E , $t(N(E))$ to the time of the event notification

The contributions of this section are (1) a set of profile transformation rules for the interaction of the meta-service with other ENS, (2) auxiliary profile definitions and rules for post-filtering, and (3) notification transformation rules. In this section, firstly we introduce the transformations for composite operators together with auxiliary profiles and post-filtering. Secondly we define transformation rules for event pattern parameters which form the basic building block for our Meta-ENS.

4.2 Profile Transformation of composite operators

This section defines the transformation rules for composite operators. The rules are presented for the transformation of filter expressions defined at the Meta-ENS towards expressions of a target system within a given group (as identified in Section 3.2). We assume the Meta-ENS supports all concepts and event patters introduced in this paper. We now iterate through the five target groups and show the necessary transformations. Due to limitations of space not all refinements of every rule are given in this paper. For further details please contact the authors.

Simple time-frame-less composite events (CE): We give the transformation between the event operators expressed for the Meta-ENS into the target group CE (see Table8). In the Meta-ENS, the operators can be timed (subscript T) or time-frame-less (subscript ∞). If necessary,we also define auxiliary profiles and post-filtering of notifications. The filter expressions of the Meta-ENS are given on the right-hand side, the ones of the source group on the left-hand side of the transformations.

Conjunction, Disjunction, and Negation are almost identical; the transformation is based on the change of time frames. Towards the Meta-ENS, the missing time frame has to be set to ∞. Towards the target system the time frame of the Meta-ENS is lost, which leads to less expressive filter expressions. Negation does not exist without a time frame. Sequence and simultaneity do not exist in this group and have to be simulated. For each $i \in \mathbb{N}$ in selection $E_1^{[i]}$, a separate transformation has to be defined. Alternative transformations for negation and selection are given in Table 10. Note that disjunction, simultaneity, and selection are undefined as timed operators and the negation is undefined as a timeless operators (cf. Section 2).

Operators	Target Group SCE — Meta-ENS timeless	timed
Conjunction	see CE in Table 8	
Disjunction		
Sequence	$(E_1, E_2) \longleftrightarrow (E_1; E_2)_\infty$	see CE in Table 8
Negation	see CE in Table 8	
Simultaneity		
Selection		

Table 9. Target system without time frames: SCE - Meta-ENS: E refers to event classes, $N(E)$ to notifications regarding an event in class E , $t(N(E))$ to the time of the event notification

Sophisticated time-frame-less composite events (SCE): Conjunction, Disjunction, and Negation are similar to the simple time-frame-less version (see Table 9). The sequence operator is now supported and is transformed analogous to the conjunction. For simultaneity, a combination of conjunction, sequence and negation can be used.

Simple time-framed composite events (TCE): Conjunction, Disjunction, and Sequence are directly supported (see Table 10). The selection is realized using transferring transformation. Negation can only be implemented in systems with a time concept; it then uses a transferring transformation.

Ordinary time-framed composite events (OTCE): Conjunction, Disjunction, Sequence, and Simultaneity are similar to TCE (see Table 11). Negation is directly supported. Simultaneity has to be constructed; Selection requires additional filtering in the meta-service.

Sophisticated time-framed composite events (STCE): Almost all operators are supported (see Table 12), only the selection requires a transformation for each $i \in \mathbb{N}$ in selection $E_1^{[i]}$.

Operators	Target Group TCE — Meta-ENS timeless	timed
Conjunction	$(E_1, E_2)_\infty \longleftrightarrow (E_1, E_2)_\infty$	$(E_1, E_2)_T \longleftrightarrow (E_1, E_2)_T$
Disjunction	$(E_1 \vert E_2) \longleftrightarrow (E_1 \vert E_2)$	—
Sequence	$(E_1; E_2)_\infty \longleftrightarrow (E_1; E_2)_\infty$	$(E_1; E_2)_T \longleftrightarrow (E_1; E_2)_T$
Negation	—	$(E_1)_T \overset{\#}{\longleftrightarrow} \overline{(E_1)_T}$ $N = \overline{N(E_1)_T}$
Simultaneity	$(E_1, E_2) \overset{\#}{\longleftrightarrow} (E_1 : E_2), t(N(E_1)) = t(N(E_2))$	—
Selection	$(E_1) \overset{\#}{\longleftrightarrow} (E_1)^{[i]}, N = (N(E_1))^{[i]}$	—

