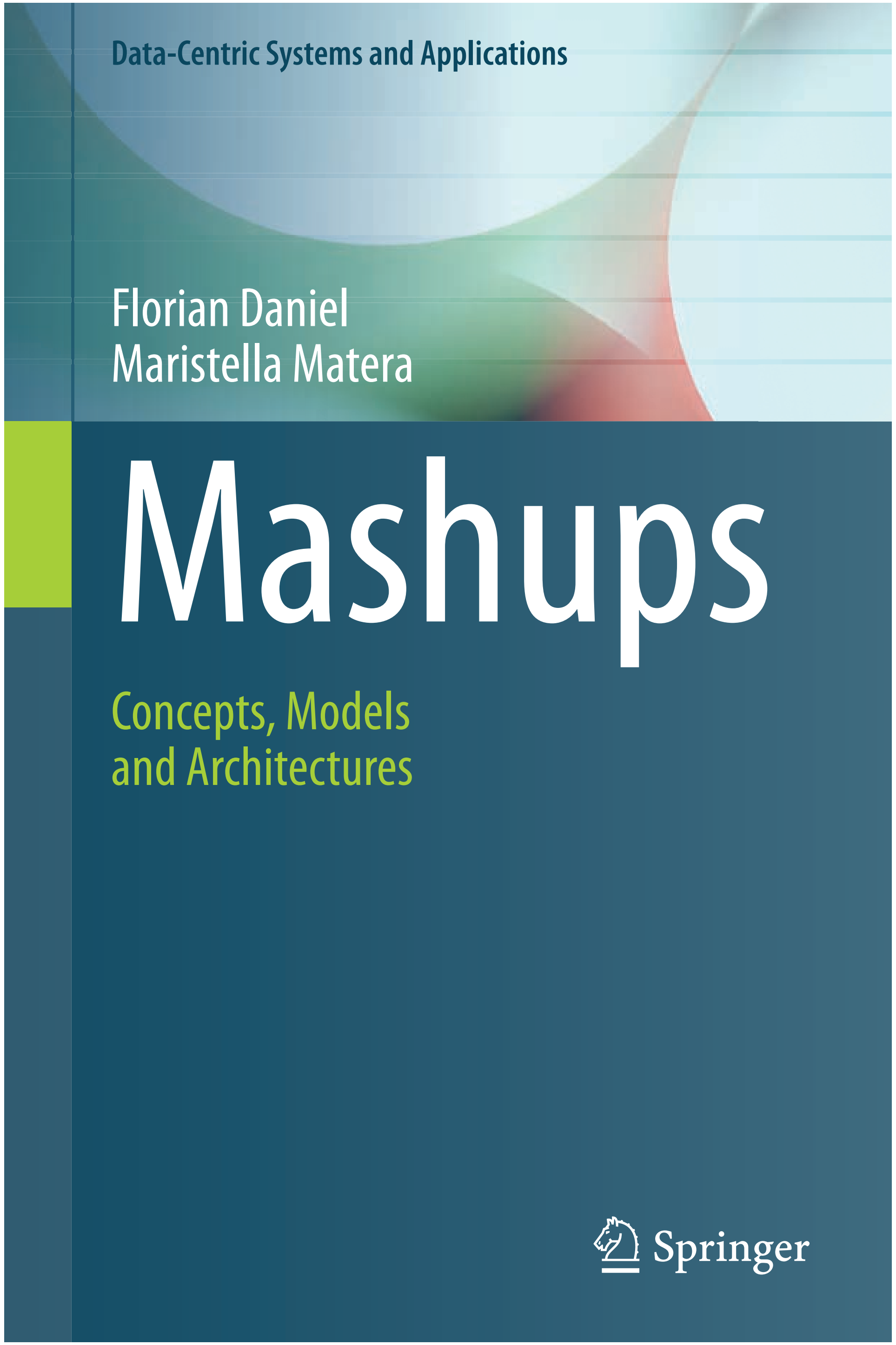

Chapter 8
Tool-Aided Mashup Development

Figures

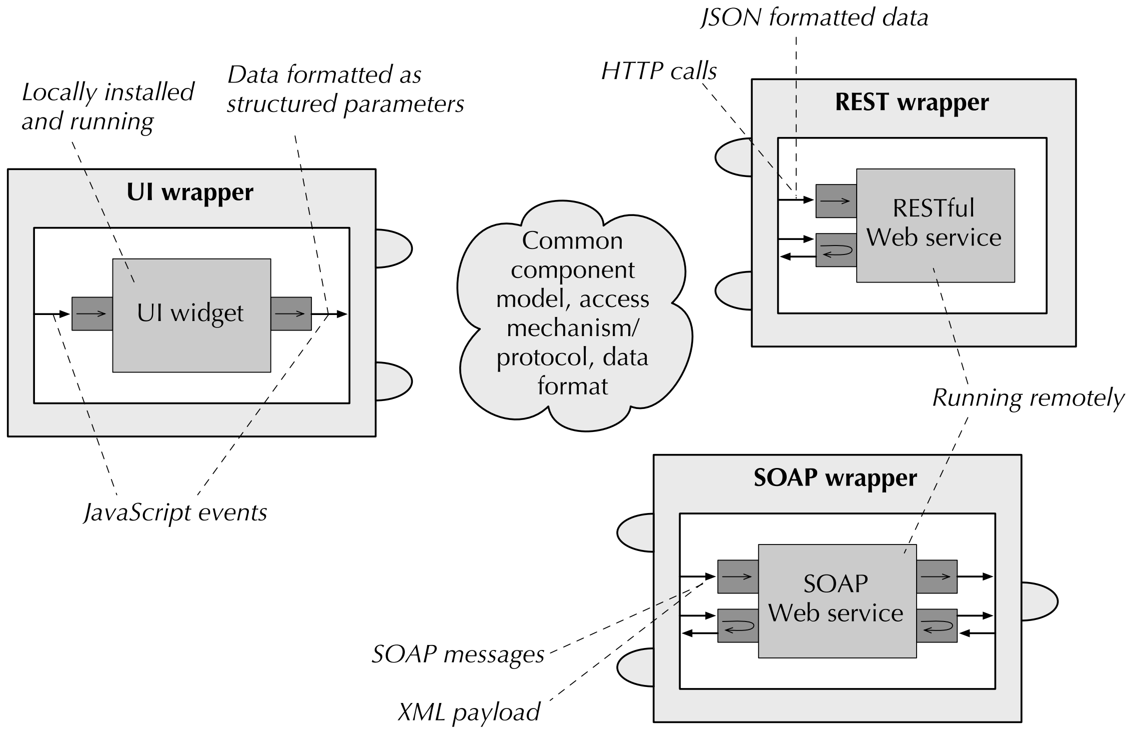

Fig. 8.1 Wrapping components into a unified view on native component models.

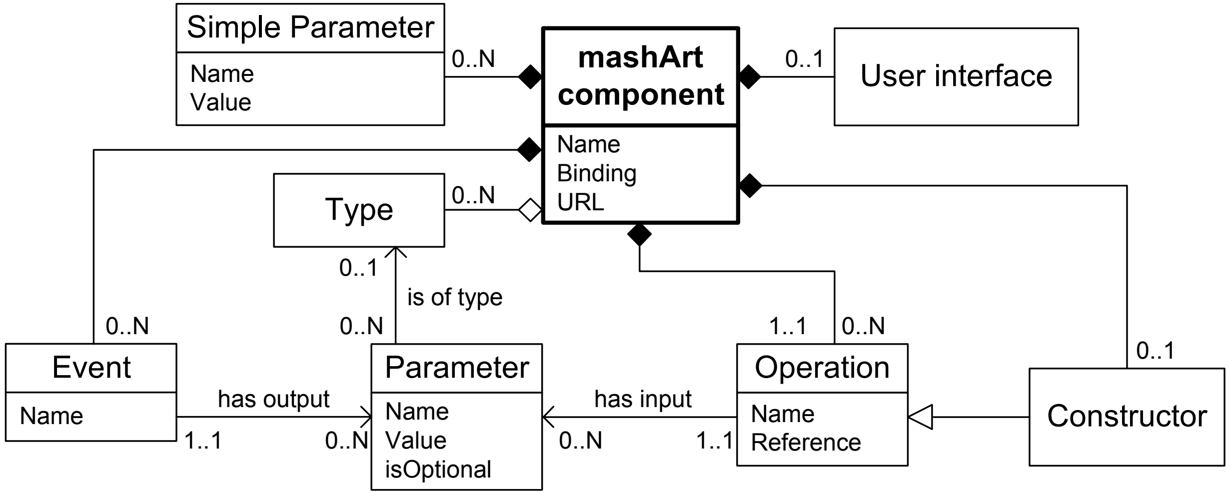

Fig. 8.2 Unified component model of mashArt components for SOAP/RESTful web services, UI components and RSS/Atom feeds [90].

