TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

Habilitationsschrift

On Quality Issues in Complex Service-oriented Systems

Eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Dr.techn. Hong-Linh Truong
Kagraner Platz 17/H2, A-1220, Wien

Wien, September 11, 2012

To Tan-Nghia, Ha-Anh, and Thu-Ha for your love, patience and devotion.
To my parents for your continously curiosity about my work.

Preface

Over the years, complex service-oriented applications have evolved rapidly, from being compute-intensive to data-intensive, and from composing purely machine-based data and computational services to composing machine-based and human-based data and computational services. Furthermore, the underlying computing systems for these applications have evolved from Grid systems of software and hardware, to Cloud systems of virtualized machine-based software and hardware, and to Cloud systems of mixed machine-based and human-based services.

Such evolutions pose several challenges for us to understand the quality of complex service-oriented applications and systems and how to evaluate, optimize and utilize the quality. The quality is no longer centered around traditional performance metrics of applications, but also associated with data and human works and with interactions among them. Furthermore, the increasing utilization of software services, data services, and human services in the same complex applications requires us to have a deep understanding of multiple quality properties associated with different things and at different levels in order to support the composition, execution and management of these applications and systems efficiently.

This habilitation thesis summarizes our research outcomes in characterizing and evaluating quality issues in complex service-oriented systems. We focus on workflow-based applications, collaboration processes, and high performance applications composed from diverse types of services and executed on Grid and Cloud systems. We present several research problems together with our approaches and solutions to evaluate performance metrics associated with workflows, to measure quality of data for workflows, to monitor and analyze human and service interactions in collaboration processes and to evaluate cost of complex applications. To consider the importance of data concerns in complex service-oriented applications and systems, we examine possible data concerns for Data-as-a-Service and present different solutions for the specification, monitoring, evaluation and publishing of data concerns. Furthermore, we also address the reconciliation of service contracts to deal with diverse non-functional properties of applications and systems as well as we propose a novel data contract model for data marketplaces.

From our understanding and analysis of quality for complex service-oriented systems, we observe newly emerging dynamic distributed systems of machine-based and human-based computing elements that call for a new research direction in elastic computing. This leads to our discussion of the future research direction in multi-dimensional elasticity for complex applications and systems in which we describe the Vienna Elastic Computing Model and some techniques for realizing multi-dimensional elasticity concepts.

Vienna, September 11, 2012 *Hong-Linh Truong*

Acknowledgments

After finishing my PhD, I knew that I would like to stay in research and academic environments. It was my mentor and advisor, Schahram Dustdar, who has continuously encouraged me to carry out my habilitation plan, right after I joined the Distributed Systems Group at the Vienna University of Technology in 2007. It is also him who shows me how to pursuit a habilitation and reminds me about important things for finishing a successful habilitation thesis. As the head of the group, Schahram has created a great environment that allows me to freely pursuit my ideas. As a colleague, Schahram has intensively shared his ideas and engaged in discussing new research fronts with me that has led to several interesting joint works partially mentioned in this thesis. I want to thank him for all the things he has dedicated to me and for his strong support for my research career.

My former PhD advisor, Thomas Fahringer at the University of Innsbruck, has offered me a chance to work on an exciting topic that also supports my work in this habilitation. I want to thank him for our great collaboration when I was at the University of Innsbruck.

This habilitation summarizes part of my research over the last few years. I have great opportunities to collaborate with several people, including colleagues and students in groups where I have been as well as my PhD internships, visitors and external collaborators. Results from several collaborations have contributed to this thesis. Among them, I would like to thank the following people for sharing, discussing and working with me in several joint research ideas and contributions mentioned in this habilitation: Kamal Bhattacharya, Peter Brunner, Marco Comerio, Vincenzo D'Andrea, Schahram Dustdar, Thomas Fahringer, G.R.Gangadharan, Lam-Son Le, Frank Leymann, Andrea Maurino, Michael Mrissa, Vlad Nae, Flavio De Paoli, Tran-Vu Pham, Reinhard Pichler, Michael Reiter, Robert Samboski, Vadim Savenkov, Martin Treiber, and Quang-Hieu Vu.

My research in this thesis has been supported by several research funds, including EU FP6 K-WfGrid, EU FP6 inContext, EU FP7 COIN, EU FP7 COMMIUS, and WWTF SODI. I would like to thank my colleagues in these projects for the opportunities to discuss ideas and for sharing applications for experiments.

Not in my office but at home, for my research on this habilitation, I have taken a lot of time and energy that should have been dedicated to my family. I am in debt to my small kids – Tan-Nghia and Ha-Anh – and my loving wife – Thu-Ha – for their understanding, patience and support. My wife has taken care for everything of my family, allowing me to freely concentrate on my research. Without their love and support, I would never be able to conclude this thesis.

Vienna, September 11, 2012

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
CRUD	Creat, Read, Update, Delete
CSE	Computational Science and Engineering
DaaS	Data-as-a-Service
DIPAS	Distributed Performance Analysis Service
eHPA	elastic High Performance Application
EU	European Union
HCE	Human-based Computing Element
HPA	High Performance Application
HPC	High Performance Computing
HBS	Human-based Service
HPS	Human-provided Service
IaaS	Infrastructure-as-a-Service
ICU	Individual Compute Unit
IT	Information Technology
MCE	Machine-based Computing Element
MPI	Message Passing Interface
NaaS	Network-as-a-Service
NFP	Non-functional properties
PaaS	Platform-as-a-Service
REST	Representational State Transfer
ROO	Return On Opportunity
SaaS	Software-as-a-Service
SBS	Software-based Service
SCU	Social Compute Unit
SLA	Service Level Agreement
SOA	Service-oriented Architecture
SOC	Service-oriented Computing
SOAP	Simple Object Access Protocol
QoD	Quality of Data
QoS	Quality of Service

WADL	Web Application Description Language
VieCOM	Vienna Elastic Computing Model
WfMS	Workflow Management System
WfPerfOnto	Workflow Performance Ontology
WS	Web service
WSDL	Web Service Description Language

Part I
Techniques for Characterizing and Evaluating Quality Issues in Complex Service-oriented Systems

Chapter 1
Introduction

1.1 The Evolution of Systems and Applications

Over the last years, large-scale computing systems have been evolving rapidly. Grid computing and similar techniques [42, 16] have enabled the aggregation of distributed computing resources and virtualized them as an integrated, large-scale computing system for complex applications. Cloud computing has not only been able to consolidate computing resources into large-scale data centers by using virtualization techniques at the infrastructure level but also provided virtualization for programming, deploying and executing platforms and software, introducing Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) models [10, 40, 43, 20]. Furthermore, data have been also provided under cloud models, establishing so-called Data-as-a-Service (DaaS) and data marketplaces. Cloud computing also presents several new business models. Social computing [29] has not only enabled people to collaborate via social networks and collaboration services but also fostered emerging human computation models, in which humans play the role of computing resources to provide computational services and data. Service-oriented Computing (SOC) and Service-oriented Architecture (SOA) technologies [71] have enabled Grid computing, social computing, and cloud computing models by simplifying the provisioning of human, data and software under the service model and the composition of them into complex applications. All of them have fostered the development of emerging distributed applications and systems that include and integrate machines, data sources, sensors and humans.

Modern compute- and data-intensive e-science and business applications involve large-scale and complex computational services, data services, and humans, as shown in existing applications [108, 30, 76, 85, 78]. On the one hand, SOC techniques have facilitated the integration and execution of software and data services. Moreover, workflow techniques have been widely adopted for complex data processing applications in large-scale distributed systems, such as Internet, Grids and Clouds. On the other hand, diverse types of data have been integrated from different sources, including real-time sensors and humans. These applications and enabling techniques establish complex service-oriented data-intensive and compute-intensive computing problems that require us to understand behavior of complex workflows, data concerns, and computing systems and to monitor, analyze and optimize them.

1.2 Complex Service-oriented Applications and Systems

Our work focuses on characterizing and evaluating quality associated with complex service-oriented foundations. In particular, the conceptual models of complex service-oriented applications throughout our study are workflow-based applications and distributed, service-based applications. Throughout the course of our study, these applications have evolved rapidly to meet the changes of their consumers and underlying computing systems.

Figure 1.1 depicts a sketch of our typical applications, their consumers and underlying computing systems. On the one hand, the consumers of our complex systems have moved from individual humans to teams and also middleware/software services. This has a strong impact on the need to understand different quality criteria expected from these types of consumers as well as suitable interfaces to provide such expected quality. On the other hand, the underlying computing systems have changed from Grid systems of purely machine-based resources, such as computational services and data services, to Cloud systems of different types of services, such as SaaS and DaaS, and to Clouds of mixed machine-based and human-based services. The changes of the underlying computing systems have a profound impact on our quality characterization and evaluation study. Not only we need to examine new quality metrics associated with the underlying systems, but we also need to relate these metrics back to the studied applications.

1.3 Quality in Complex Service-Oriented Foundations

For complex applications, their consumers, and underlying computing systems shown in Figure 1.1, numerous types of quality issues need to be characterized and evaluated. In the context of our study, we see the need to characterize and evaluate:

- performance metrics, human interactions, and quality of data (QoD) associated with complex applications.
- quality of service (QoS), data concerns, and service contracts associated with different types of services in the underlying computing systems.
- consumer quality specification and quality provisioning for services.

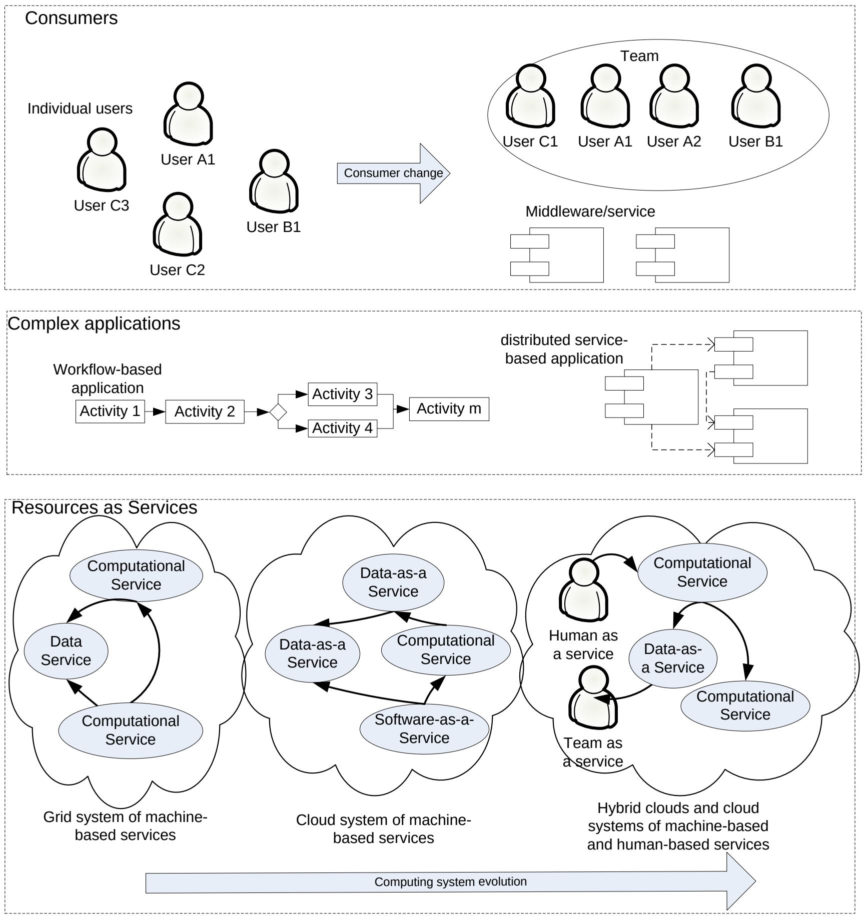

Fig. 1.1 The complex service-oriented system including resources as services, dynamic processes, and teams

Characterizing and evaluating them raise several research challenges. In this thesis, we will present some of research challenges and our approaches and solutions.

1.4 Research Journey – Research Challenges and Contributions

In general, complex applications considered in this thesis can be divided into service-based workflows, including scientific workflows and business processes, collaboration services, and high performance applications. These applications

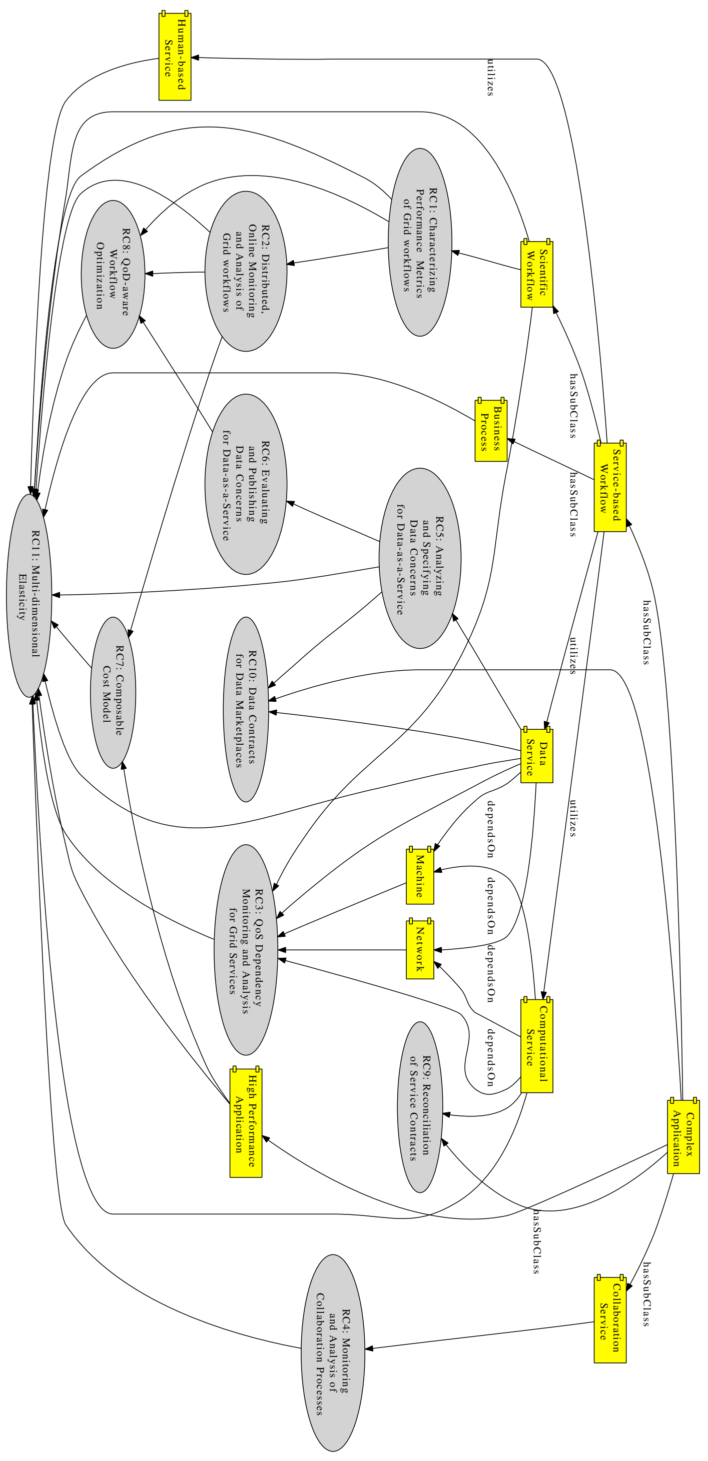
Human-based Service
utilizes
RC1: Characterizing Performance Metrics of Grid workflows
Scientific Workflow
RC2: Distributed, Online Monitoring and Analysis of Grid workflows
RC8: QoD-aware Workflow Optimization
hasSubClass
Business Process
hasSubClass
Service-based Workflow
RC6: Evaluating and Publishing Data Concerns for Data-as-a-Service
RC5: Analyzing and Specifying Data Concerns for Data-as-a-Service
RC11: Multi-dimensional Elasticity
hasSubClass
utilizes
RC7: Composable Cost Model
RC10: Data Contracts for Data Marketplaces
Data Service
dependsOn
utilizes
Machine
dependsOn
RC3: QoS Dependency Monitoring and Analysis for Grid Services
Network
dependsOn
Computational Service
dependsOn
RC9: Reconciliation of Service Contracts
High Performance Application
Complex Application
hasSubClass
hasSubClass
Collaboration Service
RC4: Monitoring and Analysis of Collaboration Processes

Fig. 1.2 complex applications, services and their research topics utilize different types of services, such as computational services, data services, and human-based services.

Figure 1.2 outlines the research journey carried out in this thesis, including (simplified) types of applications and services and their relationships, and research challenges (RCs). The research carried out in this thesis has started with scientific workflows of Grid services in Grid computing systems. For scientific workflows, we have focused on compute-intensive workflows, in which computational services (and their dependent machines and networks) are the most important things. Since the performance is a critical issue, our research was motivated by RC1 – *characterizing performance metrics for large-scale scientific workflows* – and RC2 – *how to monitor and analyze these metrics at runtime to support the user and middleware.* Our contributions are a novel workflow performance ontology and techniques for analyzing workflow performance [98, 88]. By dealing with RC1 and RC2, we identified RC3 – *monitoring and analysis of QoS of interdependent services* – and we contribute novel solutions to deal with quality of service (QoS) dependency among Grid services required by Grid workflows [104].

In our next step, we moved from scientific workflows to collaborative processes which are more adhoc and human-intensive and we examined interactions between services and humans in collaboration processes in RC4 – *monitoring and analysis of collaboration processes.* This leads to our contribution on the monitoring and analysis human and service interactions in SOA-based collaboration processes [92].

