Event monitoring and active behaviors are important aspects in many software systems and application domains, not only in database management systems. In this article, we propose an event-condition-action (ECA) approach that spans from application data to application components and behaviors. Starting from an exception manager we previously developed in the context of a workflow management system, we derived an autonomous active component capable of handling a variety of events and enacting actions in response to detected events. The ECA server runs as an autonomous engine and can be seamlessly integrated with existing systems, thus enhancing the systems’ functionalities and maintainability by separating active and nonactive design concerns.

Keywords: active application; active rule; autonomous ECA server; database design; ECA rule; event monitoring

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 ’80s, database management systems (DBMSs) provided efficient, external data management solutions, and in the ’90s, workflow management systems (WfMSs) extended this idea and extracted entire processes from still rather monolithic software systems. We believe that in a similar way, active behaviors (also known as reactive), which are present in many modern applications, can be more efficiently managed by proper active software supports, such as active rules and rule engines.

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 were first 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, & Pozzi, 1999; Combi & 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 article, 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 to the whole application as well, 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. This could be 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 reengineering 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) that 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. (The fifth sections describes the customization process, and the sixth illustrates a use case of the system.) Instead of implementing the OES system from 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); unbundling is the activity of breaking up monolithic software systems into smaller units (Gatziu, Koschel, von Bultzingsloewen, & Fritschi, 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, FORO Active Rules) 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 (OO) active databases have recently been extended to tightly related research areas such as XML (extensible markup language) repositories and ontology storage systems. To the best of our knowledge, only few works (Chakravarthy & Liao, 2001; Cugola, Di Nitto, & Fuggetta, 2001; Dittrich, Fritschi, Gatziu, Geppert, & Vaduva, 2003) try to elevate the applicability of active rules from the data level to the application level and eliminate the tedious mapping from active behavior requirements to data-centric active rules. (The eighth section discusses related works in more detail.) Besides DBMSs, there are several application areas that 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 the following.

- WfMSs or, in general, business process management systems allow one to define the system-assisted execution of office and business processes that may involve several actors, documents, and work items. Active mechanisms could be exploited for an efficient enactment of the single tasks or work items, and the management of time constraints during process execution (Combi & Pozzi, 2003, 2004).
- Web services and applications, which use Web services as data sources or incorporate their business logic (Li, Huang, Yen, & Chang, 2007), may rely on an asynchronous communication paradigm where an autonomous management of incoming and outgoing events (i.e., messages) is crucial. Suitable active rules could ease the integration of Web services with already existing applications. Active rules could further serve for the coordination of service compositions, similar to the coordination of actors and work items in a WfMS (Charfi & Mezini, 2004; Daniel, Matera, & Pozzi, 2006).
- Exception handling is gaining more and more attention as a cross-cutting aspect