Discovery and Reuse of Composition Knowledge for Assisted Mashup Development

Florian Daniel, Carlos Rodriguez, Soudip Roy Chowdhury, Fabio Casati
Department of Information Engineering and Computer Science
University of Trento, Italy

Hamid R. Motahari Nezhad
HP labs, Palo Alto,
U.S.A

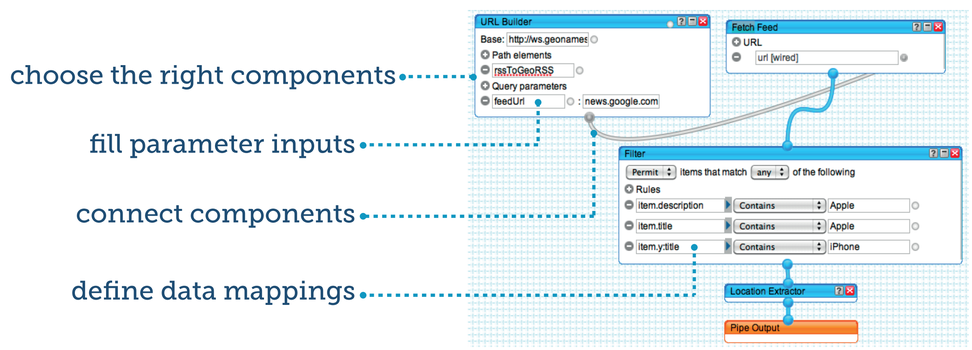
Visit us

Goal

To assist users in the development of mashups by means of interactive, contextual recommendation of reusable composition patterns

Motivation

Composing a mashup is a complex task:



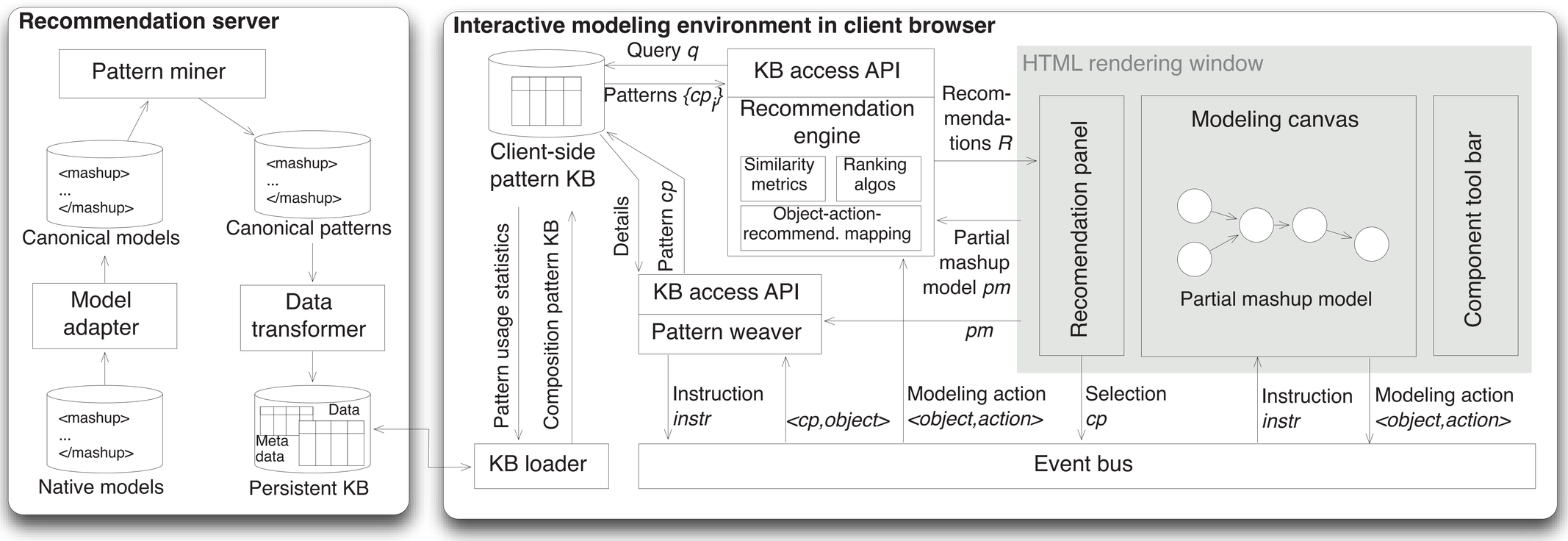
People without programming knowledge are not aware of these composition tasks

Challenges

- Discover **reusable composition knowledge** from a repository of mashups
- **Interactively assist** users during mashup development
- Query and retrieve **contextual recommendations** related to the current status of the user's composition
- **Automatically weave** composition patterns into the user's current composition

Functional architecture of our knowledge recommendation approach

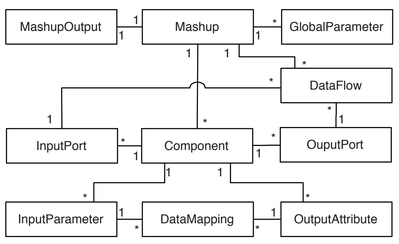
The prototype is divided into two parts: the recommendation server is where we mine composition patterns and provide composition knowledge as a service; the interactive modeling environment is where we recommend and weave composition patterns via suitable client-side extensions.



