Configurable and Collaborative Scientific Workflows

Michael Hahn and Dimka Karastoyanova

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{hahn, karastoyanova}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{INPROC-2014-42,
  author    = {Michael Hahn and Dimka Karastoyanova},
  title     = {Configurable and Collaborative Simulation Workflows},
  booktitle = {Proceedings of the Workshop on Simulation Technology: Systems
               for Data Intensive Simulations (SimTech@GI) in Conjunction with
               INFORMATIK 2014, 22. September 2014, Stuttgart, Germany},
  year      = {2014},
  pages     = {125--136},
  series    = {Lecture Notes in Informatics (LNI)},
  volume    = {P-232},
  publisher = {Gesellschaft f\"{u}r Informatik e.V. (GI)}
}
```

Configurable and Collaborative Scientific Workflows

Michael Hahn and Dimka Karastoyanova

Institute of Architecture of Application Systems (IAAS)
University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart
{hahn, karastoyanova}@iaas.uni-stuttgart.de

Abstract: The use of workflows to support and realize computer simulations, experiments and calculations is well-accepted in the e-Science domain. The different tasks and the parameters of the simulation are therefore specified in workflow models. Scientists typically work in a trial-and-error manner which means they do not know how the final workflow of a simulation has to look like. Therefore, they use a maybe insufficient workflow model as a basis and try to improve this model over multiple iterations to get a better approximation to the problem to solve. So in each iteration multiple trials are based on different variants of the same workflow model. Towards the goal of building variants of workflow models and enabling the reuse of existing scientific workflows in a controlled and well-defined manner, in this paper, we identify how configurable workflow models will support scientists to customize existing workflow models by their configuration. Therefore, we introduce possible configuration options for scientific workflows and how scientists can specify them. Furthermore, we show how configurable workflow models are a first step towards enabling the collaboration among scientists in creating scientific workflows.

1 Introduction

Workflows are well-accepted in the e-Science domain to support and realize computer simulations, experiments or calculations. Workflow models enable scientists to specify the tasks and parameters of a simulation. Due to the fact, that scientists typically work in a trial-and-error manner they use a maybe insufficient workflow model as a basis and try to improve this model over multiple iterations to get a better approximation to the problem to solve. Towards the goal of building variants of workflow models and enabling the reuse of existing scientific workflows in a controlled and well-defined manner, in this paper, we identify how configurable workflow models will support scientists to customize existing workflow models by their configuration.

Figure 1 shows an example for the configurability of a workflow model based on the executable workflow modeling language BPEL [BPE07]. On the right of Figure 1 is a set of variables which are used by the workflow model. The *maxIterations* variable specifies the maximum number of iterations of the *Create Parallel Instances* BPEL While loop activity. The model itself contains a set of service invocations (BPEL Invoke activities) that trigger the execution of existing scientific services, like the *CreateGrid* or *WriteIntermediateRe-*

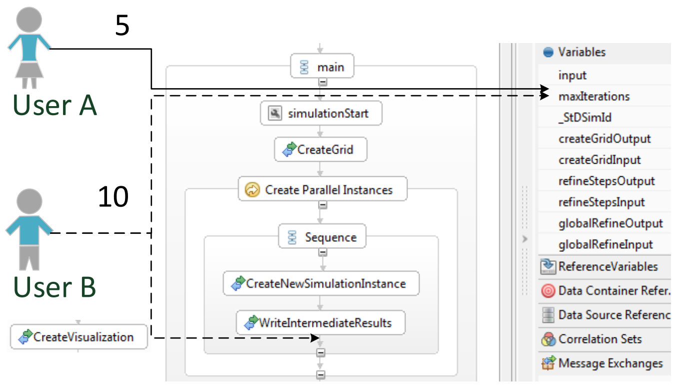

Figure 1: Example of a scientific workflow that is tailored to the needs of two different users through configuration

sults activities. On the left of Figure 1, two users (User A and User B) are shown. Both User A and B register specific values for the *maxIterations* variable. In addition to that, User B enriches the model with an additional activity (*CreateVisualization*) at the end of the while loop. If User A instantiates the workflow model the while loop is executed five times as specified over the registered configuration data. Each instance of the workflow model started by User B, executes ten loop iterations and at the end of each loop cycle the *CreateVisualization* activity is executed. We have already a prototypical realization of a multi-tenant aware workflow engine SCE^{MT}[1] that provides a subset of the configuration capabilities introduced in this paper. Furthermore, interested readers can find a demonstration[2] which shows the configuration of a workflow model with runtime data (user-specific variable values) using SCE^{MT} as described in the example.

