

DERI – Digital Enterprise Research Institute

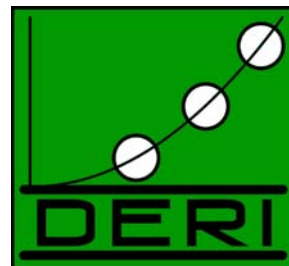
Mapping Mashup Makers to a Design Space

Ronan Fox, Manfred Hauswirth

DERI Technical Report 2010-06-02

Copyright © 2010 by the authors.

DERI – Digital Enterprise Research Institute



DERI Galway
National University of Ireland
Galway, Ireland
www.deri.ie

Mapping Mashup Makers to a Design Space

Ronan Fox, Manfred Hauswirth¹

Abstract: In this report the important dimensions that bound the design space of mashup makers are described. These identified dimensions are mapped to existing mashup makers, showing examples of how those dimensions are implemented by mashup makers in the provision of services to their user groups. 13 technical dimensions were identified with 53 classes, which were subsequently mapped to 45 mashup makers.

Keywords: mashups, design space, reference architecture

¹ Digital Enterprise Research Institute, National University of Ireland, Galway

Acknowledgements: This work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

Table of Contents

1	INTRODUCTION	4
2	THE MASHUP MAKER	5
2.1	Processing Transparency.....	5
2.2	Resource Combination Logic	5
2.3	Resource Interaction	6
2.4	Communication	6
2.5	Discovery	7
2.6	Extensibility	7
3	THE MASHUP	7
3.1	Processing Transparency.....	7
3.2	Location Transparency.....	8
3.3	Resource Access	8
3.4	Resource Use	8
3.5	Integration Level.....	9
3.6	Reuse.....	9
3.7	Connectivity.....	9
3.8	Portability	10
4	THE DESIGN SPACE MAPPED TO MASHUP MAKERS.....	10
4.1	User Support.....	10
4.2	Security	11
4.3	Flexibility	12
4.4	Infrastructure.....	13
5	RELATED WORK.....	14
6	DISCUSSION	15
7	REFERENCES	16
	APPENDIX A - TECHNICAL DIMENSIONS WITHIN THE MASHUP DESIGN SPACE	17

1 INTRODUCTION

In the selection of the various dimensions used to describe the design space of Mashup Makers we draw on research carried out into mashups [1], mashup makers [2], other design spaces [3] and the background and intensions around the development of mashups themselves [4]. Elements of end-user programming and end-user software engineering and the requirements that drive this activity within the context of mashups was also used to identify the key stakeholders and their motivations for developing and using mashups [5].

From the above work we have identified a number of key overriding concepts to be considered in the specification of dimensions:

- Consumer or Enterprise. In [2] the authors provide a classification of existing mashup makers based on the market target of the tools: Consumer or Enterprise. This provides an important question in terms of what makes a mashup maker consumer- or enterprise-oriented, and has significant impact on features of a mashup maker such as security, flexibility, and the infrastructure given that enterprise-oriented mashup makers will be made available in a controlled environment whereas consumer-oriented mashup makers will be made available on the open internet.
- Heterogeneity. In the context of integration heterogeneity acts across several aspects of a mashup maker. Heterogeneity exists in aspects of the syntax used to access a resource, in the nature of the (a)synchronous communication mode, whether the resource manages session state or if this responsibility is placed on the mashup maker, and on the nature of the data integration process – be that declarative or imperative.
- Decoupling/Reuse. When combining resources in a mashup, providing a decoupled integration mechanism makes the resources more reusable, and the mashup itself more flexible, as equivalent resources can be swapped as required and with minimised effort. Reuse is also a feature of the mashup.
- Integration Level. Within a mashup the resources determine the level at which integration takes place; combining data, processes, or UI components with data elements. Each layer places differing requirements on the mashup maker, influencing the dimensions to be selected as core tool requirements.

