

FlexMash 2.0 – Flexible Modeling and Execution of Data Mashups

Pascal Hirmer^(✉) and Michael Behringer

Institute of Parallel and Distributed Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{pascal.hirmer,michael.behringer}@ipvs.uni-stuttgart.de

Abstract. In recent years, the amount of data highly increases through cheap hardware, fast network technology, and the increasing digitization within most domains. The data produced is oftentimes heterogeneous, dynamic and originates from many highly distributed data sources. Deriving information and, as a consequence, knowledge from this data can lead to a higher effectiveness for problem solving and thus higher profits for companies. However, this is a great challenge – oftentimes referred to as Big Data problem. The data mashup tool FlexMash, developed at the University of Stuttgart, tackles this challenge by offering a means for integration and processing of heterogeneous, dynamic data sources. By doing so, FlexMash focuses on (i) an easy means to model data integration and processing scenarios by domain-experts based on the Pipes and Filters pattern, (ii) a flexible execution based on the user’s non-functional requirements, and (iii) high extensibility to enable a generic approach. A first version of this tool was presented during the ICWE Rapid Mashup Challenge 2015. In this article, we present the new version FlexMash 2.0, which introduces new features such as cloud-based execution and human interaction during runtime. These concepts have been presented during the ICWE Rapid Mashup Challenge 2016.

Keywords: ICWE Rapid Mashup Challenge 2016 · FlexMash · Data processing and integration · Pipes and Filters

1 Context and Goals

Today, data becomes more and more important throughout all domains. Especially the integration and processing of a large amount of distributed data sources can lead to valuable knowledge, which e.g., enables higher profits for companies due to an increased efficiency and effectiveness for problem solving. Oftentimes, this valuable knowledge can only be derived based on a large number of data sources, which is a great challenge. A common way to cope with this challenge are Extract-Transform-Load (ETL) processes. However, traditional ETL processes have several shortcomings: (i) they are very complex and require a lot of effort for creation, (ii) they require deep technical knowledge about the data, algorithms and technology, and (iii) they are oftentimes executed in a static execution

environment, which can only cope with specific requirements. In our previous work [7,9] we introduced FlexMash, a data mashup tool that copes with these issues by (i) enabling a fast creation of data integration and processing scenarios without the definition of complex ETL processes, by (ii) domain-specific modeling based on the Pipes and Filters patterns, which enables usage by users without deep technical knowledge, and by (iii) a flexible execution based on the non-functional requirements of the users (such as security, robustness, efficiency), which leads to a tailor-made execution for each user. Furthermore, FlexMash enables an ad-hoc and explorative approach for data processing. Through easy adaptations of the abstracted model, desired results can be achieved in a step-wise manner. For example, new data sources can be added easily to improve the results without any programming effort.

In this article, we present the new version FlexMash 2.0, which offers enhanced features to further improve the solution provided by the first version of FlexMash. Among improvements regarding the usability and the frontend as well as efficiency improvements of the backend, we introduce two new concepts, which are enhancing FlexMash 2.0: (i) automated deployment of data mashups in distributed cloud computing environments, and (ii) extending data mashups with human interaction during runtime. We will describe these concepts in this article. Furthermore, we will describe the preparation and the demonstration of the FlexMash 2.0 tool during the ICWE Rapid Mashup Challenge. By doing so, we will describe the scenario we prepared for the demo and also challenges we encountered during preparation.

The remainder of this article is structured as follows: In Sect. 2, we describe basic concepts that are necessary to comprehend the approach of this article. Section 3 introduces the main contribution of this article by introducing FlexMash 2.0 – flexible execution and modeling of data mashups. After that, Sects. 4, 5 and 6 describe details regarding the tool demonstration at the Rapid Mashup Challenge 2016, including the tool’s features, its level of maturity, and the demo scenario that was presented. In Sect. 7, we describe related work. Finally, Sect. 8 gives a summary and an outlook on future work.

2 Basic Concepts

This section introduces important concepts that serve as foundation for this article.

2.1 Data Mashup Modeling and Processing Based on Pipes and Filters

The design pattern *Pipes and Filters* [13] is well-established for building modular, highly extensible applications and inspired many application and mashup modeling approaches. The basic idea of this pattern is to create software components (the filters), also referred to as services or micro-services, that offer

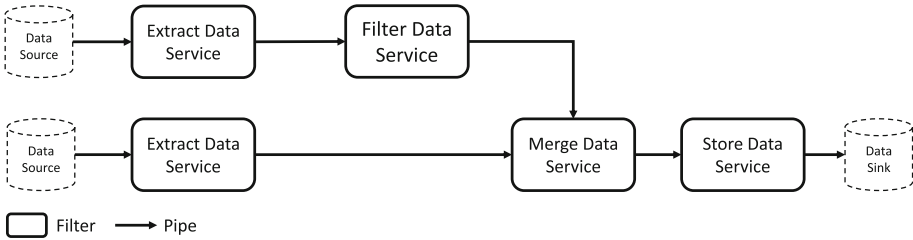


Fig. 1. Basic example of Pipes and Filters based data processing

uniform interfaces (e.g., REST) and are based on a uniform data exchange format. This enables an easy interconnection of these services through so called pipes – the connections between the filters. An example for a Pipes and Filter based data processing approach is depicted in Fig. 1.

The idea of Pipes and Filters has been adopted in many data mashup approaches. In FlexMash, we also build on the concepts of the Pipes and Filters pattern both on the modeling level and on the execution level. This means, the modeler of the data mashup first creates a graphical model such as the one depicted in Fig. 1, which is called *Mashup Plan* in the context of FlexMash. After that, this model is transformed into an executable format such as, e.g., a workflow model. The functionality of the filters is provided by services that are called in the order as defined by the data mashup model. Due to the fact that each service has uniform interfaces and works based on the same data exchange format, we can easily interconnect them arbitrarily.