Realizing that workflows are utilizing more and more data from different sources, data intensive workflows are becoming important and cloud computing models enable the provisioning of data under the services, we started to investigate data services and data concerns. We identified RC5 – *analyzing and specifying data concerns for Data-as-a-Service* – and RC6 – *evaluating and publishing data concerns.* Our solutions for RC5 and RC6 are several techniques for specifying, evaluating and publishing data concerns [91, 94], leading to other applications, such as concern-aware querying [9] and concern-aware service design [100, 89].

With the emerging cloud computing, we realized that while complex applications, in particular high performance application (HPA) for scientists in computational science and engineering (CSE), can be composed from different application models, determining cost benefits for complex applications when utilizing Clouds is crucial. This leads to our RC7 on *cost evaluation for cloud applications.* Our solutions introduce a novel composable cost model and evaluation techniques that are partially built based on the result from RC2 [93]. The experience and results from dealing with RC1, RC2, and RC6 leads us to the identification of RC8 – *QoD-aware workflow optimization* – w.r.t. how to optimize workflow composition and execution by utilizing QoD. We introduce different techniques for monitoring and evaluating QoD of workflows [78, 79].

As both business processes and scientific workflows rely more and more on third-party data and computational services, we face with several questions related to service and data contracts. We identified RC9 – *reconciliation of service contracts* - and RC10 – *data contracts for data marketplaces* – and contribute a set of techniques for reconciliation of service contracts as well as contract compatibilities and a novel data contract model [101, 102, 27, 90].

Finally, experience and results from many RCs have led to our identification of RC11 – *multi-dimensional elasticity* for service-oriented foundations. We contribute to the definition of the Vienna Elastic Computing Model and modeling and composition techniques as initial results for our future research plan [95, 103, 38, 74, 97].

1.5 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 presents research problems and solutions for quality evaluation of complex applications, covering RC1, RC2, RC3, RC4, RC7 and RC8. Chapter 3 describes research questions and our contributions for the analysis, evaluation and provisioning of data concerns associated with data-a-a-service in the Cloud, covering RC5 and RC6. Chapter 4 presents our solutions for reconciling service contracts and introduces data contracts, covering RC9 and RC10. Chapter 5 discusses initial research development for the elasticity of service-oriented foundations, covering RC11.

The thesis also includes representative published papers. Appendix A includes five papers related to the monitoring and analysis of quality for complex applications. Appendix B includes two papers describing the analysis, evaluation and publishing of data concerns. Appendix C includes two papers describing reconciliation techniques and data contracts for data marketplaces. Appendix D includes three papers describing elasticity requirements and initial solutions for multi-dimensional elasticity concepts.

Chapter 2
Characterizing and Evaluating Quality of Complex Applications

2.1 Introduction

As mentioned before, complex applications that we consider in our work are built by utilizing multiple computational and data services and executed on large-scale distributed computational infrastructures. In this chapter, we focus on the workflow-based applications atop Grid and cloud infrastructures.

First, we consider Grid workflows which have been increasingly exploited as the main programming model for addressing large-scale e-science problems, as demonstrated by a large number of Grid workflow systems [108] and applications [30, 76, 85]. As the Grid is diverse, dynamic, and inter-organizational, the execution of Grid workflows is very flexible and complex. Therefore, we focus on understanding the performance behavior of Grid workflows and techniques to monitor and analyze Grid workflow performance at multiple levels of detail to detect and correlate components that contribute to performance problems. Furthermore, as underlying Grid systems include several interdependent services at different levels of abstraction, such as machines, software and network, we focus on monitoring and analysis of QoS dependency among Grid services.

Second, we consider collaboration processes that involve different collaboration services and people in service-oriented environments. We identify and develop metrics that characterize interactions among humans and services in order to support collaboration adaptation. Third, we consider complex applications composing different application models on clouds. We address the question about how to monitor and evaluate the cost for such complex applications. Finally, as dealing complex data is one of the main challenges in scientific workflows and as supporting quality of data (QoD) in scientific workflows is indispensable but has been neglected so far, we aim at understanding how QoD impacts on the execution of multi-scale workflows, which measurement processes can be developed for QoD of workflows, how to determine QoD metrics for multi-scale workflows, and how to utilize these metrics for the optimization of the composition, execution and resource provisioning and usage of multi-scale workflows.

2.2 Characterizing Performance Metrics for Workflows

2.2.1 Research problems

To understand the performance of Grid workflows, performance metrics of the workflows have to be studied and defined. However, there is a lack of a comprehensive study of useful performance metrics which can be used to evaluate the performance of workflows executed in the Grid. Only few metrics are supported in most existing tools, and most of them being limited at activity (task) level. Moreover, performance data of workflows needs to be shared among various other tools, such as workflow composer, scheduler, and optimizer. To support a wider utilization of performance information about Grid workflows, essential performance-related concepts and their properties in Grid workflows, together with associated performance data, must be well described.

2.2.2 Approaches and solutions

Our approach is that we characterize workflows into a hierarchical view and then study performance metrics associated with Grid workflows based on this view. We then present how different performance tools for different workflow systems can utilize studied performance metrics. Our contributions [98] are as follows:

- we introduce a common, hierarchical multiple levels of abstraction model for the performance analysis of Grid workflows.
- we present a large set of performance metrics that associates with relevant concepts within Grid workflows.
- we develop a novel ontology, named **WfPerfOnto**, for describing performance data associated with Grid workflows.

Figure 2.1 describes the hierarchical structure view of a workflow. Based on this model, we can examine performance metrics associated with workflow, workflow region, activity, invoked application and code region. The general techniques we develop are that: first we need to understand execution models associated with these levels of abstraction. Second, suitable metrics are identified and defined for specific levels. Furthermore, aggregation techniques are needed for composing metrics from different levels. All the detailed metrics

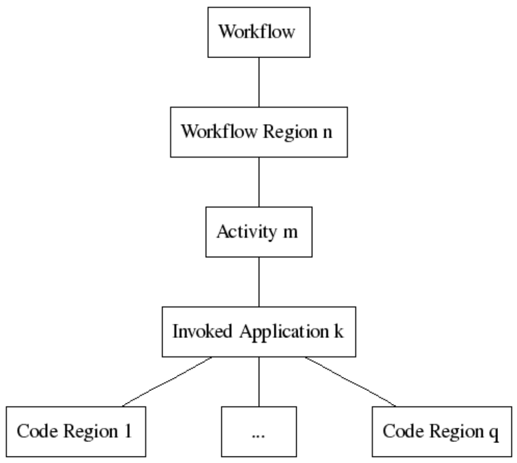

Fig. 2.1 Hierarchical structure view of a workflow.

are defined in an ontology namely **WfPerfOnto**, shown in Figure 2.2. The benefits of **WfPerfOnto** are twofold:

- *common understanding of performance results*: different performance tools can use different underlying techniques to measure metrics but using similar metric names and interpretation leading to a common understanding of performance behaviors.
- *supporting languages for specifying performance requests and metrics associated with workflows*: generic ways to specify performance requests for workflows at different levels can be developed. This will abstract the performance analysis from specific performance monitoring and workflow systems. We describe some concrete techniques in Section 2.3.

We have integrated our techniques into the ASKALON framework [41] and the EU FP6 K-WfGrid project [19] and validated our techniques with several real-world scientific workflows in the context of the Austrian Grid and K-Wf Grid projects.

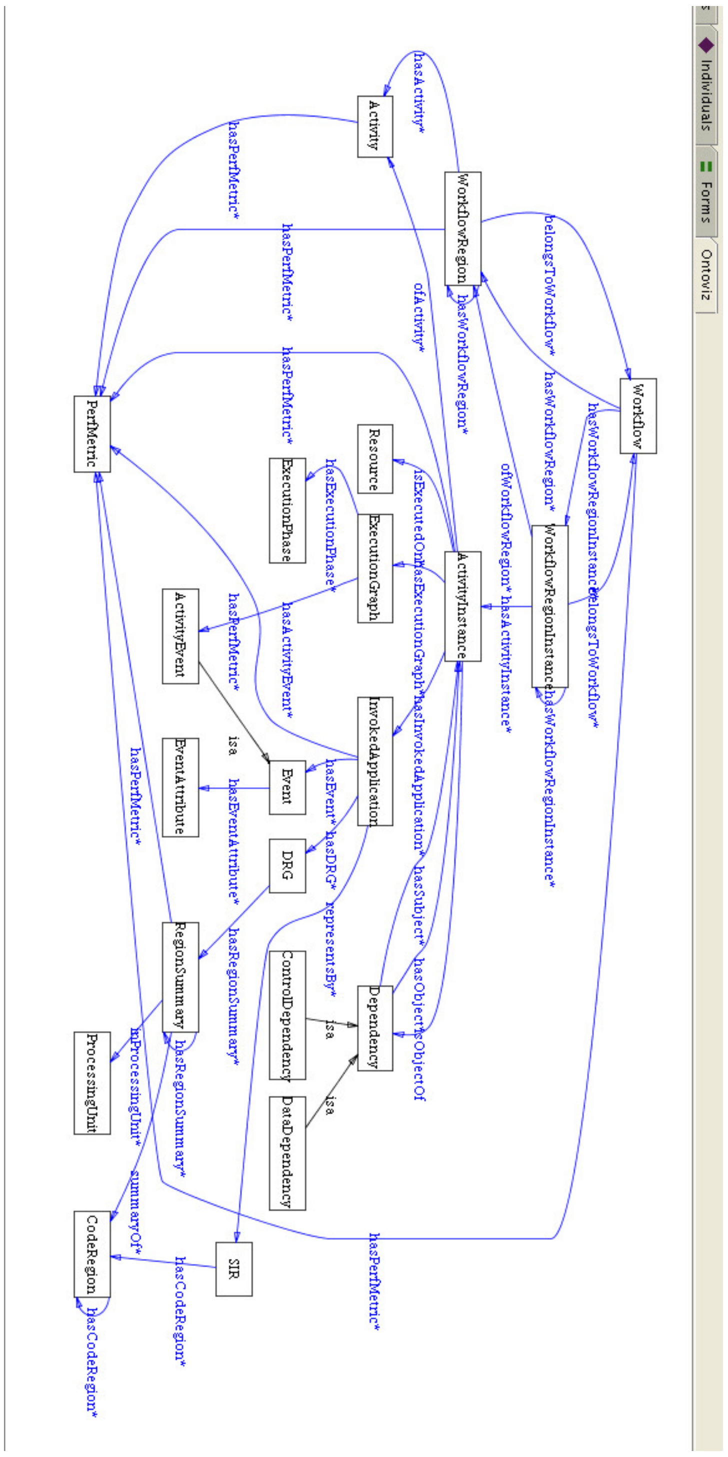

Fig. 2.2 Part of ontology for describing performance data of workflows

2.3 Distributed, Online Performance Monitoring and Analysis of Grid workflows

2.3.1 Research problems

Performance monitoring and overhead analysis of Grid workflows are required by not only the application developer - who will design, customize, develop and test workflows - but also the application user - who will execute workflows provided by the developer. Both types of users require the performance of the Grid infrastructure and the Grid workflows being monitored and evaluated. While the application user requires online monitoring of the workflow execution and online detection of performance problems, the application developer requires a full support of performance evaluation, including post-execution analysis features, in order to efficiently construct and execute workflows.

Apart from these types of users, Grid middleware services, such as the workflow execution service and the workflow scheduler, also require performance information about executed and running activities to dynamically control the execution of workflows. It is because the Grid offers a huge amount of heterogeneous computing resources, making the traditional code optimization cycle difficult. Instead, the performance optimization is shifted to Grid middleware services which aim at optimizing the execution of Grid workflows by means of the optimization of resource selections and execution control, and self-adaptive techniques [56, 83, 109, 51, 47, 50, 84]. To support this paradigm shift, we need performance analysis services which are capable of detecting performance problems of Grid workflows during runtime.

The above-mentioned objectives and user classes imply various requirements for the performance analysis support:

- *Supporting online and offline performance analysis in the same system.* Grid workflows take a long time to finish and the resources are prone to failures. Therefore, both the user and the Grid workflow middleware require runtime performance evaluation because they want to react as soon as possible to performance problems. Moreover, the application developer needs to examine in detail the performance of workflows as well as to capture performance information for further utilization. This requires the user to spend a certain amount of time to carefully examine all performance behavior. Such a required time may not be enough during runtime of a workflow.
- *Allowing specifying performance requests before and during the execution.* Since the client of the performance analysis service needs to react to performance problems immediately, the client of the performance analysis services must be able to specify performance analysis tasks, before and during runtime. For example, at a given time during the workflow execution, the user can select an activity which has not been started yet and specify analysis tasks for that activity. Based on that, the performance analy-

sis service can prepare necessary steps. When the activity is instantiated, performance analysis techniques can be applied to the activity instance immediately.
- *Providing highly interoperable interfaces for Grid workflow middleware to specify analysis requests and to obtain performance result.* Performance analysis services have to be loosely integrated with various services and different performance requests and information are exchanged between clients, middleware and performance services in the Grid environment. To make sure that Grid workflow middleware and clients can seamlessly be integrated and can interoperate with the performance analysis services, the analysis services must offer highly interoperable interfaces for other services and clients to utilize performance analysis features. Furthermore, all performance data as well as analysis requests should also be well defined, using open, standardized representations, such as XML.

2.3.2 Approaches and solutions

To address the above-mentioned requirements, our solutions include the following novelties:

- distributed on-the-fly monitoring at multiple levels of abstraction of Grid workflows, including the whole workflow, workflow region, workflow activity, invoked application and code region (based on the hierarchical workflow view in Section 2.2).
- online overhead analysis and search for performance problems for workflows based on a common, rich set of performance metrics for Grid workflows mentioned in Section 2.2.
- generic languages for querying performance data and specifying analysis requests.

Figure 2.3 depicts the architecture of our distributed performance analysis service (DIPAS) and relevant components involved in the performance analysis. DIPAS basically includes two parts: (i) DIPAS Portal, a portal for the end-user and developer, and DIPAS Client, any clients that need performance results, and (ii) DIPAS Gateway, the core service performing online overhead analysis and search for performance problems. DIPAS interacts with underlying workflow execution, monitoring and knowledge core services that control and monitor the execution of workflows.

DIPAS Gateway includes a component to query and subscribe online monitoring data - *Monitoring Data Query and Subscription* - and a component to analyze workflow representations - *Workflow Graph Analysis.* Based on these two components, a component named *Performance Overhead Analysis* will conduct the performance evaluation of workflows. This component determines performance overheads based on a well-defined classification of

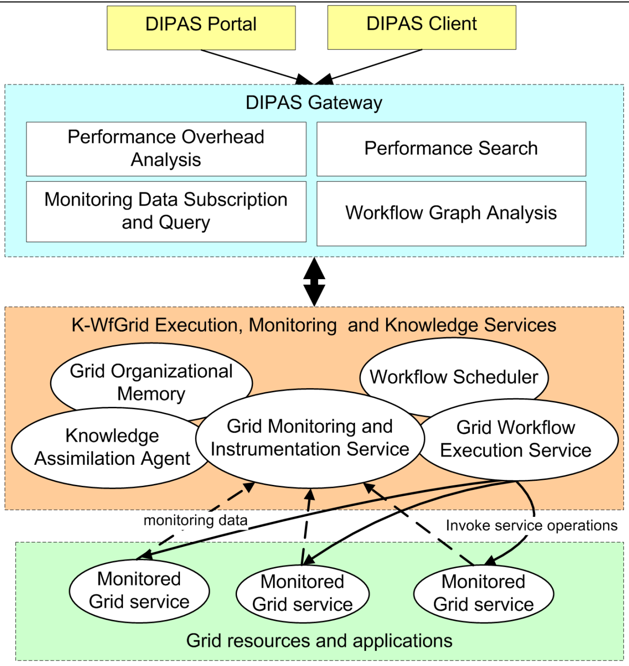
DIPAS Portal
DIPAS Client
DIPAS Gateway
Performance Overhead Analysis
Performance Search
Monitoring Data Subscription and Query
Workflow Graph Analysis
K-WfGrid Execution, Monitoring and Knowledge Services
Grid Organizational Memory
Workflow Scheduler
Knowledge Assimilation Agent
Grid Monitoring and Instrumentation Service
Grid Workflow Execution Service
monitoring data
Invoke service operations
Monitored Grid service
Monitored Grid service
Monitored Grid service
Grid resources and applications

Fig. 2.3 Overall architecture of the distributed performance analysis service (DIPAS). Arrowed black lines indicate the invocation of service operations (control and data flow). Arrowed dash lines indicate data flows.

workflow performance overheads [67]. The *Performance Search* component will perform the search and check performance problems based on performance conditions specified by clients. DIPAS also provides information for other Grid middleware services, such as the Workflow Scheduler - which uses the information to schedule the execution of activities - and the Knowledge Assimilation Agent (KAA) - which utilizes runtime and past performance information to estimate the performance of future Grid services and workflows.

DIPAS supports service interface for both the user (the application developer and user) and the Grid middleware to conduct the analysis. More importantly, the requests and responses of any performance analysis tasks are defined using a set of XML-based languages, including a generic Performance Data Query and Subscription (PDQS) language, and a Workflow Analysis Request Language(WARL). The result of performance analysis is returned in a pre-defined XML representation.