While these concepts are at too broad of a level to be of practical use as dimensions in and of themselves, they provide us with guidance on the types activities stakeholders will perform while using both the mashup maker and the resultant mashups and applications. We will find them permeating other dimensions when we consider such facilities as the discovery of resources, determining the requirements of a mashups, the support of users in building mashups, the collaboration involved in the creation of mashups and the collaboration that takes place in the use of those mashups.

From The Design Space of Wireless Sensor Networks [3]:

“It is certainly debatable which issues are important enough to be explicitly considered as dimensions in the design space and one could argue in favour of adding more dimensions or removing some

from our suggestions detailed below. In fact, we expect that this might become reasonable in the future as the field and its applications evolve. However, we have tried to ensure that our initial suggestion consisted of a sensible set of dimensions, by basing our choice on the following two principles. Firstly, there should be notable variability between applications with respect to dimensions. Secondly, a dimension should have a significant impact on the design and implementation of technical solutions.”

This is true of any design space created, especially in areas where research is ongoing, and new knowledge beachheads are being established on a daily basis. Any Design Space that is not capable of growing and morphing with the research and practice which bounds it, is useless. Likewise, any Design Space that, in one snapshot, encompasses a complete research area most likely reflects a dormant area of research. With that in mind what are the criteria we used to select the dimensions?

1. Dimensions must have a significant effect on the design and implementation of technical solutions
2. There is a notable variability in the types of applications with respect to the dimensions
3. The dimensions should be as independent as possible, and if not, this should be clearly shown.

In the following sections we describe the design space of mashup makers from two perspectives:

- Requirements placed directly on the mashup maker environment
- Requirements placed indirectly on the mashup maker environment through the requirements of users as they create and use mashups

2 THE MASHUP MAKER

This section defines the dimensions of the mashup maker design space which are directly influenced by the requirements placed on the mashup makers themselves.

2.1 Processing Transparency

Within a Mashup Maker the creation of mashups may take place on the client machine, or on a server which may be provided by the mashup maker. Occasionally a mashup is made available for use on a website, thus ensuring that while the creation took place on a desktop, the mashup maker allows the deployment of mashups to a server configuration for either download or direct execution on the server infrastructure.

The location where the processing takes place is an important factor since it affects the infrastructure that must be in place in terms of processing power, user interaction, and installation requirements.

Classes: Client, Server;

2.2 Resource Combination Logic

The basic requirement of a mashup is that it takes data or functionality from two or more resources and combines them within an ad-hoc environment. There are many ways in which these resources can combine the data; pipes and filters are the most common way of modelling the logic required to execute the mashup. Occasionally the combination logic required is to merely display resources in a unified view, maintaining distinct units of functionality without the need for direct combination logic. Other types of interaction models are also available which either attempt to decouple the resources, or to provide more efficient and more expressive means of combination. One example of this is the use of the data federation pattern to enable the selection of optimal combinations of data sources. Another interaction type is explicitly coded by the mashup developers, employing a code engine to drive the model which could be object-oriented or procedural.

The combination logic used has an effect on a number of factors when designing a mashup maker tool. Issues such as the expressivity of models, decoupling of resources, and reduction in the use of integration logic specific to each mashup, which must be firstly created and then maintained, act as input into the decision of what type of resource combination logic to use.

Classes: None, Pipes & Filters, Data Federation, Code.

2.3 Resource Interaction

When creating a mashup what facility is in place to enable interaction with the resources of concern? How do we get data from one or more resources and use it in another resource for more processing or display the results? Most mashups to date use code and/or configuration entries to extract, manage, and integrate data from resources. This code/configuration usually means that the data is either presented as is (in a homogeneous environment) or certain adjustments are made to it to ensure compatibility with another consuming resource (in a heterogeneous environment). There are options which will enable the flow of data between resources in a mashup while keeping them decoupled; publish/subscribe is one such option. Using pub/sub resources can publish data while other resources, or indeed the mashup environment itself, can subscribe to that data, process that data and/or display it.

The choice of resource interaction model has important consequences for the mashup maker in that it affects the coupling of resources, the level of user knowledge required to get resources to interact (the threshold), and the richness of the environment (the ceiling).