2.2 TOSCA

The OASIS standard Topology and Orchestration Specification for Cloud Applications (TOSCA) provides a means for automated application provisioning in cloud computing environments and for their management. To enable this means, the topology of an application has to be provided in a model called Service Template. This model contains specific information about all the components of an application ranging from infrastructure to platforms to software components. As a consequence, TOSCA unifies the cloud computing paradigms infrastructure as a service, platform as a service and software as a service. Each component is represented in the topology as so called Node Template, the connections between components are represented by so called Relationship Templates. Node and Relationship Templates are typed by Node Types and Relationship Types, respectively. Furthermore, these templates can be attached with a list of properties and so called Implementation Artifacts, which contain scripts or binaries to provision and manage the component. The topology model can then be used for application provisioning using a TOSCA runtime environment. Though, in this case, the runtime has to work in a declarative manner, which means that it contains explicit knowledge about the modeled components and can set them up

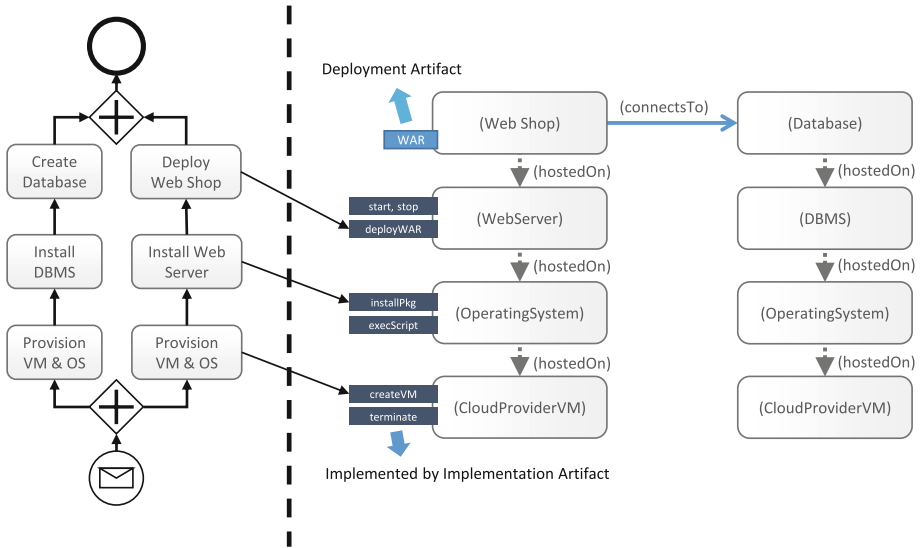


Fig. 2. Example of a TOSCA Build Plan (left) and a TOSCA application topology (right)

in the right order solely based on the topology model. However, most TOSCA runtimes do not support such a declarative approach, and thus are more generic, i.e., they are able to provision and manage arbitrary applications not only the ones known to the runtime. This is referred to as an imperative approach. To enable this, an additional artifact has to be provided with the topology. This artifact is called the Build Plan, oftentimes also referred to as Provisioning Plan. The Plan specifies exactly how and in which order the components have to be set up. For example, an application server container has to be set up first before a web application can be deployed into it. An example for a TOSCA topology and a corresponding provisioning plan is depicted in Fig. 2. In this example, a web shop application connected to a database is modeled and can be provisioned automatically by executing the Build Plan. In this article, we use the concepts of TOSCA to automatically deploy data mashups in distributed cloud computing environments to enable scalability, availability and saving costs. By doing so, we use a combination of an imperative and declarative approach, which is enabled through the TOSCA Build Plan generator as described in [4]. Our prototype shown in the demo of the Rapid Mashup Challenge is implemented based on the open-source TOSCA runtime environment OpenTOSCA.

The open source ecosystem *OpenTOSCA* [3] includes a graphical modeling tool called *Winery* [12] and a plan-based provisioning and management runtime environment, which can be used to provision and manage TOSCA applications fully automatically. The interested reader is referred to the official OASIS TOSCA specification [14], TOSCA Primer [15], or Binz et al. [2].

3 FlexMash 2.0 – Flexible Execution and Modeling of Data Mashups

In this section, we describe the main concepts of FlexMash and we show how they have been extended in contrast to previous work.

The FlexMash approach is applied through the method depicted in Fig. 3. In the first step of this method, a domain-user without any extensive programming skills models a so called Mashup Plan, a Pipes and Filters based flow model as described in Sect. 2.1 and depicted in Fig. 4. This model contains two kinds of nodes: (i) Data Source Description (DSD) nodes describing the data sources that hold the data, and (ii) Data Processing Description (DPD) nodes that describe data operations for, e.g., filtering, aggregation, data analytics, visualization or storage. As described in Sect. 2.1, these nodes can be interconnected arbitrarily due to the fact that they have the same interfaces and the same data interchange format as defined by the Pipes and Filters pattern. The model depicted in Fig. 4

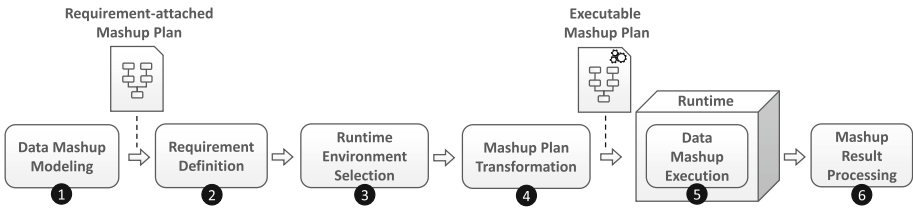


Fig. 3. The FlexMash method (based on [8])

