Data Component Based Management of Reservoir Simulation Models

Cong Zhang Amol Bakshi Viktor K. Prasanna

Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, California
Email: {congzhan, amol, prasanna}@usc.edu

Abstract

The management of reservoir simulation models has been an important need of engineers in petroleum industry. However, due to data sharing among reservoir simulation models, data replication is common and poses many challenges to model management, including management efficiency and data consistency. In this paper, we propose a data component based methodology to manage reservoir simulation models. It not only improves management efficiency by removing data replicas, but also facilitates information reuse among multiple models. We first identify the underlying structure of the simulation models and decompose them into three types of components: reservoir realization, design, and simulator configuration. Our methodology then identifies the duplicate components and guarantees that each component has one physical copy in the data repository. By separating the logical connections between the models and the components from the physical data files, our methodology provides a clean and efficient way to manage data sharing relationships among the models.

1 Introduction

Reservoir simulation models are computer models used to predict the flow of fluids (typically, oil, water and gas), through porous media in a reservoir. They store the complete input data required by a reservoir simulator. They are large datasets consisting of heterogeneous data, including physical models of a reservoir, historical data, well management information, simulator configuration data, etc. [9, 15].

Management of reservoir simulation models includes two tasks. One is the management of models' data files. In a file-based approach, the management mainly concerns the directory hierarchy for storing the models, including naming of the data files and directories. The second task is the management of relationships between the models, such as *data sharing*. Two models are considered to have a data sharing relationship if they share a portion of their model data. For example, two models might share the same historical production data and well completion data. Data sharing relationships are very common among reservoir simulation models.

However, data sharing among simulation models results in data replicas. The complete data of a model are typically stored in a repository managed by traditional approaches. If part of a model is shared by multiple models, multiple replicas of the data are created and distributed in each of those models. These data replicas cause two problems to the repository. The first is concerned with the storage efficiency. The second is related to data consistency among the models with shared data components. In an oilfield asset, reservoir simulation models are regularly updated or calibrated as historical production data of the reservoir become available (the process is called *history matching* in petroleum engineering). Due to the complexity of a reservoir simulation model, propagating changes in these models properly and efficiently is a nontrivial task. Therefore, reducing data replicas in the repository is desirable in the management of reservoir simulation models.

Data replication has been studied extensively in the area of distributed systems [19]. Previous research efforts can be classified into two categories. The first category focuses on data replication across a distributed system to improve reliability, fault-tolerance, or accessibility of the system [10, 18]. The second category addresses data consistency resulting from data replication. Replica consistency techniques and systems have been developed for this purpose [2, 17]. However, to the best of our knowledge, no existing work has addressed the data replica problem in the management of simulation models.

In this paper, we propose a methodology to manage reservoir simulation models. We focus on eliminating data replicas among reservoir models. The methodology consists of four major steps. The first step is to decompose

IEEE IRI 2008, July 13-15, 2008, Las Vegas, Nevada, USA

978-1-4244-2660-7/08/$25.00

a reservoir model into components. The components in the reservoir simulation model are then imported into the repository. An algorithm is used to guarantee that all imported components are unique, and that there are no replicas in the repository. A single physical copy of each data component is stored and indexed in the repository, while the simulation models are stored "virtually" in the repository. Each simulation model is represented by a row in a relational table, which consists of the IDs of the data components in the model. To retrieve a simulation model from the repository, the fourth step in our methodology is to recover the simulation model by assembling the data files of the components contained by the model.

We also propose a data model, used by our methodology, to decompose a reservoir simulation model. The data model defines the structure and meaning of the data components. The data model in our methodology contains three types of data components: reservoir realization, design, and simulator configuration. It represents our view of the reservoir simulation models. To faciliate the illustration of our data model, we define two concepts: *element* and *component*. The *element* represents the lowest level building block into which a model can be decomposed; the component is a high-level grouping of related elements.

The remainder of the paper is organized as follows. In Section 2, we present management of reservoir simulation models. The proposed methodology is introduced in Section 3. Section 4 discusses related work. We conclude in Section 5.