The remainder of this paper is structured as follows: Section 2 provides an introduction to multi-tenancy, its capabilities and how the multi-tenancy support of a workflow engine can be utilized to enable the configurability of scientific workflows. Section 3 discusses the configurability of scientific workflows and introduces a set of possible configuration options. Since the support of collaborative work is an important topic when modeling and executing simulations – especially if scientists from different domains work together – the paper provides a discussion how such collaboration can be supported over configuration on the level of scientific workflows in Section 4. In Section 5, we compare our research against existing works. Finally, Section 6 summarizes our findings and presents a short outlook to future work.

2 Relationship between Multi-tenancy, Isolation and Configurability

Even though multi-tenancy is an application property typically considered in the context of Cloud computing, in fact multi-tenancy aware applications or services provide a variety

[1]SCE^{MT}: `http://www.iaas.uni-stuttgart.de/scemt/`
[2]SCE^{MT} Demo: `http://www.iaas.uni-stuttgart.de/scemt/#videos`

of capabilities which are beneficial regardless of how the application or service is provided to the users. In this section we therefore shortly introduce multi-tenancy as a property of applications and its capabilities, which also serves as a basis in the following sections.

In this paper we follow the definition of *multi-tenancy* provided in [SALM12], where the authors define multi-tenancy as "the sharing of the whole technological stack (hardware, operating system, middleware and application instances) at the same time by different tenants and their corresponding users". In the same work the authors also support the claim that resource sharing comes along with/or is a prerequisite for *isolation* (e.g. data, communication) of tenant's resources, *configurability* and *scalability*.

When talking about multi-tenancy it is important to distinguish two types of users: *tenants* and *tenant users*. A tenant is used to separate users of a multi-tenant aware application or service into organizational units like companies or departments which share the same view on the application or service [KMK12], [SALM12]. Since these organizational units are not necessarily completely disjoint and a user may belong to multiple tenants at the same time, tenant users enable the distinction of users belonging to more than one tenant. Multi-tenancy implies that applications or services are able to identify the resources belonging to a tenant (e.g. an institute) or tenant user (e.g. a scientist). For example such resources are data like simulation parameters or event data, or software artifacts like workflow models or scientific services.

The isolation of tenants and their users in nearly all parts of a system is the main requirement to provide multi-tenancy support [GSH$^+$07, KMK12]. From a users point of view, isolation is an important capability of a system from which users can benefit. The following list describes the most important isolation types and some examples how the different types change the behavior of a system.

Data Isolation ensures that the data belonging to one tenant user is not accessible by any other user of the system. Chong et al. introduce three different approaches to realize data isolation on a database level [CCW06]. As a result, a multi-tenant database enables the sharing of tables between multiple users and simultaneously protects the data from any unintended access by other users.

Communication Isolation is a special kind of Data Isolation which isolates the communication (message exchanges) between two or more applications or services on a tenant user basis [SALM12]. This provides the ability to enforce the authentication of requests sent to (scientific) workflows or services by checking if an incoming request message belongs to the same tenant user as the workflow or service to invoke. Therefore, scientists can ensure that only allowed tenant users are able to invoke their scientific workflows or services.

Administration Isolation is provided if no tenant user is able to manage or administer resources which belong to another tenant user. For example, a SWfMS provides functionality for the administration of the workflow engine, workflow models and workflow instances. Administration Isolation enforces, that all resources of a tenant user can only be managed by himself.

Performance Isolation ensures that each tenant user gets the performance he paid for and that the potentially detrimental behavior of one tenant user does not adversely affect the system performance for other tenant users [WMTJ12, GSH$^+$07]. In the domain of scientific workflows, performance isolation can secure that all simulations running on a shared system are executed with predefined performance (not violating the performance of concurrently executed simulations) and the system automatically scales if the guaranteed performance can not be provided any longer.