FlexMash Builder

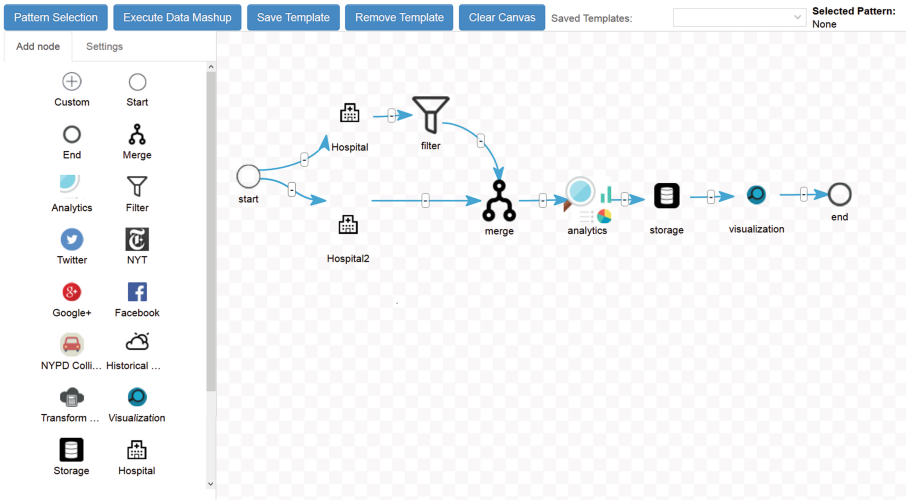
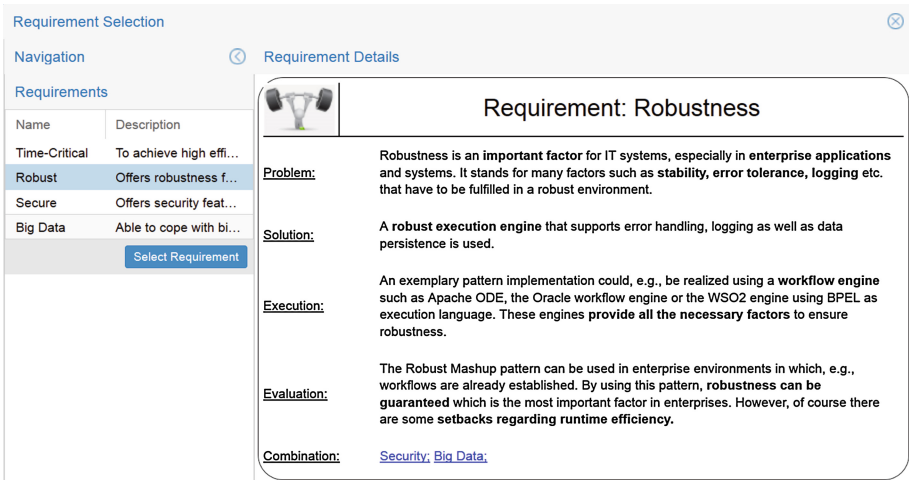


Fig. 4. Data Mashup modeling in FlexMash

gives an example for such a Mashup Plan. In this scenario, data from two different hospitals are being integrated and analytics is conducted based on the result. We assume that these hospitals store similar data, however, there are differences, e.g., one of the hospitals contains additional information that are not important for the integrated result. This information should be removed through a filter before merging. To realize this scenario, this model contains two Data Source Descriptions for the hospitals, as well as five Data Processing Descriptions for filtering, merging, analytics, storage, and visualization. Through interconnection of these nodes, this and many other scenarios can be realized using the FlexMash approach. Note that all technical details, e.g., the concrete data structure of the hospitals are abstracted, the modeler only needs to know which data is stored and not how it is stored.

After modeling, in step 2 of this method, the modeler selects non-functional requirements from the *requirement catalog* depicted in Fig. 5. This catalog contains a textual description of supported non-functional requirements containing a problem description, a provided solution, how the requirement can be fulfilled, an evaluation, and information about how the requirement can be combined with others. These descriptions are kept as free as possible from technical details so domain users can understand them and select them appropriately. Examples for non-functional requirements are robustness, security, or efficiency. Furthermore, a means is provided to combine several requirements with each other. However, some combinations have to be restricted such as robustness and efficiency. The requirements will be used to provide a tailor-made execution for each modeler that fits their specific use cases and scenarios. Note that the requirement catalog currently serves as informative service and does not provide any selection suggestions based on the modeled Mashup Plan. This is part of our future work.



The screenshot shows a web interface titled "Requirement Selection". On the left, there is a "Requirements" table with columns "Name" and "Description". The "Robust" requirement is selected. Below the table is a "Select Requirement" button. On the right, the "Requirement Details" panel is open, showing the details for "Requirement: Robustness".

Name	Description
Time-Critical	To achieve high effi...
Robust	Offers robustness f...
Secure	Offers security feat...
Big Data	Able to cope with bi...

Requirement: Robustness

Problem: Robustness is an important factor for IT systems, especially in enterprise applications and systems. It stands for many factors such as stability, error tolerance, logging etc. that have to be fulfilled in a robust environment.

Solution: A robust execution engine that supports error handling, logging as well as data persistence is used.

Execution: An exemplary pattern implementation could, e.g., be realized using a workflow engine such as Apache ODE, the Oracle workflow engine or the WSO2 engine using BPEL as execution language. These engines provide all the necessary factors to ensure robustness.

Evaluation: The Robust Mashup pattern can be used in enterprise environments in which, e.g., workflows are already established. By using this pattern, robustness can be guaranteed which is the most important factor in enterprises. However, of course there are some setbacks regarding runtime efficiency.

Combination: [Security](#); [Big Data](#).

Fig. 5. Selection of non-functional requirements in a requirement catalog

In step 3, a runtime environment is selected suitable for the requirements defined in step 2. This means that software components are put together in a modular manner depending on the combination of requirements. For example, a robust execution would also require an execution engine that provides features such as error handling or rollbacks. These software components are retrieved by the graph algorithm described in [7]. This algorithm returns a list of software components that are suitable to fulfill the selected non-functional requirements. This list serves as basis for the mashup execution. In previous work, we assume that all possible software components to run the data mashup are already set up. In this article, we show how these components can be set up on-demand and fully automatically (cf. Sect. 3.2). Common engines used for mashup execution are workflow engines that, e.g., execute BPEL workflows to invoke services that process and integrate the data.

In step 4, the non-executable Mashup Plan that was modeled in step 1 is transformed into an executable representation appropriate for the selected runtime environment (e.g., into a BPEL workflow). To realize this, several mappings are provided that transform the abstract generic Mashup Plans into concrete, executable models. The data operations and data sources to be executed by the mashup are encapsulated into services (e.g., Java web services) that are then being invoked by the executable model.

In step 5, the transformed executable representation of the Mashup Plan is executed in the appropriate engine. This engine invokes services to extract and process data in the order defined by the Mashup Plan.

Finally, in step 6, the result can be used for visualization in dashboards, analysis or further value-adding scenarios.

The presented method has been enhanced and improved with additional features that have been presented the first time at the ICWE Rapid Mashup Challenge 2016. Those are (i) the concept of sub-flows on the modeling level, (ii) the fully automated on-demand provisioning of data mashup execution components in cloud computing environments (cf. step 3 in Fig. 3), and (iii) the enabling of human interaction during runtime of data mashups. Those concepts are described in the following.