Classes: None, Event, Publish/Subscribe, Configuration, API, Combinations

2.4 Communication

Mashup Makers can access data and functionality from resources either through synchronous or asynchronous. The choice of techniques impacts on the risk associated with a badly formed resource or a badly composed mashup having an adverse affect on the working of the mashup maker.

This is an important dimension as the scalability and robustness of a mashup maker (which probably also acts as a deployment platform) can be determined by the choice of mode of communication carried out with the resources.

Classes: None, Synchronous, Asynchronous

2.5 Discovery

In an environment where there are numerous resources with which the user can work to create the mashup, discovery is a key component of the mashup maker. Supporting the user in finding relevant resources even to the point of establishing the interface of the resource and enabling the easy use of the resource in a mashup means that the user can create relevant mashups with a minimum of effort in the discovery process. Community support can be used to show individual users what other users have found useful, what combinations work, and examples of how to combine resources.

This is an important dimension as level of discovery offered by the mashup maker determines the level of mashup development support there is for the user on an individual level, and also for the community as a whole.

Classes: None, Individual, Community

2.6 Extensibility

The most commonly used resources with which mashup makers can interact are based on standards such as HTTP and REST. In some mashup makers a list of adaptors define the types of resources with which a mashup maker can interact. In other mashup makers the system will only interact with one type of (possibly standards-based) resource. A standards-based approach is designed to avoid “lock-in” for resource developers and mashup creators. Typical resources that fall outside the scope of standards-based approaches include database systems, enterprise resource planning systems, customer relationship management systems, and spreadsheets. Other mashup makers avoid lock-in by extending the capabilities of the system through the inclusion of components developed by others, or by providing wrapper technology which allows the user to define such extensions.

Extensibility provides us with an important dimension and in some mashup tools are provided with the mashup maker while other mashup makers provide mechanisms for the addition of further functionality, through for example registration facilities, eclipse-type add-ons, script libraries, and so on.

Classes: None, Fixed, Customizable

3 THE MASHUP

This section describes dimensions of the mashup makers as influenced by the requirements of the mashups users wish to create and use.

3.1 Processing Transparency

Mashup execution can take place on a client or on server infrastructure. First referenced in Section 2.1, processing is mentioned here again since, even though linked to the infrastructure provided by the mashup maker, decisions such as execution location and installation requirements impact the execution of the mashup.

This is an important dimension as the location of processing and the nature of the optimisations that can be made during the deployment and execution of the mashup must be made available through the mashup maker due to the tight coupling typical between mashups and their makers.

Classes: Execution - Client, Server; Run Time Installation - None, Browser-Add-on, Client

3.2 Location Transparency

Resources to be included in a mashup may be local to the machine on which the mashup is being created or may be remote. Each of these cases requires significant mashup maker design decisions to be made. To enable a mashup maker to take advantage of local resources as-is requires, at a minimum, that portions of the mashup maker environment be deployed to the local machine; elements of the processing must be located on the local machine. One mechanism used to overcome this limitation is enabling users to upload local resources to central mashup maker environments, making them available then as remote resources. The disadvantage here is that associated with a frequently changing local resource – how often will a user have to upload it to make sure the most recent data is included in mashups?

Selecting the locations of resource to be integrated is an important dimension since it reflects on the infrastructure required both at the server and the client.

Classes: Remote, Local, Both

3.3 Resource Access

Some resources (local and remote) may require authentication before their use. This authentication may require preregistration to gain access to the data and/or services. Simple facilities within the mashup maker may allow for integration of those authentication requirements in the calls made to those resources. Other, more complex, authentication requirements are required to access other resources, extending to the use of logins and certificates. The mashup maker may also provide facilities so that users, authenticated with the mashup maker, can gain automatic access to those resources, negotiating the authentication procedures required by individual resources.

Authentication is an important factor when designing a mashup maker. It permeates the tool and is an explicit decision that designers must take early in the process.