Configurability enables the adaptation of the behavior or appearance of an application or service without the necessity to change the underlying implementation to the needs of tenant users [KMK12], [GSH$^+$07]. Such customization is only possible if the application or service provide a set of configuration possibilities, which in turn enable users to perform ad-hoc customization of reusable artifacts to their needs. Since configurability is a main requirement for multi-tenancy to comply with the needs of all served users, multi-tenant aware applications provide natively a set of configuration options. For example, workflow models which provide commonly used functionality required in many scientific workflows can be adapted through configuration and therefore reused without the necessity to change the underlying model.

3 Configurable Scientific Workflows

Configurability enables users to customize predefined workflow models based on their needs by providing in addition a set of configuration data without the necessity to create new or adapt existing workflow models. These configuration data are used in all instances of a workflow model belonging to a single user. This allows users to customize an existing model by overriding predefined values (e.g. constants specified in a workflow model or transition conditions between activities) or registering customized control flow logic (e.g. additional activities or workflow fragments). Furthermore and because of this, workflow modelers are able to define more generic and reusable models (workflow templates) which can be adapted by the users themselves through configuration. For example, any user specific values of a scientific workflow (e.g. parameters or boundary conditions for computations) can be dynamically set upon workflow instantiation by using registered model configurations related to the user who instantiates the workflow model. Since the configuration data is stored independently of the workflow models, multiple tenant users can use the same workflow model as a basis for configuration without the necessity to maintain their own copy. If the shared workflow model has to be improved over time, only the single model has to be adapted and then the changes are dynamically reflected in all new created workflow instances of all tenant users.

We distinguish between three different types of configuration options for workflow models, the configuration of *workflow model runtime data*, *workflow model logic* and *workflow model appearance*. In the following each of these three categories and a selection of possible configuration options will be described.

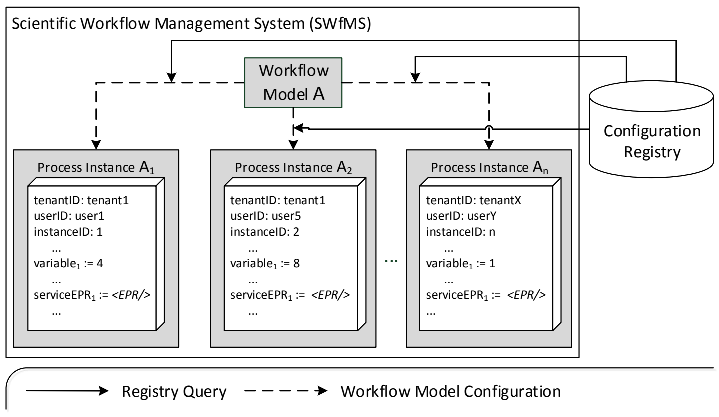

Figure 2: Example of a set of workflow instances using different runtime data which are assigned through the configuration of the workflow model

3.1 Configuration of Workflow Model Runtime Data

Figure 2 shows an example scenario for the configuration of *runtime data* of a workflow model. Runtime data can be registered for all elements of a workflow model which hold some data during the runtime (execution) of an instance of the workflow model. For example, the (initial) values of *variables* or *service endpoint references* can be specified over runtime data configurations. This means that a user can register a value for an element (e.g. a variable) which is used by the SWfMS to initialize the element whenever a new instance of the workflow model is created by the user. The registration and administration of the configuration data of the users can be realized over a *Configuration Registry*. To isolate the configuration data of all served tenant users, the registry has to provide multi-tenancy support. The configuration of variable values can be used for example to register user-specific constants or initial values. The configuration of service endpoint references makes it possible to change the target endpoint of a modeled service invocation. For example, if a tenant user hosts another (potentially adapted) version of a service invoked by the model, he can overwrite the specified service endpoint reference through configuration. Upon workflow instantiation the registered values are queried from the Configuration Registry by the SWfMS and used to initialize the corresponding workflow model elements for which they are registered. Based on the associated tenant information of the initial request which triggers the instantiation of the workflow model, the correct tenant specific runtime data is determined automatically. Since a workflow model may specify initial values for some of its elements, the configuration data has to be dynamically set during runtime to avoid that it is overwritten by such predefined initial values. As a result, the configuration of runtime data is also applicable to set/change the initial values or constants defined in a workflow model without the need to change the model. This makes it possible that multiple, parallel executed instances of the same workflow model triggered by different users, automatically use the correct configuration (runtime) data.