3.1 Subflows for Data Mashup Modeling

The first new concept we present are subflows. Subflows are Mashup Plans that can be used for modeling within other Mashup Plans. By doing so, all details of the subflow are hidden into a single node. This makes modeling a lot easier, and, furthermore, enhances the reusability of Mashup Plans. Experts could define often recurring patterns and model them as subflows. For example, the extraction and filtering of data from a SQL database is a very common pattern that is modeled many times in different Mashup Plans. As a consequence, it makes sense to provide this Mashup Plan as a pattern to be reused by others. Furthermore, we enable a conservation of knowledge. This means that experienced users can model recurring scenarios and can provide them, e.g. to their colleagues, through subflow nodes. This can save a lot of effort and can further reduce necessary expertise when modeling the Mashup Plan. An example for a subflow is depicted in Fig. 6.

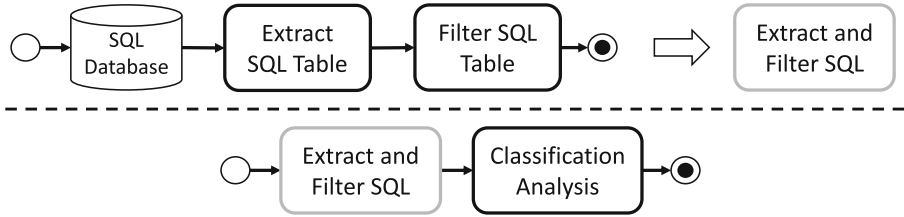


Fig. 6. Top: transformation from a Mashup Plan into a subflow node. Bottom: exemplary usage of a subflow node

3.2 On-Demand Provisioning of Data Mashup Execution Components

As depicted in Fig. 3, the execution environment is selected in step 3 of the introduced method based on the non-functional requirements of the users. In previous work, we assumed that all components that could be used for mashup execution are up and running. However, this leads to high costs due to an extensive use of resources. Especially some components are rarely used so it does not make sense to keep them running at all times. We further extend the FlexMash method to provide a means for automated, on-demand setup and execution of software components based on the Topology Orchestration Specification for Cloud Applications (TOSCA) that was introduced in Sect. 2.2.

After step 3 of our method, the selection of a suitable runtime environment for the data mashup execution, we receive a list of necessary components. These components should be set up automatically and only when they are needed for mashup execution. Based on the information provided on the list, a TOSCA topology template is created automatically using the topology completion algorithm provided in [6]. Furthermore, through the plan generator extension provided by [4], we are able to generate the TOSCA build plan based on the completed topology. This topology is then used for automated setup of data mashup execution components such as workflow engines, databases for storing intermediate results, and services for executing data operations. Detailed information about this step is provided in [8].

3.3 Extending Data Mashups with Interactivity

(Data) Mashups are a proven approach for special solutions by providing a great advantage for a small number of users, whose demand is not considered important enough for a prefabricated application [5]. The user is therefore able to build a customized solution based on his requirements with reduced technical knowledge required. Unfortunately, most applications still require an extensive knowledge of the processes and parameters used and for this reason are not adaptable for the ordinary domain experts to use.

It is undisputed that interactive elements provide several advantages, such as a deeper understanding of the characteristics of the data to be analyzed and as a

FlexMash Builder

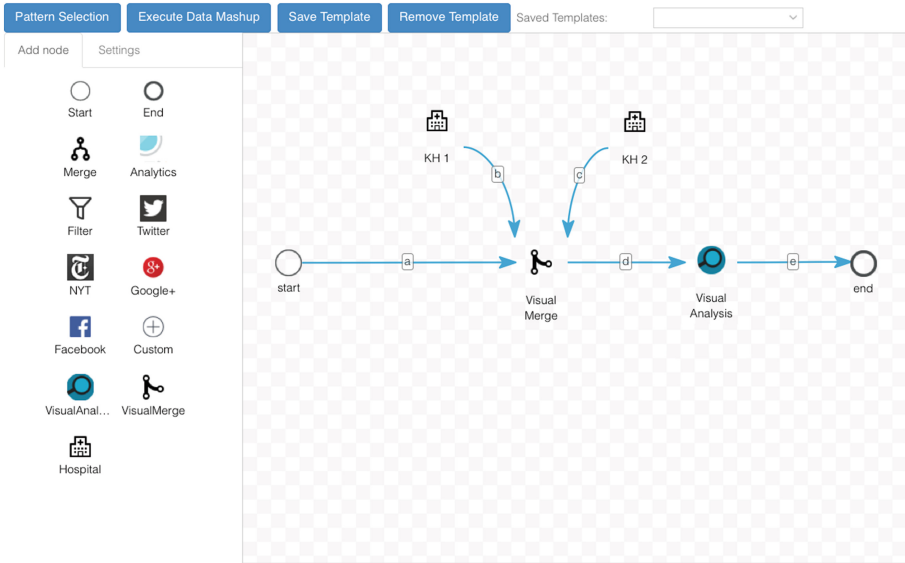


Fig. 7. Use case: health analytics

consequence, better and more accurate results as shown in different user studies (e.g., Savikhin et al. [16]). Another major advantage is the possibility to make use of the implicit domain knowledge of a user during the analysis process [18]. These advantages are, however, paid for with a partial loss of objectivity and velocity. The former can cause a problem if the user wants to prove a particular hypothesis and ignores obvious patterns in the data because they do not meet the objective [17]. The latter is unavoidable when involving the user, as this always requires more time than an automated process. Consequently, it must be considered for each use case whether the involvement of the user offers required advantages, however, in many cases an advantage is expectable. Nonetheless most applications are, regarding interactivity, limited to the modeling of workflows and specifying different parameters through a graphical user interface.