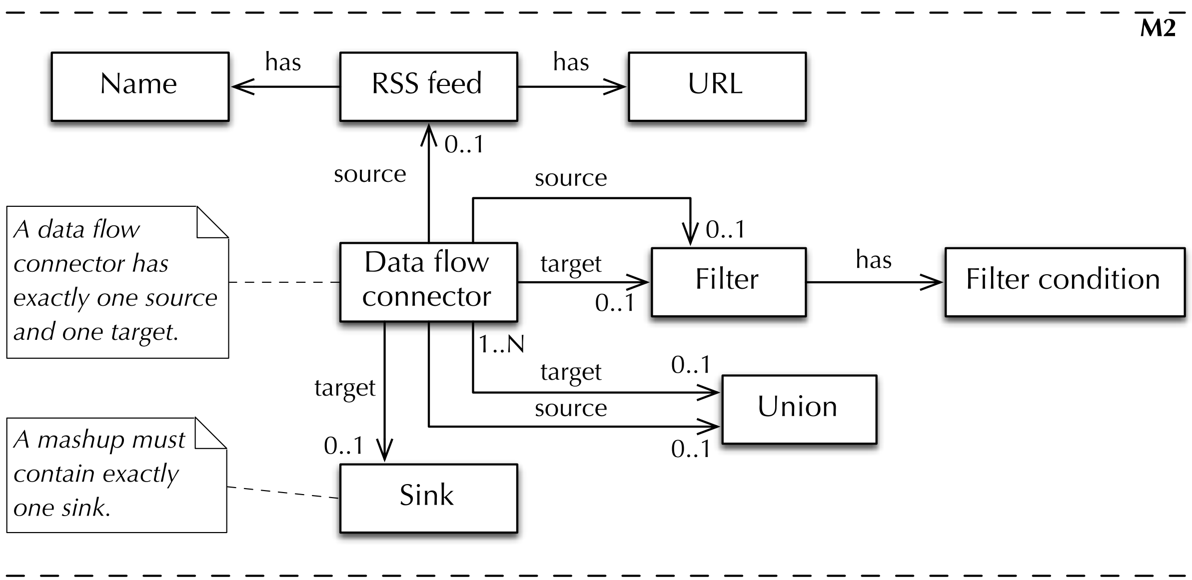

Fig. 8.3 Metamodel (M2) of a very simple data flow mashup language: it supports fetching different *RSS feeds* from the Web, computing their *union* and/or *filtering* them, and *publishing* the result again as an RSS feed on the Web (the *sink*). A *data flow connector* must always have exactly one *source* and one *target*.

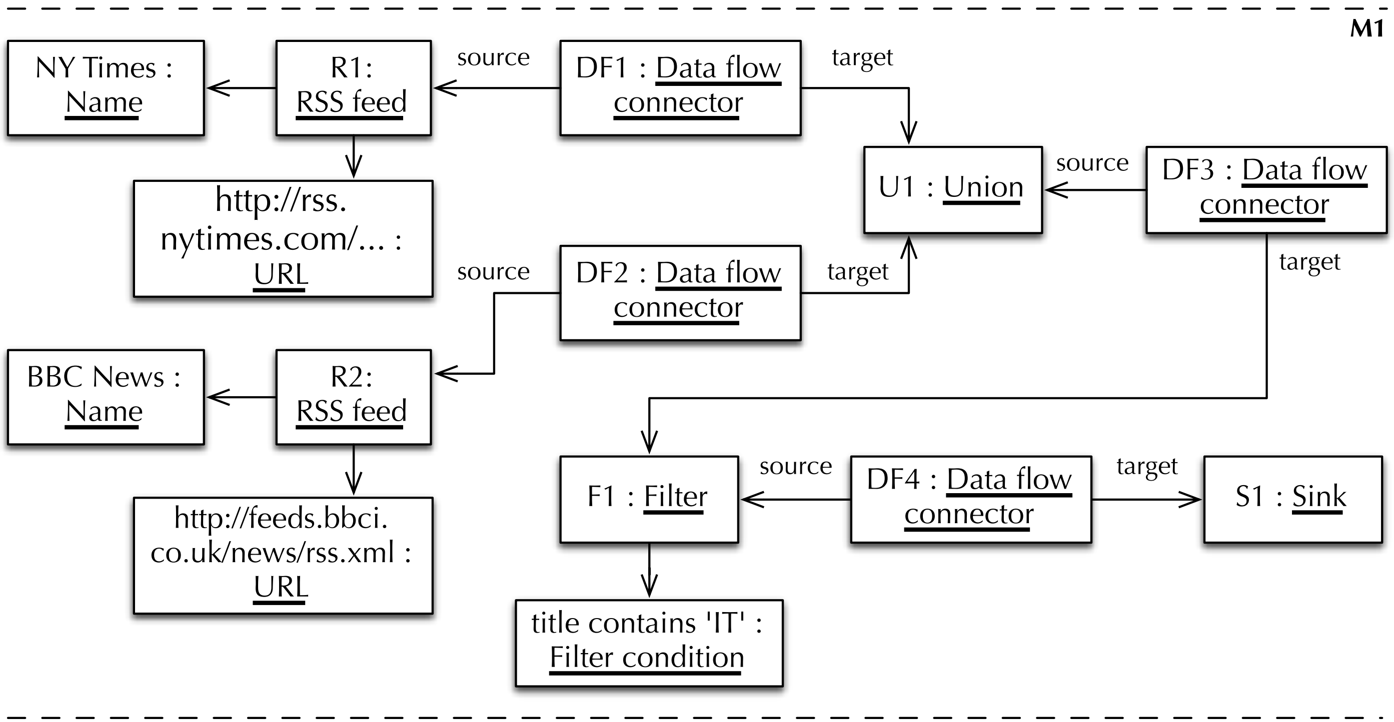

Fig. 8.4 A simple data flow model (M1) complying with the metamodel of Figure 8.3 expressed in an abstract syntax, i.e., a UML object diagram.