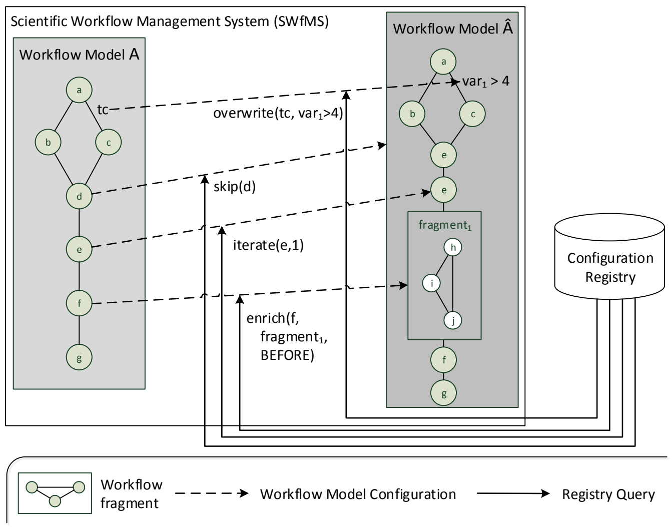

Figure 3: Example of creating a variant of a workflow model through the configuration of its control flow logic

3.2 Configuration of Workflow Model Logic

Another much more flexible and powerful option is the configuration of *workflow model logic* because users are then able to customize the control flow and therefore the behavior of the workflow. Based on the underlying workflow modeling language different configuration possibilities exist, in terms of different sets of modeling constructs which can be used for the configuration of a model. In general there are three different ways how the control flow can be configured.

First of all, the control flow can be ***enriched*** by the registration of additional workflow logic in form of workflow fragments (control flow snippets). A workflow fragment is a self-contained, well-formed, partially defined workflow model, like a sequence of activities [ELS$^+$10]. These fragments are then dynamically weaved into the workflow model at the beginning of its execution (instantiation) triggered by a tenant user. As a result, the executed instance is based on an enriched, temporary workflow model which is the aggregate of the original (template) model and all registered workflow fragments. Figure 3 shows an example for such a scenario, where *Workflow Model A* is configured by enriching it with *fragment$_1$* to a temporary *Workflow Model Â*. The exact position where a fragment should be inserted into the workflow model, is specified by referencing an existing activity and a constraint which defines if the fragment should be placed before or after this activity (*enrich(f, fragment$_1$, BEFORE)*, Fig. 3).

Second, the control flow can be ***restricted*** by adding new transition conditions between two activities or by skipping activities as shown in Figure 3. There activity *d* is skipped and therefore not represented any longer in the configured temporary model Â (*skip(d)*).

Third, the existing control flow logic can be ***adapted*** by overwriting already defined tran-

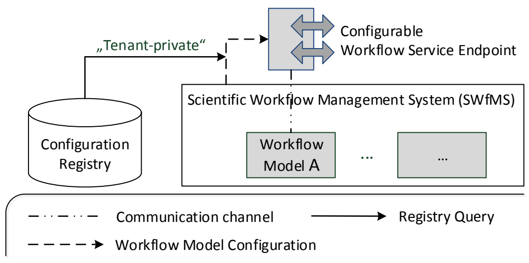

Figure 4: Example of configuring the accessibility of a workflow model

sition conditions or any other expressions (e.g. loop conditions in loop activities), or by configuring the repeated execution of modeled activities – also known as *iteration* [SK11]. Figure 3 shows an example for both adaptation possibilities. The defined transition condition *tc* of model A is configured by overwriting it with an another expression (*overwrite(tc,* $var_1 > 4$*)*) so that the link between activity *a* and *c* is only fired if $\text{var}_1 > 4$. Furthermore, activity *e* is configured to be iterated once (*iterate(e,1)*) and therefore is executed twice in each instance of the temporary workflow model Â. Sonntag et al. provide additional information on such an iteration operation for the repeated execution of activities [SK11].

All three configuration possibilities adapt the control flow of the workflow model on an instance level based on the tenant user. Available adaptation approaches from service compositions and workflows can be used to support such configuration.

3.3 Configuration of the Workflow Model Appearance

Figure 4 shows one possibility to configure the *workflow model appearance*, which means how the workflow is visible and accessible from the outside through its service interface. In the following we want to introduce and describe the configuration of the visibility and the level of Communication Isolation – more precisely the accessibility of the service interface – of a workflow model based on three constraints. These access constraints enable the owner of a workflow model to specify a scope with specific visibility and access restrictions. As a result, only well-defined groups of tenant users have access to the corresponding sets of workflow models: *public* – all tenant users of all tenants, *tenant-private* – all tenant users of one tenant and *user-private* – only one tenant user of one tenant.