In the following section, we introduce a prototypical Data Processing Description that can be used for modeling in Mashup Plans, which allows interactive merging of different data sources under full control of the user – the Visual Merge Node. This concept is based on the work of Kandell et al. [10], but with a strong focus on the specific requirements of Data Mashups and the associated integration of multiple data sources. For evaluation purposes, we used a test data set with generated values of a fictional health care scenario. The modeled use case is depicted in Fig. 7. This scenario describes the analysis of two health data sets from two different hospitals, but is also applicable for a higher number of data sources. This data sources have been previously added to FlexMash by a technical expert and placed in the modeled scenario by a domain expert through drag

Visual Merge Node

Dashboard (DataSource 1)

Data Source 1
Data Source 2

Hide Schemas

DataSource 1: gender :: name :: email :: Treatment :: diagnosis :: country
DataSource 2: id :: gender_full :: first_name :: last_name :: email :: medicine :: diagnosis :: region

Data Quality (56.68)

#	gender	name ▲	email	Treatment	diagnosis	country
1	M	Aaron Gonzalez	agonzalez49@yale.edu	Salbutamol	Chronic obstructive pulmonary disease (COPD)	France
2	M	Aaron Hamilton	ahamiltonbc@hsa.gov	Prednisolon	Chronic obstructive pulmonary disease (COPD)	Japan
3	M	Aaron Warren	awarren9@sharesale.com	Theophyllin	Chronic obstructive pulmonary disease (COPD)	United States
4	M	Adam Arnold	aarnoldcg@unicef.org	Prednisolon	Chronic obstructive pulmonary disease (COPD)	China
5	M	Adam Hall	ahallar@harvard.edu	Theophyllin	Chronic obstructive pulmonary disease (COPD)	China
6	M	Adam Robinson	arobinsona7@cornell.edu	Acetylcystein	Gastroesophageal reflux disease	Russia
7	M	Adam Shaw	ashaw38@mapy.cz	Theophyllin	Gastroesophageal reflux disease	Russia
8	M	Alan Martinez		Acetylcystein	Congestive heart failure	China
9	M	Alan Morgan		Prednisolon	Gastroesophageal reflux disease	China
10	M	Albert James	ajamesb9@furl.net	Prednisolon	Congestive heart failure	Japan
11	F	Alice Fisher	afisher5a@quantcast.com	Prednisolon	Vocal cord dysfunction	United States
12	F	Alice Ford	aford9x@networksolutions.com	Prednisolon	Chronic obstructive pulmonary disease (COPD)	Russia

Fig. 8. Visual merge node

and drop, describing the input data for our new Visual Merge Node, which is described in detail later in this section. Furthermore there is a placeholder called Visual Analysis Node depicted, which is not described in detail in this article. Both hospital sources offering a JSON-file describing patient data like name, gender, illness and optionally a diagnosis and different contact information. The exercise for the domain expert is to convert the initial two heterogeneous data sources into a combined homogeneous one by controlling every step of the data integration over an intuitive user interface.

The graphical user interface is depicted in Fig 8 and consists of different components:

Data Table – The main component of the user interface is a central data table (a) used for displaying the contents of the incoming JSON file.

Data Source-Switch – Due to the unknown number of data sources to be coped with, we implemented a menu for switching the actively displayed data source (b).

Schema Integration – Incoming data could be of very different characteristic, therefore we provide an optional menu showing the differences in schemata (c), which allows the user to evaluate attribute accordance between data sources.

Proceed Workflow – The user can continue execution (d) of the modeled workflow, once the objective quality is sufficient. Furthermore, the continuation of the processing is possible until an adequate subjective quality is achieved. The current record is then converted to the standard data exchange format for FlexMash.

Sorting – Naturally, the user interface offers simple sorting capabilities (e) allowing different views of the record.

Quality Meter – A quality measurement meter is visualized on top of the table (f) in order to help the user to evaluate the current state of integration with a single view. This quality measure is by now based on schema coincidence and empty values.

The presented prototype node provides several advantages over the previous merge-functionality. Previously, it was necessary to specify the exact attribute for a join operation without the possibility for further optimization of the records. With the new Visual Merge Node there is no need to know anything about join operations, it is an intuitive and iterative process under full control of the user. Furthermore, not all semantic errors could be automatically identified [11] and human interaction is needed to eliminate them. For this reason an improved data quality is expected if the user is involved in the analysis process.

As conclusion, it can be stated that the integration of domain experts during mashup execution has many advantages, for example the inclusion of implicit background knowledge, the recognition and correction of additional semantic errors, an increase of the understanding of the data set and a better selection of data for analysis. There is a great potential by integrating the domain expert not only in the final analysis but also in each step of the Mashup Plan. Nevertheless it must be noted that the user can create both, better and worse results, which is why further investigation in this regard is needed.

4 FlexMash 2.0 – Level of Maturity

The FlexMash tool, although being a research project developed by a small group supported by student thesis¹, has reached a high level of maturity throughout its 2 year long development. The first stable version of FlexMash was presented during the ICWE Rapid Mashup Challenge 2015 in Rotterdam, the second version, which is described in this article, was presented in the 2016 edition of this challenge in Lugano. The FlexMash implementation is provided as an open source project on GitHub¹ and is licensed under the Apache 2.0 license. We hope to increase the community of developers to even increase the level of maturity in the future.

However, although the maturity of FlexMash could have been increased with the latest features, there are still some issues and many features that are planned for the future. For example, enhanced interactivity, an extended set of data sources and data operations, the support of further non-functional requirements, and, as a consequence further execution components.

5 FlexMash 2.0 – Feature Checklist

In this section, the features of the current state of FlexMash’s implementation are described based on the ICWE Rapid Mashup Challenge checklist, which contains

¹ <https://github.com/hirmerpl/FlexMash>.

information about important properties and design choices of mashup tools to enable their categorization. The feature checklist is based on related work and is subdivided into two parts: (i) an overall *mashup feature checklist* as described in [5] (Chap. 6), and (ii) a *mashup tool feature checklist* as described in [1]. The detailed information about the single entries are provided in these references.

- Mashup Feature Checklist
 - **Mashup Type:** Data mashups
 - **Component Type:** Data components
 - **Runtime Location:** Both client and server
 - **Integration Logic:** Orchestrated integration
 - **Instantiation Lifecycle:** Stateless
- Mashup Tool Feature Checklist
 - **Targeted End-User:** Non programmers
 - **Automation Degree:** Semi-automation
 - **Liveness Level:** Level 3 – Automatic compilation and deployment, requires re-initialization
 - **Interaction Technique:** Visual language (Iconic)
 - **Online User Community:** None (yet)

6 ICWE Rapid Mashup Challenge

This section describes all aspects of the Rapid Mashup Challenge (RMC) 2016 in Lugano, ranging from the scenario we presented, the demo flow, the preparations for this challenge, and discussions and findings we collected.

