

Operationalizing Application Descriptions in DTC: Building Applications with Generalized Markup Technologies ¹

J.L. Sierra, A. Fernández-Valmayor, B. Fernández-Manjón, A. Navarro
 Dpto. Sistemas Informáticos y Programación, Facultad de Informática,
 Universidad Complutense de Madrid, Avda. Complutense S/N, 28040 Madrid, Spain
 {jlsierra, alfredo, balta, anavarro}@sip.ucm.es

Abstract: This paper describes the *operationalization process* (i.e. the step from application descriptions to executable applications) followed in DTC (structured Documents, document Transformations and software Components), an approach to develop applications using generalized markup technologies. DTC encourages the definition of XML-based *domain-specific languages* (DSLs) for describing each relevant aspect of the application. These DSLs are composed to obtain a single application DSL. Structured documents describing the application are the input for an operationalization process that yields a component-based artifact implementing the application. Operationalization process is performed in terms of a flexible architecture, where software components interact for assembling the application software in a collaborative, domain-dependent, way. Main benefits of our approach are software reuse and maintenance. These benefits are obtained through: a) the separation between high-level application description and application implementation and b) the provision of a flexible architecture, technologically neutral, enabling multiple implementation strategies.

Keywords: Content-based Applications, Application Development, Domain Specific Languages, Markup Technologies, Software Components, XML.

1 Introduction

There are application domains where the provision of the contents to be processed by the application is a critical part of the development process. We call this kind of applications *content-based applications*, because they process highly structured contents provided by domain experts, whose prior knowledge in software development is not guaranteed. This situation has been largely described in domains such as knowledge based systems [16], or, in our case, in the development of educational applications [2][3][9]. We consider that *domain-specific languages* (DSLs) [17] are crucial for the successful development of these content-based applications. A good definition for a DSL is that given in [1]: *a domain-specific language (DSL)*

is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. For content description, we are also interested in DSLs with more descriptive domain oriented meanings [4].

DTC (structured Documents, document Transformations and software Components) is our approach for building content-based applications using generalized markup technologies. Building an application according to DTC starts with the description of its most relevant aspects (e.g. contents, presentation, interaction) in terms of a collection of structured documents. Then, these documents are processed to obtain the executable application. The description of a DTC application is achieved by firstly selecting or devising suitable DSLs for each of these different application aspects. All these languages must be described with the same meta-language, so the same repertory of technologies can be used for their integration in a single description framework. DTC uses XML (eXtensible Markup Language) [19] as the common syntactic framework.

DTC encourages the explicit separation between content documents and documents involved with the description of the application's behavior. So a first distinction between *content* and *application* DSLs arises. These languages are intended for different users: content DSLs are used by domain experts, while application DSLs are used by software developers. The integration of both kinds of DSLs will be done by means of document transformations written by developers.

Application DSLs are composed for obtaining a single *application description* DSL. This language guides an *operationalization process*, so descriptions conforming it can be turned onto executable applications.

This paper describes the DTC approach focusing on the operationalization process. The structure is as follows. Section 2 outlines DTC. Section 3 describes the operationalization framework used for turning DTC descriptions onto executable applications made of discrete

¹ The EU project Galatea (TM-LD-1995-1-FR89) and the Spanish Committee of Science and Technology (TIC97 2009-CE,TIC98-0733 and TIC2000-0737-C03-01) have partially supported this work.