Public: Workflow models that are configured as *public* do not have any access restrictions and therefore are visible to/accessible by all users of all tenants. Any incoming request sent to the service interface of a public workflow model is forwarded without any authentication. These workflow models can be used by any user of any tenant without any restrictions. The fact that a workflow model is public does not mean that its instances are also accessible by any user. If a public workflow model is instantiated by a tenant user, the instance is directly associated to this user. This secures all tenant-specific instances and their data from any unauthorized entities (Data Isolation). If a public workflow model is instantiated without any provided tenant user information, the instance is also public. In this case, everyone is able to manage these instances via the administration functionality of the workflow engine. This access constraint enables for example the provisioning and sharing of workflow models which provide some useful domain-specific functionality to all tenants and their users while still enforcing Data Isolation on the workflow instance level. Furthermore, this constraint enables backward compatibility because non multi-tenant aware applications or services are able to send requests to public workflow models.

Tenant-private: This constraint isolates a workflow model on a tenant basis. Therefore the workflow model is only visible/accessible by the users of one specific tenant. The workflow engine only processes requests which reference the same tenant as the one associated to the workflow model to which the request is sent. For example, if the workflow model is associated to tenant A only requests of users of tenant A are processed, any other requests are rejected by the workflow engine. Requests without any associated tenant information have to be rejected. Similar to the public constraint case, the created instances of a tenant-private model belong to a single entity identified by the tenant information of the initial request sent to the service interface of a workflow model.

User-private: Workflow models which are configured as *user-private* are only visible to/ accessible by one specific user. The workflow engine only processes requests referencing the same tenant and tenant user as the one associated to the workflow model to which the instantiating request was sent. Requests without any associated tenant information are rejected. The created instances of a user-private model belong to a single entity identified by the tenant information of the initial request sent to the service interface of a workflow model. This is the default access constraint which is used when no configuration data is specified for the accessibility of the service interface of a workflow model.

3.4 Associating Configuration Data to Workflow Model Elements

In general, users should be able to identify and as a result, reference any element of a workflow model for which they want to specify configuration data. For example, an id, the name of the element or an XPath expression for XML[3] based model languages can be used to identify an element of a workflow model. The configuration possibilities of a workflow model range from complete configurability of the entire model to complete restriction of configurability (only some elements are configurable through a predefined set of configuration data). On the one extreme, the users of the workflow model want to be able to configure all parts of the model, while on the other, the providers of a workflow model want to restrict the configurability to some parts of the workflow using preset values in terms of security and control.

The configurability of a workflow model can be defined through the workflow model itself by specifying corresponding meta-data which mark all configurable elements and configuration possibilities. If there are no configurable (marked) elements modeled, a user is not able to configure anything. The enrichment of a model with additional activities can also be restricted using placeholders. Therefore, the workflow modeler should be able to

[3]W3C, Extensible Markup Language (XML) 1.0:`http://www.w3.org/TR/xml/`

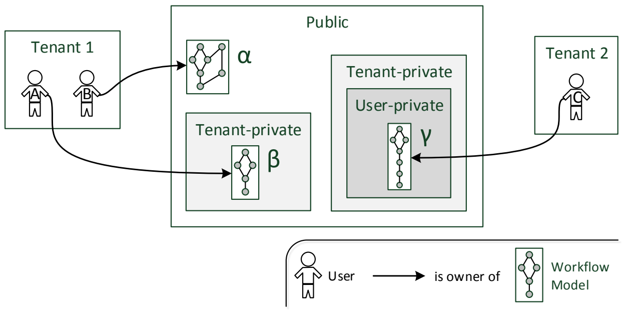

Figure 5: Specifying access scopes for workflow models based on predefined access constraints

insert such placeholders into a workflow model. The users can then register customized control flow logic only for a set of predefined positions in a workflow model which are represented by the modeled placeholders. The definition of different kinds of restrictions, how they can be specified in a workflow model and enforced during configuration time is out of the scope of this paper.

