Computers & Geosciences 53 (2013) 13–20

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

A feature model of coupling technologies for Earth System Models

Rocky Dunlap*, Spencer Rugaber, Leo Mark

College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA

A R T I C L E I N F O

Article history:
Received 15 February 2011
Received in revised form
23 June 2011
Accepted 10 October 2011
Available online 25 October 2011

Keywords:
Climate model
Coupler
Feature analysis
Software architecture
Code generation

A B S T R A C T

Couplers that link together two or more numerical simulations are well-known abstractions in the Earth System Modeling (ESM) community. In the past decade, reusable software assets have emerged to facilitate scientists in implementing couplers. While there is a large amount of overlap in the features supported by software coupling technologies, their implementations differ significantly in terms of both functional and non-functional properties. Using a domain analysis method called feature analysis, we explore the spectrum of features supported by coupling technologies used to build today's production ESMs.

1. Introduction

Model coupling is essential for implementing multi-physics models made up of two or more interacting computer simulations. A quintessential example of a coupled model is an Earth System Model (ESM), which involves several interacting components simulating the Earth's atmosphere, oceans, land, and sea ice systems. The software components that link together and mediate interactions between these models are called *couplers*. Couplers are well-known abstractions in the geophysical and other scientific communities, although their implementations differ vastly and no standardized reference architecture has emerged. Functions typically associated with couplers include managing data transfer between two or more models (often in parallel), interpolating field data (e.g., fluxes of heat, momentum, or water) when two models' resolutions or discretization schemes differ, ensuring conservation of physical quantities across the coupling boundary, accumulating and averaging of physical quantities (when model time steps differ), and coordinating execution of the constituent models. Examples of couplers actively used in the climate modeling community include the Community Earth System Model's (CESM), CPL7 (Craig, 2010), and the OASIS coupler (Redler et al., 2010).

Because coupling numerical models is a common need, a number of technologies have emerged in the form of reusable software assets to facilitate building coupled scientific applications. Reuse is an important software engineering concept and the potential benefits of reduced development costs, decreased time-to-solution, and increased reliability and quality have already been recognized by the geoscience communities (Marshall et al., 2006; Peckham, 2010).

Designing couplers is both a scientific and a software engineering activity. The design and implementation of reusable coupling technologies is an interesting challenge because, while a common set of capabilities required for coupling numerical models has been identified, the design of a "good" coupler is intimately dependent on the specific scientific and technical properties of the models being coupled. This brings into question what code can be effectively reused and what code should remain custom to a specific coupled model. Furthermore, differences in the software architecture of existing models imply that no simple approach can be universally applied.

Couplers can be seen as members of a family of software components with similar requirements. For example, they coordinate data communication among models, transform and interpolate field data, and manage use of parallel computing resources. *Generative Programming* is a software engineering technique for automatically generating members of such software families by assembling reusable components into final products based on a declarative requirements specification (Czarnecki and Eisenecker, 2000). If a capability is common among couplers, it is a good candidate for software reuse. While recognizing the commonality of capabilities in couplers, we also observe significant differences in the design, architecture, and scope of existing couplers. In fact, our analysis shows several different, yet viable approaches to coupling. At this point, the underlying scientific and computational justifications for the different approaches are not obvious.

* Corresponding author. Tel.: +1 404 894 8450; fax: +1 404 385 2295.
E-mail addresses: rocky@cc.gatech.edu (R. Dunlap), spencer@cc.gatech.edu (S. Rugaber), leomark@cc.gatech.edu (L. Mark).

0098-3004/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2011.10.002

This paper reports on a feature analysis of the software engineering aspects of coupling technologies we conducted in preparation for automatically generating couplers for numerical ESMs. In Section 2, we explain Generative Programming and describe a domain analysis mechanism called *feature analysis*. We then give a brief example of a feature diagram. In Sections 3 and 4 we describe the specific process that we undertook to arrive at a feature diagram for coupling technologies and give a brief overview of the technologies analyzed. In Section 5 we present the results of our feature analysis in the form of a series of feature diagrams with a brief description of each feature. Although the result of our analysis is a comprehensive picture of the coupler product family, the analysis also surfaced some important issues, which the ESM community has yet to address. These issues are discussed in Section 6.

