Content Is Capricious: A Case for Dynamic System Generation

Hans-Werner Sehring, Sebastian Bossung, and Joachim W. Schmidt

Software Systems Institute (STS)
Hamburg University of Science and Technology (TUHH)
{hw.sehring, sebastian.bossung, j.w.schmidt}@tuhh.de

Abstract. Database modeling is based on the assumption of a high regularity of its application areas, an assumption which applies to both the structure of data and the behavior of users. Content modeling, however, is less strict since it may treat one application entity substantially differently from another depending on the instance at hand, and content users may individually add descriptive or interpretive aspects depending on their knowledge and interests. Therefore, we argue that adequate content modeling has to be open to changes, and content management systems have to react to changes dynamically, thus making content management a case for dynamic system generation.

In our approach, openness and dynamics are provided through a compiler framework which is based on a conceptual model of the application domain. Using a conceptual modeling language users can openly express their views on the domain’s entities. Our compiler framework dynamically generates the components of an according software system. Central to the compiler framework is the notion of generators, each generating a particular module for the intended application system. Based on the resulting modular architecture the generated systems allow personalized model definition and seamless model evolution.

In this paper we give details of the system modules and describe how the generators which create them are coordinated in the compiler framework.

1 Introduction

Most data-intensive applications serve, one way or another, as information systems (ISs) and call for some kind of persistence technology. High volumes of data and large user communities require additional functionality (query support, concurrency, recovery etc.) which nowadays comes nicely packaged as off-the-shelves database models and database technology.

Database modeling is rather strict in the sense that it is based on the assumption of a high regularity of its application areas. This assumption applies to both the structure of data and the behavior of users. Therefore, database models rest on a small set of agreed upon computational base types (numbers, strings, . . . ) and a few structuring mechanisms (mostly records and sets) used to design schemata shared by the entire application and its community. In an
enterprise database, for example, the view on a company employee is defined before employee records are instantiated, and users of the database have to share the company’s view.

Content modeling, however, is more capricious since it may treat each represented entity substantially differently depending on the instance at hand, and content users may individually add descriptive or interpretive aspects depending on their knowledge and interests. For example, when considering a particular piece of art, some users may be interested in the artist who created it, the material used and the prices achieved while others are more concerned about details on the period in which it was created, its meaning, etc.

Therefore, we argue that adequate content models have to be open to changes, and content management systems have to be dynamic to reflect such model changes. In other words, content management systems are seen as a case for dynamic system generation while database management system usually get away with the technically less ambitious case of generic system implementation.

ISs inherit the restrictions of fixed schemata and a uniform user community from the underlying database technology. The development of ISs usually accommodates to these restrictions: in an intensive phase of domain analysis the database schema is defined once and for all. Application logic and presentation are implemented with respect to the schema and the domain model. Because IS implementation relies heavily on certain schema information, later changes to this schema affect all parts of an IS, an aspect nearly prohibitive to any effort of dynamic system evolution or to any attempt of system personalization.

For content management systems such inflexibility cannot be tolerated. Content is viewed by users in different contexts with individual conceptual models in mind. Furthermore, users have to be able to define suitable models or adapt existing ones during the lifetime of a content management system. Therefore, model changes have to be integrated dynamically, without additional development steps which include manual intervention.

In our approach, openness and dynamics of content management systems is provided through a compiler framework which is based on a conceptual model of the application domain. In our conceptual modeling language users can openly express their views on the domain’s entities, and based on such views our compiler framework dynamically generates the components of the implementing software system. Central to the compiler framework is the notion of generators, each generating a particular module for the application system and collectively implementing the intended application. Based on the resulting modular architecture the generated system allows personalized model definition and seamless model evolution.

In this paper we give details about the system modules as well as the generators which create them. We also describe how the generators are interconnected in the compiler framework.

The paper is organized as follows: in sec. 2 we give a brief account of conceptual modeling for content-intensive application systems and of contemporary approaches to model-based system generation. The additional support required