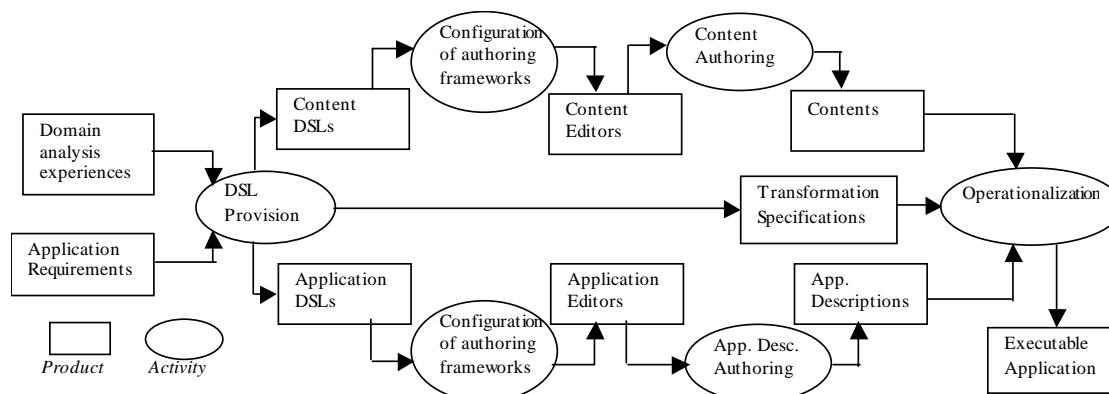


Fig. 1. Main products and activities involved on application development according DTC. From application requirements and domain analysis experiences, suitable DSLs are provided. These DSLs are devoted both for content description and for describing another application aspects. Transformations are also provided for mapping contents onto interpretation DSLs. Definition of suitable authoring syntaxes enables authoring frameworks to be configured. The results are editors tailored both for content and application description authoring. Content descriptions, transformation specifications and application descriptions are the inputs to an operational process producing the desired executable application.

software components instances. Section 4 gives an example. Section 5 discusses some related work. Finally, section 6 gives some conclusions and outlines future work.

2 The DTC approach

We have defined our approach DTC (structured Documents, document Transformations and software Components) [10][13][14] for building content-based applications. DTC makes an extensive use of the DSL principle. Building an application, according to DTC, can be largely viewed as the process of formulation, transformation, composition and operationalization of DSLs. At a very high level, DTC splits application development into two broad activities: one giving an explicit description of the application in a suitable DSL and a second, providing operational support for the DSL used to describe the application. In addition, when describing the application, DTC encourages an explicit separation between the description of contents and the description of the other application aspects.

Contents are described in terms of one or several DSLs, called *content DSLs*. Ideally, these DSLs are *use-neutral*, in the sense that they are involved with the descriptive and structural aspects of the information in the application domain, instead of dealing with how this information is going to be *used* or *processed* by the application. So, content DSLs can be formulated using a vocabulary close to the domain experts providing the contents. Therefore, descriptions conforming these content DSLs can be reused in multiple ways, either inside the same application or between different applications in the same domain area. However, for enabling particular uses of a given content, additional contextual or use dependent information will be required. For example, in a transport network application, we can have a DSL for describing the relevant data of the transport network domain (e.g. structure and timing). But, at this level it is not important the inclusion of additional data for describing presentational information (e.g. metrical

coordinates of the connected places, fonts and colors) that will be necessary for visualizing the transport network. DTC contemplates the description of this kind of use enabling information, providing additional content DSLs, called *use dependent content DSLs* (this information could be provided by a different expert).

Once content description has been decided, the uses of this information in the application must be stated. This is achieved again by selecting DSLs for describing how contents must be interpreted inside the application. Because this fact, we call these DSLs *interpretation DSLs*. The link between content and interpretation DSLs is actually described with *transformations* written by developers. For instance, the search of routes in a transport network can be easily reformulated as a route search problem in a weighted directed graph. In this way, the DSL describing transport networks can be interpreted in terms of a language for describing weighted directed graphs. The actual interpretation is stated by writing a transformation from the transport network DSL onto the graph description DSL.

Other application aspects, not directly related with content interpretation, must be described in terms of additional *application DSLs*. These DSLs allow the description of other aspects, such as GUI structure and layout, user interaction, and control processing. Interpretation DSLs are considered just as a particular kind of application DSLs. Finally, an appropriate composition of the application DSLs results in a single *application description DSL*. Application itself is described in terms of this final composite DSL.