2 Management of Reservoir Simulation Models

2.1 Terminology

Reservoir simulation is the process of using computer models to predict the flow of fluids (typically, oil, water and gas) through porous media. *Reservoir simulators* solve a complex set of coupled nonlinear partial differential equations over hundreds of thousands to millions of gridblocks [9, 15]. A reservoir simulator offers sophisticated simulation components encapsulating complex mathematical models of the physical interactions in the reservoir, such as geomechanics, chemical reactions, and fluid flow processes.

A *reservoir simulation model* is the complete input data set required by a reservoir simulator. It consists of seismic data, historical production data, well management information, simulator configuration data, etc. [9, 15]. A reservoir simulation model is specific to a reservoir simulator, in terms of model specification and storage. Most reservoir simulators use their own keyword systems to specify input data. A reservoir model is stored in a proprietary file format. Usually a reservoir model is stored in a large single file or a tree of files.

A *reservoir realization*, also referred to as reservoir characterization in petroleum engineering, is a possible representation of the reservoir. It is a key data component of a reservoir simulation model. It incorporates all the characteristics of the reservoir.

A *design* in field development is usually linked to a planning or an investment decision. Typical designs in field development include 1) number and location of wells, 2) reservoir recovery process, 3) well configuration and type, 4) subsea and platform development, and 5) surface facility capacity. In reservoir simulations, a design constrains the dynamics of fluid flow inside the reservoir.

2.2 Challenges

Data replication is common across reservoir simulation models. Initially, a few base reservoir simulation models were available for a reservoir, e.g., P10, P50, and P90 models [1]. These models are used extensively used to explore and evaluate different field development plans or field management strategies. As a result of exploration or evaluation, many more models are derived from these base reservoir models. A derived reservoir model is usually created in two steps: 1) copying a base reservoir model; 2) modifying parameter values in the copy. Furthermore, the derived models can also be used to derive new models.

Data replication poses many challenges to reservoir simulation model management. The first challenge is the efficiency of management. Data replication results in new relationships among models. The new relationships can be created based on data components shared by various models. Managing these relationships results in overhead. For example, additional metadata used to track the relationships must be captured and stored. The second challenge caused by data replication is maintaining data consistency among various reservoir simulation models. Generally, the more one replicates, the more points of divergence are created and the more one is subject to incorrect behaviors. On the other hand, base reservoir simulation models are regularly updated or calibrated as historical production data of the reservoir become available (the process is called *history matching* in petroleum engineering). Due to the complexity of a reservoir simulation model, propagating changes in these base models properly and efficiently is nontrivial.

3 Methodology

In this section, we present our methodology for reservoir simulation model management, which aims to achieve data component reuse and eliminate data replication among

models. We first describe the intuitive idea behind our methodology, then define the data model of reservoir simulation models. Next, we describe in detail the steps in the methodology.

3.1 Separation of Concerns

Our methodology is motivated by the principle of *separation of concerns*, which can be considered as one of the key principles in software engineering [14, 20]. This principle states that a given problem involves different kinds of concerns, which should be identified and separated to cope with complexity, and to achieve required engineering quality factors such as robustness, adaptability, maintainability, and reusability [20]. The principle is applied in many areas, and is considered a ubiquitous software engineering principle.

We apply the principle of separation of concerns in our methodology in two ways. First, the separated concerns in the management of reservoir simulation models are represented as data components constituting the models. Data components are defined in such a way that they are independent of each other, while the data contained in a component are closely related and represent a piece of knowledge about the reservoir model, e.g. geoscience data, engineering data or production data [7]. Second, the principle leads to two major steps in our methodology. One of the steps is to decompose a reservoir simulation model into components that can be handled separately. The granularity of the components is discussed in Section 3.2. Through the decomposition, we can 1) identify data components that are replicated across reservoir simulation models; 2) identify unique data components. The unique and replicated components are stored in different locations, and are indexed separately. The index information of the components is used in each reservoir simulation model. As a result, duplicate components in each model are deleted to reduce redundancy. Details of the steps in our methodology are presented in Section 3.3.

Note that separation of concerns is meaningful in our methodology only if components can be composed into a reservoir simulation model. Otherwise models managed in the repository cannot be used directly by other applications because these models are not stored in their entirety. Therefore, another step in our methodology is to combine various data components to recover decomposed reservoir simulation models or to generate new models. This process is called "synthesis."