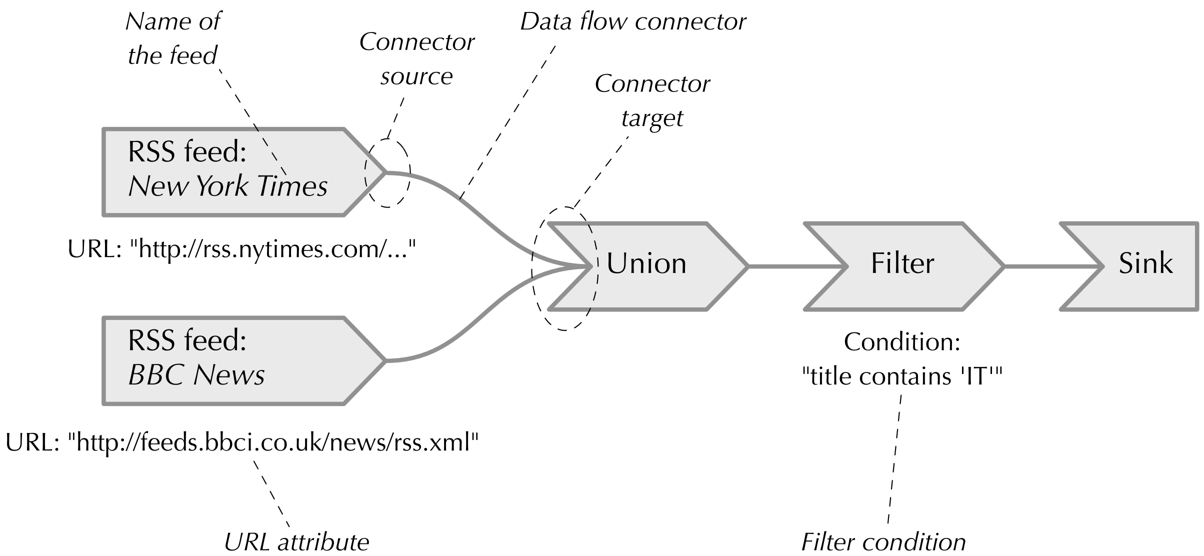

Fig. 8.5 The simple data flow model (M1) of Figure 8.4 expressed in a concrete syntax that highlights the semantics of the constructs and eases readability.

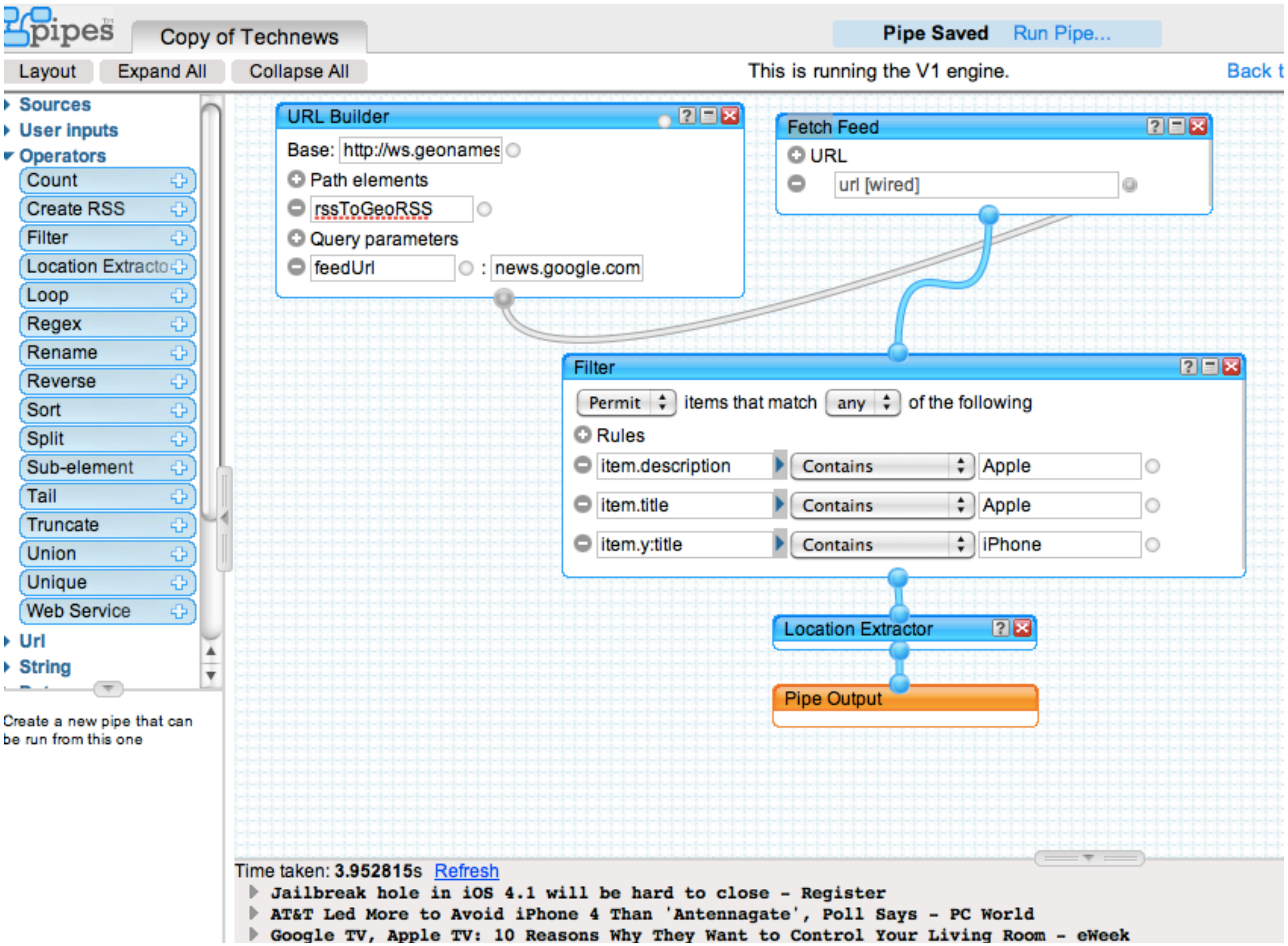

Fig. 8.6 A simple pipe that enriches an RSS feed with geographical location information and plots it on a map (the *Location Extractor* component).

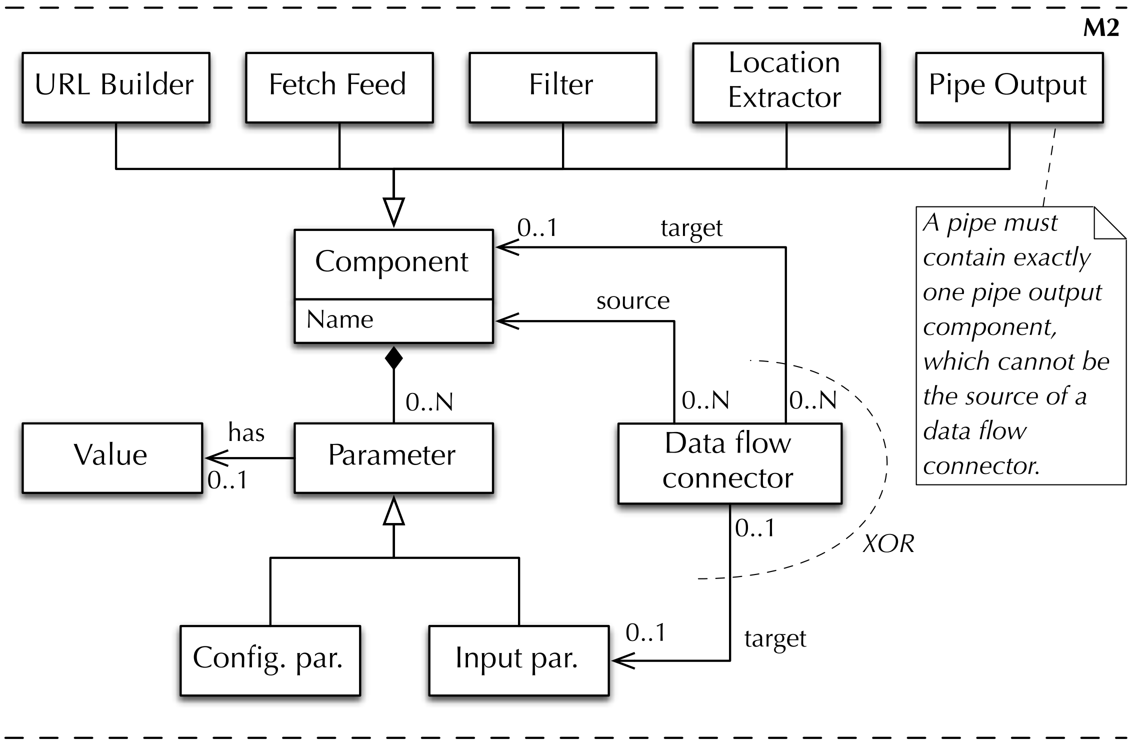

Fig. 8.7 A simplified metamodel of Yahoo! Pipes for the pipe in Figure 8.6.