4 Discussion

Especially for scientific workflows, where a variety of scientists are working together to realize coupled simulations (e.g. multi-scale, multi-physics or multi-domain), collaboration is an important aspect. In our view, multi-tenancy can be used as a basis to enable the collaboration for scientific workflows. As described in Section 2, multi-tenancy provides already the required functionality because it makes it possible to establish the connection between users and their resources, isolate them in a defined manner and enables the user-based configuration of these resources. By default multi-tenancy realizes the strict isolation of resources among tenant users. If we support that this isolation can be configured in a well-defined and secure scope, then resources can be accessed and used in collaboration by multiple users. Therefore, tenant users should be able to specify access permissions for their resources (e.g. workflow models or instances) and any operations (e.g. read, change or delete) which are applicable to these resources to enable a secure collaboration between multiple tenant users. Furthermore, these access permissions should be easily assignable across tenants' boundaries, dynamically defined during all phases of a workflow model or workflow instance (e.g. modeling, deployment, runtime) and defined in a fine-grained manner on an operation level.

As a first step to enable the collaborative work on the level of workflow models, which means sharing one workflow model between multiple users, a mechanism to share them across tenant boundaries is required. The ability to configure the accessibility of a service interface described in Section 3 is such a mechanism which enables the sharing of a workflow model based on a set of predefined access constants. The SWfMS uses these constants to check if the request send to a service interface belongs to a tenant which is contained in the configured access scope of the workflow model.

Figure 5 shows how the access scopes can be used to enable the collaboration on a workflow model level. *User A* of *Tenant 1* is the owner of one tenant-private workflow model (*Workflow Model* β) which can be used by all users of the tenant (Users A and B). In contrast to that, *User B* is the owner of a public workflow model (*Workflow Model* α) which is visible/accessible by all users (A, B, C) of both tenants (1 and 2). *User C* of *Tenant 2* is the owner of a user-private workflow model (*Workflow Model* γ) which is isolated from all other users and therefore is only visible to User C. Based on Figure 5 consider the following scenario. User B has specified workflow model α which provides some useful and reusable functionality. User C requires exactly the same functionality as provided by workflow model α, but is not able to realize an equivalent model on his own. With the help of the introduced collaboration mechanism, User B is now able to simply enable User C to use his model without the necessity to transfer the workflow model to User B. So all users, no matter if they belong to the same organizational unit (tenant), can share their workflow models with others and as a result benefit from the joint use.

5 Related Work

Van der Aalst, Gottschalk et al. introduce configurable process models as a basis for reference modeling in [vdADG$^+$06] or [GvdAJV07]. They point out that a reference model which provides a general solution can be specialized for a concrete scenario over configuration. This enables the creation of new models by selecting specific parts of a configurable (reference) model that are relevant to the user [vdADG$^+$06, GvdAJV07]. Based on that, they describe "configuration as the inverse of inheritance" which means instead of enriching a model with new functionality, configuration restricts a general model to a concrete model. Therefore, they introduce two different techniques for the restriction of the control flow of a model, called *blocking* and *hiding*. In our work, we take into account additional configuration possibilities like the customization of runtime data, accessibility of the model or the enrichment of the control flow of a workflow model.

La Rosa et al. state that "a configurable process model captures the commonalities and the variability of a family of process models in a consolidated manner" [LRDTHM11]. As a result, users are able to specify values to individualize a configurable process model based on their requirements. Therefore the individualization of the configurable process models can be done automatically which removes a tedious and error-prone step in case the user requirements are reflected over the manual adaptation of the process model. They agree with the definition of configurable process models provided in [vdADG$^+$06] and [GvdAJV07]. In addition, they describe the shortcomings of existing configurable process modeling approaches and their lack of mechanisms for enable configurability not only for the control flow of process models. Therefore, they extend the Configurable Event-Driven Process Chain (C-EPC, [RvdA07]) notation to enable the definition of roles (e.g. a organizational unit) and objects (e.g. data). Based on that, they introduce a range of configurations for the way how those roles and objects and their association to tasks can be configured.

In [LL07] the authors introduce a formal model to define different customization options for BPEL processes. Based on that and on the idea of reference models as defined in the works mentioned above, they provide the ability to specify reference process models containing possible customization options.