3.2 Data Model of a Reservoir Simulation Model

To decompose the reservior simulation models, we must first identify the underlying data model. A data model defines the structure and meaning of the data to be decomposed. The data model contains various building blocks representing the users' understanding of the domain, such as the hierarchy and organization of the domain concepts. Multiple building blocks can be assembled to form higher-level building blocks.

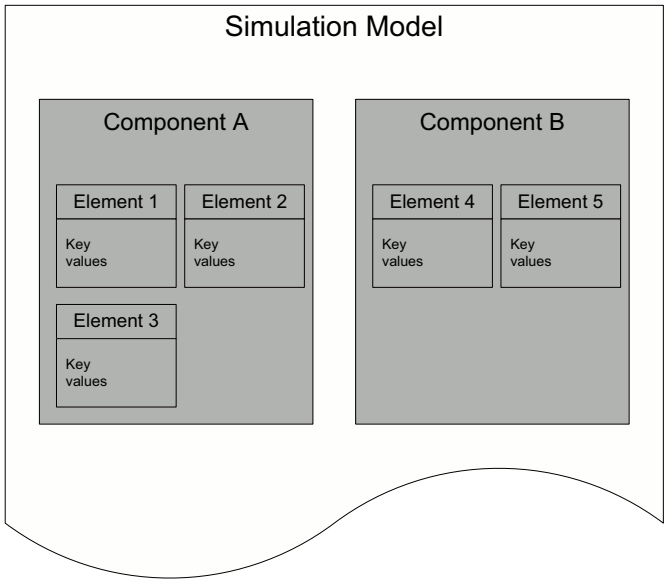

Figure 1. Data model of reservoir simulation model

To faciliate specification of data models, we define two concepts: *element* and *component*. The *element* represents the lowest level building block into which a model can be decomposed. The *component* is a high-level grouping of related elements.

The data model represents our view of the reservoir simulation models. In other words, any reservoir simualtion model can be viewed as a series of elements. Each element contains a portion of the model data.

Elements can be defined at different granularities, which can range from a single character to a line in the data file of the model. In our data model, the element is defined by a model parameter, and its associated values in the simulation model, as shown in Figure 2. Each model parameter is identified by a keyword.

Another concept in the data model is the *component*, which represents high-level groupings of related elements included in a reservoir simulation model. This type of grouping is necessary due to the large number of elements constituting a reservoir simulation model. For example, a typical simulation model for the CHEARS reservoir simulator [6] would consist of over four hundred elements. As

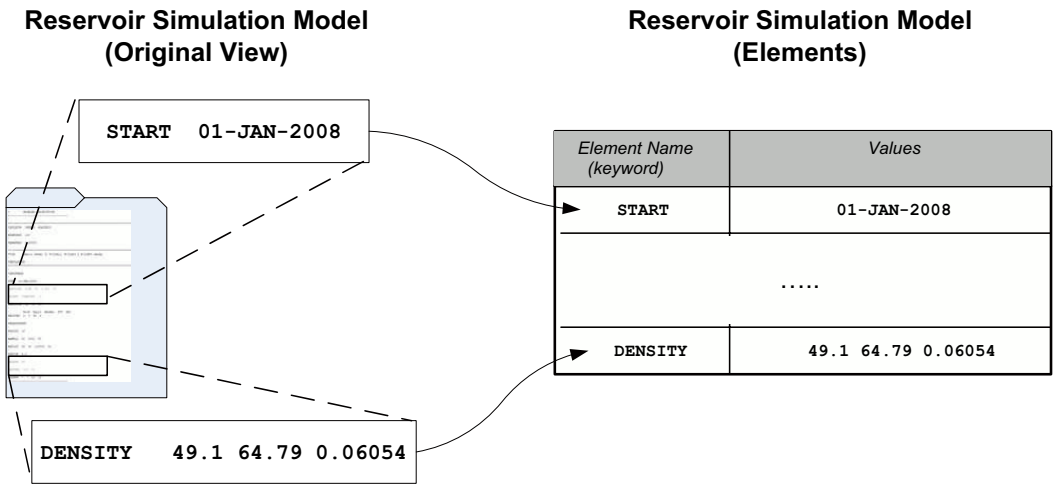

Figure 2. A reservoir simulation model in terms of elements

a result of this type of grouping, one component usually contains hundreds of elements, and one reservoir simulation model contains multiple components. The meaning of each component is dependent upon how the elements are grouped. For example, each component may be linked to a task that can be accomplished by a user with all the elements in it. These tasks include reservoir characterization by petroleum geologists and geophysicists, well management scheduling by production engineers, simulation configuration by reservoir simulation engineers, etc.