6.1 Scenario and Demo Flow

This section introduces the scenario we presented during the Rapid Mashup Challenge 2016, which focuses on data integration and analytics. This scenario analyzes if weather conditions influence the amount and types of traffic accidents. Basis for the analyses are the New York City Police Department Motor Vehicle Collisions² data set and a data set extracted from the Open Weather API³. Consequently, we focus our analyses on New York City exclusively, however, due to the generic nature of the analysis, further data sets are also easy applicable.

The NYPD Motor Vehicle Collisions data set contains all documented accidents from 2012 until today. Currently there are 800,000 entries in the data set. Each entry contains the following information: (i) date and time of the accident, (ii) location information containing street name, zip code, borough as well as latitude and longitude, (iii) whether and how many persons (separated in drivers, cyclists, motorists, pedestrians) have been injured or killed, (iv) the cause for the accident (if known), and (v) the amount and types of vehicles (e.g., bus, taxi,

² <https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95>.

³ <http://openweathermap.org/api>.

FlexMash Builder

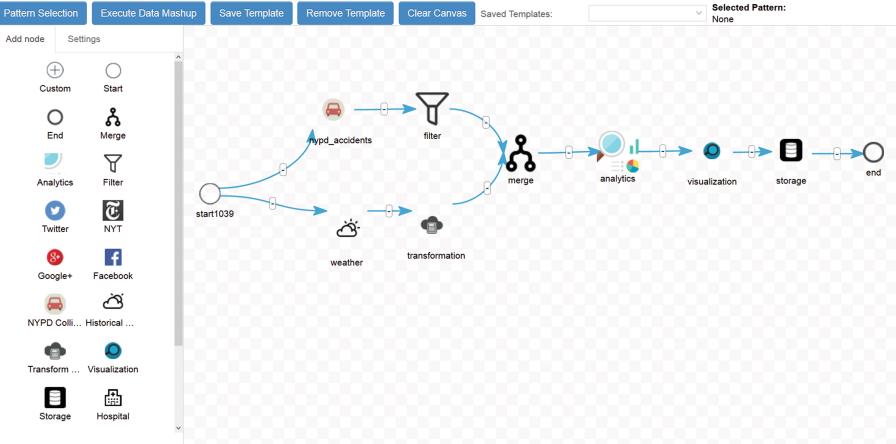


Fig. 9. Mashup Plan to run the analyses of the scenario shown at the RMC

bicycle, passenger vehicle, and many more) involved. The data from the Open Weather API for New York City provides the following information: (i) date, (ii) temperature, (iii) amount of precipitation, (iv) amount of snowfall, and (v) wind speed. In our scenario, we analyze the following questions based on the given data:

1. Does the temperature influence the amount of occurring accidents?
2. Does the precipitation influence the amount of occurring accidents?
3. Does the wind speed influence the amount of occurring accidents?
4. Are certain vehicle types influenced more by weather conditions (e.g., buses)?

To run these analyses, we use our data mashup tool FlexMash. The basis for this is the Pipes and Filters based Mashup Plan depicted in Fig. 9. To model these analyses, we have to create the depicted model step by step using the FlexMash modeling tool. First, a start node is inserted into the canvas. Start nodes mainly serve overview purposes and are not a necessary means to model these Mashup Plan. The entry points can also be determined automatically through topological sorting. Second, we insert the data sources, i.e., the NYPD Motor Vehicle Collisions data set provided through a REST interface, and the Open Weather Data provided by a CSV file. The data sources are abstracted through two Data Source Descriptions that extract the data as a whole from these two data sets. These Data Source Descriptions are also inserted into the canvas and are connected to the start node. Third, the NYPD accidents DSD is connected to a filter node (DPD), which is necessary to realize the fourth analysis, which investigates different types of vehicles. Through the settings of this filter node, the type of vehicle can be chosen accordingly. Furthermore, we connect the weather data DPD to a transformation node, which can be configured to transform the temperature provided in Fahrenheit to Celcius. These nodes are inserted and

FlexMash Result View Flow #1

9 Accidents occurred at a temperature between 0 degree Celcius and 5 degree Celcius
21 Accidents occurred at a temperature between 5 degree Celcius and 10 degree Celcius
48 Accidents occurred at a temperature between 10 degree Celcius and 15 degree Celcius
0 Accidents occurred at a temperature between 15 degree Celcius and 20 degree Celcius
0 Accidents occurred at a temperature above 20 degree Celcius

Fig. 10. Analytics result of the FlexMash tool shown at the RMC

connected as depicted in Fig. 9. Next, the data is merged to receive a common data set. The merging of the data is done in a straight-forward manner by a join over the date and time columns. This can be done easily because there is no overlap in the data, and therefore there are no conflicts possible. Furthermore, the data of these columns are structured in a similar manner. To model this merge operation, we insert a merge node DPD into the canvas and connect it accordingly. Based on the merged data, we conduct a simple analysis that groups the accidents according to the weather conditions. Which weather condition is used for grouping, e.g. temperature or wind speed, can be chosen in the settings of the analytics node. After that, we connect the analytics node to a visualization node that depicts the analytics results as HTML (cf., Fig. 10). This HTML text is then stored in a file by connecting it to the storage node. This node can be configured in different ways: data can be stored in files or in different databases. Based on this model, the four different analyses can be executed only by slightly adjusting the settings of the nodes. This especially shows the ad-hoc capabilities of FlexMash and, furthermore, that an explorative behavior is possible due to an easy adaptation and re-execution of the model.

6.2 Challenge Preparations

We had four weeks to prepare for this challenge. To do so, we had to conduct several preparations regarding the FlexMash tool and the presented scenario. First, we did a refactoring of FlexMash's code to enable a more modular, loosely coupled application, which is easier to extend. Next, we provided DSDs and DPDs for the scenarios and we implemented corresponding services that realize the data processing operations. Because of the high extensibility of FlexMash, the preparations did not take a lot of time. Due to the limited complexity of the scenario, three weeks of implementation were sufficient. The implementation of the interactive node has been finished prior to the challenge preparations. In the

last week before the challenge, we primarily executed tests of the implementation and cleaned the code.

6.3 Discussion and Findings

During the preparation for this challenge and the challenge itself, we discovered several findings that are discussed in this section.