Table 10. Target system with time frame support: TCE - Meta-ENS: E refers to event classes, $N(E)$ to notifications regarding an event in class E , $t(N(E))$ to the time of the event notification

Operators	Target Group OTCE — Meta-ENS	
	timeless	timed
Conjunction	see TCE in Table 10	
Disjunction		
Sequence		
Negation	—	$\overline{(E_1)_T} \longleftrightarrow \overline{(E_1)_T}$
Simultaneity	see TCE in Table 10	
Selection	see TCE in Table 10	

Table 11. Source system with time frame support: OTCE - Meta-ENS: E refers to event classes, $N(E)$ to notifications regarding an event in class E , $t(N(E))$ to the time of the event notification

Operators	Target Group STCE — Meta-ENS	
	timeless	timed
Conjunction	see TCE in Table 10	
Disjunction		
Sequence		
Negation	see OTCE in Table 11	
Simultaneity	$(E_1 : E_2) \longleftrightarrow (E_1 : E_2)$	—
Selection	see TCE in Table 10	

Table 12. Source system with time frame support: STCE - Meta-ENS: E refers to event classes, $N(E)$ to notifications regarding an event in class E , $t(N(E))$ to the time of the event notification

4.3 Transformation of Operator Parameters

The group definitions given in Section 3.3 abstracted from the parameters of consumption mode and duplicate handling strategy since these parameters are rarely explicitly supported in the considered systems. In Table 13, we show the influence of considering parameter transformations on operator transformations (as introduced in the previous section). We show which transformation are possible when translating the parameter set of one system (y-axis in Table 13) into the parameter set of another system (x-axis in Table 13). That is, for all possible combinations of the duplicate and selection parameter we state whether the transformation is not possible (indicated by a dash) or equivalent (indicated by $\longleftrightarrow$) or only possible in one of the given directions while creating a larger result set (indicated by $\xleftarrow{+}$ and $\xrightarrow{+}$, where the arrow orientation defines the direction of the possible transformation).

This section presented our answer to the second problem stated in the introduction: translating filter expressions between languages that use different operators and semantics (Problem 2). We provided a set of transformation rules that form the core of the proposed Meta-ENS for integrating heterogeneous event notification services. The transformation rules presented here have also been implemented in a prototype transformation component that can be used with any given ENS .

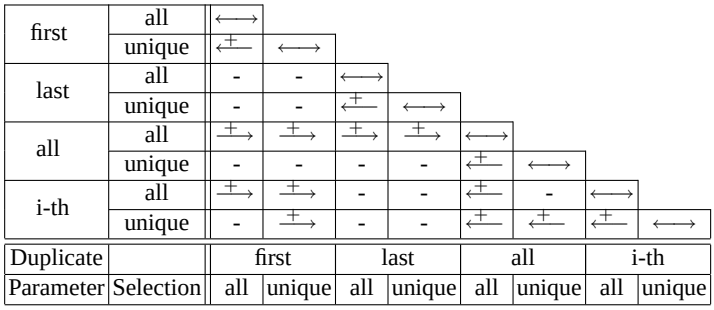


Table 13. Parameter transformations: Duplicate handling and consumption mode parameter

5 Conclusion and Outlook

In this paper, we proposed the concept and design of a Meta Service for Event Notification. In detail, we presented the answers to the following two research problems: Firstly, subscribers of heterogeneous event notifications services are forced to subscribe the same profile to a number of services using different filter languages. Secondly, composite events combining events from different providers that are handled by different services have to be identified by a subscriber-based post-filtering.

As a solution to these two problems we proposed the detailed design of a Meta-Event Notification Service based on transformation rules. In particular, this paper presented the following contributions: Firstly, we presented a survey of filter languages for event notification. Secondly, we introduced a classification schema for profile definition languages. Thirdly, we identified five categories of profile languages. Fourthly, we proposed detailed transformation rules for translating profiles defined at the Meta-ENS into languages of system from the five categories (and vice versa for notifications). An extended description of our findings can be found in [13].

