ICWE RMC 2016: FlexMash 2.0

Pascal Hirmer

Institute of Parallel and Distributed Systems, Universität Stuttgart,
`pascal.hirmer@ipvs.uni-stuttgart.de`
`http://www.ipvs.uni-stuttgart.de`

Abstract. Today, a multitude of highly-connected applications and information systems hold, consume and produce huge amounts of heterogeneous data. In order to conduct, e.g., data analysis, visualizations or other value-adding scenarios, it is necessary to integrate specific, relevant parts of data into a common source. Due to oftentimes changing environments and dynamic requests, this integration has to support ad-hoc and flexible data processing capabilities. Furthermore, an iterative and explorative trial-and-error integration based on different data sources has to be possible. To cope with these challenges, we developed the data mashup tool FlexMash. In 2015, we presented FlexMash at the ICWE Rapid Mashup Challenge in Rotterdam. During this event, we gained important feedback and insights, which inspired us to develop an enhanced version – FlexMash 2.0. The new version of FlexMash was not only improved in regard to robustness and efficiency, but also supports more data sources, data operations and thus a wider range of scenarios FlexMash 2.0 can be applied to. During the RMC 2016 we want to show these enhancements based on a new data mashup scenario.

Keywords: ICWE RMC, Data Mashups, FlexMash, Patterns

1 Goals

The overall goal of the FlexMash [4] approach is creating an easy-to-use data mashup [2] tool that enables domain-users to extract information from different, heterogeneous data sources. For example, FlexMash can be used to efficiently and effectively find information in so called *data lakes* [3]. Flexibility is an important aspect of FlexMash. That is, the way of executing a data mashup should depend on the use case scenario. This enables a tailor-made mashup execution behavior. For example, if the execution should be especially robust, a workflow engine should be used for mashup execution, if the execution should be efficient, a more lightweight execution engine should be used. As a consequence, the execution environment of FlexMash needs to be highly dynamic.

In summary, the main goals of FlexMash are: (i) data mashup modeling by domain-users, and (ii) flexible mashup execution dependent on the use case scenario. How these goals can be achieved is described in the next section.

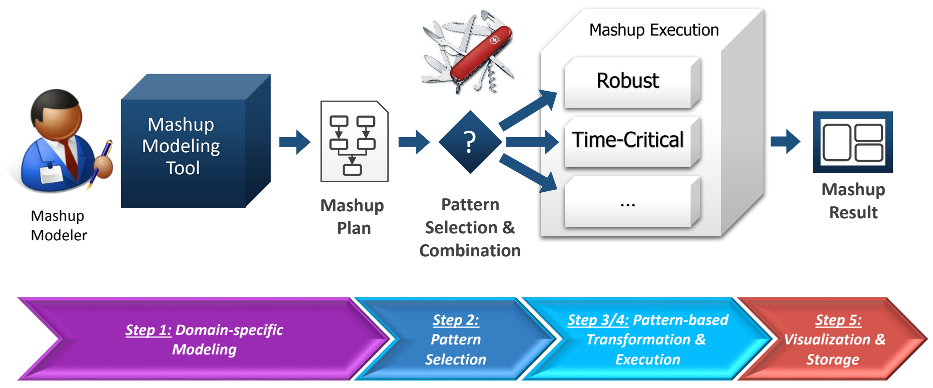
Mashup Modeler
Mashup Modeling Tool
Mashup Plan
?
Pattern Selection & Combination
Mashup Execution
Robust
Time-Critical
...
Mashup Result
Step 1: Domain-specific Modeling
Step 2: Pattern Selection
Step 3/4: Pattern-based Transformation & Execution
Step 5: Visualization & Storage

Fig. 1. The FlexMash Method [4]

2 Proposed Solution

We developed the method depicted in Fig. 1 to achieve the described goals. In the first step, a domain-user models a so called *Mashup Plan*, a graphical, abstract model that contains the data sources and data operations to be used for the mashup. It is important to note that we offer an abstraction of technical details of data sources by so called Data Source Descriptions (DSD) and of data operations by so called Data Processing Descriptions (DPDs), that is, a domain-user only models elements s/he is familiar with. By combining data sources and operations in a *pipes-and-filters* manner, domain users can easily model data mashup scenarios they are interested in without necessary deep technical knowledge. After modeling of the Mashup Plan, the domain-user can attach it with so called *Transformation Patterns*, which can be selected in a *pattern catalog*. Transformation Patterns offer a means to domain-expert to influence the way the data mashup is executed. Those patterns describe non-functional requirements to be considered during mashup execution. For example, we offer patterns such as *Security*, *Robustness* or *Efficiency*. Most patterns can be combined, however, some patterns are not compatible, e.g., the *Robustness* and *Efficiency* patterns. Transformation Patterns can be parameterized to enable a more detailed description. After the Transformation Patterns are selected, a suitable execution environment, e.g. an execution engine, are found using so called *pattern graphs* as described in [4]. In pattern graphs, patterns are hierarchically structured in sub-patterns and in implementations. By traversing the pattern graphs, the most suitable implementation, i.e. mashup execution environment, can be found based on parameter matching. After the suitable execution environment is found, it can be set up dynamically using cloud computing deployment approaches such as TOSCA [5]. The Mashup Plan is then transformed in a suitable execution format, e.g., by connecting workflow fragments. After that, the data mashup is executed as modeled in the Mashup Plan. The results can be used for visualization in dashboards, analytics, or other value-adding scenarios. In addition to the functionality we presented last year at the RMC, we support more DSDs, DPDs, i.e., a wider range of scenarios, dynamic mashup provisioning using TOSCA, and an enhanced, more robust frontend and backend. In addition, we support new *visual analytics* nodes that allow human interaction during mashup execution.

