Supporting the Composition of Domain-Specific Software via Task-Specific Roles

Brigitte Boden
brigitte.boden@dlr.de
German Aerospace Center (DLR)
Intelligent and Distributed Systems
Cologne, Germany

Robert Mischke
robert.mischke@dlr.de
German Aerospace Center (DLR)
Intelligent and Distributed Systems
Cologne, Germany

Alexander Weinert
alexander.weinert@dlr.de
German Aerospace Center (DLR)
Intelligent and Distributed Systems
Cologne, Germany

Andreas Schreiber
andreas.schreiber@dlr.de
German Aerospace Center (DLR)
Intelligent and Distributed Systems
Cologne, Germany

ABSTRACT

Nowadays, complex technical systems are frequently developed by composing discipline-specific tools into an automated high-level workflow. Constructing, executing, and maintaining this workflow together with the infrastructure supporting it involves several distinct roles. We argue that awareness of these roles and providing explicit software support for them accelerates these processes and enables fast iteration on the design of the workflow itself.

CCS CONCEPTS

• **Social and professional topics** → **Automation**; • **Computing methodologies** → **Modeling methodologies**; • **Software and its engineering** → **Software usability**; • **Applied computing** → *Engineering*.

KEYWORDS

Multi-Disciplinary Design, Collaboration, Engineering Processes

ACM Reference Format:
Brigitte Boden, Robert Mischke, Alexander Weinert, and Andreas Schreiber. 2020. Supporting the Composition of Domain-Specific Software via Task-Specific Roles. In *Companion Proceedings of the 4th International Conference on the Art, Science, and Engineering of Programming (<Programming'20> Companion), March 23–26, 2020, Porto, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3397537.3399576

1 INTRODUCTION

The continued development of complex systems such as airplanes or spacecraft is only possible due to the collaboration of numerous engineers. Each engineer is typically an expert for some of the involved disciplines, but the overall design goals can only be achieved through an iterative process involving all engineers. At the core of this process are individual numerical simulations which are implemented as *discipline-specific tools*. For the remainder of this work, we use the qualifier "discipline-specific" only to resolve ambiguities and omit it otherwise. Each of these tools can be a custom compiled software, a script, or a specific setup involving standardized software. To implement such a tool, the *tool developer* needs deep insight into the respective domain.

The composition of these tools results in an automated *(design) workflow* which comprises the individual tools as well as a formal description of the data dependencies between them. Such a workflow is usually developed and executed using a *workflow design framework*. In contrast to the development of a discipline-specific tool, constructing the overall workflow requires working knowledge of all disciplines involved as well as an overview over the interfaces between them. This construction is handled by the *workflow designer*. For a concrete example of the work of the roles described here, please refer to work by Moerland et al. [2] as well as work by Boden et al. [1].

While the tool developers and the workflow designer share the common goal of developing a simulation workflow, the task of developing a discipline-specific tool for inclusion in a workflow is conceptually different from composing several tools into a workflow. Both roles, however, use the same workflow design framework for their work. In Section 2, we argue that having the workflow design framework provide robust abstractions for all roles involved in constructing a workflow is central to enabling effective and efficient collaboration between these roles. Particularly, they enable fast iteration on the constructed workflow and the individual tools, which in turn supports agile development of complex automated simulations. In Section 3, we discuss the application in our software *RCE*.

2 ABSTRACTIONS FOR DEVELOPING AN INTEGRATED DESIGN WORKFLOW

In the scenario outlined above, multiple tool developers collaborate with a workflow designer to construct a comprehensive design workflow. This is an overly simplistic view. Most modern design workflows use not only tools purpose-built for inclusion in the workflow, but mix and match such tools with off-the-shelf ones. Thus, there are typically *tool integrators*, which are responsible for

<Programming'20> Companion, March 23–26, 2020, Porto, Portugal

ACM ISBN 978-1-4503-7507-8/20/03.
https://doi.org/10.1145/3397537.3399576

formalizing and implementing the interfaces between discipline-specific tools and the workflow design software.