2. Feature analysis

A prerequisite to generating couplers automatically is to understand the features that existing couplers provide. A *feature* is a unit of user-visible value. Features may be *functional*, such as when a coupler provides interpolation, or they may be *non-functional*, as when a coupler uses on-the-fly compression to reduce data transmission costs. In order to generate couplers automatically we need to know what features are common across couplers and what features vary. A key step in Generative Programming is *feature analysis,* the systematic examination of existing members of the application domain for which generation is proposed. The output of feature analysis is a *feature model* that identifies a concise and descriptive set of common and variable properties of domain concepts. Once a feature model has been produced, elements can be selected to produce a *configuration*, describing a desired family member. From the configuration, an automated generator can then be used to produce the actual code for that member.

The results of a feature analysis can be expressed as a *feature diagram*—an annotated tree in which nodes denote features. Nodes are connected with directed edges, and edges have decorations that define the semantics between parent and child nodes. Fig. 1 shows a simple feature diagram for a car.

The root node of a feature diagram is called the *concept* node. The example diagram describes the concept Car. All nodes directly below the concept node represent features, and lower nodes represent subfeatures. *Mandatory* features are denoted by a simple edge ending with a filled circle. In the example diagram, both Transmission and Engine are mandatory features. *Optional* features are denoted by a simple edge ending with an open circle. In the example, the Navigation System feature is optional. Subsets of features may be *alternatives* to each other; meaning that exactly one member of the subset is included in any configuration. This possibility is represented in a feature diagram by connecting the edges pointing to alternative features with an arc. The Transmission feature has two alternative subfeatures: Automatic and Manual. If an arc connecting edges pointing to two or more features is filled in, it indicates that the set of features are *or-features*. Within a set of or-features, any non-empty subset of the features can be included in a configuration. In the example, if the optional Navigation System feature is included, then it will either be Voice Activated, Touchscreen Activated, or both.

We have extended the feature diagram notation in two ways. First, we allow a diagram to be split into pieces: a box in a diagram may have its background shaded. This means that the corresponding feature and its subfeatures are elaborated in a separate diagram. Second, when a feature has many subfeatures, each of which is not further elaborated, then, instead of using boxes, we present the subfeatures as a bulleted list under the given feature.

3. Feature analysis process

The feature analysis we conducted is based on information found in technical documentation that accompanies the coupling technologies as well as peer-reviewed articles that describe the technologies and their uses. The initial feature analysis was conducted in a bottom-up fashion by gathering a large list of features that couplers support. The resulting feature diagrams contained over one hundred features at the leaf level. We dealt with this complexity by abstracting related subfeatures into common higher-level features, sometimes producing a hierarchy several levels deep. During this process, we have, in effect, defined a vocabulary that describes the space of features supported by coupling technologies for ESMs. When alternative terms were found in the literature, we either chose one of the terms or selected a different term, which we felt best described the semantics of the set of alternative features.

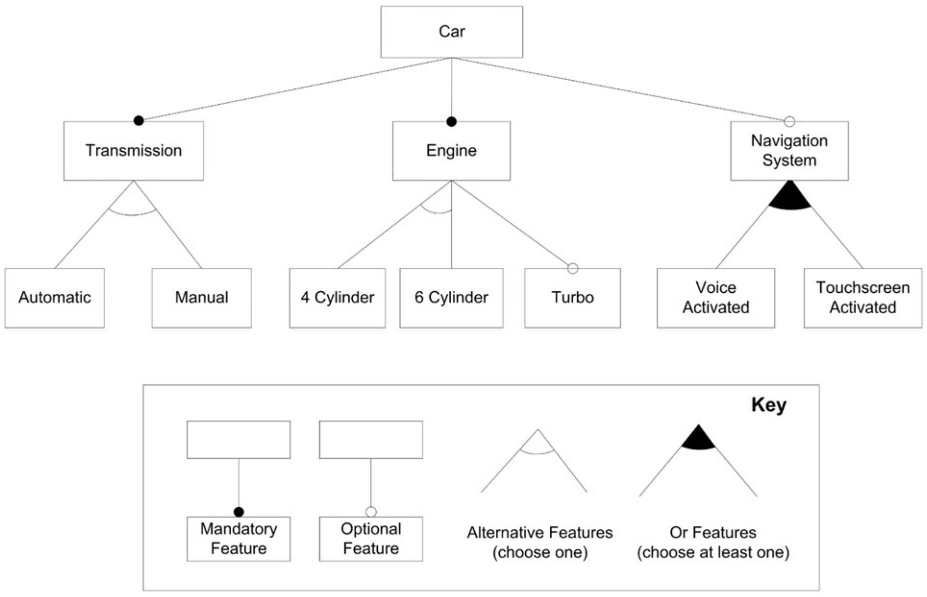

Fig. 1. Example feature diagram.