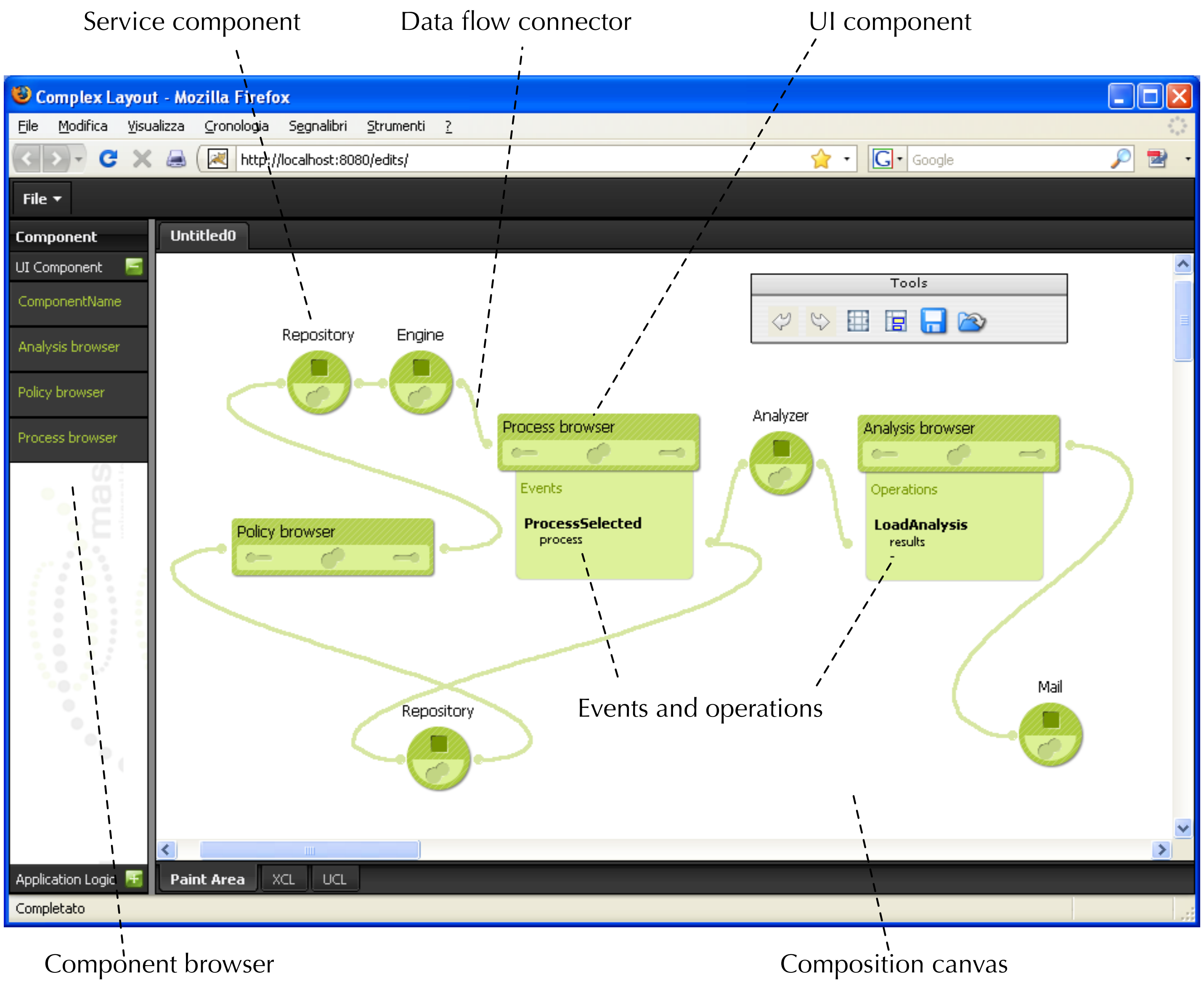

Fig. 8.8 A mashup modeled in mashArt [90]. The model represents a simple application for the monitoring of compliance.

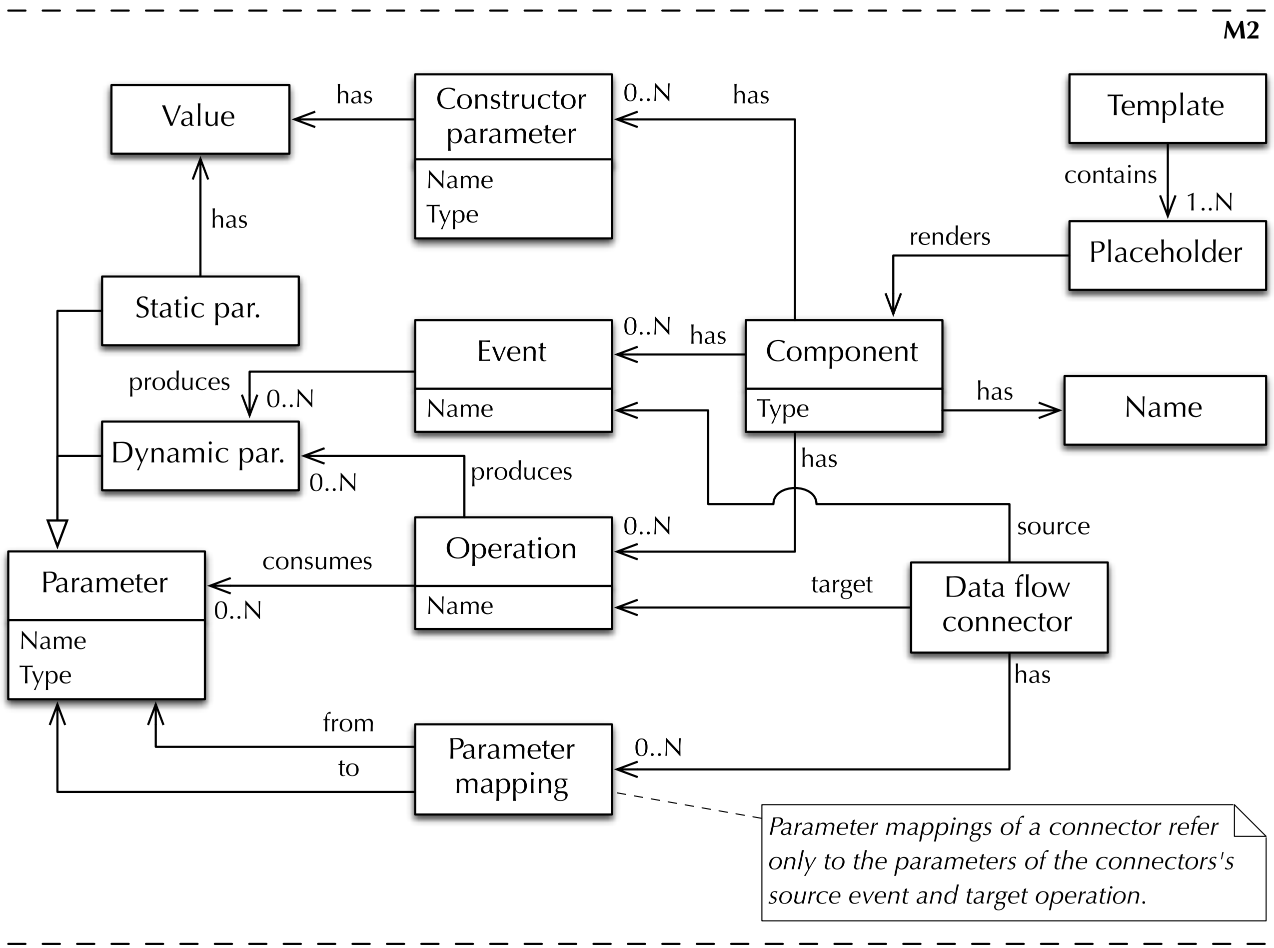

Fig. 8.9 Metamodel of the mashArt modeling notation based on the unified component model of Figure 8.2.