Several online performance online features, such as execution tracing, performance overhead analysis, and search for performance problems, at multiple levels of workflow abstraction, together with extensible performance analysis interfaces have enabled different middleware to utilize DIPAS. DIPAS is the online performance analysis service for Grid workflows in the EU FP6 K-

Wf Grid project [19]. We show the benefits of DIPAS via several real-world applications, such as Coordinated Traffic Management, Flood Forecasting Simulation, and Enterprise Resource Planning workflows, in the context of the K-WfGrid project.

2.4 QoS Dependency Monitoring and Analysis for Grid Services

As we have discussed Grid workflows before, we have observed that the execution of a Grid workflow relies on many Grid services which are in different layers but have dependencies. With the Grid, the user is able not only to harness a powerful pool of services but also to choose the best services suitable for their tasks from various similar services. In order to use these services efficiently, users must be able to select the best services for their tasks, based on quality aspects of resources. Not surprisingly, utilizing QoS parameters in the Grid has got a lot of attention as QoS parameters play a key role in selecting resources efficiently as well as in negotiating service level agreements (SLAs) between clients and services [66, 17, 18, 109, 8, 63]. QoS parameters are associated with various aspects, considered as what is normally called *non-functional* properties. In the context of our research, we focus on runtime dependency monitoring QoS metrics of Grid services.

2.4.1 Research problems

Most of existing works focus on specifying and modeling QoS of Grid services and workflows [32, 70, 24] or on using QoS in composition of services, service negotiation, resource management systems, service discoveries, and schedulers [73, 53, 63, 22, 23, 18, 77]. In the first case, various languages specifying QoS have been developed, and QoS models for workflows and composite Web services (WS) are proposed. Nevertheless, typically each framework presents only a subset of selected QoS attributes. In the latter, existing frameworks usually indicate how they use QoS, e.g. for managing resources and scheduling jobs. Among these works, many presents the conceptual framework validated through simulation, not real QoS monitoring systems, e.g. in [22]. Also there are some QoS ontologies [32, 70] for service-based applications. These ontologies describe QoS vocabularies and metrics and their relationship to resources, rather than present which QoS metrics are suitable and how to monitor them. Moreover, we observed little work on monitoring and analyzing QoS of Grid resources that consider the dependencies among these resources. In fact, we argued monitoring techniques for QoS attributes in the Grid have not been focused and there is a lack of QoS monitoring and analysis tools for the Grid.

We observed that while many QoS attributes are discussed for WS and Grid applications/workflows, existing techniques for measuring and monitoring QoS attributes are limited and inadequate. Using extra service proxies/wrappers and instrumented client and services to monitor QoS attributes are widely used for WS [80]. However, this method cannot be applied for different types of services developed in the Grid. [61] uses fault injection method to evaluate the dependability of services but it does not provide dependability metrics. Many frameworks use QoS attributes but are just based on simulation without the support of a real QoS monitoring tool.

2.4.2 Approaches and solutions

To address the lack of frameworks for monitoring and analyzing QoS attributes of Grid services, our goal is to deal with three important issues named monitoring, analysis and management of QoS attributes. Our solutions [104] are based on the following key points:

- First, we develop a QoS classification for Grid resources that combines existing QoS metrics with new Grid-based QoS metrics. Since Grid services are diverse, any single method cannot be used for monitoring various types of Grid services. Therefore, we have to utilize different measurement methods for different services and unify these methods into a single framework. To this end, we classify types of monitored resources, as services, into machine, network path, middleware and application. Machines are places on which middleware and applications are deployed and executed. We focus on middleware/applications built based on SOA, in particular based on Web services technologies. The monitoring and analysis of computational services and networks are necessary in order to establish the dependency among middleware/applications. We do not employ a single method to collect monitoring data but apply different methods for different types of services, for example, QoS of machines and network paths provided by well-known system and network tools, and instrumentation and black-box monitoring can be used for application and middleware. Table 2.1 gives examples of monitored services and measurement methods.
- Second, the Grid introduces complex interactions among various services. As a result, we develop monitoring and analysis techniques that consider the dependencies among Grid services and to support the monitoring and analysis of QoS based on these dependencies. We model dependent Grid services as a directed graph in which a node n represents a Grid service whereas an edge $e(n_i, n_j)$ means that n_j is dependent on n_i (n_i can cause some effect on n_j). A node n is associated with QoS metrics and *status of service* including *functional* and *operational* status information. An edge $e(n_i, n_j)$ representing the dependency between two services. Dependencies can be (i) *functional relationship* (affect only the functional status of a

Monitored Resources	*Measurement Method*	*Metrics*
machine	using ping	availability
network path	using TCP connection, ping	availability, reliability
middleware	using GRAM, GridFTP, log files	availability, reliability
application	WS/WSRF/WSDM interfaces, SOAP message, instrumentation	availability, reliability, manageability, performance

Table 2.1 Example of monitored services and measurement methods

service) or *operational relationship* (affect the operational status of a service) and (ii) *casual relationship* (a change of a service affects another) or *mutually-exclusive relationship* (a change of a service might not affect another). The graph of Grid services is used in monitoring service status and in determining QoS parameters during runtime. During runtime, monitoring data is updated. Subsequently, a status of a service can change. A change in status of a service can cause the change of other services which depend on the service.

With our solution, during runtime, QoS parameters of Grid services are determined based on monitoring data of these services. Moreover, the computation of QoS parameters is dependent on *client local view* or *system global view*. Given a service and the same request for QoS parameters, we can have different results associated with the service due to the fact that requesters are disparate in the Grid and their views to the service are different. For example, for a client the availability of a Web services may be 90%, but for another client the service availability may be only 50% due to the network differences. This enables a user to have different views when the user encounters a QoS problem.

We have validated our techniques and demonstrated the usefulness of our QoS dependency monitoring and analysis in the EU FP6 K-WfGrid project and the AustriaGrid using several Grid services from distributed locations.

2.5 Monitoring and Analyzing Collaboration Processes

While our previous studied Grid workflows consist of only software-based activities and do not include interactive activities, in e-science and business environments there exist collaboration processes (also called collaborative workflows) that consist of software-based activities and human interactions. In these processes, people belonging to different organizations collaboratively work together to achieve a common goal. In this context, they establish a *team*, often a *virtual team* [59], and conduct a *collaborative process* implementing their common goal. The current trend is to rely on SOA (Service-Oriented Architecture) to implement tools and services for virtual teams and their collaborative processes because SOA offers many technologies to simplify the integration and interoperability of services belonging to different organizations and to allow the user to easily access existing services and tools. Examples of such SOA-based collaboration tools and services are the inContext [96], ECOSPACE [39] and COIN [26] systems. Given these systems, one important aspect is to understand how *people and software services interact* in order to adapt activities of people and software services to the change of their operating environments as well as to allow them to self-manage their behaviors during their collaboration. Hence, metrics and patterns associated with interactions are a valuable source of information.

2.5.1 Research problems

While existing research has been focused on defining and detecting patterns in workflows [13, 6, 110, 7, 48, 35, 36], support for runtime analysis of patterns in dynamic collaboration environments has got little attention. In complex, dynamic collaboration environments, ad-hoc collaboration processes are not pre-defined; their dynamic, ad-hoc activities are defined on-demand. These activities may be combined with well-defined workflows and composition patterns, but not necessarily. In such environments, metrics and patterns characterizing the collaboration and its activities are relevant because they can provide valuable insights into the collaborative process to support runtime adaptation. However, existing offline mining techniques are not suitable because they are not designed (and cannot be tested) with evolving collaborative processes. Existing mining techniques typically require complete log data and do not deal with runtime aspects, such as runtime processing data from various services and runtime provisioning of interaction metrics and patterns. Furthermore, most existing work focus on either human-to-human interactions (e.g., social networks analysis) or service-to-service interactions (e.g., performance analysis or service interaction pattern analysis), whereas SOA-based collaboration environments include diverse types of interactions among humans and services that should be considered together.

Runtime analysis of interactions in such environments poses many research challenges. First, data is obtained and analyzed while the collaborative process just continues to evolve as new activities emerge. This requires us to deal with different types of events collected at different levels, such as activities, interactions and service-specific events. Secondly, metrics and patterns have to be determined for both humans and software services according to different needs of clients, such as determining metrics and patterns based on collaboration contexts, e.g., in individual, group or the whole collaboration levels, and on user-specific conditions, e.g., in particular time period and with a specific threshold. All these challenges imply that runtime analysis frameworks must be flexible and customized: new metrics and patterns analyses can be easily added and analysis requests are user-customized.

2.5.2 Approaches and solutions

Our solutions to online interaction analysis for collaborative processes in SOA-based environments are (1) to develop a classification of interaction metrics and patterns covering human-to-human, human-to-service, and service-to-service interactions at different levels, and (2) to introduce a flexible and customizable software framework supporting runtime interaction analysis. Figure 2.4 illustrates our approach. The upper part of Figure 2.4 illustrates the dynamic collaboration environment. In such an environment, both humans and software services exist. Humans use services to perform their activities, while services can interact with each other to fulfill requests from humans. Services will follow the SOA model, thus they can be easily integrated together, providing seamless access virtually from anywhere. The SOA-based service model also simplifies the monitoring and maintenance of services, making the acquisition of multiple sources of log information at different levels in the widely distributed system possible and easier.

Using services, people perform their collaboration, of which processes are typically not modeled beforehand. Furthermore, the execution of collaborative processes is continuous and evolving, thus in many cases we will not know in advance when a process finishes. Interactions between humans and services are highly concurrent and distributed as multiple activities are executed in parallel. These activities involve services and people spanning different geographical locations and organizations. Therefore, offline analysis, which either requires complete, centralized information or the completion of the process in order to analyze the process, is not suitable. In addition, the need for adapting activities of and resources for collaboration is required at runtime, due to highly dynamics of modern collaborations (e.g., team member is often on the move). Thus, online analysis techniques are more suitable. The middle part of Figure 2.4 shows that the *Online Interaction Analysis* analyzes log events obtained from services and provides metrics and patterns back to the services in the dynamic collaboration environment.

Our online interaction analysis for such an environment aims to answer the following challenging questions:

- Which metrics and patterns associated with interactions are useful for optimizing collaborative work and resources used for the work at runtime?
- How can we correlate metrics and patterns from different levels of collaboration context, such as individual, group and the whole collaboration?
- How do we support the customization of metrics and patterns analysis so that different clients can utilize the metrics and patterns differently?
- How to handle diverse events from various services?

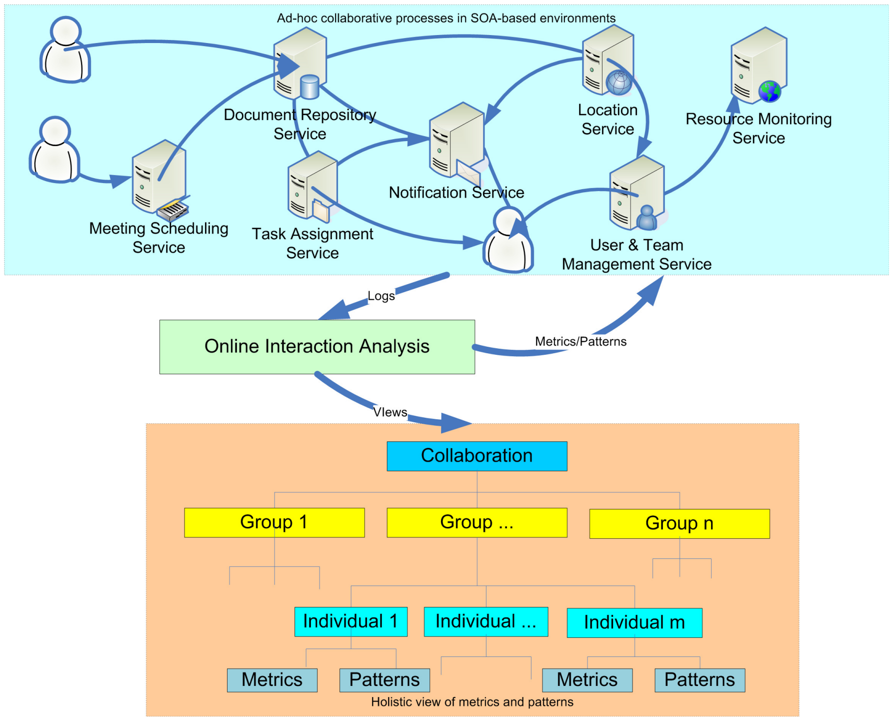
Ad-hoc collaborative processes in SOA-based environments
Document Repository Service
Location Service
Resource Monitoring Service
Notification Service
Meeting Scheduling Service
Task Assignment Service
User & Team Management Service
Logs
Online Interaction Analysis
Metrics/Patterns
Views
Collaboration
Group 1
Group ...
Group n
Individual 1
Individual ...
Individual m
Metrics
Patterns
Metrics
Patterns
Holistic view of metrics and patterns

Fig. 2.4 Online interaction analysis environment and metrics/patterns view

- How to manage and provide these metrics and patterns to concurrent clients at runtime?

Our solutions for these questions include:

- a holistic view of patterns and metrics associated with services and interactions at multiple levels, ranging from the *individual* (services or human), to the *group* (of humans or services or the mix of them), and to the *whole collaboration*. The bottom part of Figure 2.4 shows the holistic view of interaction metrics and patterns that are associated with interactions.
- complex event processing rules for determining metrics and patterns based on different levels, client requests and periods of time
- management and provisioning of XML-based metrics and patterns at runtime.

We show that with capabilities of performing runtime analysis of interaction patterns and metrics spanning different levels, including individual, group, and collaboration views, and that characterizes different types of interactions, including service-to-service, human-to-service, and human-to-human, our system is flexible and customizable, allowing for the inclusion of new analysis and supports client-customized mining. Useful interaction metrics can be utilized for runtime adaptation in collaborative working environments. Our interaction analysis is conducted within the FP6 EU inContext project and is validated using the FP6 EU inContext testbed.

2.6 Composable Cost for Cloud-based Applications

2.6.1 Research problems

Cloud computing promises to eliminate obstacles due to the management of IT resources and to reduce the cost on infrastructure investments. As a result, besides business customers, cloud computing is also attractive to many scientists from small research groups in the computational science and engineering (CSE) field. However, still there are many unclear questions about how cloud computing can help CSE scientists [20, 11]. Among them, the need to determine the actual cost when using cloud computing is evident. In general, given the offer from the cloud computing, scientists want configure the execution of their applications in different ways: local (e.g., for testing), partial cloud (when dealing with sensitive information) and fully cloud/cluster (e.g., for performance test). The requirement is that depending on the cost, performance and urgency, they could use full resources from the cloud or partial resources from the cloud and their own resources (e.g., clusters from the university). Using information provided by the vendor, it is very difficult to calculate the cost of an application because scientists need to map the computation and data transfer requirements associated with their CSE applications to primitive prices of CPUs, storage and network transfer. Scientists expect to have cost models associated with application models, e.g., OpenMP, MPI, and workflows. Furthermore, a scientific experiment might include different parts, each may have a different application model and might or might not be executed in the cloud.

2.6.2 Approaches and solutions

Cost estimation, monitoring and analysis for different applications are needed. But we should note that before utilizing cloud resources, scientists already have vast knowledge about their applications. Therefore, our intuition for developing cost models is that, before deciding to move to the cloud, scientists use their known performance characteristics of their applications, e.g., transferred data and execution times, to provide many application-specific parameters that can be used for estimating costs. Our approach and solutions [93] can be summarized as follows:

- develop several basic cost models for different application execution models, such as OpenMP, MPI and workflows. This can be achieved by also leveraging previous knowledge in performance estimation, monitoring and analysis, e.g., [88, 45, 55]. As an example, Table 2.2 describes basic cost models. The first four models, M_{ds}, M_{cm}, M_{dfi} and M_{dfo} are basic models that are provided by cloud providers in their pricing specifications. By using these models and well-established execution models of sequential, parallel and workflow applications, corresponding cost models are introduced.
- develop runtime cost estimation, monitoring and analysis tools by utilizing (i) online performance monitoring data from cloud providers and from application monitoring, and (ii) cost models. It is important to note that cost estimation, monitoring and analysis algorithms are composable: each application can have different parts whose costs are evaluated differently.
- provide cost analysis by presenting near-realtime costs and the difference between application view and provider view (detecting cost overhead).

By applying our techniques to real-world applications, such as Bone Structure Simulation, Next Generation Sequencing Analysis, and Remote Sensing, on private clouds and Amazon EC2, we have shown that with our solutions, we can detect cost elements that the user may not be fully aware of in advance or may pay due to the lack of attention (e.g., the output of print commands). Furthermore, it shows that it is not trivial to determine some types of costs due to the complex dependency among services in the cloud (e.g., the case of cloud instances and storage mounted in the local file system in the instances) or due to the overhead for determining the costs (e.g., monitoring all I/O calls in an application could introduce severe performance overhead). Our composable cost model work is conducted in the context of the WWTF SODI project and we are extending it for the new FP7 EU project CELAR.

