Chapter 8

Implementing Adaptivity

To improve is to change;
to be perfect is to change often
Winston Churchill

As illustrated in the starting chapters of this thesis, new applications have emerged that call for an event notification service that flexibly supports various and changing applications and sources, as well as efficient filter algorithms that keep high performance under changing system load. The preceding chapters addressed selected aspects of an integrating service: adaptable semantics for composite events, influence of observation and time-stamping on the accuracy of event composition, and performance for the filtering of primitive and composite events. In this chapter, we address the question of how to implement *adaptivity* in an integrating ENS. As emphasized before, adaptation is needed, e.g., for the semantics of user profiles and for the employed filter algorithms. We distinguish *qualitative* and *quantitative* adaptivity.

In Section 8.1, the implementation issues for qualitative adaptation are discussed – in particular, for semantic variations and accuracy awareness in event composition (introduced in the chapters 5 and 6). We introduce the concepts of *source profiles* and *application profiles*. We discuss how these additional profiles can be used to adapt the filter characteristics to changing sources and applications. Here, we concentrate on the conceptional level of qualitative adaptivity. Our implementation of a basic approach is described in the next chapter; its application is illustrated in a case study in Chapter 10.

In Section 8.2, the implementation issues for quantitative adaptation are discussed. We study the adaptation of our filter algorithms for primitive and composite events (introduced in Chapter 7). Special focus is given to the detection of the distributions of attribute values in events and profiles. Finally, we extend the ENS architecture presented in our reference model (cf. Chapter 3) with an *analyzer and optimizer part* that controls the adaptation processes described here.

8.1 Qualitative Adaptation

In this section, we discuss strategies for adapting the event composition to changing application requirements (*qualitative adaptation*). In Chapter 5, we introduced our event algebra that supports a flexible handling of event composition. We now show how this algebra can be used to adapt the event composition. First, the shortcomings of simpler solutions for adaptation are emphasized. Then, we analyze the factors influencing the event filter semantics in detail. Finally, we propose an event filtering strategy that takes of characteristics of event sources, clients, and applications into account. This strategy also allows for consideration of composition accuracy as introduced in Chapter 6.

We consider the following parameters as discussed in Chapter 5:

EIS – event instance selection
EIC – event instance consumption
EET – event evaluation time

In other systems, a rigid filter behavior is defined implicitly either by some (often inconsistent) language constructs or by the filter implementation. Our parameterized algebra flexibly allows for changes of profiles depending on the context. Thus, as we shall see, our system supports integrating event-based applications and changing event sources without the need to redefine client profiles. The following example illustrates the danger of rigid filter behavior. Subsequently, we discuss our adaptation strategy.

Example 8.1 (Rigid Profile Evaluation)
An example profile from our facility management scenario is 'Notify if the temperature rises above $35°C$ *within 7 days after a failure in the air conditioning system.' Using our event algebra, this profile could be defined as follows:*

$$\Big(\big(all\_dup(E_1), first\_dup(E_2)\big)_{7days}\Big)_{all\_pairs} \text{ with } EIC = final$$

where, E_1 *refers to the class of failure events in the air conditioning and* E_2 *refers to the class of temperature events as required for the profile. Here, we focus on the detection of the temperature events. The events are provided by two different sources A and B covering the buildings A and B, respectively (see Figure 8.1). Source A employs a surveillance sensor that sends the current temperature value every 30 seconds, Source B employs a warning system that alerts the system in case of critical temperature changes.*

To be notified only of critical changes, the client would have to define separate profiles for each source: In the Profile p_A *for Source A, only the* first *event in a series of duplicate events is necessary. In the Profile* p_B *for Source B,* all *events are of interest. Defining only Profile* p_A *leads to missed events from Building B, defining only Profile* p_B *leads to false alarm in Building A.*

Simple approach. A simple and direct approach to event filtering evaluates the client profiles directly as they have been defined by the clients. Using this approach, clients are notified about events that match their profile queries. However, we have argued that these events may carry different semantics, which may lead to false notifications. Alternatively, the client may have to separately define profiles for each of