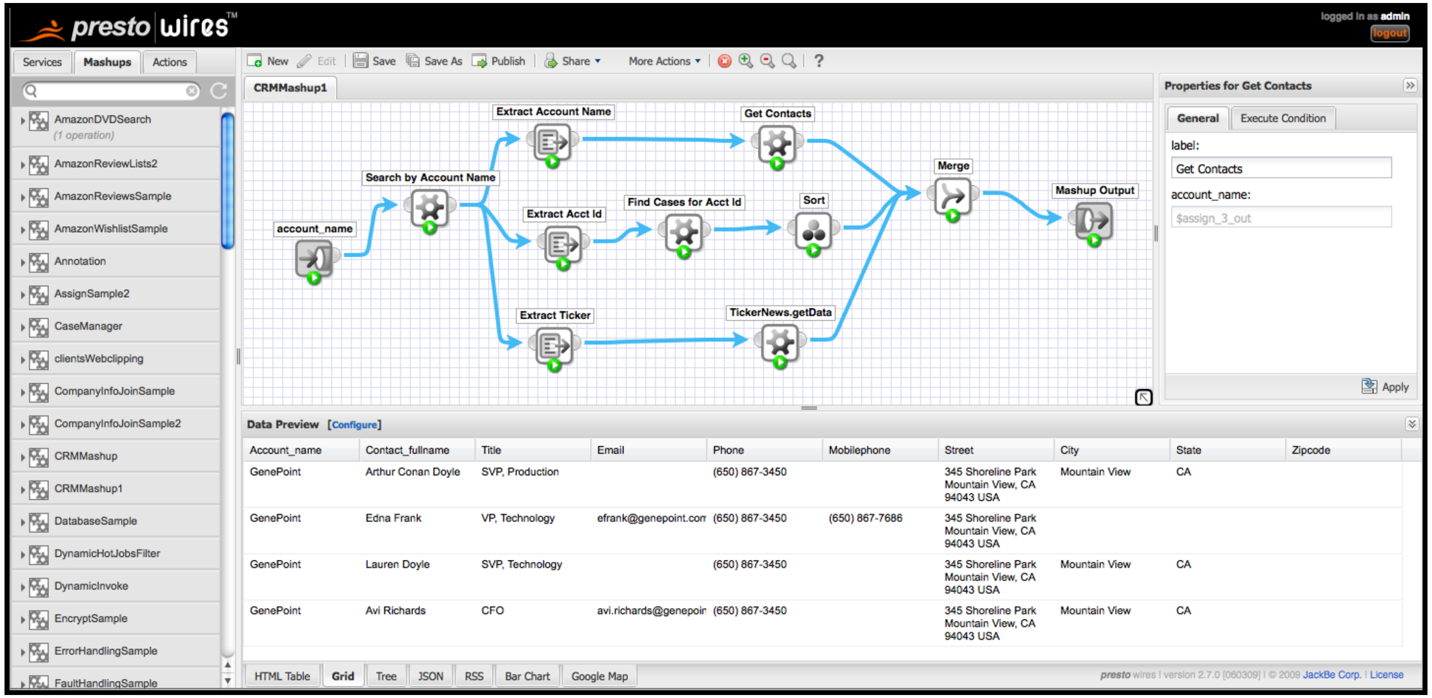

Fig. 8.10 A screenshot of Presto Wires for data mashups serialized in EMML.

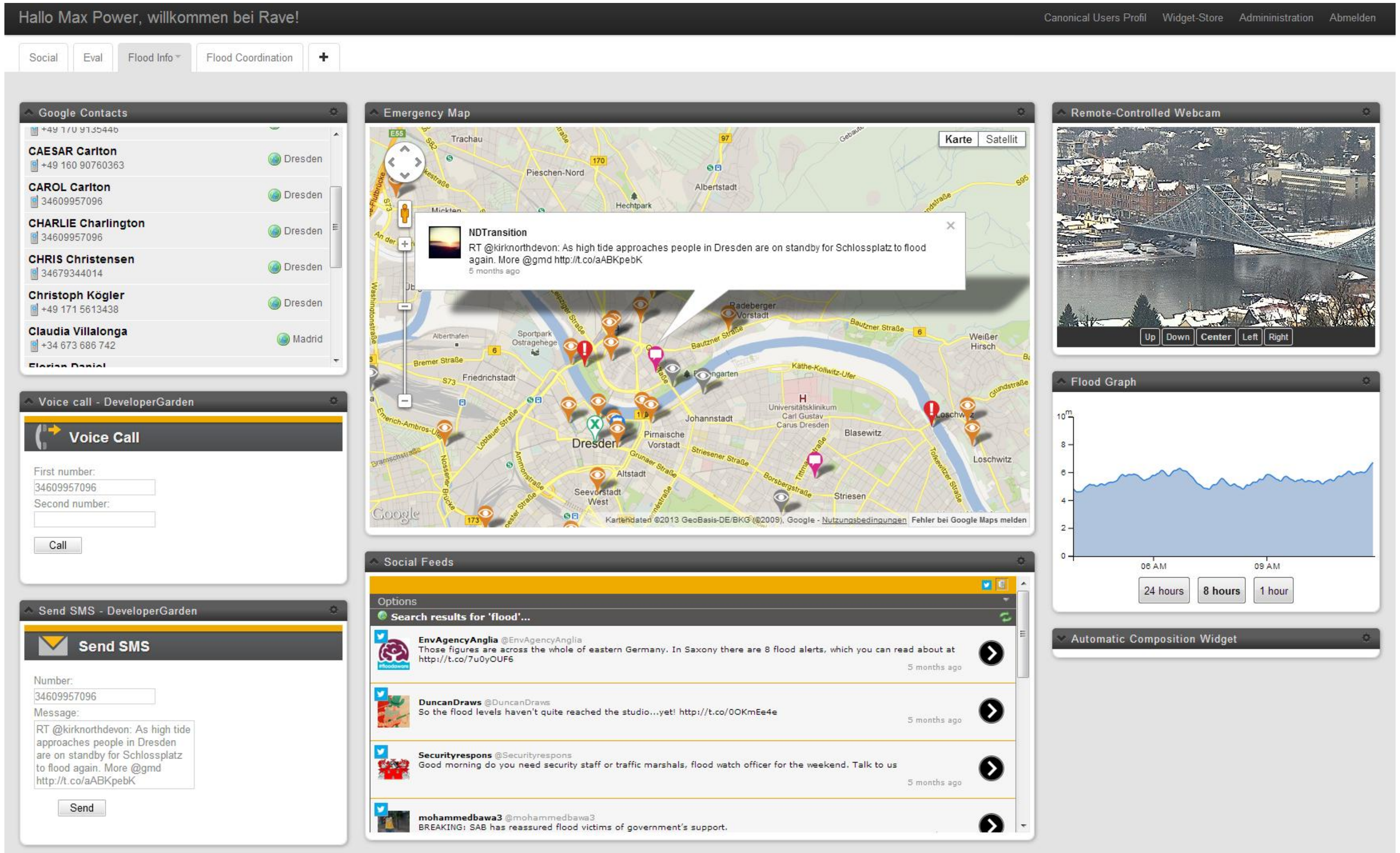

Fig. 8.11 A screenshot of the Apache Rave mashup environment extended by the EU FP7 project OMELETTE to import/export OMDL-compliant workspaces [262].