2.7 Quality of Data aware Workflow Optimization

2.7.1 Research problems

Recently we focus on multi-scale workflows which combines (i) sub-workflows which are implemented with different underlying fundamental computational models, although these models can be in the same domain, and (ii) sub-workflows from different domains. With the first type of the workflow, we see that data resulted from a scale-specific sub-workflow will be the input for another scale-specific sub-workflow, and each scale-specific sub-workflow needs a different resource provisioning strategy. However, when data obtained in a scale-specific level is not met the requirement of the next scale-specific level, several problems can arise: workflow activities in the next scale-specific

Model	Activities	Cost
M_{ds}	Data storage	$size(total) \times t_{sub} \times cost(storage)$ where t_{sub} is the subscription time
M_{cm}	Computational machine	$cost(machine)$
M_{dfi}	Data transfer into the cloud	$cost(transfer_{in})$
M_{dfo}	Data transfer out to the cloud	$cost(transfer_{out})$
M_{sd}	Single data transfer without the cost for machines performing the transfer	$size(in) \times M_{dfi} + size(out) \times M_{dfo}$
M_{sm}	Sequential/multi-threaded program or single data transfer with the cost for machines performing the transfer (cost monitoring)	$t_e \times M_{cm} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$
M_{se}	Sequential or multi-threaded program (cost estimation)	$f_{pi} \times M_{cm} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$ where f_{pi} is an estimated performance improvement function when n expected threads to be used. f_{pi} can be provided by performance prediction tools or scientists. In our case, currently, we use an ideal parallel performance improvement $f_{pi} = \frac{p}{n} \times t_e(p)$ where p is the number of threads used to obtain $t_e(p)$. p and $t_e(p)$ are known knowledge.
M_{pm}	Parallel/MPI programs on multiple machines (cost monitoring)	$n \times M_{cm} \times t_e + size(out) \times M_{dfo} + size(in) \times M_{dfi}$
M_{pe}	Parallel/MPI programs on multiple machines (cost estimation)	$n \times M_{cm} \times f_{pi} + size(out) \times M_{dfo} + size(in) \times M_{dfi}$ where f_{pi} is an estimated performance improvement function when n processes are used.
M_{wm}	Workflows (cost monitoring)	$\sum_{i=1}^{k} (size(in_i) \times M_{dfi}) + \sum_{i=1}^{l} (size(out_i) \times M_{dfo}) + \sum_{i=1}^{n} (M_{cm} \times t_e(machine_i))$
M_{we}	Workflows (cost estimation)	$\sum_{i=1}^{nwr} cost(wr_i)$. For a workflow region wr_i, $cost(wr_i) = \sum_{j=1}^{q}(cost(activity_j))$ where $cost(activity_j)$ is determined based on M_{mp}, M_{sm}, and M_{sd}, when the activity $activity_j$ is a parallel activity, sequential activity, or a data transfer activity, respectively.

Table 2.2 Costs. t_e and $t_e(p)$ are the total elapsed time for executing computational task or data transfer and the execution time obtained with p parallel processes/threads, respectively. n is the number of machines used or to be used. $size(in)$ and $size(out)$ are the size of the data transferred into and out to a cloud.

level can yield useless results after consuming a lot of resources or can fail when handling inputs with unexpected quality. Domain scientists currently have no solution to specify, control and measure the influence of QoD on the execution of different scale-specific sub-workflows in an integrated manner. A common solution is to break sub-workflows into different separate workflows and the scientists manually examine inputs and outputs for individual scale-specific workflows. But this is a tedious effort as it does not support the automation of simulations and data processes and does not speedup the performance of scientist work. Fortunately, in this case, still domain scientists have knowledge about possible QoD metrics and how to evaluate them (e.g., using (semi)automatic tools or human activities). But such metrics and evaluation methods must be modeled and integrated into the workflow execution.

The second type of workflows introduces further challenges. Data resulted from a domain-specific workflow will be used as input in another domain-specific workflow. In this case, data and quality metrics of a domain are not easy to be understood and mapped to another domains. Data mapping techniques are well supported in workflows but scientists in one domain do not understand QoD in another domain well as they lack knowledge about other domains. In current solutions, typically, scientists from many domains will manually examine the QoD of each domain-specific workflows. When they want to perform multi-scale workflows integrating several domain-specific parts, they do not have a solution to make sure that QoD can properly be checked. Unlike in the first type of multi-scale workflows, this type of multi-domain workflow needs to capture QoD metrics from different domains.

In general, in multi-scale workflows, several data models and data sources are used and they are complex not only in terms of space (volume) but also in terms of the QoD. As a workflow-specific part relies heavily on certain input data for its execution and produces resulting data for other workflow-specific parts, if the quality of the input is not guaranteed at a certain degree, the output data might not be useful, as well as if the quality of the output data is not guaranteed, next workflow activities should not be started or further activities and service resources should be allocated to handle low quality input data.

Unfortunately, although several scientific workflows have been developed and workflows for e-science have been discussed intensively, such as in [86], QoD for workflows has not been well-supported in current use of workflows. We believe this is very important for multi-scale workflows due to the complexity and scale of data, computational models and processing algorithms. On the one hand, while current (scientific) workflows focus on handling data and resources, as well as performance monitoring, we lack techniques to support the model and evaluation of QoD associated with the data movement in workflows and to adapt and optimize the composition and execution of and resource provisioning for workflows (e.g., change data sources, refine workflow structures, and decide whether to trigger the execution of another workflow-specific part) based on QoD metrics. On the other hand, today's many workflows rely on different data sources provided by different data providers that have a multitude of quality differences. While some effort have been spent for supporting data compliance in workflows, such as data privacy [46], most existing workflow techniques have not provided QoD evaluation and optimization.

2.7.2 Approaches and solutions

2.7.2.1 Approaches

Domain-specific scientists also know tools and mechanisms to determine QoD for certain types of data. However, they lack supporting tools and techniques in order to examine QoD in their workflows in an automated and integrated manner. Furthermore, scientists expect tools that can automatically adapt and optimize their workflows based on the specified quality. Therefore, we need to address the following research questions:

- What are main QoD metrics, what are the relationship between QoD metrics and other service level objectives, and what are their roles and possible trade-offs?
- How to support different domain-specific QoD models and link them to workflow structures?
- How to model, evaluate and estimate QoD associated with data movement into, within, and out to workflows? When and where software or scientists can perform automatic or manual QoD measurement and analysis
- How to optimize the workflow composition and execution based on QoD specification?
- How does QoD impact on the provisioning of data services, computational services and supporting services?

In order to answer these questions, we propose to research and develop novel models, techniques and algorithms:

- *Core models, techniques and algorithms to allow the modeling and evaluating QoD metrics for multi-scale workflows*: Our idea is that QoD metrics should be modeled, specified, estimated and evaluated in different levels of abstraction of workflows, including the workflow activity, workflow region, sub-workflow, and the whole workflow. At the workflow activity level, the user must be able to specify and establish constraints based on QoD for the input data and output data of workflow activities and the workflow system must be able to support the selection and assurance of these constraints by monitoring, estimating and evaluating QoD metrics and by adapting the workflow activities.
- *QoD-aware composition and execution*: After having the core layer, we can add QoD-aware flexibility and adaptation features into the workflow composition and execution. This includes techniques using QoD requirements and evaluated QoD metrics to steer the execution of the workflow and to (semi)automatically refine the workflow composition.
- *QoD-aware service provisioning and infrastructure optimization*: Another layer is to adapt service provisioning and infrastructure based on the QoD. Dependent on the quality associated with some data flows, different strategies for resource provisioning can be employed. One example is due to the

low quality of the data produced by the first domain-specific sub-workflow, it is expected that the second domain-specific sub-workflow will need more data storage, machines, and middleware services because some of activities in the second sub-workflow will perform more computation and data handling. Thus, elastic resource provisioning is needed during runtime.

2.7.2.2 Solutions

As the first step toward quality of data -aware workflow optimization, we concentrate on modeling and evaluating QoD metrics for workflows. Figure 2.5 describes a general view for evaluating QoD metrics at the activity level. Similar to the performance metric analysis, but we need to capture data exchange among activities and using QoDEvaluation tool. This general model can also be applied at the invoked application, in which data is intercepted at specific points in the invoked application (e.g., using instrumentation techniques) and evaluated on-the-fly.

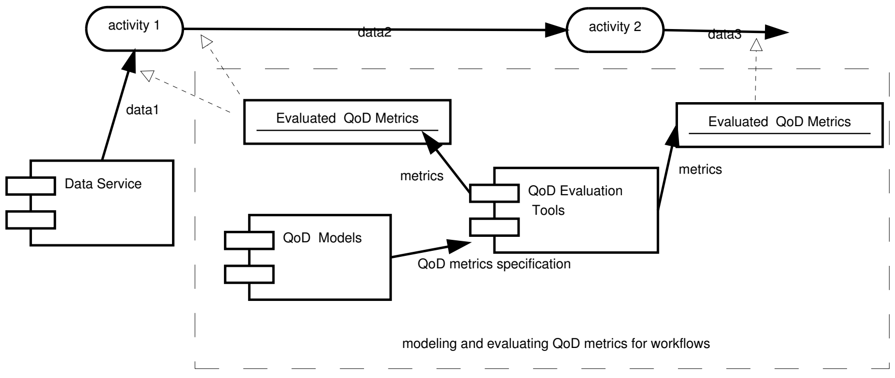

Fig. 2.5 Modeling and evaluating QoD metrics for workflows

Our initial solutions are described in [78, 79] and can be summarized in the following points:

- Instrument workflows at multiple level to capture data to be evaluated. The places where data to be captured can be associated with workflow region, workflow activity, invoked applications and code regions.
- The QoD can be evaluated by different QoD Evaluators, which can be software or humans. The rationale is that in many cases only humans can determine the QoD of simulation data.

Our solutions have been applied to real-world simulation workflows in the context of SimTech[1]and we have demonstrated how measuring and utilizing QoD could save time and effort.

2.8 Publications and Notes

The following publications describe the research challenges and our contribution in this chapter:

- **Hong-Linh Truong, Robert Samborski, Thomas Fahringer: Towards a Framework for Monitoring and Analyzing QoS Metrics of Grid Services. e-Science 2006: 65** [104]
- **Hong-Linh Truong, Schahram Dustdar, Thomas Fahringer: Performance metrics and ontologies for Grid workflows. Future Generation Comp. Syst. 23(6): 760-772 (2007)** [98]
- **Hong-Linh Truong, Peter Brunner, Vlad Nae, Thomas Fahringer: DIPAS: A distributed performance analysis service for grid service-based workflows. Future Generation Comp. Syst. 25(4): 385-398 (2009)** [88]
- **Hong-Linh Truong, Schahram Dustdar: Online Interaction Analysis Framework for Ad-Hoc Collaborative Processes in SOA-Based Environments. T. Petri Nets and Other Models of Concurrency 2: 260-277 (2009)** [92]
- **Hong-Linh Truong, Schahram Dustdar: Composable cost estimation and monitoring for computational applications in cloud computing environments. Procedia CS 1(1): 2175-2184 (2010)** [93].
- Michael Reiter, Uwe Breitenbucher, Schahram Dustdar, Dimka Karastoyanova, Frank Leymann, Hong-Linh Truong: A Novel Framework for Monitoring and Analyzing Quality of Data in Simulation Workflows. eScience 2011: 105-112 [78]
- Michael Reiter, Hong-Linh Truong, Schahram Dustdar, Dimka Karastoyanova, Robert Krause, Frank Leymann, Dieter Pahr, On Analyzing Quality of Data Influences on Performance of Finite Elements driven Computational Simulations, 18th International European Conference on Parallel and Distributed Computing (EuroPar 2012), 27-31 August, 2012, Rhodes Island, Greece. [79]

The original ideas of research topics in this chapter have been developed by the author and the ideas have been elaborated and implemented together with several colleagues and students. The work mentioned in Section 2.2 is a joint work with Schahram Dustdar and Thomas Fahringer, published in [98] and

[1] Stuttgart Research Center for Simulation Technology `http://www.simtech.uni-stuttgart.de/`

included in Appendix A. The work mentioned in Section 2.3 was conducted with Peter Brunner, Vlad Nae and Thomas Fahringer, published in [88] and included in the Appendix 2.3. Peter Brunner was a master research student working under the author supervisor while Vlad Nae was a PhD student who contributes to the prototype implementation. The work mentioned in Section 2.6 and in Section 2.5 were conducted together with Schahram Dustdar and are included in Appendix A. The work mentioned in Section 2.4 is carried out together with Robert Samborski in the context of his master research under the supervision of the author. The work mentioned in Section 2.7 is a joint work with Michael Reiter, Uwe Breitenbuecher, Schahram Dustdar, Dimka Karastoyanova, Robert Krause, Frank Leymann, and Dieter Pahr, published in [78, 79]. The work has mainly been performed via several internships of Michael Reiter carried out with the author.

Chapter 3
Analyzing, Evaluating and Provisioning Data Concerns for Data-as-a-Service

3.1 Introduction

Taking advantage of Web services technologies, the software as a service model [15] and cloud computing [20], recently, various research efforts have concentrated on the development of the concept of data as a service (DaaS) [28]. Whether a service is a DaaS can depend on specific context, for example, a DaaS can simply allow consumers to create, store and manage their own data according to their specific data models or can provide data assets needed by consumers. However, DaaS have a common property: they mainly provide data capabilities based on common data CRUD (Create, Read, Update, Delete) commands rather than computation on data and they allow their consumers to acquire or provide data under the service model, regardless of whether the offerings are free or commercial. Over the last few years, various providers have provided (and claimed) services as DaaS. Yet still from an outlook of a consumer, it is difficult to distinguish a DaaS from other types of services. It is partially due to the fact that currently there is a lack of well-defined and -understood description models that are able to characterize concerns for DaaS. Most of today's DaaS in the Web just provide WSDL- or REST-based interfaces describing their operations and static Web pages about pricing and usage permission. DaaS are still described in terms of typical QoS, but not of specific concerns related to the data provided by DaaS, while data is the main ingredient that makes DaaS different. This problem has hindered the consumer from the selection and utilization of DaaS due to the lack of knowledge about offered data.

In our work, we focus on relationships among data consumers, DaaS providers and data providers, shown in Figure 3.1. In our model, one DaaS provider can interact with multiple data providers. A data provider can pass data to a DaaS on-demand, based on requests from the DaaS provider. A data consumer can use a pay-as-you-go model to request data from a DaaS, which, in turn, searches its own data assets or different data providers, if they can

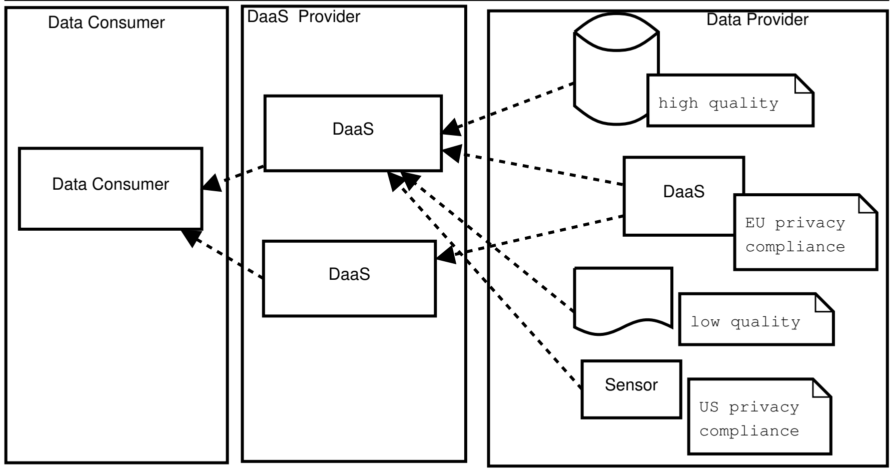

Fig. 3.1 Interactions in the DaaS model

provide some types of data, in order to fulfill the request. The data providers can set different constraints on the data assets they provide. Therefore, the *DaaS provider* is not necessarily the same as the *data provider*.

We found that the DaaS model is very different from conventional (software) services as it can be characterized into three different levels: (i) data service as a whole, (ii) DaaS APIs and (iii) data asset. As a result, quality concerns at each level must be analyzed. However, contemporary research focus very much on service level (e.g., in service [3, 81]) or data assets (such as quality of data from database perspectives [64, 68, 75, 14]) but not DaaS. Furthermore, several concerns due to the intersection between services and data in DaaS have not been analyzed. This chapter we describe research challenges, our approaches and solutions for analysis, specification, evaluation and publishing of data concerns associated with DaaS.

3.2 Analyzing and Specifying Concerns for Data as a Service

3.2.1 Research problems

We argue that a DaaS should be described and published in a way that it is able to highlight distinguishable characteristics of the data it provides, for example, whether a DaaS supports fundamental requirements for data governance, which meta-data is associated with the data a DaaS provides, whether the data can be used freely for commercial purpose, to name just a few. A DaaS must, therefore, be characterized by not only traditional QoS but also quality of data (QoD) and other data-specific concerns.

While QoD has been extensively studied in database research, how to associate QoD with DaaS is not defined yet, let alone the combination of QoS and QoD for DaaS. We further argue that characterizing QoS and QoD is not enough and we also need to address other concerns such as data usage, service context, and data source concerns as well as the license issue associated with DaaS. Among these issues, only QoS has been extensively studied for services that can be utilized for DaaS. The issues of QoD, data service licensing and other concerns, and their combination for DaaS remain open. A systematic approach to the description of QoD/QoS and service/data license for DaaS is missing.

3.2.2 Approaches and solutions

The use of a DaaS is bound to various concerns. Some concerns are technical specific to the data and the service, for example, QoD and QoS. However, there are also many other concerns related to business, regulatory and compliance aspects, such as pricing, copyright, and law enforcement. All of these concerns are critical for searching, comparing and selecting DaaS. We approach this problem by identifying important concerns that should be explicitly considered when publishing a DaaS, and by proposing a model specifying these concerns.