We have defined three types of components in the data model: reservoir realization, design, and simulator configuration. Definitions of reservoir realization and design data are given in Section 2. Simulator configuration data include historical production and injection data of a reservoir, simulator output specification, etc. The reason for having these three components is based on the following observations of the reservoir simulation models. First, simulator configuration data are usually fixed for the reservoir simulation models. Second, the reservoir simulation models are characterized by different reservoir realizations. Third, the three components represent three major tasks, accomplished by different engineers in the petroleum industry. Reservoir characterization is a task for petroleum geologists and geophysicists; designs for field development are mostly studied by production engineers; simulation configuration is typically performed by reservoir simulation engineers.

3.3 Steps of the Methodology

Our methodology consists of four steps, as shown in Figure 3.

Step 1: Model decomposition

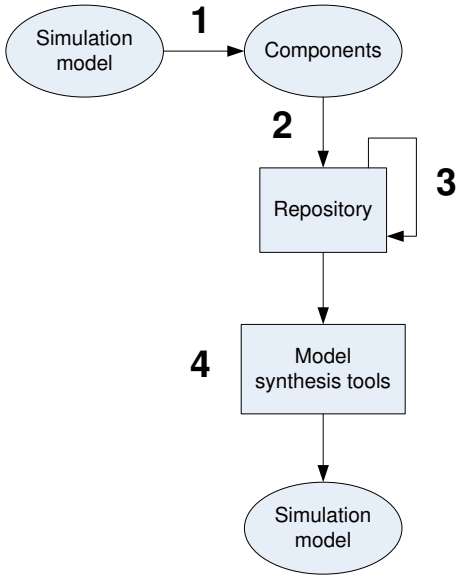

Figure 3. Data component based methodology

The algorithm for decomposing a reservoir simulation model is given in Algorithm 1. We assume that the data files for a reservior simulation model are stored in ASCII format. This assumption implies that, either manually or programmatically, a reservoir simulation model can be broken into smaller pieces, each of which can be stored in a new data file. Such an assumption holds for most existing reservoir simulators, such as the Eclipse reservoir simulator developed by Schlumberger [8] and CHEARS by Chevron.

Algorithm 1 Model decomposition

read the reservoir simulation model from its data file(s)
convert the reservoir simulation model into a list of elements

{*cluster the elements*}
initialize components for the reservoir simulation model to be empty
for each element on the list of elements **do**
  get the name of the element
  look up the element name in the data model to find the component that the element belongs to
  add the element, including its name and values, to the corresponding component
end for

{*persist the components*}
for each non-empty component created for the model **do**
  store its content in a data file
end for

Algorithm 2 Importing a component into the repository

let c be the the data file of the component
read c
get the type of the component from the metadata stored in c

{*check if the component already exists in the repository*}
$X =$ all the data files in the repository that have the same type as the component
for $x \in X$ **do**
  if $x = c$ **then**
    exit the program {*the component exists in the repository*}
  end if
end for

{*store the data file in the repository*}
copy c into the repository
assign a unique identifier to c
store the type and the identifier of c in the repository

As can be seen in the algorithm, the data model of the reservoir simulation model plays a key role in this step. It guides the model decomposition process by providing the definitions of the elements and components comprising a reservoir simulation model. For example, in the component initialization, the data model dictates the number and types of components needing to be initialized.

The format used to store the data components can be the same as the one used for the reservoir simulation model. Different data storage formats can also be used such as XML. In this case, a conversion tool is needed to extract the values contained by the elements and store them in a specified format. In the data file where a component is stored, metadata describing the properties of the component are also stored. One of the key metadata is the type of the component.

Step 2: Importing components

In this step, the components in the reservoir simulation model are imported to the repository. The algorithm is described in Algorithm 2. It guarantees that all imported components are unique, and that there are no replicas in the repository.