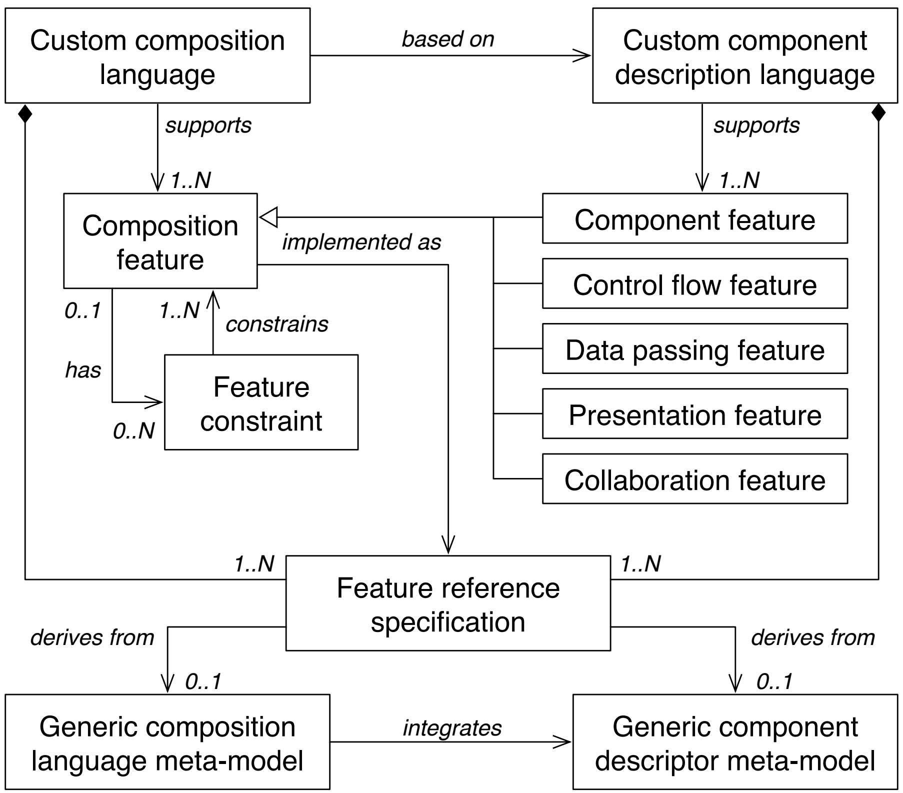

Fig. 8.12 Conceptual approach to developing custom mashup languages [254].

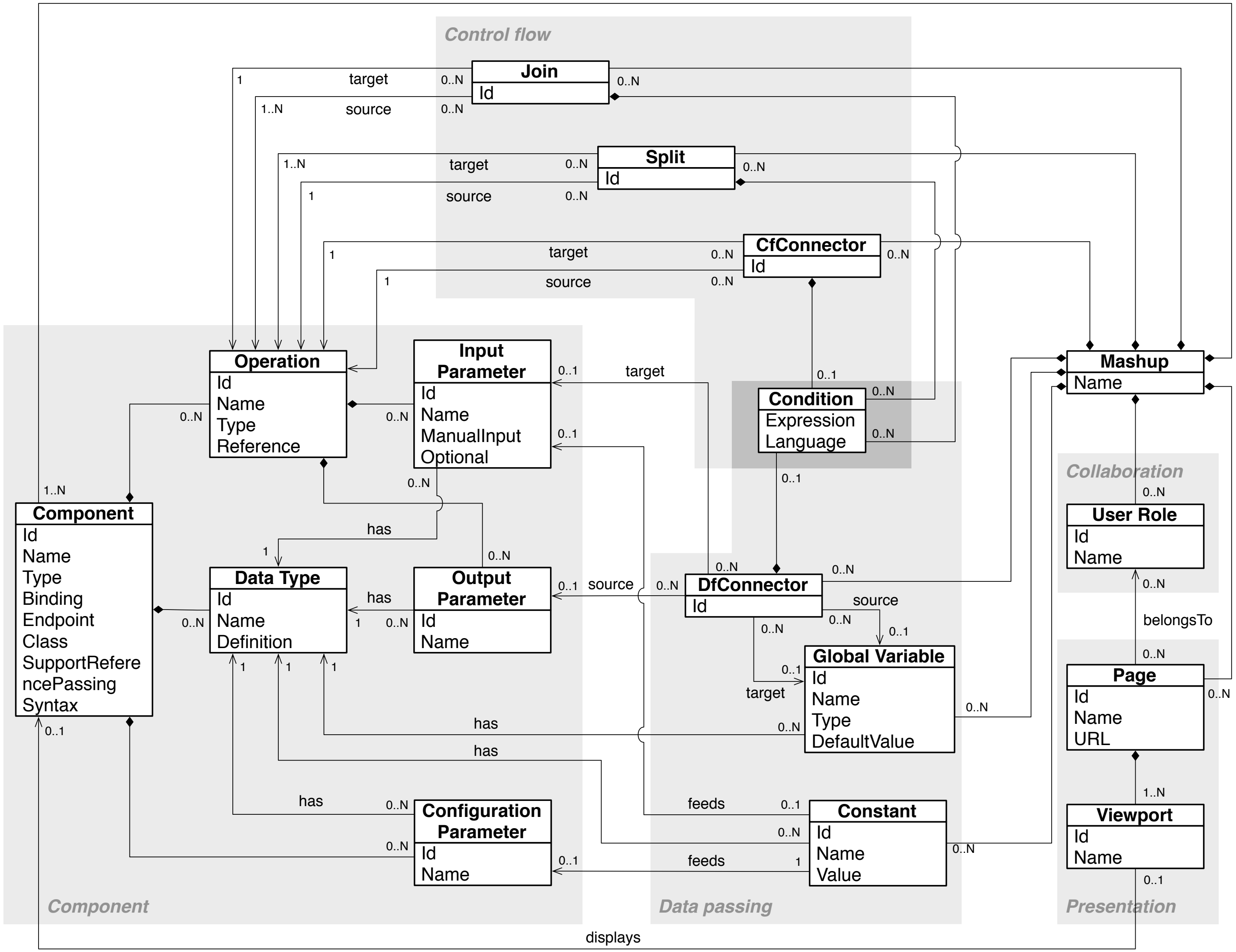

Fig. 8.13 The generic mashup language model bringing together the most common mashup features [253].