Concerns	Read-only DaaS	CRUD Daas
QoD	Important factor for the selection of DaaS. For example, the accuracy and completeness of the data, whether the data is up-to-date	Expected some support to control the quality of the data in case the data is offered to other consumers
Data source	Important factor for the trustworthiness of the DaaS	
Data & Service Usage	Important factor, in particular, price, data and service APIs licensing, law enforcement, and IPRs	Important factor, in particular, price, service APIs licensing, and law enforcement
Data Governance		Important factor, for example, the security and privacy compliance, data distribution, and auditing
QoS	Important factor, in particular availability and response time	Important factor, in particular, availability, response time, dependability, and security
Service Context	Useful factor, such as classification and service type (REST, SOAP), location	Important factor, e.g. location (for regulation compliance) and versioning

Table 3.1 Important DaaS concerns for consumers

Table 3.1 identifies important DaaS concerns associated with (1) Read-only DaaS which only provides data based on existing data sources, and (2) CRUD DaaS which allows the consumer to create, retrieve, update and delete data. In the second category, DaaS can be infrastructure-based in which services typically just provide a storage capability and it is up to consumers to define their own data schema and/or to publish their data.

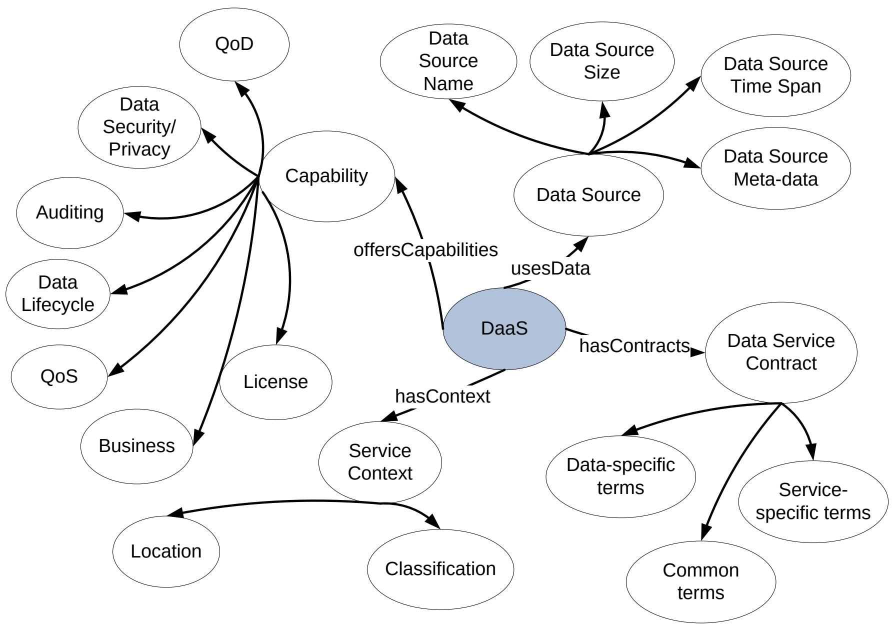

Fig. 3.2 DaaS's concerns and contracts

Based on our identification, we develop a specification for publishing information characterizing DaaS. This results in a well-structured data concern model that can be managed and associated with DaaS. This goes beyond existing works which (i) tend to consider DaaS as normal services whose publishing information is based on service interfaces (described by WSDL and REST API/WADL) and QoS only and (ii) neglect the data aspect which is the core of DaaS. Figure 3.2 depicts main concerns associated with DaaS that we divide into four main categories: capability, service context, data source, and data service contract. The first three categories represent DaaS properties and the last category, built upon the first three concerns, represents conditions established on properties under different circumstances.

To validate our techniques, we have analyzed several real-world DaaS for enterprises and e-science. The work mentioned in this section is one of the main contributions of the WWTF SODI project[1] that provides foundations for several other works in this project.

3.3 Evaluating and Publishing Data Concerns for Data as a Service

3.3.1 Research problems

DaaS providers and data providers need to evaluate and provide data concerns for consumers to understand constraints applied to the data. Without explicit information about data concerns, data consumers also face information overloading as irrelevant results can be obtained. While several tools exist to support the development and publishing of data under the DaaS model, currently there is the lack of techniques and tools for the evaluation and publishing of data concerns for DaaS.

3.3.2 Approaches and solutions

Our approach is to develop a methodology and corresponding tools that facilitate the integration of the evaluation of data concerns into the publishing of data concerns in the DaaS model. As data concerns will change as long as the data changes or new knowledge about the data is obtained, such data concerns have to be also evaluated, managed and published on-the-fly. We address the above-mentioned issues by contributing (i) a novel, generic data concern-aware service engineering process for DaaS and (ii) a framework for evaluating and publishing quality of data metrics for DaaS, as an implementation of our proposed process. Our process and our framework cover different evaluation scopes, modes and integration models for data concerns.

In order to ensure that data concerns associated with provided data to be available and searchable to the DaaS consumer, the DaaS and data providers have to utilize several components and to conduct several activities. Figure 3.3 shows main components and activities that we have identified for supporting data concern-aware service engineering. Typically, in order to expose data to DaaS, the data provider has to perform the *Wrapping Data* activity, which defines service operations for accessing and managing data assets, and the *Publishing Interfaces* activity, which publishes service operations and exposed data assets into service registries. These two activities are supported by a majority of tools for engineering DaaS. The exposed data is then accessed by *Data Consumers* via the *Selecting Data* activity.

[1] `http://www.dbai.tuwien.ac.at/proj/InfInt/`

However, in order to provide also data concerns associated with data assets, other activities are needed: the *Evaluating Data Concerns* activity is used to determine data concerns while the *Describing Data Concerns* activity will utilize evaluated concerns and configurations to determine data concerns to be published. Then, data concerns can be associated with service interfaces or data returned to data consumers in the *Selecting Data*. Furthermore, when data is updated, in *Updating Data*, data concerns can be evaluated.

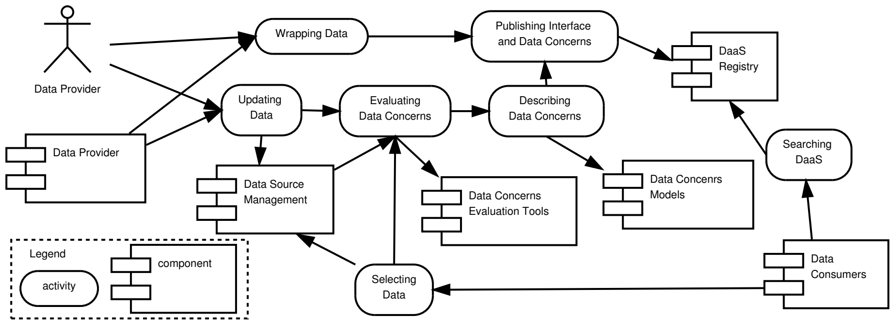

Fig. 3.3 Activities in the data concern-aware service engineering process. Arrow lines indicate control flows.

The goal of the *Evaluating Data Concerns* activity aims at providing extra information about concerns for such data assets by evaluating data concerns based on the movement of data resources from DaaS to data consumers. Certainly, some data concerns can be determined before the data is exposed through DaaS or before the data is accessed via DaaS operations, such as when the data is produced or is updated via the *Updating Data*. In our approach we evaluate data concerns during the *Selecting Data* activity and consider pre-defined data concerns as inputs for tool-specific evaluations in our process. In our process, three important aspects in evaluating data concerns are (i) the *scope* of the evaluation, (ii) the *mode* of the evaluation, and (iii) the *integration model* of evaluation tools with data answering.

Evaluation scopes: Data concerns can be determined and evaluated based on three scopes namely *data assets* (data concerns representing individual data resource(s)), *service operation* (data concerns representing all data provided by specific service operations), and *the service as a whole* (data concerns representing the data service as a whole. In all scopes, data concerns can be determined by tool-specific implementations and composition rules.

Evaluation modes: Two modes for the evaluation of data concerns are *off-line* and *on-the-fly*. In the off-line mode, data concerns can be determined before the data is accessed. In the on-the-fly, data concerns are evaluated for data movement requested through service operations. In some cases, data concerns are evaluated by using both off-line and on-the-fly.

Evaluation integration models: The integration model can be *pull* or *push.* In the pull model, DaaS operations will invoke *Data Concerns Evaluation Tool* by passing data asset references or data asset values. In this model, the data is accessed as normally and data concerns can be separately evaluated (at the same time, before, or after the data access). It is easier to include new data concerns tools and to utilize pre-evaluated data concerns. This integration model can support generic data assets (e.g., relational, XML, and file-based data). This can support on-the-fly evaluation and data concerns evaluation tools can be an external software-as-a-service as well. In the pull, pass-by-value evaluation model, data is captured and specific tools are used to evaluate data concerns in the DaaS service APIs. This is suitable for specific data concerns (e.g., anonymity operation on privacy data) and requires a tightly coupling between DaaS operations and evaluation tools due to the passing of data values. In the push integration model, data is pushed to evaluation tools which determine concerns and pass the evaluated concerns to the DaaS service operation. This model is suitable for active data sources or subscribed data which is continuously pushed to the DaaS service operation.

Our solutions are part of the WWTF SODI project. Using our solutions, we show that we can apply them for on-the-fly data concerns in Clouds. We validate and demonstrate our techniques with several real-world DaaS. Data concerns can also be used for data quality monitoring. Furthermore, our solutions are applied to QoD monitoring of simulation workflows discussed in Section 2.7.

3.4 Utilizing Data Concerns of Data-as-a-Service

Understanding, evaluating and publishing data concerns of DaaS have led to several benefits. The first utilization of data concerns is that we can use data concerns to optimize queries in data integration [9]. In this work, data concerns are associated with data services. Then when a query is made, data concerns can be associated with a query to search suitable data. Several data concern-aware querying models have been developed [9].

Second, data concerns are used for the development of RESTful Web services for data-intensive services [100]. First, we analyze relationships among context, quality, and relevance, as well as their impact on the design and composition of Web services, in particular at the data asset level. Then, we propose several techniques to incorporate and couple context and quality descriptions into REST APIs and RESTful services publishing. Context and quality can be described using the model in Section 3.2. This includes publishing quality and context information for service discovery, specifying context and quality parameters in REST APIs, and obtaining context and QoD in REST APIs. By implementing these features, RESTfulWeb services could allow the consumer to specify and query context and quality information associated with services and data assets, thus fostering the provision of high relevant data assets. We also applied data concerns for dealing with information overloading in service composition [89].

Third, from the DaaS model, we also develop conceptual framework for privacy which is a specific type of data concerns [65]. In the context of this work, we consider three levels of data privacy capabilities (i)the service as a whole: privacy capabilities apply to all data assets returned by any service invocation, (ii) service operation: privacy capabilities apply to all invocations of specific service operation, and (iii) data assets: privacy capabilities apply to data assets.

Finally, recently we have linked different types of data concerns at DaaS and data assets level with QoS service and data and service contract (see Chapter 4) in order to describe DaaS in data marketplaces.

3.5 Publications and Notes

- **Hong-Linh Truong, Schahram Dustdar: On analyzing and specifying concerns for data as a service. APSCC 2009:87-94** [91]
- **Hong-Linh Truong, Schahram Dustdar: On Evaluating and Publishing Data Concerns for Data as a Service. APSCC 2010:363-370** [94]
- Hong-Linh Truong, Schahram Dustdar, Andrea Maurino, Marco Comerio: Context, Quality and Relevance: Dependencies and Impacts on RESTful Web Services Design. ICWE Workshops 2010: 347-359 [100]
- Hong-Linh Truong, Marco Comerio, Andrea Maurino, Schahram Dustdar, Flavio De Paoli, Luca Panziera: On Identifying and Reducing Irrelevant Information in Service Composition and Execution. WISE 2010: 52-66 [89]
- Michael Mrissa, Salah-Eddine Tbahriti, and Hong-Linh Truong. 2010. Privacy Model and Annotation for DaaS. In Proceedings of the 2010 Eighth IEEE European Conference on Web Services (ECOWS '10). IEEE Computer Society, Washington, DC, USA, 3-10. DOI=10.1109/ECOWS.2010.11 http://dx.doi.org/10.1109/ECOWS.2010.11 [65]
- Quang Hieu Vu, Tran Vu Pham, Hong-Linh Truong, Schahram Dustdar, Rasool Asal, DEMODS: A Description Model for Data-as-a-Service, (c)IEEE Computer Society, The 26th IEEE International Conference on Advanced Information Networking and Applications (AINA-2012), Fukuoka, Japan, March 26-29, 2012 [106].
- Schahram Dustdar, Reinhard Pichler , Vadim Savenkov, Hong-Linh Truong, "Quality-aware Service-Oriented Data Integration: Requirements, State of the Art and Open Challenges", SIGMOD Record, Vol. 41, Number 1, March 2012 [37].

The original ideas of the research topics discussed in this chapter has been developed by the author of this thesis. The works mentioned in Sections 3.2

and 3.3 were conducted with Schahram Dustdar, published in [91, 94] and are included in Appendix B. The works on utilizing data concerns analysis and evaluation mentioned in Section 3.4 were conducted with several colleagues, including Muhammad Intizar Ali, Marco Comerio, Schahram Dustdar, Andrea Maurino, Flavio De Paoli, Tran-Vu Pham, Reinhard Pichler, Michael Mrissa, and Quang-Hieu Vu.

Chapter 4

Service and Data Contracts for Complex Service-oriented Systems

4.1 Introduction

As shown in Chapters 2 and 3, complex applications utilize various services and data from different providers. Furthermore, the way they use data and computational services tends to be more and more elastic: data and computational services can be added into and removed from the applications on-demand, under different constraints imposed by consumers and providers. This elastic way can work only when the contracts between the applications and the services to be used can be managed[1]. While adding and removing data and computational services into an application have attracted many researchers, the dynamicity of service and data contracts has not been well supported. This naturally hinders the elastic utilization of data and computational services in large-scale infrastructures, such as Clouds.

While several researches have been devoted for modeling and specifying service contracts, we deal with service and data contracts from the need of complex applications: utilizing data and services from different providers whose contracts are heterogeneous. Such applications require mechanisms to guarantee that they can handle and compare multiple service and data contracts to ensure that they do not have any incompatibility when using different services and data. From that perspective, we have found that several issues, including (i) there is a strong separation between contracts discussing technical, measurable conditions and business legal, (ii) there is a strong difference among different contract terms that would hinder multiple contracts utilization, (iii) contracts are mostly applied to the service as a whole but not data, and (iv) there is a lack of data contracts. Our work described in this chapter addresses some of the above-mentioned problems. In this di-

[1] Contracts define agreements between service providers and consumers and can cover different aspects of how data and service can be used. In literature, the terms contracts, policies, licenses, and service level agreements (SLAs) are used, sometimes, interchangeably. In this thesis, we use the term "contract" to indicate all of them.

rection, we contribute two important results (i) reconciliation and mapping among different contracts, service and data contract compatibility and (ii) data service contracts [27, 101, 102, 90].

4.2 Reconciliation of Service Contracts

When we use a service we need to comply with constraints imposed by the service provider. It is obviously not-so-trivial but feasible to deal with individual providers in an isolated manner. However, today we often utilize different services from different providers in order to construct converged services. As a consequence, we have to ensure that we comply with constraints imposed by different vendors. Unfortunately, for this compliance we have to consider all contract constraints imposed by different service providers.

4.2.1 Research problems

In cloud computing environments, individual providers tend to define their own service contract models, using different specifications and terminologies, to maximize the utilization of their own services. Therefore, making the service contract compatibility evaluation among different providers is hard. Existing research has neglected contracts of composite services when performing service composition by considering only functional parameters (service interfaces) or assume that contracts associated with services being composed are described by a single language. Furthermore, past research has not focused on tools and algorithms dealing with contract compatibility evaluation when combining different services from different providers. Typically, they deal with only contract between consumer and service in a point-to-point manner.

4.2.2 Approaches and solutions

One of the first step is to understand the relationships between service properties and contracts and which information can be included in services. In our study of different types of contract information we categorize contractual terms into:

- quality of service and quality of data terms (e.g., response time, accuracy and availability),
- legal terms (e.g., fair use and copyrights),
- intellectual rights terms (e.g., allowing or denying composition), and
- financial terms (such as payment and tax)

Typically these terms are described in forms of conditions established on the basis of non-functional properties associated with services. As there is no unique way of describing these terms, service providers and consumers can represent these terms in their own ways, causing strong ambiguity and redundancies in terms descriptions and preventing the right interpretation of contract terms in multi-provider service-oriented environments.

In dealing with with the above-mentioned issues, we have analyzed the complexity of current contractual concerns and posed several research questions:

- What would be the best way to manage non-functional properties and service contracts associated with service instances
- Can we have a single language to describe all contractual concerns?
- Do the existing languages/standards satisfy the requirements for representing contractual concerns?
- Can we use contract negotiation/compatibility algorithms, even though we do not have a unified specification approach?
- Is real time monitoring and enforcement of contractual concerns, in particular level and intellectual right terms of dynamic services possible?

To answer some of these questions, we have concretely proposed some approaches and solutions for some of the above-mentioned questions.

- First of all, we propose the management of non-functional properties and service contracts by using a data representation based on hierarchical service catalogs [87] for services. This model enables the management of several service properties, including contracts, at the whole service level. Techniques introduced in [87], however, are not suitable for describing contracts associated with DaaS, which have different levels of abstraction. Therefore, we devise a new model describing non-functional properties and service/data contracts at the DaaS service level, data API level and data asset level [94, 106]. Furthermore, the same way is applied for specifying privacy constraints of DaaS [65].
- Second, we address contract mapping and contract compatibility issues for heterogeneous contract specifications by (i)modeling and mapping service contract terminologies using a reference ontology, (ii) developing different mapping techniques for different contracts into a common contract models, and (iii) developing different compatibility algorithms based on the common model [27].
- Third, we examine how existing languages/standards can support requirements for representing data-specific terms. We found that data contracts have been neglected, although complex applications can utilize data in an elastic manner by accessing data from DaaS and data marketplaces. This leads to our research topics described in Section 4.3.