3 Level of Maturity

We presented a first prototype of FlexMash at the ICWE Rapid Mashup Challenge 2015, which we further extended with additionally implemented concepts. The result was a more robust and powerful data mashup tool, which is open source[1] and is, e.g., used within the project SitOPT. Most of the functionality is implemented, however, there are still some limitations. We currently only support a specific, limited amount of data sources and operations, however, we provide a means for an easy adding of new ones. Furthermore, currently, we are still working on a more sophisticated pattern-implementation-matching algorithm based on the introduced pattern graphs. In our prototype, the implementation of this algorithm is realized in a straight-forward manner. Our goal is to enhance our prototype as much as possible until the RMC 2016.

4 Feature Checklist

The following feature checklist is based on [1] and available online[2].

- **Mashup Type:** Data mashups
- **Component Type:** Data components
- **Runtime Location:** Both Client and Server
- **Integration Logic:** Orchestrated integration
- **Instantiation Lifecycle:** Stateless
- **Targeted End-User:** Non Programmers
- **Automation Degree:** Semi-automation
- **Liveness Level:** Level 3
- **Interaction Technique:** Visual Language (Iconic)
- **Online User Community:** None

5 Demo

FlexMash is a web-based data mashup tool based on JavaScript (client) and Java (server) using the modeling framework AlloyUI[3]. The tool is fully hosted on the platform-as-a-service provide IBM Bluemix. Currently, we support as data sources: CSV files, MySQL, PostgreSQL, MongoDB, CouchDB, Text files, RSS Feeds, and Twitter Feeds. As data operations we support: join, aggregate, analytics

[1] https://github.com/hirmerpl/FlexMash
[2] http://challenge.webengineering.org/feature-checklist/
[3] http://www.alloyui.com/

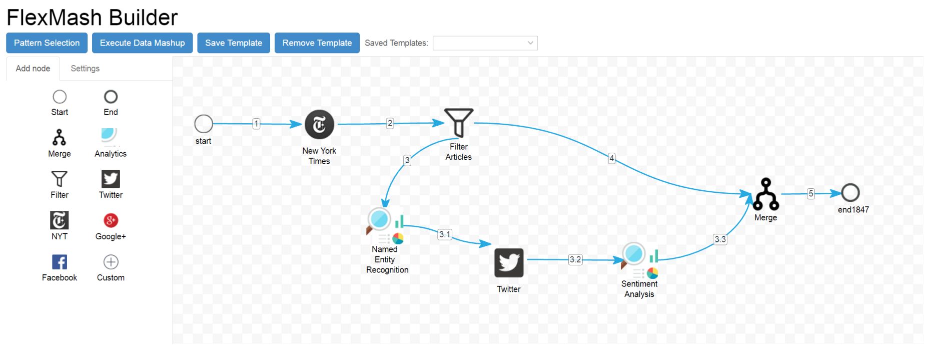

Fig. 2. Screenshot of the scenario presented at the RMC 2015

(e.g., sentiment, clustering, association rules, visual analytics, ...), merge, and filter. Furthermore, we provide a *pattern catalog* for the transformation patterns. After modeling, the transformation and execution can be invoked through the modeler's UI. In our demo, we will model an integration scenario based on the challenge's tasks using our graphical data mashup modeling editor and we will execute the mashup in a pattern-based manner.

Figure 2 depicts the scenario we showed at the ICWE RMC 2015. In this scenario, Twitter "Tweets" are searched based on keywords extracted from New York Times articles. Based on these Tweets, the sentiment of people regarding the corresponding topic can be determined. These information are merged and visualized. In the last year, we were able to gain important feedback to enhance our data mashup tool FlexMash 2.0 and created the new version FlexMash 2. This year, we hopefully have the possibility to present FlexMash 2.0 at the RMC 2016 and gain additional feedback to make FlexMash even better.

References

1. Aghaee, S., Nowak, M., Pautasso, C.: Reusable decision space for mashup tool design. In: Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. pp. 211–220. EICS '12, ACM, New York, NY, USA (2012)
2. Daniel, F., Matera, M.: Mashups - Concepts, Models and Architectures. Data-Centric Systems and Applications, Springer (2014)
3. Heudecker, N., White, A.: The data lake fallacy: All water and little substance. Gartner Report G 264950 (2014)
4. Hirmer, P., Mitschang, B.: FlexMash - Flexible Data Mashups Based on Pattern-Based Model Transformation. In: Daniel, F., Pautasso, C. (eds.) Rapid Mashup Development Tools, Communications in Computer and Information Science, vol. 591, pp. 12–30. Springer International Publishing (2016)
5. 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), submitted