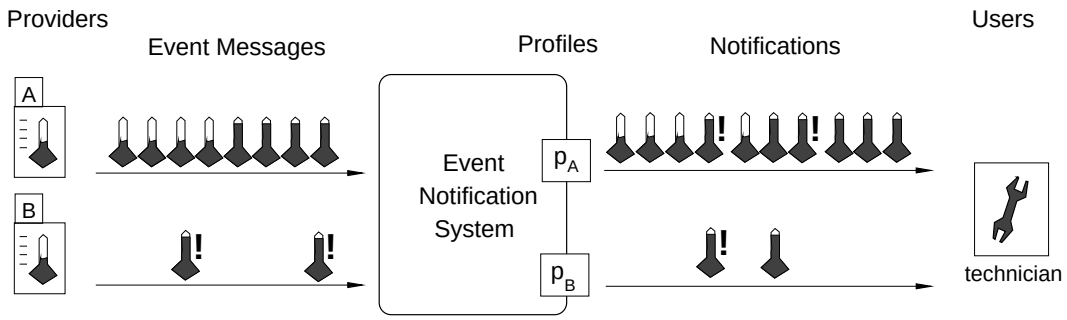

Figure 8.1: Illustration of Example 8.1 for false notifications

the available event sources and applications. Because of the large numbers of sources and applications and their possible changes, this alternative would overburden the client. Therefore, the simple approach is not sufficient and a different handling of profiles for each source and application context is needed.

Wrapper approach. The established approach for the integration of information from inhomogeneous sources (e.g., for data warehouses [CGL+98]) is the employment of wrapper objects. At each event source, a wrapper is used to translate the event messages according to common event semantics for the service. For our context, wrappers can be used for ENS that have a *fixed* profile evaluation strategy. In this case, wrappers are a conceivable solution. However, the following disadvantage remains: For systems with changing application characteristics as considered here, new wrappers would have to be provided for each source and application.

Our solution. We propose a more flexible solution – the employment of three profile types: *client profiles* as introduced before, *source profiles* to describe the available event types, and *application profiles* to capture the applications' characteristics (see Figure 8.2). Now, for each system context (e.g., available sources and applications), a specific profile evaluation strategy may be applied. Subsequently, we describe the factors influencing event composition and present our adaptation approach.

8.1.1 Influences on Event Composition

The semantics of a composite-event profile defined by a client is not fixed, but depends on event sources and application context. We identify the following system players: *event sources* as offered by the providers, *clients*, and *applications*. Here, we analyze the influence of each player on the event composition and use the results for our adaptation approach.

Event Source. Following our analysis in Chapter 6, we distinguish active and passive observation of events, which may result in different event semantics. Active observation may lead to *direct duplicates*, i.e., a number of events that have identical attribute values except for the time-stamp attribute. For passive observation, some event sources may only forward events if the current value differs significantly from an earlier one, i.e., these providers employ additional filters and therefore prevent direct duplicates. Duplicate events that belong to the same event class but do not have all identical attribute values, we call *semantic duplicates*. In Example 8.1, Provider A delivers direct and semantic duplicates, Provider B

delivers only semantic duplicates. Hence, the characteristics of the event source profile directly influence the effect of the parameter for event instance selection (EIS).

Characteristics of the event observation and timestamping determine the accuracy of event information provided by an event source, e.g., the accuracy of the time-stamp. For passive observation, an event source may offer information about the event detection accuracy. The event sources can only describe specifics of primitive events; composite events are constructed within the event notification service. Therefore, the event sources do not influence the effect of the parameter for event instance consumption (EIC).

Client Interest and Preferences. The client's interest and preferences are specified within the client profile. Client profiles may lead to notifications about duplicate events, i.e., events that are elements of the same event class as defined by the client profile. Events may be duplicates according to certain client profiles. Note that these duplicate events are not necessarily direct duplicates.

The client profile influences the effect of the parameter for event instance selection EIS in two ways: First, the parameter is defined for each referenced event within the profile. Second, the notion of duplicates (which are referenced to by the parameter EIS) is based on the profile query, which defines an event class. The parameters for event instance consumption EIC and event evaluation time EET may be defined within the client profile for each composite event.

The client profile may specify the required accuracy of event information, e.g., that information about a sequence of events must not exceed a given tolerance range.

ENS Application Scenario. The evaluation strategies for composite events differ depending on the application field. For example, in a surveillance application, anomalous events are of high interest while for a long-term monitoring application, every event has to be logged. Thus, the typical settings for the parameters (e.g., EIS, EIC, EET) depend on the application field. These application characteristics can be only partly obtained in a formalized way, e.g., by deriving typical parameter settings for certain event types based on the source type. Other information, such as preferred or trustworthy event sources and composition parameter settings, may only be available to an application expert.

In summary, the three system players differently influence the event composition: All three influences have to be considered.