Our techniques for reconciliation of contractual terms have been validated with several real services. They contribute to the work on the FP7 COIN IP project[2] and the WWTF SODI project.

4.3 Data Contract

As discussed in Chapter 3, data is associated with multiple concerns. To date, relationships between data contract and service contract in the ecosystem of DaaS have been neglected in current research. In fact, when a DaaS provides rich types of data, then service contracts cannot be used to specify data contracts as (i) a DaaS offers facilities for multiple data providers, (ii) a data provider has multiple types of data, and (iii) each type of data can be associated with multiple data contracts. Thus, service contracts applied to the DaaS as a whole will not be suitable. We believe that it is required to define data contract models that can be used separate from service contract or in combination with service contract.

4.3.1 Research problems

First, for rich data assets in cloud data marketplaces, we need open and common contractual terms that can be reused and composed and allows new terms to be defined and integrated into our model. This requires us to understand common data contract terms and to develop a different approach for data contract terms.

Second, data contracts must be designed suitable for DaaS and data marketplaces, for example, ODRL [52] [69] allows specifying data terms but it is not designed for data assets in data marketplaces. In SOA, QoS models for Web services have been well-researched and various techniques, methods and tools to support QoS modeling for Web services have been proposed [58, 77, 107]. However, they mainly focus on operational aspects of services like performance, reliability, availability, and security, while the data aspects related to data publishing are largely ignored. On the other hand, much effort has been spent on QoD from database perspectives and many metrics characterizing data quality have been proposed [75, 14]. Nevertheless, there is a lack of integration between data contract terms and service contract terms. In fact, no standard model of data contracts that could serve as a basis for the DaaS specification is available so far. Similarly, existing service licensing and service level agreements (SLAs), see e.g, [44, 54], are mainly for "opera-

[2] `www.coin-ip.eu`

tional" service APIs and they do not include mechanisms to deal with data contract terms.

4.3.2 Approaches and solutions

We have performed a detailed analysis of data contracts and found that most data contracts are not well defined. We found that categories of data contract terms are limited, however, contract terms are diverse. In particular, data contract terms are contextual (e.g., based on laws of geographical regions and the domain of data assets). Furthermore, in many cases, data contract term values and their measurement units are also complex and contextual, e.g., one needs to make sure that the value "Austria" can be interpreted as a sub element of "EU" (European Union) in some specific contexts. Therefore, we do not expect that a unified specification for data contracts, with pre-defined term names, will be available and sufficient. In order to deal with data contracts in data marketplaces, we propose a different approach centered on (i) a combination of community and people-centric collaboration, and and (ii) an abstract model for data contract and its development approach and techniques.

First, we propose to enable community users to participate in defining (i) fundamental elements in data contracts, such as term categories, term names, term values and term units, (ii) rules for data contracts, such as syntax validation and evaluation rules, and (iii) common contracts and contract fragments (see Figure 4.1). Note that community users should be understood as experts in specific domains who understand contractual terms suitable for data in their domain, not novice users. The combination of community and people-centric collaboration is powerful for solving the heterogeneity of data contract terms. By employing people-centric approach in establishing and developing data contracts, we propose that data providers and consumers can utilize fundamental elements to define their own contracts and evaluation techniques.

Second, to provide abstract data contract models we determine possible representations for data contract terms. From our analysis, we describe possible ways to model data contract terms for different categories. Overall, we can represent a data contract term as a tuple of $(termName, termValue)$ in which $termName$ is either common terms established via standards/communities or user-specific terms and $termValue$ are the assigned values for $termName$. $termValue$s can be a set, a single value, or a range. We support the following categories: Data Rights, QoD, Compliance, Pricing Model and Control & Relationship. Figure 4.2 presents our abstract data contract structure. By "abstract" we mean two aspects. First, this structure is not the final form of a data contract as it represents only contractual conditions without specifying on which data assets it is applied to. Second, the structure is not a

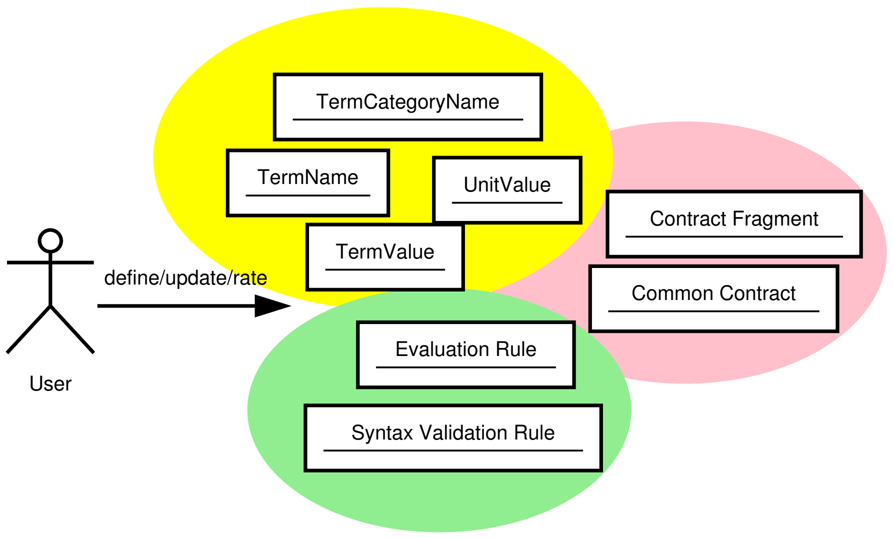

Fig. 4.1 Community contributions in data contracts

proposal for final and concrete data contract specification, which can be seen and obtained by data consumers in data marketplaces, but it can be used to make abstract contracts from which concrete specifications will be generated for data consumers and providers.

With our data contract framework, we demonstrate how we can facilitate the development of data constraints in different scenarios. We also show that data contract compatibility can be developed and data contracts can be exchanged easier. Our data contract model is part of our research in the context of our Pacific Controls Cloud Computing Lab (PC3L)[3].

4.4 Publications and Notes

The following publications are related to the research questions and contributions mentioned in this chapter.

- Hong-Linh Truong, G.R. Gangadharan, Martin Treiber, Schahram Dustdar, Vincenzo D'Andrea, "On Reconciliation of Contractual Concerns of Web Services,", 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'08), co-located with The 6th IEEE European Conference on Web Services, CEUR Vol-411, November 12, 2008, Dublin, Ireland.

[3] `http://pc3l.infosys.tuwien.ac.at/`

- **Hong-Linh Truong, G.R. Gangadharan, Marco Comerio, Vincenzo D Andrea, Flavio De Paoli and Schahram Dustdar. "Reconciliation of Contractual Concerns of Web Services." In Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions, ed. Stephan Reiff-Marganiec and Marcel Tilly, 298-321 (2012), 2012. doi:10.4018/978-1-61350-432-1.ch013** [101].
- Marco Comerio, Hong-Linh Truong, Flavio De Paoli, Schahram Dustdar: Evaluating Contract Compatibility for Service Composition in the SeCO2 Framework. ICSOC/ServiceWave 2009: 221-236 [27]
- Hong-Linh Truong, Schahram Dustdar, Joachim Goetze, Tino Fleuren, Paul Mueller, Salah-Eddine Tbahriti, Michael Mrissa, Chirine Ghedira: Exchanging Data Agreements in the DaaS Model. APSCC 2011: 153-160 [99].
- Hong-Linh Truong, G.R. Gangadharan, Marco Comerio, Schahram Dustdar, Flavio De Paoli, "On Analyzing and Developing Data Contracts in Cloud-based Data Marketplaces", The 2011 Asia-Pacific Services Computing Conference (IEEE APSCC 2011), (c) IEEE Computer Society, December 12 - 15, 2011, Jeju, Korea [102]
- **Hong-Linh Truong, Marco Comerio, Flavio De Paoli, G.R. Gangadharan, Schahram Dustdar, "Data Contracts for Cloud-based Data Marketplaces", International Journal of Computational Science and Engineering, Vol. 7, No. 4, 2012.** [90].

The original ideas of research topics in this chapter have been developed by the author and in collaboration with several colluagues. The work in Section 4.2 is conducted together with G.R. Gangadharan, Marco Comerio, Vincenzo D'Andrea, Flavio De Paoli, Schahram Dustdar, and Martin Treiber. This work has started since G.R. Gangadharan and Marco Comerio were PhD internships working with the author and Martin Treiber was a PhD student working with the author. The work mentioned in Section 4.2 were published in [27, 101] and we include the paper [101] in Appendix C. The work mentioned in Section 4.3 was conducted together with G.R. Gangadharan, Marco Comerio, Flavio De Paoli, and Schahram Dustdar, and is published in [90] and included in Appendix C.

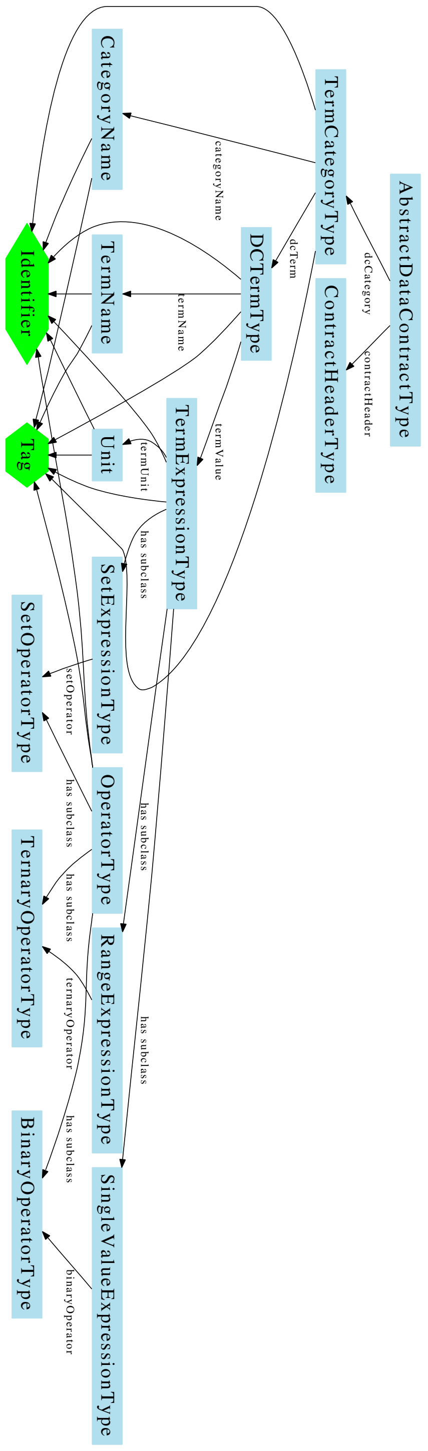

Fig. 4.2 (Simplified) abstract structure of data contracts

Chapter 5
Challenges in Elastic Computing

5.1 Introduction

Instead of wrapping up our results described in Chapters 2–4 as a conclusion for this thesis, this chapter will discuss our view on our future research direction that largely emerges from the study of quality of complex service-oriented applications and systems in this thesis. What we have observed is that the user of complex applications really cares about a few properties, such as quality of results (e.g., cost, response time and QoD). However, we have observed rapid changes in service provisioning with the emerging cloud computing and social computing. There are several service models have been offered, including IaaS, PaaS, and SaaS. Moreover, not only software but human can also be provisioned and incorporated into complex applications. These service models introduce diversity and complexity of service-based resources and NfPs. Several complex applications can utilize them in different ways and can be elastic in different manners to meet the user's expected properties.

However, the current view in elasticity is very much traditional, centered on the elasticity of machine-based resources [25, 1, 2]. We believe that elastic techniques will play a major role in utilizing resources and satisfying requirements from consumers as well as in the mapping of user requirements to complex resources and quality properties. But to go beyond contemporary techniques, we need to foster the concept of elastic processes based on different dimensions, such as price, resource, and quality [34]. However, supporting multi-dimensional elasticity requires us to characterize and evaluate properties of complex applications and systems in order to understand which properties based on that the elasticity can be built as well as how to model and reason these properties at the application as well as the underlying computing system levels. In this research direction, considered a future plan, we will discuss some initial results and approaches on understanding multi-dimensional elasticity, integration and virtualization of human-based and machine-based elements and elasticity engineering.

5.2 Novel Multi-dimensional Elasticity

Resource Elasticity: The Internet and underlying powerful, pervasive machine-based computing systems have enabled us to harness vast human capabilities around the world. Human capabilities can be exploited in a similar way to machine-based computing elements (MCEs): humans perform certain types of computing activities. For example, in the search of Jim Gray missing [49] as well as on situational analysis in disaster scenarios [105], several images taken from satellites are processed by various software (machine-based computing elements) in order to detect patterns required for specific purposes (such as missing people or victims in a disaster). Another scenario is that several long-running scientific simulations can be performed by MCEs, but scientists are required to analyze quality of intermediate simulation results during the simulations. In general, it is well-known that we need to have humans in the loop is due to the complexity of the problems that the software cannot handle. Essentially, we need to build the elasticity of hybrid computing elements within these complex applications but these applications must be *proactive* in scaling in/out of MCEs and HCEs to solve problems under *several constraints*, such as time, quality of results, and compliance laws. These applications should be able to decide when and how to take into account HCEs instead of MCEs actively, rather than just posting the tasks to crowds and waiting for some HCEs to take the tasks, as shown in current crowdsourcing systems. As a result of these requirements, we need techniques to support the integration of HCEs and MCEs into a single system that goes beyond contemporary crowdsourcing models. We must virtualize software and human in such a way that facilitates the management of software and human and enables the programming of them in a similar way.

Resource and Quality Elasticity: We examined the need from small CSE research groups in general, and research and teaching groups in developing countries and we found that they need techniques to support quality elasticity together with resource quality [95]. The major obstacle of these groups is that they need computational resources which are currently not enough but the way they use resources a very different: depending on different factors, with the same application, some scientists may want to be delivered at the very good quality in a very short time, while others may prefer to use it with a minimal cost, regardless of service quality and time. In short, requirements from scientists are *elastic* in different dimensions, including resource, quality, cost, available time and usage rights [74].

Cost, Rewarding and Incentive Elasticity: We examined the case of using on-demand Social Computing Units (SCUs) [33] for solving complex problems, e.g., incident responses or software management. One interesting aspect is that when solving problems, new problems can emerge or old problems will be resolved. This leads to an interesting aspect is that the SCU is also expanded or reduced accordingly. However, when the SCU is expanded or reduced, the tasks of the SCU members have to do are different. This requires the cost, rewarding and incentive models for the SCU to be changed to support the work. This reflects cost, rewarding and incentive elasticity support in complex applications.

To deal with that, we believe that a well understood multiple elastic properties must be presented in order to support. Figure 5.1 describes elasticity dimensions.

- *Resource elasticity*: resources can be elastic by taking into account MCEs and HCEs from multiple clouds. Resources can be provided under the service models; thus we have Software-based Service (SBS) units built atop MCEs, Human-based Service (HBS) units built atop HCEs and mixed service units composing SBS and HBS units. From functionalities, resources can act as, for example, data service, network service and computational service. To enable this, we need to virtualize HCEs and MCEs in a similar manner. Furthermore, techniques for multiple clouds must be extended for HBSs.
- *Quality elasticity*: quality can be elastic by multiple of quality properties, including two major classes namely QoS and QoD. We need to model and evaluate these quality dimensions that characterizes data, computation and human behaviors.
- *Costs and Benefits elasticity*: cost and incentive factors can be elastic by switching from one model to another model as well by combining different models. This dimension covers most types of (economical and social) costs and benefits that can be associated with SBS and HBS units.

5.3 The Vienna Elastic Computing Model

To support the development of applications with built-in multi-dimensional elasticity capabilities, we introduce the Vienna Elastic Computing Model (VieCOM) which is built based on the following key points:

- supporting multi-dimensional elasticity: our computing model supports the principles of elastic processes [34]. By considering these principles, we take into account different dimensions of elasticity. In particular, we propose criteria in these dimensions to suitable for both MCEs and HCEs. We consider scaling out/in MCEs and HCEs in our processes. Thus, we consider hybrid systems of MCEs and HCEs in our applications [38]. Furthermore, we support price/reward/incentive models and quality models in the context of multiple clouds [82].