For defining DSLs in DTC we are currently using XML. In this way, each DSL is conceived as a XML based markup language. XML provides a common syntactic framework suitable for representing, in a structured way, all the information relevant for application's deployment. In addition, XML is accompanied with a set of complementary technologies that make easier a DSL-based application development with independence of particular,

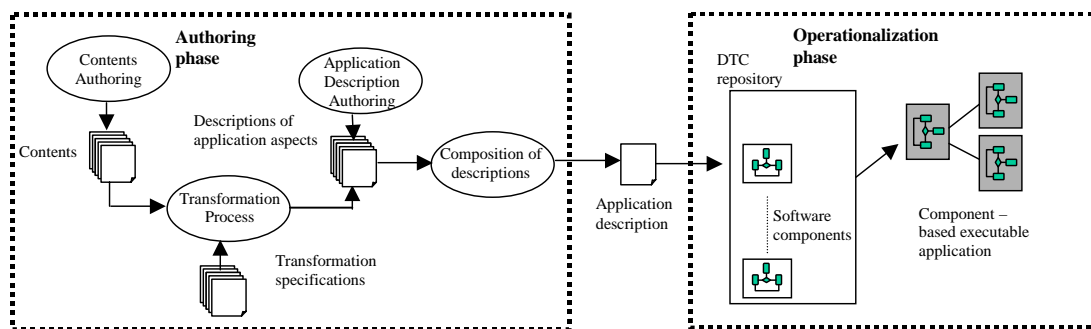


Fig.2. Schema of the building process for a particular DTC application instance.

vendor-oriented, implementation platforms. However, for some application domains, direct provision of XML structured documents could be inconvenient (e.g. providing all the description of a huge transport network directly in terms of raw XML). Thus DTC also introduces the concept of *document authoring*. This way, generic *authoring frameworks* (e.g. visual editors) can be configured by defining *authoring syntaxes* together with translation relationships between these syntaxes and the specific XML-based markup languages.

Finally, it is the *semantic* associated to interpretation and application DSLs. DTC is mainly concerned with *operational semantics*. In this way, each application DSL must be accompanied by a computational artifact (i.e. a component) giving one (or a set) of operational(s) meaning(s) to the information conforming this language. Regarding content DSLs, DTC does not prescribe a particular way for describing the associated semantics (i.e. it could be either given in an informal document or established using some suitable formal technique). Indeed, when transformations between content DSLs into interpretation ones are defined, operational semantics of the interpretation languages could be automatically attributed to the content ones.

Fig 1. outlines the main activities and products involved in the development of an application according DTC. In this work we are mainly interested on the process that makes DTC application descriptions operational. Next section describes our solution.

3 Construction of applications using DTC descriptions

Fig.2 sketches how a particular DTC application instance is built. In this process we identified two main phases:

- The *authoring* phase, where application descriptions are provided. Contents are authored and, then, they are transformed onto interpretations. Description of other application aspects relies on additional authoring. Finally, authored application descriptions and derived interpretations are composed in an overall application description.
- The *operationalization* phase, where the actual executable program is obtained from documents with

the application description and a set of appropriate components. The ingredients in this phase are: a) the application description produced in the authoring phase, b) suitable *software components* giving operational support to the DSLs, and c) a *DTC repository* grouping all the components.

Next subsections go inside the details of the operationalization process.

3.1 The operationalization ingredients

The operational meaning of a DTC application is obtained by assigning computational artifacts to descriptions. Because DTC does not compromise itself with any particular DSL, this operationalization process must be necessarily open. In this way, the existence of a universal engine that, given any DTC application description, yields the associated artifact is clearly meaningless. On the other hand, DTC encourages the formulation of description languages by an appropriate combination of simpler languages, each one for the description of a particular application aspect. Thus, applications can be implemented using the components attached with those simpler languages. Hence, in DTC, language composition has an operational counterpart in composition of software components.

A more in-depth view of the language composition approach followed in DTC reveals the convenience of maintaining the semantics of some languages *parametric* in some of their aspects. For instance, a language for describing control and interaction with a stated-based formalism can be parametric in the repertory of control actions. Those aspects can, in their turn, be described by other DSLs. Actually, the ability for finding and using suitable description languages able to be parameterized is a key aspect for enabling the compositional approach encouraged by DTC. On the implementation side, the direct consequence of this approach is the distinction between two different categories of software components:

- *Atomic components*. Components that perform entirely their functionalities without the need of delegating further tasks in other artifacts. These components give support for non-parametric application DSLs. An example of this sort of components, in the application

