Mashups have emerged as an innovative software trend that re-interprets existing Web building blocks and leverages the composition of individual components in novel, value-adding ways. Additional appeal also derives from their potential to turn non-programmers into developers. Daniel and Matera have written the first comprehensive reference work for mashups. They systematically cover the main concepts and techniques underlying mashup design and development, the synergies among the models involved at different levels of abstraction, and the way models materialize into composition paradigms and architectures of corresponding development tools. The book deliberately takes a balanced approach, combining a scientific perspective on the topic with an in-depth view on relevant technologies. To this end, the first part of the book introduces the theoretical and technological foundations for designing and developing mashups, as well as for designing tools that can aid mashup development. The second part then focuses more specifically on various aspects of mashups. It discusses a set of core component technologies, core approaches, and architectural patterns, with a particular emphasis on tool-aided mashup development exploiting model-driven architectures. Development processes for mashups are also discussed, and special attention is paid to composition paradigms for the end-user development of mashups and quality issues. Overall, the book is of interest to a wide range of readers. Students, lecturers, and researchers will find a comprehensive overview of core concepts and technological foundations for mashup implementation and composition. Even without low-level coding details, practitioners like software architects will find guidance on key implementation concepts, architectural patterns, and development tools and approaches. A related website provides additional teaching material which can be used either as part of a course or for self study.

This book is timely, provides a thorough scientific investigation and also has practical relevance in the general area of composition and mashups. It is of particular interest to researchers and professionals wishing to learn about relevant concepts and techniques in service mashups, composition, and end-user programming.

From the Preface by Boualem Benatallah, University of New South Wales, Sydney
The users start from a set of ready services, i.e., the mashup components, in which they create the mashups in a context characterized by multi-stakeholders, each one participating with own skills and competencies. In this way, the users benefit from the expert developers (such as IT programmers, service providers, or so on), who are directly involved in constructing such mashups, but they benefit from the shorter turnaround time for new applications.

Fig. 9.1 Two main mashup development scenarios. (a) Expert developers exploit mashup tools “centrally” to deliver applications quickly. (b) Users exploit such tools to create mashups in a “distributed” fashion, starting from a set of ready services.
9.4.1 Component discovery and selection

The mashup composer starts with an idea that addresses personal needs and preferences and then selects services that can provide the necessary data, application logic, or user interfaces. Discovery and selection is a life-cycle activity which is peculiar of mashup applications. It precedes mashup composition and implicitly incorporates requirements analysis and specification. The mashup idea itself, which leads the composer to discover components, can be indeed considered an informal expression of the application requirements. The selected mashup components then “specify” these requirements in terms of the capabilities offered by the selected services, thus proving in a lightweight manner the idea’s feasibility and providing a draft of the mashup organization that then, dynamically and iteratively, can be easily evolved into the final application.

However, it might be complicated for a non-expert composer looking at a programmatic interface or at a technical documentation to guess how a component can be used, which of its features can be adopted and which effect each service may have on the overall composition. Therefore, in a context of EUD mashup development, specific attention must be paid to the adoption of adequate service representations. For example, showing the behavior of services in terms of the data, functionality and UI they are able to offer would help the users understand how the service can be used and integrated in a mashup. Technical representations, for example highlighting input and output parameters, that are far from the behavior of the service that can be observed during the mashup execution, are difficult to master by the end-user. This of course entails the adoption of adequate component models, that can mask the technology heterogeneity of components and expose only the...

Fig. 9.2 Life-cycle models of (a) current Web applications and (b) mashups. The model for the end-user development of mashups presumes the availability of a dedicated mashup platform and toolkit, along with a set of open Web services that provide functionality and data.
The WYSIWYG composition editor of the PEUDOM platform [58, 183].
Fig. 9.4 Internal architecture for WYSIWYG composition in PEUDOM [58, 183].

© Copyright 2014 by F. Daniel and M. Matera. Reproduction for classroom use and teaching allowed if source is properly cited.
Fig. 9.5 Model of University of Trento’s internal department evaluation procedure modeled in ResEval Mashup [155].
Fig. 9.6 A meta-design approach for mashup creation. The bottom layer outlines the environments for end-users, the middle and the top layers the environments for experts developers and domain experts who operate customizing the platform [18].
Fig. 9.7 Methodology for the development of domain-specific mashup tools with the Meta-DMT described in [253].
9.5 EUD Dimensions for Mashup Tools

Component Registry

Quality Vectors
Association Rules
Quality Broker
Event Handler

Component Recommender

Front-end

Back-end

Component Registry

Quality Vectors
Component Descriptors
Wrappers

Quality Broker

Composition Model
Compat. & Similarity Matrices

Data Services and APIs
Web

Adding a component
Component ranking
new component
new listener

Fig. 9.8 Modules for quality-aware recommendations in PEUDOM [61].
Fig. 9.9 Simplified architecture of the assisted modeling environment with client-side knowledge base and interactive recommender proposed in [73].
Fig. 9.10 Screen shot of the Baya plug-in for Yahoo! Pipes at work [74]: mashup model patterns are recommended in the panel at the right-hand side and woven into the model in the canvas by dragging and dropping them onto the canvas.