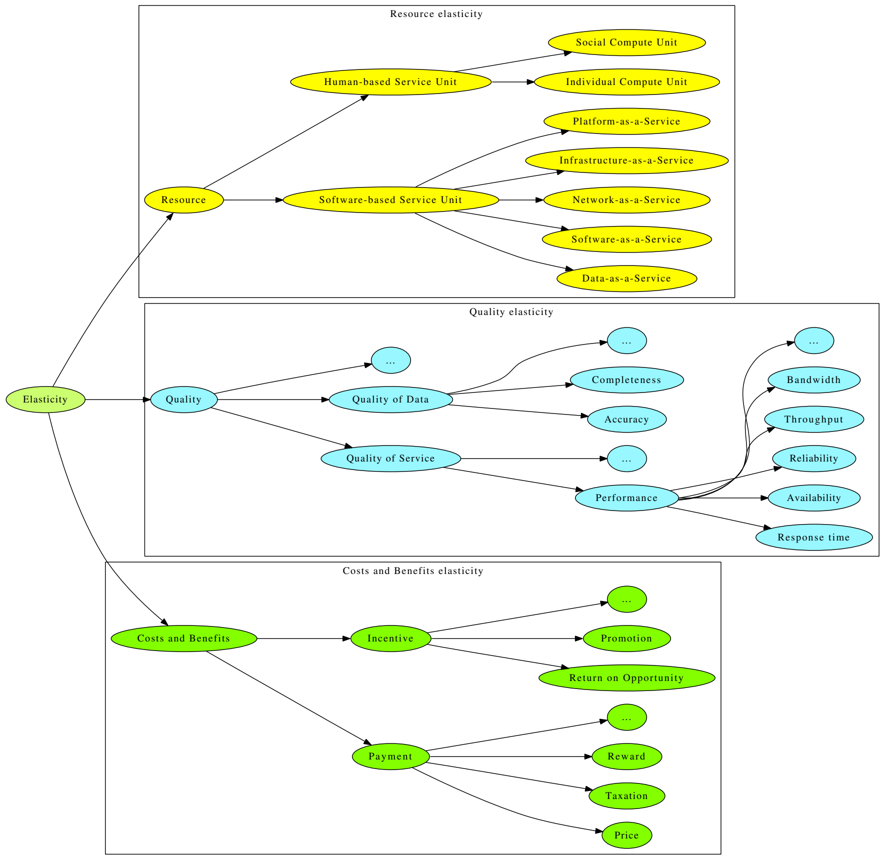

Fig. 5.1 (Simplified) elasticity dimensions

- following the SOC model: our computing model utilizes existing SOC techniques and concepts to virtualize MCEs and HCEs under the service model and to provision them under different forms of cloud systems [97].
- supporting an end-to-end approach, from modeling to execution: our computing model aims at providing techniques to cover different layers ranging from modeling to runtime management [21]. This aims at closing the gap between the modeling techniques and runtime management techniques for elastic computing.

Figure 5.2 outlines layers in the Vienna Elastic Computing Model. We briefly explain them in the following:

- *Computing Element*: describes fundamental computing elements in VieCOM which can be machines or humans.

- *Virtualization*: describes virtualization techniques and frameworks that provision HCEs and MCEs under IT services whose properties can be elastic.
- *IT Elastic Service*: abstracts HCEs and MCEs under the service model. *IT Elastic Services* can be selected based on their properties and be invoked based on their well-defined service interfaces, such as based on SOAP or REST. *IT Elastic Services* can be elastic, e.g., their costs and quality can be dynamically changed, depending on specific contexts.
- *IT Elastic Process*: specifies IT processes built atop *IT Elastic Services*. An *IT Elastic Process* can be described in the form of workflows or distributed components.
- *Elastic Service*: specifies business service which consists of non IT services and of IT processes.
- *Elastic Runtime Management*: includes techniques and platforms to manage and execute *IT Elastic Services*, *IT Elastic Processes* and *Elastic Services* provisioned from different clouds.
- *Elastic Alignment*: specifies techniques and frameworks to support the alignment between *Elastic Service* and its corresponding IT process (specified by *IT Elastic Process*), and between *IT Elastic Process* and its *IT Elastic Service*.
- *Elasticity Modeling*: specifies techniques and frameworks for modeling elastic properties and trade-offs for services and processes.
- *Elasticity Monitoring*: specifies techniques and frameworks for monitoring the elasticity of services, processes and computing elements.
- *Application*: specifies different types of applications that are built by utilizing *Elastic Services*.

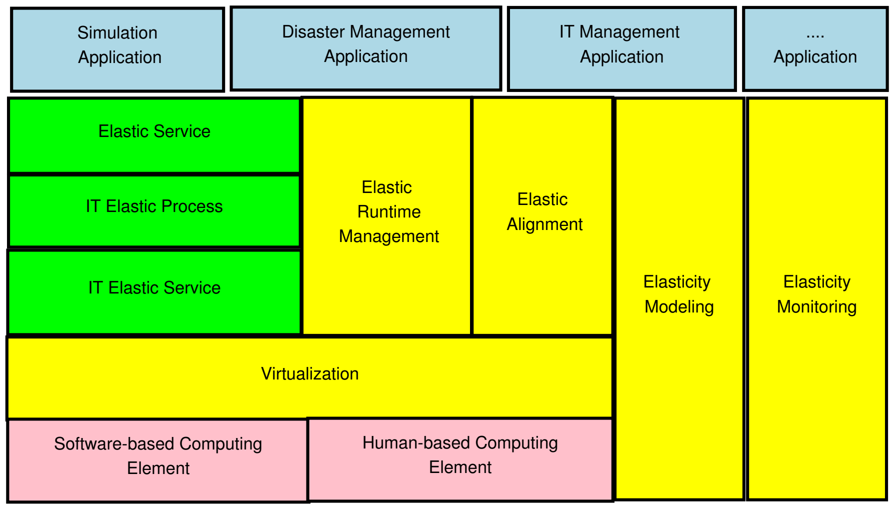

Fig. 5.2 Conceptual view of the Vienna Elastic Computing Model

5.4 Modeling, Composing and Reasoning Elastic Processes

As mentioned, the elasticity is the future, therefore, several ongoing works are currently carried out. We outline some of our early contributions on modeling and composition techniques for elastic processes.

- *Elastic High Performance Applications:* To enable scientists to develop research applications that can be executed on different and multiple cloud platforms, including private as well as public clouds, we focus on elastic high performance applications (eHPAs) [74]. eHPAs are composed from different types of components, including services, libraries, self-made codes, and virtual machines. Due to the availability and diversity of such components, we consider them as elastic components: a component itself can be elastic to eHPA composer and scientists. Based on characteristics of HPAs, we model elastic components using several properties characterizing resources, costs, quality, usage rights and time. Based on these properties and user needs (e.g. budget and time), an eHPA can be composed from different elastic components. As a result, an eHPA can also be characterized via several elastic properties. As the couplings between an eHPA and its constitute elastic components have different degrees, we further partition the eHPA structure into different partitions. Although we have not developed adaptation techniques for eHPAs, depending on eHPA elastic properties and partitions, suitable and different strategies could be developed and applied for eHPA partitions and components, making eHPA execution elastic in different clouds under different elastic conditions.
- *Elastic Modeling for Business/IT Services*: To consider the elastic alignment from elastic business service to IT Elastic Process to IT Elastic Service, we need to model and reason elastic properties at different levels, including the business level, the IT process level and IT service/resource level. We approach this by considering few elastic properties associated with business services whose include several business steps [57]. A business step is mapped to a set of IT resources including both machine-based and human-based services. Thus we need to map the few business elastic properties to a vast resource elastic properties. Our initial development shows that elasticity reasoning relies on a general approach including (i) modeling elastic properties at different layers, (ii) mapping elastic properties one layer to another layer using rules, (iii)runtime reasoning based on rules and runtime elastic properties, and (iv) scaling in/out resources based on workflow regions and activity levels. Although not all steps have been implemented and validated, our initial results show that it is a promising way to deal with multi-dimensional elasticity across different layers.

5.5 Programming Hybrid Services in the Cloud

The emerging computing model in which HBS and SBS are interconnected in different ways could support different programming models; there are different ways to develop applications atop such a new computing model. In current research approaches, human-based capabilities are usually provisioned via "crowdsourcing" platforms [31] or specific human-task plug-ins [12, 60, 62]. These approaches achieve human and software integration mainly via (specific) platform integration but essential programming elements representing SBS and HBS cannot be programmed directly into applications. Furthermore, these approaches does not provide a uniform view of SBS and HBS, and let the developer perform the complex tasks of establishing relationships between SBS and HBS.

Our goal is to program HBS and SBS together in an easier way because several complex applications need to utilize SBS and HBS in a similar way. Thus, we conceptualize human capabilities under the service model and combine them with software establishing clouds of hybrid services. We explore novel ways to actively program and utilize human capabilities in a similar way to software services. By enabling cloud provisioning models for human capabilities, HBS can be requested and initiated on-demand under different quality, cost and benefit models. To support programming hybrid services in cloud, we have researched and developed [97]:

- a novel model for clouds of HBS and hybrid services provisioning
- a framework for solving complex problems using clouds of hybrid services
- programming primitives for hybrid services

In our initial results, we have illustrated our programming concepts via several examples of using our cloud APIs and existing cloud APIs for SBS.

5.6 Publications and Notes

As we discuss before, elastic processes and techniques for elastic computing are an ongoing work. The following papers present main publications of which some are summarized in this chapter:

- **Hong-Linh Truong, Schahram Dustdar: Cloud computing for small research groups in computational science and engineering: current status and outlook. Computing 91(1): 75-91 (2011)** [95]
- Tran Vu Pham, Hong-Linh Truong, Schahram Dustdar: Elastic High Performance Applications - A Composition Framework. APSCC 2011: 416-423 [74]
- Hong-Linh Truong, Tran-Vu Pham, Nam Thoai and Schahram Dustdar. "Cloud Computing for Education and Research in Developing Countries."

In Cloud Computing for Teaching and Learning: Strategies for Design and Implementation, ed. Lee Chao, 64-80 (2012), doi:10.4018/978-1-4666-0957-0.ch005 [103].

- **Schahram Dustdar, Hong-Linh Truong, "Virtualizing Software and Humans for Elastic Processes in Multiple Clouds - a Service Management Perspective", Invited paper, International Journal of Next-Generation Computing, March, 2012.** [38]
- **Hong-Linh Truong, Schahram Dustdar, Kamal Bhattacharya, "Programming Hybrid Services in the Cloud", 10th International Conference on Service-oriented Computing (ICSOC 2012), (c) Springer-Verlag, November 12-16, 2012, Shanghai, China** [97]

The original ideas of the topics presented in this chapter have been developed by the author and coauthors. The elasticity requirements mentioned in Section 5.2 were conducted together with Schahram Dustdar, Tran-Vu Pham and Nam Thoai and published in different scientific papers [95, 103]; the paper [95] is included Appendix D. The work mentioned in Section 5.3 was conducted with Schahram Dustdar, is published in [38], and is included in Appendix D. In Section 5.4, eHPA composition is conducted when Tran Vu Pham was a visitor hosted by the author and published in [74]. The service provisioning [57] is conducted in the context of collaboration between the author and Lam-Son Le, Aditya Ghose and Schahram Dustdar. The work in programming hybrid services [97], included in Appendix D, is conducted in collaboration between the author and Schahram Dustdar and Kamal Bhattacharya. As the elasticity is the future direction, several research topics are currently being investigated. The author would like to thank for fruitful discussion about elasticity. Furthermore, the author would like to thank several other colleagues and students, Kamal Bhattacharya, Hoa Khanh Dam, Yike Guo, Frank Leymann, Vitaliy Liptchinsky, Mirela Riveni, Benjamin Satzger, Ognjen Scekic, and Rostyslav Zaboloznyi for their fruitful discussions in the topics of elasticity.

Part II
Representative Published Scientific Work

References

1. Elastic computing. `http://en.wikipedia.org/wiki/Elastic_computing`. Last access: 3 April 2012
2. Elasticity (data store). `http://en.wikipedia.org/wiki/Elasticity_(data_store)`. Last access: 3 April 2012
3. QoS for Web Services: Requirements and Possible Approaches, http://www.w3c.or.kr/kr-office/tr/2003/ws-qos/
4. First International Conference on e-Science and Grid Technologies (e-Science 2005), 5-8 December 2005, Melbourne, Australia. IEEE Computer Society (2005)
5. Second International Conference on e-Science and Grid Technologies (e-Science 2006), 4-6 December 2006, Amsterdam, The Netherlands. IEEE Computer Society (2006)
6. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., de Medeiros, A.K.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W.E., Weijters, A.J.M.M.: Prom 4.0: Comprehensive support for *eal* process analysis. In: J. Kleijn, A. Yakovlev (eds.) ICATPN, *Lecture Notes in Computer Science*, vol. 4546, pp. 484–494. Springer (2007)
7. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14**(1), 5–51 (2003)
8. Al-Ali, R.J., Amin, K., von Laszewski, G., Rana, O.F., Walker, D.W., Hategan, M., Zaluzec, N.J.: Analysis and provision of qos for distributed grid applications. J. Grid Comput. **2**(2), 163–182 (2004)
9. Ali, M.I., Pichler, R., Truong, H.L., Dustdar, S.: Incorporating data concerns into query languages for data services. In: R. Zhang, J. Zhang, Z. Zhang, J. Filipe, J. Cordeiro (eds.) ICEIS, *Lecture Notes in Business Information Processing*, vol. 102, pp. 132–145. Springer (2011)
10. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
11. Armbrust, M., Fox, A., Grifth, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Tech. rep., University of California at Berkeley (2009). URL `http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html`
12. Barowy, D.W., Berger, E.D., McGregor, A.: Automan: A platform for integrating human-based and digital computation. Technical Report UMass CS TR 2011-44, University of Massachusetts, Amherst (2011). Http://www.cs.umass.edu/ emery/pubs/AutoMan-UMass-CS-TR2011-44.pdf
13. Barros, A.P., Dumas, M., ter Hofstede, A.H.M.: Service interaction patterns. In: W.M.P. van der Aalst, B. Benatallah, F. Casati, F. Curbera (eds.) Business Process Management, vol. 3649, pp. 302–318 (2005)

14. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. ACM Comput. Surv. **41**(3) (2009)
15. Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L., Munro, M.: Service-based software: the future for flexible software. Asia-Pacific Software Engineering Conference **0**, 214 (2000). DOI http://doi.ieeecomputersociety.org/10.1109/APSEC.2000.896702
16. Berman, F., Fox, G., Hey, A.J.G., Hey, T.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Inc. (2003)
17. Bhatti, S.N., Sorensen, S.A., Clark, P., Crowcroft, J.: Network QoS for Grid Systems. International Journal of High Performance Computing Applications **17**(3), 219–236 (2003). DOI 10.1177/1094342003173009. URL `http://hpc.sagepub.com/cgi/content/abstract/17/3/219`
18. Brandic, I., Benkner, S., Engelbrecht, G., Schmidt, R.: Qos support for time-critical grid workflow applications. In: e-Science [4], pp. 108–115
19. Bubak, M., Unger, S.: K-Wf Grid: the knowledge-based workflow system for Grid applications : Cracow '06 Grid Workshop : proceedings, October 15-18, 2006 Cracow, Poland. Proceedings of Cracow '06 Grid Workshop, October 15 - 18, 2006, Cracow , Poland. Academic Computer Centre Cyfronet AGH (2007). URL `http://www.cyf-kr.edu.pl/cgw06/info/K-WfGrid_PROCEEDINGS.pdf`
20. Buyya, R., Yeo, C.S., Venugopal, S.: Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. High Performance Computing and Communications, 10th IEEE International Conference on **0**, 5–13 (2008). DOI http://doi.ieeecomputersociety.org/10.1109/HPCC.2008.172
21. Candra, M.Z.C., Zaboloznyi, R., Truong, H.L., Dustdar, S.: Handbook on Web Services, chap. Virtualizing Software and Human for Elastic Hybrid Services. Springer-Verlag (2012). Accepted Sep 2012. To appear
22. Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: GECCO, pp. 1069–1075. ACM (2005)
23. Cardoso, J., Sheth, A.: Semantic e-workflow composition. J. Intell. Inf. Syst. **21**(3), 191–225 (2003). DOI http://dx.doi.org/10.1023/A:1025542915514
24. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. J. Web Sem. **1**(3), 281–308 (2004)
25. Chiu, D.: Elasticity in the cloud. Crossroads **16**, 3–4 (2010). DOI http://doi.acm.org/10.1145/1734160.1734162. URL `http://doi.acm.org/10.1145/1734160.1734162`
26. COIN: Enterprise Collaboration and Interoperability: http://www.coin-ip.eu/. Last access: 28 Nov 2008
27. Comerio, M., Truong, H.L., Paoli, F.D., Dustdar, S.: Evaluating contract compatibility for service composition in the seco$_2$ framework. In: L. Baresi, C.H. Chi, J. Suzuki (eds.) ICSOC/ServiceWave, *Lecture Notes in Computer Science*, vol. 5900, pp. 221–236 (2009)
28. Dan, A., Johnson, R., Arsanjani, A.: Information as a service: Modeling and realization. Systems Development in SOA Environments, 2007. SDSOA '07: ICSE Workshops 2007. International Workshop on pp. 2–2 (2007). DOI 10.1109/SDSOA.2007.5
29. Dasgupta, S.: Social Computing: Concepts, Methodologies, Tools, and Applications. Hershey, PA, USA. IGI Global (2010). URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-984-7`
30. Deelman, E., Plante, R., Kesselman, C., Singh, G., Su, M.H., Greene, G., Hanisch, R., Gaffney, N., Volpicelli, A., Annis, J., Sekhri, V., Budavari, T., Nieto-Santisteban, M.A., O'Mullane, W., Bohlender, D., McGlynn, T., Rots, A.H., Pevunova, O.: Grid-based galaxy morphology analysis for the national virtual observatory. In: SC, p. 47. ACM (2003)
31. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM **54**(4), 86–96 (2011)