Firstly, we confirmed that FlexMash offers high extensibility. Extending the functionality with the analytics functionality needed for this scenario was easy due to the concept of DSDs, DPDs, and the execution of the operations within modular services. However, there were also some challenges while preparing for the challenge. First, the data had to be cleansed so it can be used for reliable analysis, i.e., errors, missing fields etc. had to be deleted first, which was a cumbersome task. Second, it was hard to find the weather data for the location at the specific time range due to the fact that a lot of weather APIs only provide the data for a payment. As a consequence, we did not find weather data for the whole time range of the accident data. Because of that, we could only analyze a subset of the data.

There are still some limitations regarding the implemented scenario although it only serves demonstration purposes: the analyses are implemented in a straight-forward manner, i.e., we only count the occurrences of accidents for a specific range of temperature, precipitation, and so on. Furthermore, the nodes are not very generic, that is, they currently only work for this specific scenario. In the future, we will provide more generic nodes for data extraction and for analytics.

7 Related Work

This section introduces related work and describes a detailed separation of FlexMash with other approaches that have the same goal – efficient processing and integration of heterogeneous data. These are approved ETL (Extract Transform Load) tools such as Pentaho⁴, data analytics tools such as KNIME⁵ or RapidMiner⁶ and information integration tools such as informatica⁷. We use the following criteria to compare the FlexMash approach with related work:

- **Extensibility.** We define an approach as fully *extensible*, if it provides a means to extend the set of supported data sources and data operations.
- **Cloud support.** We define an approach as fully *cloud supported*, if it provides a means to execute the data processing in a virtual cloud environment. More precisely, it has to support automated provisioning, distribution of load, scalability, and availability.

⁴ <http://www.pentaho.com/>.

⁵ <https://www.knime.org/>.

⁶ <https://rapidminer.com/>.

⁷ <https://www.informatica.com>.

Criteria \ Approach	ETL Tools	Data Analytics Tools	Information Integration Tools	FlexMash
Extensibility				
Cloud support				
Graphical modeling / Usability				
Flexible execution				
User interaction during runtime				

No Support
 Low Support
 Medium Support
 High Support
 Full Support

Fig. 11. Classification of the FlexMash approach

- **Usability.** We define an approach as fully *usable*, if it provides an easy modeling of data integration and processing scenarios, preferably graphically and support for users outside the IT area, i.e., users without extensive programming expertise.
- **Flexibility.** We define an approach as fully *flexible*, if it provides adaptations tailor-made for the specific user. For example, by providing a configurable execution and not a fixed, static one.
- **User interactivity during runtime.** We define an approach as fully *interactive*, if the user has the possibility to monitor and control the execution during modeling and during runtime.

For each approach or tool related to FlexMash, we will provide examples to be able to provide a fair comparison (Fig. 11).

7.1 Extensibility

We compared the extensibility of different approaches and we found out, that ETL tools and information integration tools offer a good extensibility. For example, new data sources can be added easily to these platforms to be integrated and processed. However, in regard to the extension of data processing operations in the sense that new algorithms or self-made data operations can be added is not supported by most platforms. More precisely, the concept of *bring your own code* is oftentimes not supported. In contrast, data analytics tools such as Knime and RapidMiner support the extension with new data operations, e.g., new analytics algorithms. These can, for example, be added using the programming language *R*. However, the extension with new data sources on the other hand is not fully supported. As a consequence, related approaches are only extensible to a certain degree, either enabling the integration of new data sources or data operations but not both. Because of that, we classify these approaches as medium extensible.

FlexMash offers a high degree of extensibility through its generic approach. By enabling an easy adding of services to extract data from various data sources

and processing data, FlexMash enables bringing own code into the system. However, a fully generic approach also increases the complexity of the extension of new data sources and operations. As a consequence, we classify FlexMash as highly extensible. In the future, we will provide an easy-to-use interface to extend the set of data sources and data operations.

7.2 Cloud Support

When examining the related approaches, we found out that most of them offer great support of virtualized cloud environments, even providing a means for automated distribution of load and scaling of virtual machines. Especially the platforms Pentaho, informatica, and the IBM analytics services of their cloud platform Bluemix cope with the cloud-based processing of data very efficiently. FlexMash also supports cloud-based execution as well as the distribution of load on different scalable virtual machines as described in Sect. 3.2. Consequently, we classify all approaches as fully cloud supported.

7.3 Usability

Due to the fact that usability cannot be easily measured and requires extensive user studies, we only focus on the aspect whether in our opinion users have to have knowledge about programming and algorithms or if domain users such as business experts can use these tools. Information integration tools focus on the usability by non-programmers, i.e., by domain-experts in enterprises. Due to this fact, the technical details are well hidden from the users so we can classify information integration tools as fully usable. ETL tools and data analytics tools in contrast expect a lot of specific knowledge from their users. Although the source code is mostly hidden from the user, extensive, detailed parameterization has to be provided on the modeling level. This parameterization, however, can only be provided if the algorithms, data sources and data operations are fully understood. Because of that, in our opinion, we classify these tools as lowly usable (for domain-experts). FlexMash offers a full abstraction from technical details by the introduced Mashup Plans that are transformed into executable representations. The parameterization is kept simple and in a way that domain-experts can understand. Furthermore, FlexMash provides means for visual interaction during runtime (cf. Sect. 3.3). As a consequence, FlexMash can be classified as fully usable.

7.4 Flexibility

In our examination of the different tools, we found out that there is a huge gap regarding the flexibility of the execution. More precisely, all available tools only offer one way to execute the integration and processing of data. However, the execution should depend on the use case scenario, and the requirements of the tool's users. Although some tools offer a means to configure the execution, there

is no support for a tailor-made execution specific to the users' needs. FlexMash enables this flexibility by enabling a tailor-made execution of data integration and processing based on non-functional requirements as described in Sect. 3. Therefore, we classify FlexMash as fully flexible.

7.5 Interactivity

Another gap we discovered in the approaches we examined was the interactivity during modeling and execution. Especially interaction during runtime is not supported at all. We close this gap by introducing an approach to extend FlexMash with interactivity during runtime (cf., Sect. 3.3) so that users can easily monitor and control the execution, which is expected to lead to improved results. This will be especially addressed in our future work.

7.6 Conclusion

We found out that although there are many tools that have the same goal as FlexMash, they cannot fulfill all criteria that are in our opinion important. FlexMash can cope with most of these criteria and therefore provides a comprehensive solution. We are aware that commercial products can provide a wider selection of data sources and data operations as well as a more efficient processing through several years of code optimizations. However, we discovered several gaps they cannot cope with. FlexMash mainly serves as proof of concept that these gaps can be closed.