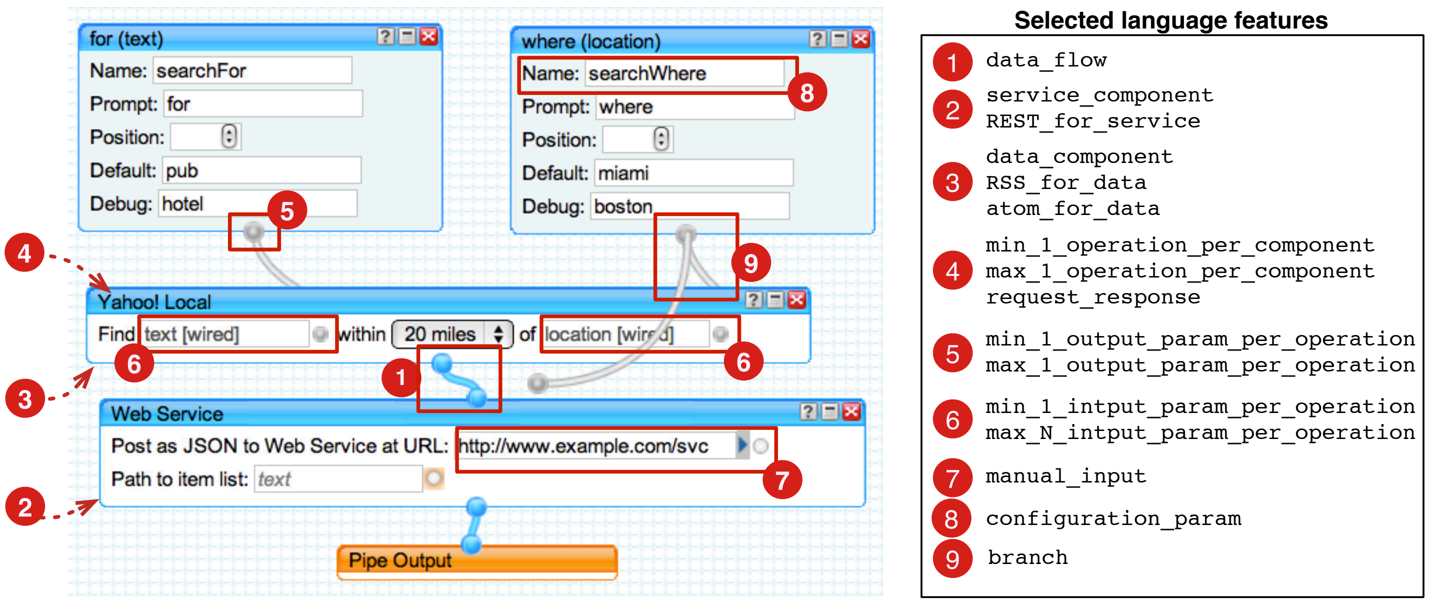

Fig. 8.14 Yahoo! Pipes example composition and set of respective language features

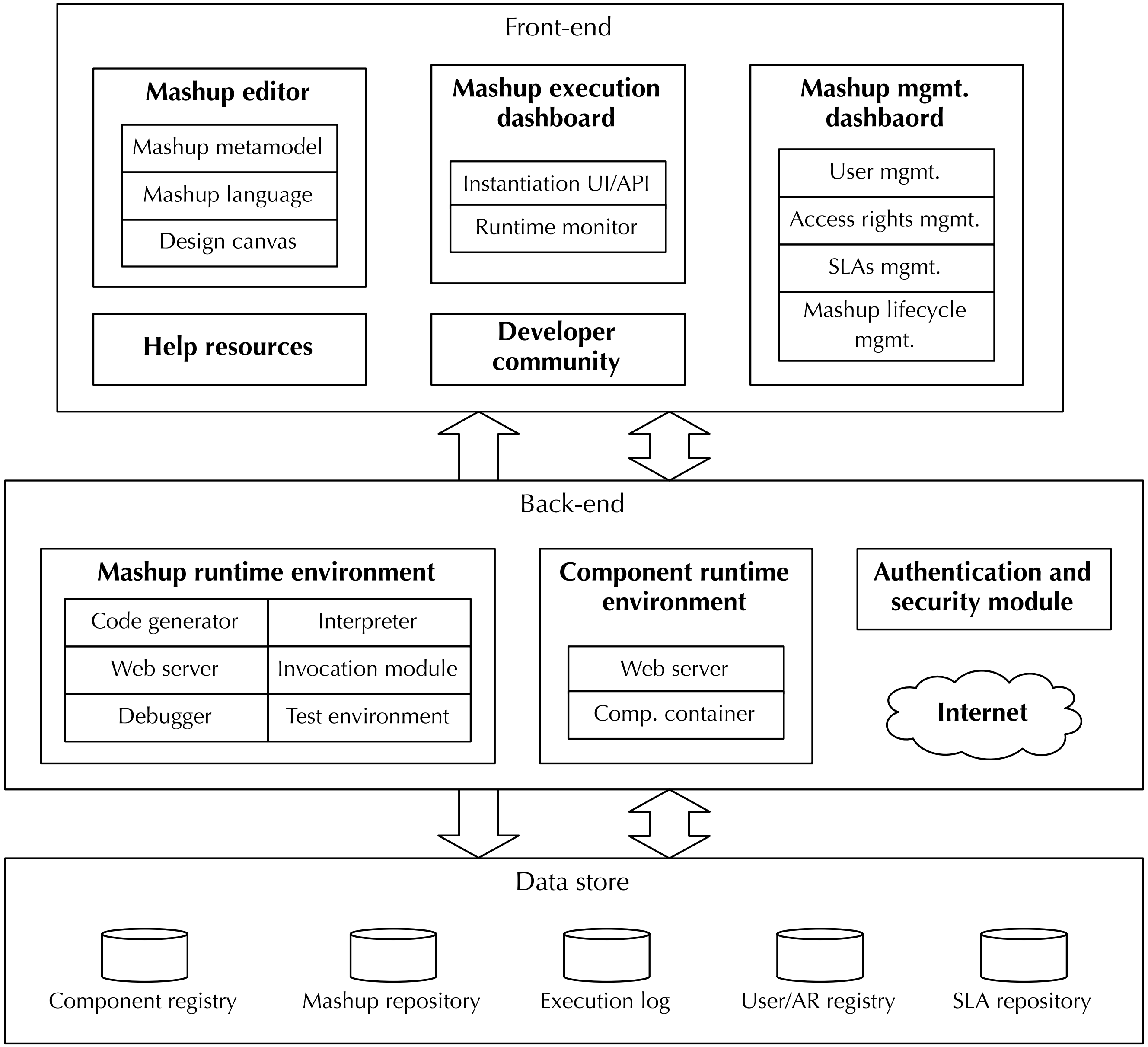

Fig. 8.15 Conceptual reference architecture of a mashup platform articulated into front-end, back-end and persistent data store.