Mining patterns

Canonical Model

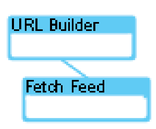
It represents generic constructs of dataflow mashup models. It allows us to map different native mashup models into it and to run only a single set of mining algorithms, in order to extract knowledge from all of them.



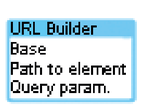
Composition Knowledge Patterns

They are the output of the mining process and provide insight into how to solve repetitive modeling problems. They are at the same level of abstraction as data-flow-based, interactive mashup tools.

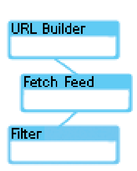
Example patterns:



Component cooccurrence pattern



Parameter value pattern

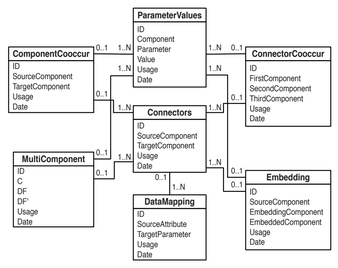


Multicomponent pattern

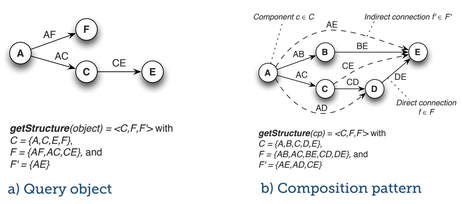
Recommending patterns

In order to recommend patterns in the client side, we store them in a knowledge base that is optimized for fast pattern search and retrieval.

We use exact matching for retrieving patterns with a fixed and known structure, such as the component co-occurrence pattern.We use approximate matching for multi-component patterns, whose structure we do not know in advance



Pattern knowledge base



***getStructure**(object) = <C,F,F'>* with *C = {A,C,E,F}*, *F = {AF,AC,CE}*, and *F' = {AE}*

a) Query object

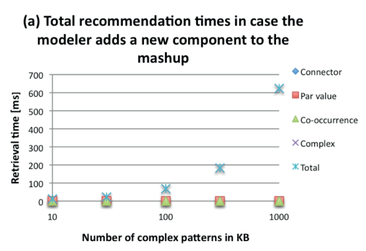
***getStructure**(cp) = <C,F,F>* with *C = {A,B,C,D,E}*, *F = {AB,AC,BE,CD,DE}*, and *F' = {AE,AD,CE}*

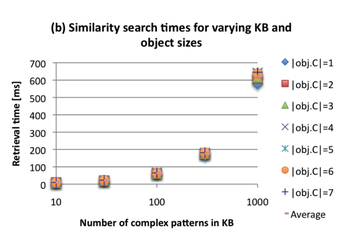
b) Composition pattern

	A	B	C	D	E	F
A	1	-	-	-	-	-
B	-	1	-	-	-	0.5
C	-	-	1	-	-	-
D	-	-	-	1	-	-
E	-	-	-	-	1	-
F	-	0.5	-	-	-	1

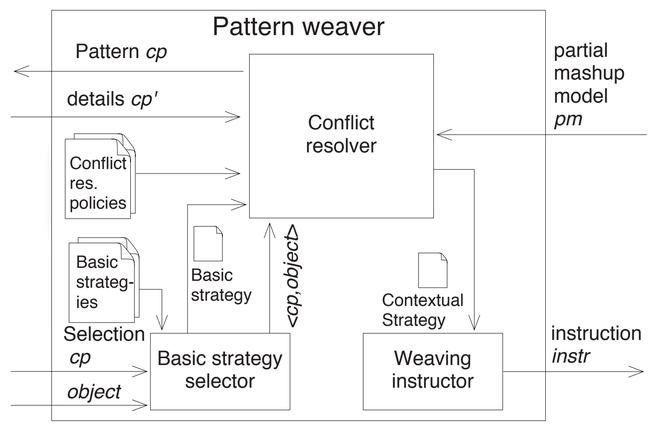
b) Similarity matrix

Even with 1000 multi-component patterns in the knowledge base, it takes only 608 ms to retrieve matching patterns.





Weaving patterns



Upon selection of a recommended pattern, the pattern weaver weaves the pattern into the partial mashup by:

- identifying a basic weaving strategy, consisting of a set of platform- and composition context-independent basic mashup operations;
- deriving a contextual weaving strategy, consisting of a set of platform-independent but composition-contextual mashup operations;
- applying the contextual weaving strategy to the partial mashup, using a set of platform-specific mashup operations.