8.1.2 Adaptive Event Composition

We now discuss how additional profiles can be used to adapt the filter characteristics to changing sources and applications. Information about the nature of the observed events that are provided by a given event source (e.g., a temperature sensor) may be defined in a *source profile* (see Figure 8.2).

Definition 8.1 (Source Profile)
A source profile describes the event types available at a particular source. Additional information may refer to the characteristics of the event observation, such as observation method and frequency.

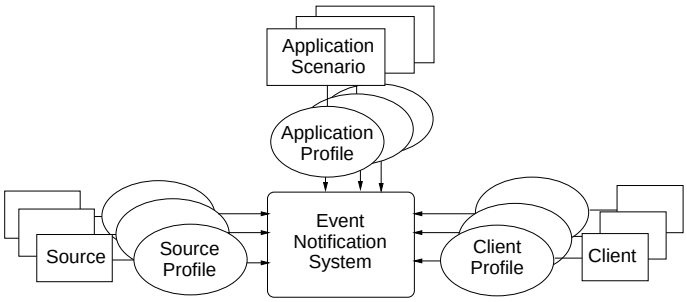

Figure 8.2: System players and their profile types

For each event source, a source profile has to exist at the event notification server. These profiles enable clients to subscribe to events provided by the service – preventing client profiles regarding events that are not covered by the service.

To prevent that a client has to define multiple client profiles (to cover each source) that have to be updated whenever a new source joins the service, a translation between client profiles and source profiles is needed. This translation may be controlled by an application expert, e.g., in defining translation rules. This is especially useful for the inclusion of new sources with different characteristics. Additionally, the translation may cover application-specific aspects, such as tolerances, accuracy bounds, and typical composition parameters.

We introduce the concept of *application profiles* to store information about the filter semantics required by applications. An application profile contains the transformation rules to translate profiles that have been defined for the characteristics of one specific source into profiles for other source characteristics.

Definition 8.2 (Application Profile)
An application profile is a set of query transformation rules for translating queries (client profiles) according to source profiles into new queries.

Application profiles are inspired by concepts from query transformation in database systems (see, e.g., [RCD94]) and predicate rewriting in Information Retrieval [CGMP99] The transformation of client profiles changes the event message streams directed to the clients. The queries are translated such that for different source profiles, a query with equivalent semantics is created for each of the available sources (see Figure 8.3). Using our parameterized algebra, most transformation rules only require the application of new parameter values. The parameterized form of our algebra supports the flexible adaptation of the filter.

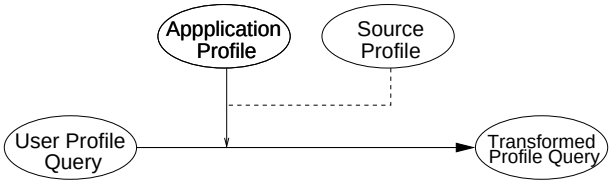

Figure 8.3: Client profile transformation

One approach is to define each client profile with an explicit reference to a certain source type. Then, an application profile defines $N_s \times (N_s - 1)$ transformation rules for translating the queries for each source accordingly (N_s refers to the number of distinct source profiles). This approach is unfavorable, because the definition of the numerous profile variations is left to the clients. Additionally, new applications and sources may not be known in advance.

We follow another approach that is more appropriate for realistic application contexts: For each application, a typical (default) source type is defined by the application expert. It is assumed that all client profiles of the application are defined relative to that default source type. Based on this default source type for the profiles, $(N_s - 1)$ transformation rules are defined in the application profile. In both cases, the transformation rules depend on the application context; an automatic creation of the rules based on the source profiles is only feasible for selected and simple characteristics.

As argued in the previous section, the application of the event instance selection parameter EIS depends on client preferences, available source types, and the application. The application of the parameters for event instance consumption EIC and evaluation time EET depend on client preferences and system restrictions only: Due to limited system resources, the buffer of event instances that await matching composite partners may have to be restricted or their evaluation may be postponed.

In this section, we described the conceptional level of profile adaptation. We implemented the concept introduced here in two versions. In our prototype, we implemented a *basic approach suitable for facility management applications*. The application and effectiveness of this approach is illustrated in a case study in Chapter 10. Additionally, we studied profile transformations for *integrating event information from external ENS systems*. Our integrating service may use other ENS as 'intelligent' event sources by forwarding client profiles to these services. The event operators supported by these services may differ from our service. Therefore, the profiles have to be translated. For example, a sequence profile may have to be translated into a conjunction to be processed by a service that only supports simple composite profiles. Similarly, the notifications sent by these services have to be processed, e.g., the notifications announcing two conjunctive events have to be filtered such that only valid sequences remain. We implemented this mechanism in a separate prototype [Jun03].