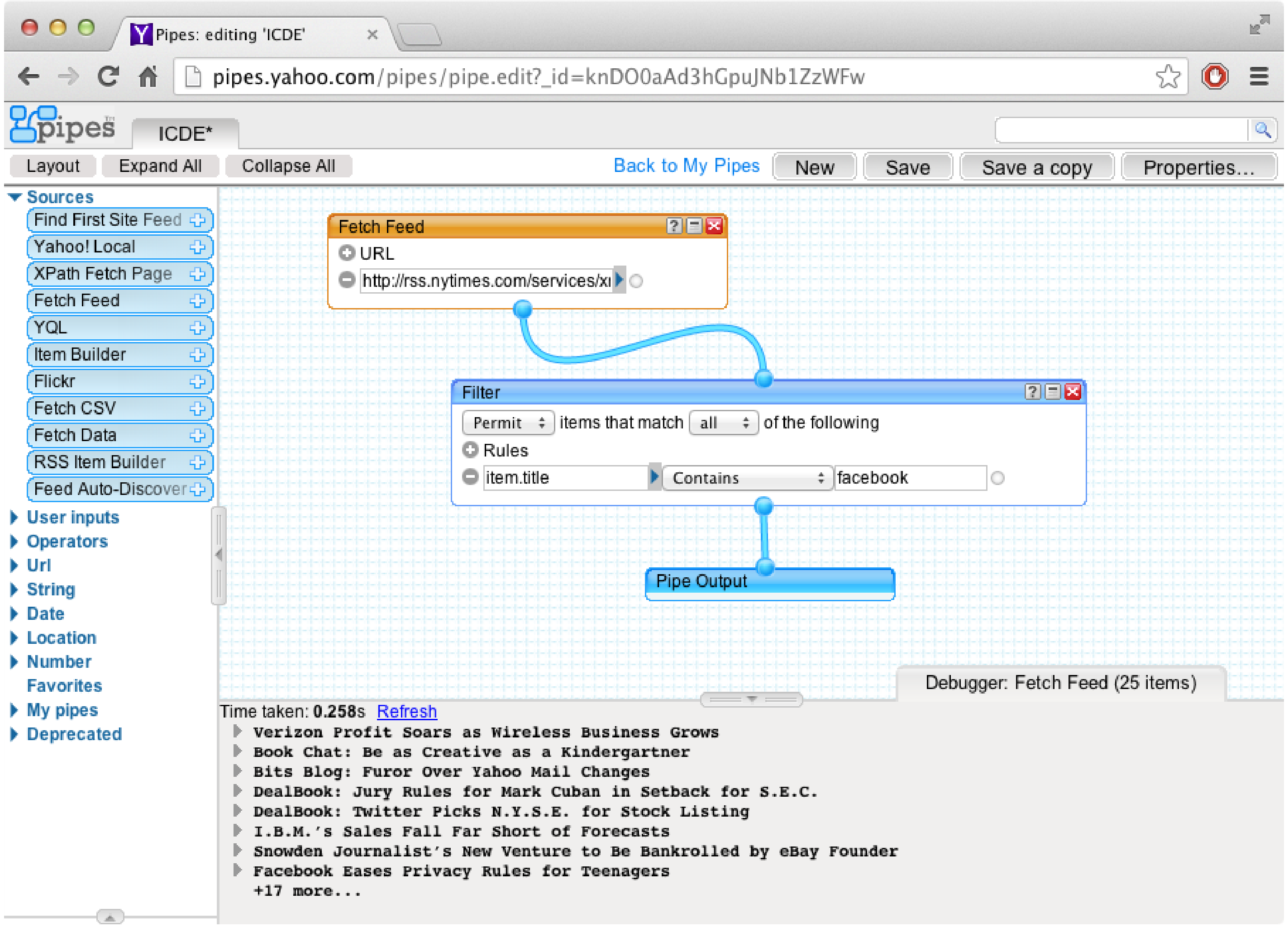

Fig. 8.16 Screen shot of Yahoo! Pipes with the debugging tab open (bottom). The content of the tab is the output of the Fetch Feed component selected in the canvas.

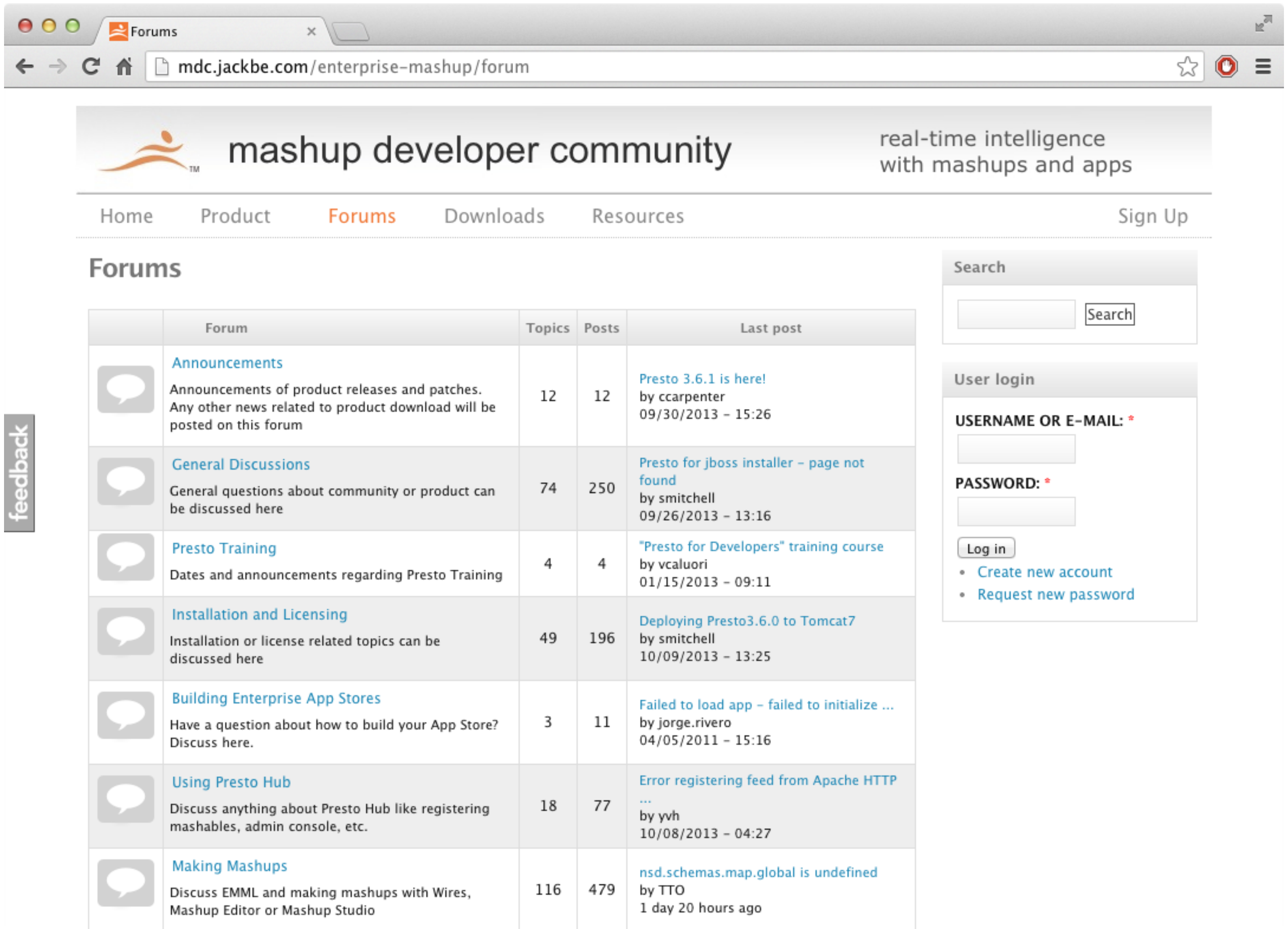

Fig. 8.17 Screen shot of the JackBe Presto developer community website.

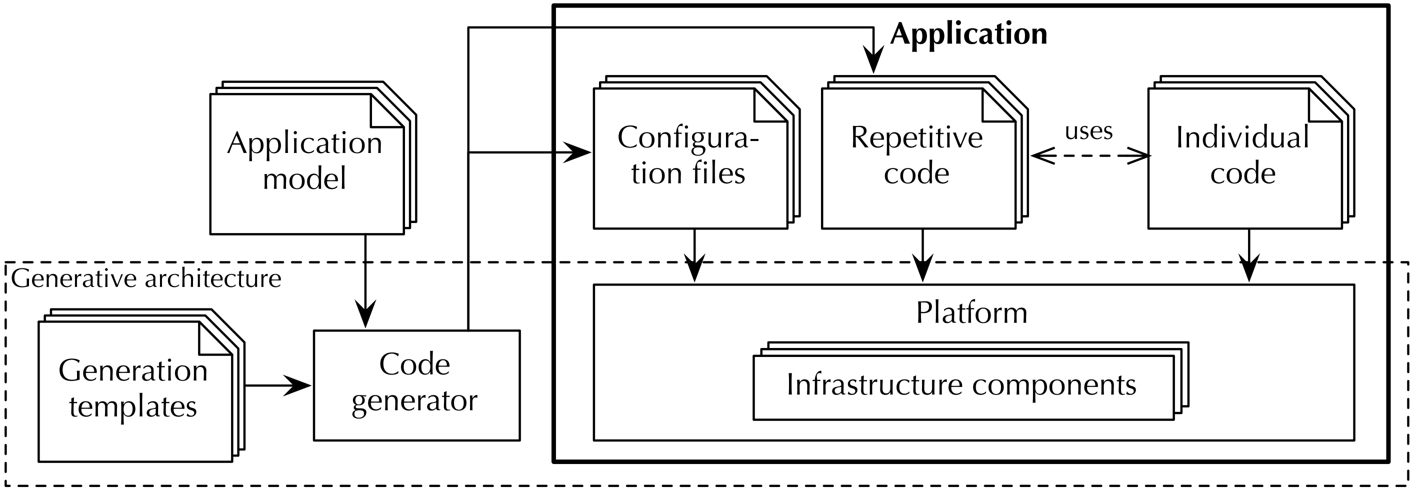

Fig. 8.18 Code generation in architecture-centric MDSD. The generative architecture corresponds to the runtime environment in Figure 8.15.