Further, the tools involved in a workflow are not necessarily executed on the same computer, but distributed onto multiple systems due to performance considerations, licensing requirements, or dependencies on different operating environments. Hence, there are *network administrators*, who are responsible for designing, setting up, and maintaining the connections between the involved machines. They are not concerned with the implementation or the interface of the involved tools, nor with the overall design of the workflow.

While all roles collaborate via the same workflow design framework, each role involved in the construction of an integrated design workflow has vastly different requirements towards the workflow design software. A tool integrator should not need to ensure that the machine they use to provide their tool is connected to that of the workflow designer. Conversely, a network administrator is not concerned with the tools provided by the machines that she connects.

To facilitate the work of each individual role, it is necessary to provide strong abstractions for the concepts that are not required for their particular task. Abstractions have enabled the development of complex discipline-specific tools by enabling separation of concerns between developers.

When developing a workflow design framework, developers must first identify the roles that will interact with the resulting system. The number and focuses of these roles will vary depending on the generality as well as the scope of the envisioned workflow design framework. They must then identify the core concepts that each role interacts with. These include the discipline-specific tool itself for the role of tool developer, the interface of a tool for the role of the tool integrator, the network connections between administrated machines for the role of network administrator, and the dependencies between the integrated tools for the role of workflow designer. Subsequently, the developers of the workflow design framework must define the interfaces between the involved roles.

It is then the task of the developers of the workflow design framework to design abstractions of all concepts that are on the "correct" level for each involved role. These abstractions do not only simplify the work of each individual user, but also allow for fast iteration on the workflow. A workflow designer can easily explore new designs of a workflow if she does not have to worry about the connections between the involved machines or the implementations of the involved tools.

3 PRACTICAL IMPLEMENTATION

We develop the workflow design framework RCE [1] that allows tool integrators to integrate discipline-specific tools and make them available to other users via a peer-to-peer network of RCE instances. Workflow designers can then build complex automated workflows comprising discipline-specific tools and standard tools provided by RCE. By abstracting from technical details, RCE allows users to focus on those aspects that are relevant to their specific work.

The workflow designer, e.g., only sees the integrated tools as abstract components with specified inputs and outputs. The implementation details of these tools (e.g. their programming language, the operating system they are running on or their network location) are of no concern to the workflow designer and thus not visible to her.

Similarly, details of the underlying peer-to-peer network are hidden by default. Different instances of RCE may be connected via a direct connection, through a local communication hub, or even over the Internet. Although these approaches are quite different technologically, all remote resources are displayed uniformly, as these differences are irrelevant for most roles. For network administrators, however, the statuses and detailed properties of network connections and remote machines are easily accessible.

Our work on RCE and discussions with users allowed us to identify some of the roles involved in operating a workflow design framework. In future work, we aim at developing a comprehensive overview of all roles involved in this task together with a definition of their respective tasks and responsibilities, as well as the concepts they require to perform these responsibilities. Using this catalogue, we will explore further improvements in the user experience of RCE, particularly by providing even more specialized interfaces for the individual roles. Additionally, we intend this catalogue of roles to serve as a basis for further discussion on the general use of workflow design systems in practice.

REFERENCES

[1] Brigitte Boden, Jan Flink, Robert Mischke, Kathrin Schaffert, Alexander Weinert, Annika Wohlan, and Andreas Schreiber. 2019. RCE: An Integration Environment for Engineering and Science. *CoRR* abs/1908.03461 (2019). arXiv:1908.03461 http://arxiv.org/abs/1908.03461

[2] Erwin Moerland, Till Pfeiffer, Daniel Böhnke, Jonas Jepsen, Sebastian Freund, Carsten M. Liersch, Gabriel Pinho Chiozzotto, Carsten Klein, Julian Scherer, Yasim J. Hasan, and Jan Flink. 2017. On the Design of a Strut-Braced Wing Configuration in a Collaborative Design Environment. In *17th AIAA Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2017-4397