8.1.3 Influencing Composition Accuracy

As shown in Chapter 6, the accuracy of the temporal event information (i.e., the timestamps) depends on the event observation methods and the time system of the event sources. Therefore, event sources have to provide information about the used observation and timestamping methods within the *source profile* (source meta-data). The accuracy requirements defined by the *client profiles* require the system to calculate the accuracy of composite event information. As discussed in Chapter 6, a strategy for handling uncertain event information has to be implemented. For different applications, different accuracy restrictions may hold. Therefore, the handling of composition accuracy has to be controlled by the *application profile*. Notification about composite events that exceed the tolerance range defined in the client profile may have to be discarded or the notification has to be accompanied by a warning.

8.2 Quantitative Adaptivity

The attribute values of events and profiles are not uniformly distributed over time: Critical events, such as a system failure, typically occur with varying frequency and less often than regular events. In the contrary, client profiles refer more often to critical events than to regular ones. In Chapter 7, we have shown how these characteristics can be used to improve the performance of primitive and composite-event filtering. In this section, we discuss strategies for obtaining information about value distributions of events and profiles and for adapting the event filter process to these distributions in order to gain maximum performance (*quantitative adaptation*).

8.2.1 Measuring the Distributions

The distributions of profiles and events are either known to the application administrators or they have to be observed. If the system load does not change over time, i.e., the application's profile and event distributions remain relatively stable, these application characteristics can be used to directly adjust the system for best performance.

The distributions may change when (a) the application changes, (b) a different use case is temporarily applied, or (c) the system load (events and profiles) changes. The system load may change regularly based on certain external conditions. For example, the number of people at a festive evening provides a peak for the event observation in the security system of a building. The cases (a) and (b) require only basic adaptation to new distributions of profiles and events. The third case requires detailed consideration.

If the system load changes over time (c), the distributions of events and profiles have to be observed. For the profiles, this is a relatively simple task, because the required information is directly available at the system side. In a distributed system, for each of the event servers, the distribution of the profile values within each attribute domain have to be analyzed. For a detailed profile analysis, a profile history has to be maintained. For the events, the distributions of event values within the attribute domains have to be observed by collecting and evaluating the event information in a persistent history of events.

The storage and evaluation of the history of events and profiles may directly influence the system's performance due to the performance costs for event persistency, the method of distribution measurement, and the handling of the algorithm adaptation.

How shall we determine the current event distribution at runtime? The events and profiles may be stored as a history and analyzed later, e.g., using techniques from data mining [FPSSU95] and query optimization [PHIS96], and adaptive query rewriting [IFF+99]. An option is to store a certain number of events in main memory with regular access to persistent storage during times of low system load. Alternatively, events and profiles may be directly analyzed for their value distributions, which causes a certain performance overhead [MF02, VN02].

For both cases, the distributions are not to be computed over the complete system runtime but for selected intervals. The length of the interval determines the topicality of the computed values but carries the danger of over- or undervaluation of load changes. The length of the interval has to be defined by the application administrator.

8.2.2 Adapting the Algorithms

The filter algorithms have to be adapted, if the distributions of events and profiles differ more than a certain epsilon from distributions measured before. To prevent oscillations in an adaptive system, the evaluation interval for the distribution evaluation has to be defined with sufficient length. Additionally, a temporal hysteresis and a threshold may be defined: Changes in the algorithms are only realized after the new conditions hold for at least the length of the hysteresis value or the changes are larger than the threshold. Reorganization of the tree may be applied to an off-line tree which is then exchanged into the running system.

8.3 Extended System Architecture

In this section, we propose an extension of the ENS architecture introduced in our reference model (cf. Chapter 3, Page 28). We introduce an analyzer and optimizer part and an extended profile repository to additionally store source and application profiles. The new architecture is presented in Figure 8.4.