32. Dobson, G., Lock, R., Sommerville, I.: Qosont: a qos ontology for service-centric systems. In: EUROMICRO-SEAA, pp. 80–87. IEEE Computer Society (2005)
33. Dustdar, S., Bhattacharya, K.: The social compute unit. IEEE Internet Computing **15**(3), 64–69 (2011)
34. Dustdar, S., Guo, Y., Satzger, B., Truong, H.L.: Principles of elastic processes. IEEE Internet Computing **15**(5), 66–71 (2011)
35. Dustdar, S., Hoffmann, T.: Interaction pattern detection in process oriented information systems. Data Knowl. Eng. **62**(1), 138–155 (2007)
36. Dustdar, S., Hoffmann, T., van der Aalst, W.M.P.: Mining of ad-hoc business processes with teamlog. Data Knowl. Eng. **55**(2), 129–158 (2005)
37. Dustdar, S., Pichler, R., Savenkov, V., Truong, H.L.: Quality-aware service-oriented data integration: requirements, state of the art and open challenges. SIGMOD Rec. **41**(1), 11–19 (2012). DOI 10.1145/2206869.2206873. URL `http://doi.acm.org/10.1145/2206869.2206873`
38. Dustdar, S., Truong, H.L.: Virtualizing software and humans for elastic processes in multiple clouds – a service management perspective. International Journal of Next-Generation Computing (IJNGC) (2012). URL `http://www.infosys.tuwien.ac.at/staff/truong/publications/2012/truong-vecm-ijngc-2012.pdf`. To appear
39. ECOSPACE: eProfessionals Collaboration Space: http://www.ip-ecospace.org/. Last access: 14 April 2008
40. Erdogmus, H.: Cloud computing: Does nirvana hide behind the nebula? IEEE Software **26**(2), 4–6 (2009). DOI http://doi.ieeecomputersociety.org/10.1109/MS.2009.31
41. Fahringer, T., Prodan, R., Duan, R., Hofer, J.g., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J.n., Siddiqui, M., Truong, H.L., Villazon, A., Wieczorek, M.a.: Askalon: A development and grid computing environment for scientific workflows. In: I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (eds.) Workflows for e-Science, pp. 450–471. Springer London (2007). URL `http://dx.doi.org/10.1007/978-1-84628-757-2_27`
42. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
43. Foster, I.T., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. CoRR **abs/0901.0131** (2009)
44. Gangadharan, G.R., D'Andrea, V.: Licensing services: Formal analysis and implementation. In: Proc. ICSOC 2006, *Lecture Notes in Computer Science*, vol. 4294, pp. 365–377. Springer (2006)
45. Geimer, M., Shende, S., Malony, A.D., Wolf, F.: A generic and configurable source-code instrumentation component. In: G. Allen, J. Nabrzyski, E. Seidel, G.D. van Albada, J. Dongarra, P.M.A. Sloot (eds.) ICCS (2), *Lecture Notes in Computer Science*, vol. 5545, pp. 696–705. Springer (2009)
46. Gil, Y., Cheung, W.K., Ratnakar, V., kin Chan, K.: Privacy enforcement in data analysis workflows. In: Proceedings of the AAAI Workshop on Privacy Enforcement and Accountability with Semantics (PEAS), held in conjunction with the Sixth International Semantic Web Conference (ISWC) and the Second Asian Semantic Web Conference (ASWC). Busan, Korea (2007). URL `papers/gil-etal-peas07.pdf`
47. Glatard, T., Montagnat, J., Emsellem, D., Lingrand, D.: A Service-Oriented Architecture enabling dynamic services grouping for optimizing distributed workflows execution. Future Generation Computer Systems **24**(7), 720–730 (2008). URL `http://dx.doi.org/10.1016/j.future.2008.02.011`
48. Gombotz, R., Dustdar, S.: On web services workflow mining. In: C. Bussler, A. Haller (eds.) Business Process Management Workshops, vol. 3812, pp. 216–228 (2005)
49. Hafner, K.: Silicon valley's high-tech hunt for colleague. New York Times. `http://www.nytimes.com/2007/02/03/technology/03search.html?ex=1328158800&en=e58764b50c8a4508&ei=5090&partner=rssuserland&emc=rss` (2007)
50. Heinis, T., Pautasso, C., Alonso, G.: Design and evaluation of an autonomic workflow engine. In: ICAC, pp. 27–38. IEEE Computer Society (2005)

51. Huang, L., Walker, D.W., Rana, O.F., Huang, Y.: Dynamicworkflow management using performance data. ccgrid **0**, 154–157 (2006). DOI http://doi.ieeecomputersociety.org/10.1109/CCGRID.2006.37
52. Iannella, R.: Open digital rights language (odrl) version 1.1 (2002). URL `http://www.w3.org/TR/odrl/`
53. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: QoS Aggregation for Service Composition using Workflow Patterns. In: Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC 2004), pp. 149–159. IEEE CS Press, Monterey, California, USA (2004)
54. Keller, A., Ludwig, H.: The wsla framework: Specifying and monitoring service level agreements for web services. J. Network Syst. Manage. **11**(1) (2003)
55. Knüpfer, A., Kranzlmüller, D., Mohr, B., Nagel, W.E.: M09 - program analysis tools for massively parallel applications: how to achieve highest performance. In: SC, p. 223. ACM Press (2006)
56. Kyriazis, D., Tserpes, K., Menychtas, A., Litke, A., Varvarigou, T.: An innovative workflow mapping mechanism for grids in the frame of quality of service. Future Gener. Comput. Syst. **24**(6), 498–511 (2008). DOI http://dx.doi.org/10.1016/j.future.2007.07.009
57. Le, L.S., Truong, H.L., Ghose, A., Dustdar, S.: On elasticity and constrainedness of business services provisioning. In: The 9th International Conference on Service Computing (SCC 2012). Hawaii, USA (2012)
58. Lee, K., Jeon, J., Lee, W., Jeong, S.H., (eds.), S.W.P.: QoS for Web Services: Requirements and Possible Approaches (2003). URL `http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/`. W3C Technical Report
59. Lipnack, J., Stamps, J.: Virtual teams: reaching across space, time, and organizations with technology. John Wiley & Sons, Inc., New York, NY, USA (1997)
60. Little, G., Chilton, L.B., Goldman, M., Miller, R.C.: Turkit: tools for iterative tasks on mechanical turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '09, pp. 29–30. ACM, New York, NY, USA (2009). DOI 10.1145/1600150.1600159. URL `http://doi.acm.org/10.1145/1600150.1600159`
61. Looker, N., Xu, J.: Assessing the dependability of ogsa middleware by fault injection. In: SRDS, pp. 293–302. IEEE Computer Society (2003)
62. Marcus, A., Wu, E., Karger, D., Madden, S., Miller, R.: Human-powered sorts and joins. Proc. VLDB Endow. **5**, 13–24 (2011). URL `http://dl.acm.org/citation.cfm?id=2047485.2047487`
63. Menascé, D.A., Ruan, H., Gomaa, H.: A framework for qos-aware software components. In: WOSP, pp. 186–196. ACM (2004)
64. Moen, W.E., Stewart, E.L., McClure, C.R.: Assessing metadata quality: Findings and methodological considerations from an evaluation of the u.s. government information locator service (gils). Advances in Digital Libraries Conference, IEEE **0**, 246 (1998). DOI http://doi.ieeecomputersociety.org/10.1109/ADL.1998.670425
65. Mrissa, M., Tbahriti, S.E., Truong, H.L.: Privacy model and annotation for daas. In: A. Brogi, C. Pautasso, G.A. Papadopoulos (eds.) ECOWS, pp. 3–10. IEEE Computer Society (2010)
66. Musunoori, S.B., Eliassen, F., Staehli, R.: Qos-aware component architecture support for grid. In: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004), Modena, Italy, pp. 277–282. IEEE Computer Society Press (2004)
67. Nerieri, F., Prodan, R., Fahringer, T., Truong, H.L.: Performance Analysis of Grid Workflow Applications. In: Proceedings of The 7th IEEE/ACM International Conference on Grid Computing (Grid'06). IEEE Computer Society Press (2006)
68. Nichols, D.M., Chan, C.H., Bainbridge, D., McKay, D., Twidale, M.B.: A lightweight metadata quality tool. In: JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries, pp. 385–388. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1378889.1378957

69. ONIX-PL: `http://www.editeur.org/21/ONIX-PL/` (2011)
70. Papaioannou, I.V., Tsesmetzis, D.T., Roussaki, I.G., Anagnostou, M.E.: A qos ontology language for web-services. AINA **1**, 101–106 (2006). DOI http://doi.ieeecomputersociety.org/10.1109/AINA.2006.51
71. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. IEEE Computer **40**(11), 38–45 (2007)
72. Park, J.J., Nikolaou, C., Cao, J. (eds.): 2011 IEEE Asia-Pacific Services Computing Conference, APSCC 2011, Jeju, Korea (South), December 12-15, 2011. IEEE (2011)
73. Patel, C., Supekar, K., Lee, Y.: A qos oriented framework for adaptive management of web service based workflows. In: DEXA, *Lecture Notes in Computer Science*, vol. 2736, pp. 826–835. Springer (2003)
74. Pham, T.V., Truong, H.L., Dustdar, S.: Elastic high performance applications - a composition framework. In: Park et al. [72], pp. 416–423
75. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Communications of the ACM **45**, 211–218 (2002)
76. Plale, B., Gannon, D., Brotzge, J., Droegemeier, K., Kurose, J.F., McLaughlin, D., Wilhelmson, R., Graves, S., Ramamurthy, M., Clark, R.D., Yalda, S., Reed, D.A., Joseph, E., Chandrasekar, V.: Casa and lead: Adaptive cyberinfrastructure for real-time multiscale weather forecasting. IEEE Computer **39**(11), 56–64 (2006)
77. Ran, S.: A model for web services discovery with qos. SIGecom Exch. **4**(1), 1–10 (2003). DOI http://doi.acm.org/10.1145/844357.844360
78. Reiter, M., Breitenbucher, U., Dustdar, S., Karastoyanova, D., Leymann, F., Truong, H.L.: A novel framework for monitoring and analyzing quality of data in simulation workflows. In: eScience, pp. 105–112. IEEE Computer Society (2011)
79. Reiter, M., Truong, H.L., Dustdar, S., Karastoyanova, D., Krause, R., Leymann, F., Pahr, D.: On analyzing quality of data influences on performance of finite elements driven computational simulations. In: C. Kaklamanis, T.S. Papatheodorou, P.G. Spirakis (eds.) Euro-Par, *Lecture Notes in Computer Science*, vol. 7484, pp. 793–804. Springer (2012)
80. Rosenberg, F., Platzer, C., Dustdar, S.: Bootstrapping performance and dependability attributes of web services. In: IEEE International Conference on Web Services (ICWS'06) (2006)
81. Sabata, B., Chatterjee, S., Davis, M., Sydir, J.J., Lawrence, T.F.: Taxomomy of qos specifications. In: WORDS '97: Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems - (WORDS '97), p. 100. IEEE Computer Society, Washington, DC, USA (1997)
82. Scekic, O., Truong, H.L., Dustdar, S.: Modeling rewards and incentive mechanisms for social bpm. In: A. Barros, A. Gal, E. Kindler (eds.) Business Process Management, *Lecture Notes in Computer Science*, vol. 7481, pp. 150–155. Springer Berlin / Heidelberg (2012)
83. Shi, Z., Dongarra, J.: Scheduling workflow applications on processors with different capabilities. Future Generation Comp. Syst. **22**(6), 665–675 (2006)
84. Singh, G., Kesselman, C., Deelman, E.: Optimizing grid-based workflow execution. Journal of Grid Computing **3**(3-4), 201–219 (2005). DOI 10.1007/s10723-005-9011-7. URL `http://dx.doi.org/10.1007/s10723-005-9011-7`
85. Sloot, P.M.A., Tirado-Ramos, A., Altintas, I., Bubak, M., Boucher, C.A.: From molecule to man: Decision support in individualized e-health. IEEE Computer **39**(11), 40–46 (2006)
86. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M.: Workflows for e-Science: Scientific Workflows for Grids. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
87. Treiber, M., Truong, H.L., Dustdar, S.: Semf - service evolution management framework. In: EUROMICRO-SEAA, pp. 329–336. IEEE (2008)

88. Truong, H.L., Brunner, P., Nae, V., Fahringer, T.: Dipas: A distributed performance analysis service for grid service-based workflows. Future Generation Comp. Syst. **25**(4), 385–398 (2009)
89. Truong, H.L., Comerio, M., Maurino, A., Dustdar, S., Paoli, F.D., Panziera, L.: On identifying and reducing irrelevant information in service composition and execution. In: L. Chen, P. Triantafillou, T. Suel (eds.) WISE, *Lecture Notes in Computer Science*, vol. 6488, pp. 52–66. Springer (2010)
90. Truong, H.L., Comerio, M., Paoli, F.D., Gangadharan, G., Dustdar, S.: Data contracts for cloud-based data marketplaces. International Journal of Computational Science and Engineering **7**(4) (2012)
91. Truong, H.L., Dustdar, S.: On analyzing and specifying concerns for data as a service. In: M. Kirchberg, P.C.K. Hung, B. Carminati, C.H. Chi, R. Kanagasabai, E.D. Valle, K.C. Lan, L.J. Chen (eds.) APSCC, pp. 87–94. IEEE (2009)
92. Truong, H.L., Dustdar, S.: Online interaction analysis framework for ad-hoc collaborative processes in soa-based environments. T. Petri Nets and Other Models of Concurrency **2**, 260–277 (2009)
93. Truong, H.L., Dustdar, S.: Composable cost estimation and monitoring for computational applications in cloud computing environments. Procedia CS **1**(1), 2175–2184 (2010)
94. Truong, H.L., Dustdar, S.: On evaluating and publishing data concerns for data as a service. In: APSCC, pp. 363–370. IEEE Computer Society (2010)
95. Truong, H.L., Dustdar, S.: Cloud computing for small research groups in computational science and engineering: current status and outlook. Computing **91**(1), 75–91 (2011)
96. Truong, H.L., Dustdar, S., Baggio, D., Corlosquet, S., Dorn, C., Giuliani, G., Gombotz, R., Hong, Y., Kendal, P., Melchiorre, C., Moretzky, S., Peray, S., Polleres, A., Reiff-Marganiec, S., Schall, D., Stringa, S., Tilly, M., Yu, H.: incontext: A pervasive and collaborative working environment for emerging team forms. In: SAINT, pp. 118–125. IEEE Computer Society (2008)
97. Truong, H.L., Dustdar, S., Bhattacharya, K.: Programming hybrid services in the cloud. In: 10th International Conference on Service-oriented Computing (I CSOC 2012). Shanghai, China (2012)
98. Truong, H.L., Dustdar, S., Fahringer, T.: Performance metrics and ontologies for grid workflows. Future Generation Comp. Syst. **23**(6), 760–772 (2007)
99. Truong, H.L., Dustdar, S., Götze, J., Fleuren, T., Müller, P., Tbahriti, S.E., Mrissa, M., Ghedira, C.: Exchanging data agreements in the daas model. In: Park et al. [72], pp. 153–160
100. Truong, H.L., Dustdar, S., Maurino, A., Comerio, M.: Context, quality and relevance: Dependencies and impacts on restful web services design. In: F. Daniel, F.M. Facca (eds.) ICWE Workshops, *Lecture Notes in Computer Science*, vol. 6385, pp. 347–359. Springer (2010)
101. Truong, H.L., Gangadharan, G.R., Comerio, M., Dâ??Andrea, V., De Paoli, F., Dustdar, S.: Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions, chap. Reconciliation of Contractual Concerns of Web Services, pp. 298–321. Hershey, PA, USA. IGI Global (2012). URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-61350-432-1.ch013`
102. Truong, H.L., Gangadharan, G.R., Comerio, M., Dustdar, S., Paoli, F.D.: On analyzing and developing data contracts in cloud-based data marketplaces. In: Park et al. [72], pp. 174–181
103. Truong, H.L., Pham, T.V., Thoai, N., Dustdar, S.: Cloud Computing for Teaching and Learning: Strategies for Design and Implementation, chap. Cloud Computing for Education and Research in Developing Countries, pp. 64–80. Hershey, PA, USA. IGI Global (2012). URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-0957-0.ch005`

104. Truong, H.L., Samborski, R., Fahringer, T.: Towards a framework for monitoring and analyzing qos metrics of grid services. In: e-Science [5], p. 65
105. Voigt, S., Kemper, T., Riedlinger, T., Kiefl, R., Scholte, K., Mehl, H.: Satellite image analysis for disaster and crisis-management support. IEEE T. Geoscience and Remote Sensing **45**(6-1), 1520–1528 (2007). URL `http://dblp.uni-trier.de/db/journals/tgrs/tgrs45.html#VoigtKRKSM07`
106. Vu, Q.H., Pham, T.V., Truong, H.L., Dustdar, S., Asal, R.: DEMODS: A Description Model for Data-as-a-Service. In: The 26th IEEE International Conference on Advanced Information Networking and Applications (AINA-2012). IEEE Computer Society Press (2012)
107. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A qos-aware selection model for semantic web services. In: A. Dan, W. Lamersdorf (eds.) ICSOC, *Lecture Notes in Computer Science*, vol. 4294, pp. 390–401. Springer (2006)
108. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. Journal of Grid Computing **3**(3-4), 171–200 (2005). DOI 10.1007/s10723-005-9010-8. URL `http://dx.doi.org/10.1007/s10723-005-9010-8`
109. Yu, J., Buyya, R., Tham, C.K.: Cost-based scheduling of scientific workflow application on utility grids. In: e-Science [4], pp. 140–147
110. Zdun, U., Hentrich, C., van der Aalst, W.M.P.: A survey of patterns for service-oriented architectures. IJIPT **1**(3), 132–143 (2006)