Chapter 12
Aiding the Development of Active Applications: A Decoupled Rule Management Solution

Florian Daniel
University of Trento, Italy

Giuseppe Pozzi
Politecnico di Milano, Italy

ABSTRACT
Active applications are characterized by the need for expressing, evaluating, and maintaining a set of rules that implement the application’s active behavior. Typically, rules follow the Event-Condition-Action (ECA) paradigm, yet oftentimes their actual implementation is buried in the application code, as their enactment requires a tight integration with the concepts and modules of the application. This chapter proposes a rule management system that allows developers to easily expand its rule processing logic with such concepts and modules and, hence, to decouple the management of their active rules from the application code. This system derives from an exception manager that has previously been developed in the context of an industry-scale workflow management system and effectively allows developers to separate active and non-active design concerns.

INTRODUCTION
Until the emergence of the first operating systems and high-level programming languages allowed developers to disregard hardware peculiarities, computers had to be programmed directly in machine code. Then, only in the eighties, Database Management Systems (DBMSs) provided efficient, external data management solutions, and in the nineties Workflow Management Systems (WfMSs) extended this idea and extracted entire processes from still rather monolithic software systems. We believe that in similar way also active (also known as reactive) behaviors, which are present in many modern applications (see for instance Section 2), can be more efficiently managed by proper active software supports, such as active rules and rule engines (Section 3).

The basic observation underlying this idea is that, when abstracting from the particular application and domain, most of the active behaviors in software systems adhere to the rather regular and
stable ECA (Event-Condition-Action) paradigm. ECA rules have first been introduced in the context of active DBMSs, where operations on data may raise events, conditions check the status of the database, and actions perform operations on data. Our previous experience in the field of WfMSs (Casati, Ceri, Paraboschi, and Pozzi, 1999; Combi and Pozzi, 2004) allowed us to successfully apply high-level ECA rules to WfMSs for the specification and handling of expected exceptions that may occur during process execution. By leveraging this experience, in this paper, we propose an ECA paradigm accompanied by a suitable rule language, where events represent data, temporal, application or external events, conditions check the state of data or of the application, and actions may act on data, applications, or external resources. Active rules may thus not only refer to the data layer, but as well to the whole application, comprising data and application-specific characteristics. Elevating active rules from the data layer to the application layer allows designers to express a broader range of active behaviors and, more importantly, to address them at a suitable level of abstraction (Section 4). This could turn out beneficial for example in requirements engineering approaches, such as the ones described by Loucopoulos and Kadir (2008) or by Amghar, Meziane, and Flory (2002), as well as in re-engineering approaches like the one described in Huang, Hung, Yen, Li, and Wu (2006).

For the execution and management of ECA rules, we further propose an open ECA server (OES), which runs in a mode that is completely detached from the execution of the actual application, so as to alleviate the application from the burden of event management. OES is highly customizable, which allows developers to easily add application- or domain-specific features to the rule engine (Section 5 describes the customization process, Section 6 illustrates a use case of the system). Instead of implementing the OES system from the scratch, we shall show how we unbundled and reconfigured the necessary components from a previously developed exception manager for a WfMS (Casati et al., 1999) (Section 7) – unbundling is the activity of breaking up monolithic software systems into smaller units (Gatziu and Koschel, 1998). We thus move from the ECA server we developed within the EC project WIDE to manage exceptions in the context of Sema’s FORO commercial WfMS, where the exception manager (FAR) was tightly bundled into FORO.

**Rationale and Background**

Active mechanisms or behaviors have been extensively studied in the field of active DBMSs as a flexible and efficient solution for complex data management problems. Many of the results achieved for relational or object-oriented active databases have recently been extended to tightly related research areas such as XML repositories and ontology storage systems. To the best of our knowledge, only few works (Dittrich, Fritschi, Gatziu, Geppert, and Vaduva, 2003; Chakravarthy and Liao, 2001; Cugola, Di Nitto, and Fuggetta, 2001) try to elevate the applicability of active rules from the data level to the application level and to eliminate the tedious mapping from active behavior requirements to data-centric active rules (Section 8 discusses related works in more detail). Besides DBMSs, there are several application areas, which could significantly benefit from an active rule support that also takes into account their application- or domain-specific peculiarities. Among these application areas, we mention here:

- WfMSs or in general business process management systems allow one to define the system-assisted execution of office/business processes that may involve several actors, documents, and work items. Active mechanisms could be exploited for an efficient enactment of the single