All the discussed related works use a reference model as a basis to enable configurability. The reference model therefore has to be specified using a modeling language which implicitly provides a set of configuration possibilities to individualize the model for a concrete scenario. In contrast to the business domain, scientific workflows are ever-changing models because scientists work in a trial-and-error manner to specify and improve the modeled simulations. In this case, it is not possible to predefine all variants of a scientific workflow in a reference model and as a result enabling configurability based on reference models is not an applicable approach for the scientific domain. Therefore, in this work we focus on enabling the configurability of workflow models without requiring a reference model specified with an enriched modeling language. This allows users to seamlessly build multiple variants of a workflow model or even of previously created variants just through the association of configuration data to the workflow model (see Sec. 3). As a result, any existing workflow model can be tailored on a per-user-basis through configuration using e.g. restriction, enrichment, replacement, overwriting.

6 Conclusion and Future Work

Our main goal in this work is to introduce the concept of configurable scientific workflow models. We described how multi-tenancy can be used as a basis to support the configuration of workflow models in a secure and well-defined manner. Furthermore, we introduced three different types of configuration options, which refer to the customization of different layers of a workflow model. For each of these three options, we presented concrete configuration possibilities. As a first step towards collaboration in scientific workflows, we presented our vision about how collaboration can be enabled through configuration.

Our current work focuses on the extension of our existing multi-tenant aware workflow engine SCE^{MT} to support all introduced configuration options and their validation using a case study.

In the future, we aim at supporting *configuration views*, in order to enable the customization of nearly all parts of a workflow model using the introduced configuration options. These views provide to the users a graphical representation of the resulting virtual models, which are the aggregate of the initial model and all specified configuration data.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310) at the University of Stuttgart.

References

[BPE07] Organization for the Advancement of Structured Information Standards (OASIS). Web Services Business Process Execution Language Version 2.0. OASIS Standard, 2007.

[CCW06] Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-Tenant Data Architecture, 2006.

[ELS$^+$10] Hanna Eberle, Frank Leymann, Daniel Schleicher, David Schumm, and Tobias Unger. Process Fragment Composition Operations. In *Proceedings of APSCC 2010*, pages 1–7. IEEE Xplore, Dezember 2010.

[GSH$^+$07] Chang Jie Guo, Wei Sun, Ying Huang, Zhi Hu Wang, and Bo Gao. A Framework for Native Multi-Tenancy Application Development and Management. In *E-Commerce Technology and CEC/EEE 2007*, pages 551–558, 2007.

[GvdAJV07] Florian Gottschalk, Wil MP van der Aalst, and Monique H. Jansen-Vullers. Configurable Process Models – A Foundational Approach. In Jrg Becker and Patrick Delfmann, editors, *Reference Modeling*, pages 59–77. Physica-Verlag HD, 2007.

[KMK12] Rouven Krebs, Christof Momm, and Samuel Kounev. Architectural Concerns in Multi-tenant SaaS Applications. In *Proceedings of CLOSER'12*, pages 426–431, 2012.

[LL07] Alexander Lazovik and Heiko Ludwig. Managing process customizability and customization: Model, language and process. In *Web Information Systems Engineering–WISE 2007*, pages 373–384. Springer, 2007.

[LRDTHM11] Marcello La Rosa, Marlon Dumas, Arthur HM Ter Hofstede, and Jan Mendling. Configurable Multi-perspective Business Process Models. *Information Systems*, 36(2):313–340, 2011.

[RvdA07] Michael Rosemann and Wil MP van der Aalst. A configurable reference modelling language. *Information Systems*, 32(1):1–23, 2007.

[SALM12] Steve Strauch, Vasilios Andrikopoulos, Frank Leymann, and Dominik Muhler. ESB^{MT}: Enabling Multi-Tenancy in Enterprise Service Buses. In *Proceedings of CloudCom'12*, pages 456–463, 2012.

[SK11] Mirko Sonntag and Dimka Karastoyanova. Enforcing the Repeated Execution of Logic in Workflows. In *Proceedings of BUSTECH'11*, pages 1–6. IARIA, September 2011.

[vdADG$^+$06] Wil MP van der Aalst, Alexander Dreiling, Florian Gottschalk, Michael Rosemann, and Monique H. Jansen-Vullers. Configurable Process Models as a Basis for Reference Modeling. In *Business Process Management Workshops*. Springer, 2006.

[WMTJ12] Stefan Walraven, Tanguy Monheim, Eddy Truyen, and Wouter Joosen. Towards Performance Isolation in Multi-tenant SaaS Applications. In *Proceedings of MW4NG'12*, page 6. ACM, 2012.