A key operation in the algorithm is determining the equality between two components. Two components are *equal* if and only if they have the same set of elements; two elements are *equal* if and only if they have the same element name and values.

Step 3: Component reuse

Components in the repository are used to create simulation models in the following three scenarios: (1) after the components of a base simulation model have been either imported to the repository or discarded as replicas; (2) when creating a new base simulation model; (3) when creating a new working simulation model.

In our methodology, the simulation models are stored "virtually" in the repository. In other words, they are not stored as data files containg the complete data of the models. Instead, each simulation model is assigned a row in a relational table only, which stores the containment relations between the simuation models and the components. Each simulaton model is represented by a row in the table, which consists of the ID and the name of the model, and the IDs of the data components in the model.

To create a simulation model, a user selects a set of components from the repository. The user is first presented with a list of all the components that have been imported to the repository. If a component needed by the user to create the model does not exist in the repository, it must be created, and then imported using Step 2. After the component selection is over, the repository adds a new row for the created model in the table storing the simulation models.

Step 4: Model retrieval

During the retrieval of a reservoir simulation model in the repository, the data file for the model must be created dynamically. This is because the simulation models are stored "virtually," as discussed in Step 3. In other words, there are no physical data files for the simulation models in the repository. However, data files containing the complete data of the models are required by simulators.

Therefore, a critical operation in retrieving models in the repository is the synthesis of a simulation model from its contained components. Algorithm 3 describes an algorithm

used in our methodology for model synthesis. The algorithm first finds all the components comprising a simulation model. It then creates the complete data file for the simulation model by synthesizing the data files of the contained components.

Algorithm 3 Model synthesis

```
{find the IDs of the components in the model}
let m be the model to be synthesized
look up m in the repository
if m is a reservoir simulation model then
   look up m in the repository
   X = identifiers of the components in m
end if

initialize the data file for m
for x ∈ X do
   look up x in the repository
   get data access information of the component
   read data file of the component
   update data file of m using the data file of the component
end for
```

3.4 Advantages of the Methodology

Our methodology not only improves the storage efficiency of the repository for the simulation models, but also offers many benefits to the end users of the repository:

- **Increased organization of model data**. Splitting the models logically makes it easier for engineers to find model data, including the reservoir realization, design data, application specific data, etc.
- **Facilitated data reuse**. Data in a model are split into three components that operate largely independently of each other. This allows reuse of the components in other models, hence saving much of the engineers' effort in model building later.
- **Shared model data between models**. By carefully separating the data file of a simulation model into multiple files, it is possible for multiple models to share some data files without duplication. Through such file sharing, any change made to the shared file or files for one model affects the other models immediately. Therefore, all models can be guaranteed to use the most up-to-date versions of data.
- **Split modeling responsibilities among engineers**. It is impractical for more than one engineer to make changes to a single file at the same time. Our methodology separates a large single file into multiple files so that each engineer can work on a separate part of the model without affecting the work of the other engineers.

4 Related Work

Little research has focused on reservoir simulation model management. As far as we know, file-based management still prevails. In such ad-hoc solutions, models are stored as flat files in a file system. Relationships among the models are not explicitly managed. Instead, they are managed to a limited extent through the directory hierarchy and file names.

Recently, several aspects of reservoir simulation model management have been addressed by petroleum software companies and research communities. In [12], the authors focus on data management in data-driven reservoir simulation studies. To store and retrieve large-scale reservoir models and reservoir simulation outputs, they apply DataCutter [4, 3, 5, 13, 11], which is a middleware infrastructure focusing on the storage, retrieval and processing of multiple large scientific datasets on remote archival storage systems.

The Model Catalogue, an integral part of the IFM platform developed by Petroleum Experts [16], focuses on two aspects in reservoir model management: archiving and versioning. It gives the engineer a structured archiving system to store and retrieve models in an oilfield. The engineer can therefore maintain a complete history of each version of a model as it is developed and maintained over the field life. Based on database technology, the Model Catalogue is designed to work as the central repository of the "official" or "published" production models of a production team within a company. In this way, only the latest, most up-to-date models of each production element are used for analysis in the production system. By comparison, the Model Catalogue focuses mainly on the versioning of simulation models, while the work in this paper focuses on eliminating data replication in the repository of simulation models.