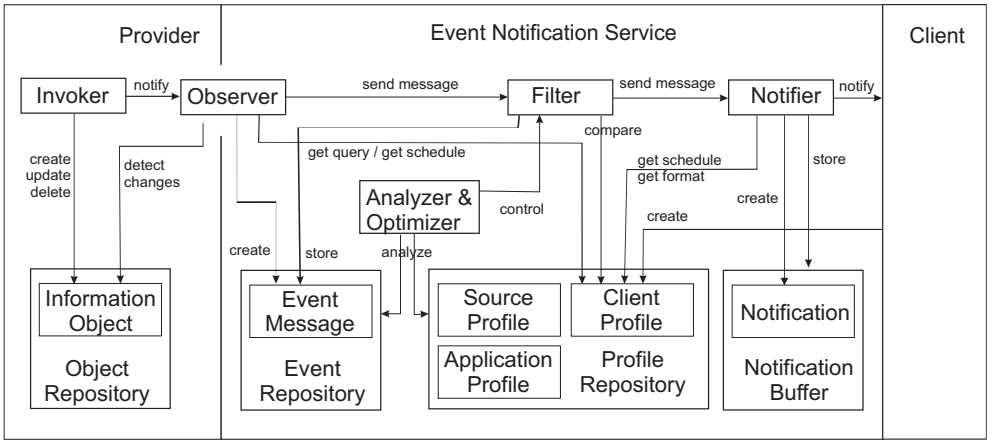

Figure 8.4: Extended reference model for adaptive event notification systems

The *extended profile repository* stores three kinds of profiles: client profiles as defined in the original model, source profiles, and application profiles. The definition of the application profile by an application expert is not shown in the model.

The *analyzer & optimizer part* analyzes the event messages and user profiles in the repositories for their distributions. Based on this analysis, it may trigger a change of the filter algorithms.

The event detection and filtering process is driven by the observer: Whenever an event message arrives at the passive observer or is detected by the active observer, the message is parsed, a timestamp is attached to it, and the event is forwarded to the filter engine. The information about the event is additionally logged by the analyzer. The core of the system is the interplay of analyzer component and filter engine. The algorithms used in the filter engine are controlled by the analyzer output.

The analyzer has access to information about events and profiles: It logs the event history and has access to the profile tree. Based on this information, the analyzer computes the distributions of events

and profiles. This information is made available to the system administrator or to an optimizer component. The administrator may trigger the adaptation of the filter strategy at the analyzer component. The adaptation can be automatically triggered by the system (via the optimizer). Then, the optimizer controls the necessary reordering of the filter tree for primitive-event filtering as well as changed strategies for composite event evaluation. Once the matching profiles have been identified for an event message, a notification is sent to the profile's client. In Chapter 10, we present test results and case studies that show the effectiveness of the qualitative and quantitative adaptation introduced here.

8.4 Related Work

As discussed in Chapter 4, none of the existing ENS provide the required flexible adaptation to application characteristics. Some systems use related concepts: Simple source profiles have been used to advertise event types in Siena [CRW01] and Elvin [SAB+00]. However, the source profiles in Elvin and Siena are used only to reject those events to which no client has subscribed and to prevent profiles without matching event types.

The notion of adaptation is a well known concept in software development for the conversion of interfaces [GHJV95]. Two design patterns are used in this context: `adapter` and `proxy`. An adapter converts between different types and leaves the semantics unchanged, while a proxy modifies the semantics of a call without changing its type. In our work, qualitative adaptation acts similarly to a proxy (in the application profile) while quantitative adaptation uses an adapter object to change between filter algorithms (in the analyzer). Automatic selection of adaptors gains increasing attention in mobile computation and ad-hoc networks [Vay01].

Recently, methods for user adaptivity have gained strengthened attention: Client profiles are built based on the user action (see, e.g., [BM02]). Such methods could be included in an ENS, but are not the focus of our work.

8.5 Summary

This chapter analyzed methods for implementing adaptivity in an ENS. We proposed two methods to adapt a system to changing application context and new sources. First, the translation of client queries according to application profile and source profiles has been described. In a similar way, accuracy tolerances can be taken into account for event composition. Second, the selection of the filter algorithms for both primitive and composite events depends on the distribution of events and profiles. We discussed strategies to determine these distributions and adapt the algorithms. We extended the reference model of ENS with both an analyzer & optimizer part that controls the system's adaptation and an extended profile repository.

Selected methods for adaptation have been implemented in our A-mediAS system. The system's implementation is described in the next chapter. The effectiveness of the qualitative adaptation of profiles in A-MEDIAS is shown in a case study in Chapter 10. The effect of the quantitative adaptation of the filter algorithms to changing event and profile distributions is also demonstrated in Chapter 10.