8 Summary and Outlook

In this article, we presented the data mashup tool FlexMash, more precisely, the second version of FlexMash that was presented during the ICWE Rapid Mashup Challenge 2016. FlexMash is a tool that enables flexible execution of data mashups based on the user's non-functional requirements. Furthermore, FlexMash offers domain-specific modeling based on the Pipes and Filters pattern that enables usage by domain-experts, even without extensive programming knowledge. As mentioned in the introduction, FlexMash can serve as a way to deal with the Big Data problem, which is defined through *the three Vs: Variety, Velocity and Volume*. As thoroughly explained throughout this paper, our approach can handle the Variety issue quite well. The Volume issue can also be handled through the high scalability of the approach enabled by cloud-based execution. Each data processing operation can be conducted in different virtual machines that can be scaled independently. Consequently, large data sets can be handled. However, we found out that some execution engines, especially those based on BPEL, cannot cope with transferring large data sets from one service to another. To cope with this issue, we provide caches to store intermediate data. Based on this, only the key to the corresponding entry in the cache is handed from one service to another to cope with the issues of some engines handing over

large data sets. The Velocity issue, meaning that data changes frequently, can be partially handled by FlexMash. Once data changes, the Mashup Plan has to be re-transformed and re-deployed to involve the new data. However, with data sets that change very often, this leads to efficiency issues. Currently, we are working on a solution for that by computing intermediate results during modeling time so the whole Mashup Plan does not have to be re-transformed and re-executed. In addition to the original definition, a wide variety of other *V*'s are existing, most commonly *Veracity* and *Value* whose support by FlexMash is hereinafter described. Our approach to bring in the user into the process allows a deeper and more detailed understanding of the data characteristic and content. For this reason, we can expect the user to be aware of the data quality as well as data trustworthiness, i.e., we fulfill Veracity. In respect to the last *V*, Value, our approach offers more control, steering the analysis process based on the objective of the domain expert and, as a consequence, more satisfying and more accurate results.

For the challenge, we showed the capabilities of FlexMash using a scenario that examines whether traffic accidents correlate with weather conditions. By doing so, we examined the conditions precipitation, snowfall, and temperature. Additionally, we showed how FlexMash can be enhanced with interactivity during runtime. We further described the preparation for this challenge and which findings we could discover by participating. For future work, we will focus on the interaction during mashup runtime because this is an interesting concept and can highly improve the usability. Furthermore, we are working on a new and improved user interface. Our goal is presenting these results during the next Rapid Mashup Challenge 2017.

References

1. Aghaee, S., Nowak, M., Pautasso, C.: Reusable decision space for mashup tool design. In: 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2012), Copenhagen, Denmark, pp. 211–220, June 2012
2. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: TOSCA: portable automated deployment and management of cloud applications. In: Bouguettaya, A., Sheng, Q.Z., Daniel, F. (eds.) *Advanced Web Services*, pp. 527–549. Springer, New York, Januar 2014. http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INBOOK-2014-01&engl=0
3. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA – a runtime for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013*. LNCS, vol. 8274, pp. 692–695. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45005-1_62
4. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining declarative and imperative cloud application provisioning based on TOSCA. In: *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, pp. 87–96. IEEE Computer Society, März 2014. http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2014-21&engl=0

5. Daniel, F., Matera, M.: *Mashups - Concepts Models and Architectures. Data-Centric Systems and Applications*. Springer, Heidelberg (2014)
6. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F.: Automatic topology completion of TOSCA-based cloud applications. In: *Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*. LNI, vol. 232, pp. 247–258. Gesellschaft für Informatik e.V. (GI), Bonn. http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2014-66&engl=0
7. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F.: FlexMash – flexible data mashups based on pattern-based model transformation. In: Daniel, F., Pautasso, C. (eds.) *Rapid Mashup Development Tools*. CCIS, vol. 591, pp. 12–30. Springer, Cham (2016). doi:[10.1007/978-3-319-28727-0_2](https://doi.org/10.1007/978-3-319-28727-0_2).
8. Hirmer, P., Mitschang, B.: TOSCA4Mashups - enhanced method for on-demand data mashup provisioning. In: *Proceedings of the 10th Symposium and Summer School on Service-Oriented Computing* (2016)
9. Hirmer, P., Reimann, P., Wieland, M., Mitschang, B.: Extended techniques for flexible modeling and execution of data mashups. In: *Proceedings of the 4th International Conference on Data Management Technologies and Applications (DATA)*, April 2015
10. Kandel, S., Heer, J., Plaisant, C., Kennedy, J., van Ham, F., Riche, N.H., Weaver, C., Lee, B., Brodbeck, D., Buono, P.: Research directions in data wrangling: visualizations and transformations for usable and credible data. *Inform. Vis.* **10**(4), 271–288. <http://ivi.sagepub.com/lookup/doi/10.1177/1473871611415994>
11. Kemper, H.G., Baars, H., Mehanna, W.: *Business Intelligence - Grundlagen und praktische Anwendungen*. Vieweg+Teubner, Wiesbaden (2010). <http://link.springer.com/10.1007/978-3-8348-9727-5>
12. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – a modeling tool for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) *ICSOC 2013*. LNCS, vol. 8274, pp. 700–704. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-45005-1_64](https://doi.org/10.1007/978-3-642-45005-1_64)
13. Meunier, R.: *The pipes and filters architecture*. In: *Pattern languages of program design* (1995)
14. OASIS: *Topology and Orchestration Specification for Cloud Applications* (2013)
15. OASIS: *TOSCA Primer*. <http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.pdf>
16. Savikhin, A., Maciejewski, R., Ebert, D.S.: *Applied Visual Analytics for Economic Decision-Making* (2008)
17. Shneiderman, B.: Inventing discovery tools: combining information visualization with data mining. *Inform. Vis.* **1**(1), 5–12 (2002). <http://ivi.sagepub.com/content/1/1/5.abstract>
18. Wang, X., Jeong, D.H., Dou, W., Lee, S.W., Ribarsky, W., Chang, R.: Defining and applying knowledge conversion processes to a visual analytics system. *Comput. Graph.* **33**(5), 616–623 (2009)