As proof of concept, we have implemented a transformation component for the proposed language transformations. The implementation was carried out using Prolog. The transformation component currently supports the operator transformations. The next version of the transformation component will incorporate the proposed parameter transformation. Future research will see the close integration of the transformation component into our prototypical event notification system A-mediAS [9]. The transformation can be used for the role of a Meta-ENS in the communication with other ENS (as providers) and for the mediation between ENS (as providers and subscribers).

References

1. S. Brandt and A. Kristensen. Keryx: Internet notification service for dynamic web applications. (slide presentation), presented to W3C, 1997.
2. A. Carzaniga, D. Rosenblum, and A. Wolf. Interfaces and algorithms for a wide-area event notification service. Technical Report CU-CS-888-99, University of Colorado, Department of Computer Science, 1999.

3. C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate rewriting for translating boolean queries in a heterogeneous information system. *ACM Transactions on Information Systems (TOIS)*, 17(1):1–39, 1999.
4. D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes - A notification service for digital libraries. In *Proc. of the ACM JCDL*, Roanoke, VA, 2001.
5. L. Fiege, G. Mühl, and F. C. Gärtner. A modular approach to build structured event-based systems. In *Proc. of the ACM SAC Symposium on Applied Computing*, Madrid, Spain, 2002.
6. A. Geppert and D. Tombros. Event-based distributed workflow execution with EVE. Technical Report ifi-96.05, University Zurich, Computer Science Department, 1996.
7. R. Gruber, B. Krishnamurthy, and E. Panagos. The architecture of the READY event notification service. In *Proc. of the IEEE ICDC Middleware Workshop*, Austin, TX, 1999.
8. R. Gruber, B. Krishnamurthy, and E. Panagos. CORBA notification service: Design challenges and scalable solutions. In *Proc. of the IEEE ICDE*, Heidelberg, Germany, 2001.
9. A. Hinze. *A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service*. PhD thesis, Freie Universitaet Berlin, Department of Computer Science, July 2003.
10. A. Hinze and A. Voisard. Composite events in notification services with application to logistics support. Technical Report tr-B-02-10, Freie Universität Berlin, Department of Computer Science, 2002.
11. A. Hinze and A. Voisard. A parameterized algebra for event notification services. In *Proc. of Symposium on Temporal Representation and Reasoning*, Manchester, UK, 2002.
12. Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.
13. D. Jung and A. Hinze. Analysis and transformation of profile definition languages for event notification services. Technical Report 12/2004, Computer Science Department, University of Waikato, New Zealand, August 2004.
14. L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):610–628, 1999.
15. L. Liu, C. Pu, W. Tang, and W. Han. Conquer: A continual query system for update monitoring in the WWW. *International Journal of Computer Systems, Science and Engineering*, 14(2):99–112, 1999.
16. C. Ma and J. Bacon. COBEA: A CORBA-based event architecture. In *Proc. of the COOTS Conference on Object-Oriented Technologies and Systems*, Berkeley, CA, 1998.
17. Jeff Magee, Naranker Dulay, and Jeff Kramer. A Constructive Development Environment for Parallel and Distributed Programs. In *In Proc. of the International Workshop on Configurable Distributed Systems*, Pittsburgh, March 1994.
18. G. R. Malan, F. Jahanian, and S. Subramanian. Salamander: A push-based distribution substrate for internet applications. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, Monterey, California, 1997.
19. M. Mansouri Samani and M. Sloman. GEM: A generalised event monitoring language for distributed systems. *IEE/IOP/BSC Distributed Engineering Journal*, 4(2):96–108, 1997.
20. D. Mishra. *Snoop: Am Event Specification Llanguage for Active Database Systems*. Masters thesis, University of Florida, 1991.
21. Douglas W. Oard. A comparative study of query and document translation for cross-language information retrieval. In *AMTA*, pages 472–483, 1998.
22. B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proc. of the AUUG Australian UNIX and Open Systems User Group Conference*, Queensland, Australia, 1997.
23. T. W. Yan and H. Garcia-Molina. Duplicate removal in information dissemination. In *Proc. of the VLDB*, Zurich, Switzerland, 1995.
24. D. Zimmer and R. Unland. On the semantics of complex events in active database management systems. In *Proc. of the IEEE ICDE*, Sydney, Australia, 1999.