5 Concluding Remarks

In this paper, we have proposed a data component based methodology for simulation model management. We have studied the underlying structure and meaning of the simulation models, identified their basic building blocks, and defined three types of components.

In our methodology, all the simulation models are decomposed into three components. When imported to the repository, all duplicate components are eliminated, and only the unique components are preserved. The relationships among the components, hence the data sharing and derived relationships among the simulation models, are also

stored in the repository. Therefore, users can easily recover the original simulation models, or generate new ones by selecting and synthesizing appropriate components from the repository.

Future work includes a) studying data models of different granularities and comparing their performance in eliminating data replicas; b) based on the methodology, developing a framework for the management of reservoir models based on data components.

Acknowledgment

This research was funded by CiSoft, (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence & Academic Training and a joint venture between the University of Southern California and Chevron. We are grateful to the management of CiSoft and Chevron for permission to present this work.

References

[1] J. A. Acuna, M. A. Parini, and N. A. Urmeneta. Use a large reservoir model in the probabilistic assessment of field management strategies. In *20th Workshop on Geothermal Reservoir Engineering*, Stanford, California, January 28-30, 2002.

[2] G. Belalem and Y. Slimani. A hybrid approach for consistency management in large scale systems. In *Proceedings of the International Conference on Networking and Services (ICNS)*, 2006.

[3] M. Beynon, R. Ferreira, T. M. Kurc, A. Sussman, and J. H. Saltz. Datacutter: Middleware for filtering very large scientific datasets on archival storage systems. In *17th IEEE Symposium on Mass Storage Systems*, pages 119–133, 2000.

[4] M. Beynon, T. M. Kurc, A. Sussman, and J. H. Saltz. Design of a framework for data-intensive wide-area applications. In *Proceedings of the 9th Heterogeneous Computing Workshop (HCW2000)*, pages 116–130, 2000.

[5] M. D. Beynon, T. Kurc, A. Sussman, and J. Saltz. Optimizing execution of component-based applications using group instances. *Future Generation Computer Systems*, 18(4):435–448, 2002.

[6] Chevron. `http://www.chevron.com/`.

[7] C. F. Conaway. *The Petroleum Industry: A Nontechnical Guide*. Pennwell Books, 1999.

[8] ECLIPSE Reservoir Engineering Software. `http://www.slb.com/`.

[9] J. R. Fanchi. *Principles Of Applied Reservoir Simulation*. Gulf Professional Publishing, 2 edition, 2001.

[10] T. Ho and D. Abramson. A unified data grid replication framework. In *2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, Netherlands, Dec. 4- 6, 2006.

[11] H. Klie, W. Bangerth, X. Gai, M. F. Wheeler, P. L. Stoffa, M. Sen, M. Parashar, U. Catalyurek, J. Saltz, and T. Kurc. Models, methods and middleware for grid-enabled multiphysics oil reservoir management. *Engineering with Computers*, 22(3):349–370, 2006.

[12] T. Kurc and U. C. et al. A simulation and data analysis system for large-scale, data-driven oil reservoir simulation studies. *Concurrency and Computation: Practice and Experience*, 17(11):1441–1467, 2005.

[13] S. Narayanan, T. M. Kurc, U. V. Catalyurek, and J. H. Saltz. Database support for data-driven scientific applications in the grid. *Parallel Processing Letters*, pages 245–271, 2003.

[14] H. Ossher and P. Tarr. Multi-dimensional separation of concerns in hyperspace. Technical Report RC 21452(96717)16APR99, 1999.

[15] D. W. Peaceman. *Fundamentals of Numerical Reservoir Simulation*. Elsevier Scientific Publishing Company, 1 edition, 1977.

[16] Petroleum Experts. `http://www.petex.com/`.

[17] Y. Saito and M. Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, 2005.

[18] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and object replication in data grids. In *10th IEEE Symposium on High Performance and Distributed Computing (HPDC2001)*, San Francisco, California, August 7-9, 2001.

[19] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[20] P. L. Tarr, H. Ossher, W. H. Harrison, and S. M. S. Jr. N degrees of separation: Multi-dimensional separation of concerns. In *International Conference on Software Engineering*, pages 107–119, 1999.