Classes: Anonymous, Login, Certificate

3.4 Resource Use

Once the user is authenticated by a resource he/she must have authorisation to carry out the specific activities required for the execution of the mashup. Authorisation and access control can be advertised by resources and thus the negotiation of the capabilities can be

automated based on policies advertised and authentication presented. The mashup maker may also have capabilities to degrade gracefully in cases where the authorisation to carry out a specific request is denied or limited.

This is an important dimension as enabling it within a mashup maker, and deploying it to resultant mashups, requires significant investment in infrastructure and core functionality. For the purposes of this design space we have simplified the classes to Access Control and Trust. How do third party resources trust the current user of a deployed mashup, and within a collaborative environment where data and functionality may be shared between users of a mashup maker environment, how do users trust one another? This can be done either through explicit trust mechanisms or through access control.

Classes: None, Access Control, Trust

3.5 Integration Level

Integration within mashups can take place at the data layer, business layer, or at the presentation layer. Of course there is the possibility that a mashup maker may provide support for combinations of layers.

This is an important dimension since it affects the core services to be offered by the mashup maker in enabling data access, integrating business log, and/or working with UI components in composing a user interface.

Classes: None, Data, Functionality, Application, Web Page Customizations/Clipping, Combinations

3.6 Reuse

Reuse is a key metric of the success of a mashup. This reuse requirement reflects directly on the functionality offered by the mashup maker in terms of how it enables the reuse of the mashup either through making the developed mashup available for reuse by different users with the same data sets, or through the provision of mashups as templates for use by other users with other sets of data.

This is an important dimension to be considered when designing mashup makers as the infrastructure required to store the mashups for either download or direct reuse or to be made available for the users to plug in their own data sets is significant and requires significant design effort both in terms of how the functionality is presented to the user and how to enable discovery of the created mashups.

Classes: None, As Templates, Integrated Copy with Data, Explicit Application Upload

3.7 Connectivity

How explicit is the relationship between the mashup and the mashup maker in which it was created? If the mashup needs to be deployed on mobile platforms, how much does it rely on the mashup maker infrastructure to execute? Most mashup makers are deployed on fixed infrastructures, forcing the mashups themselves to be executed in the same manner.

As with other dimensions categorized here under the mashup, the connectivity required of the mashup while deployed will have a significant impact on the mashup maker itself. Introducing mobility adds requirements in terms of the mashup (and hence the mashup maker) supporting occasional connectivity, and potentially mobile devices.

Classes: Immobile, Partly, All; Continuous, Occasional

3.8 Portability

Once a mashup or a component is created, can it be used with other mashup tools? Portability can be supported through the application of a resource or widget standard or through the use of export features which will allow the functionality or data be used in another platform.

Portability is an important feature in the design of a mashup maker; standards decisions must be taken early as they can permeate the design and implementation.

Classes: None, Standards, Abstraction, Exports

4 THE DESIGN SPACE MAPPED TO MASHUP MAKERS

Rather than try to present all of the 13 technical dimensions in this section (with their relevant classes - 53) and the mashup makers reviewed (of which there are 45) in a single matrix, the mappings are themed to give examples of what dimensions mashup makers have implemented and the techniques used as follows:

1. User Support - Discovery, Mashup Reuse
2. Security - Resource Access, Resource Use
3. Flexibility - Resource Combination Logic, Resource Interaction, Communication, Extensibility
4. Infrastructure - Processing Transparency, Location Transparency, Integration Level, Connectivity

Throughout this section we provide references for those mashup makers which do not have publically accessible URLs, and provide URLs for those which do not have references. Also, please note that the mashup makers described in this section represent a summary of the mashup makers reviewed. For the complete tabulation of the results refer to APPENDIX A - Technical Dimensions within the Mashup Design Space.

4.1 User Support

Implementing Reuse and Discovery dimensions within a mashup maker provide support for the user in the creation of mashups and are grouped in Table 1.

Mashup Maker	Discovery	Reuse
Potluck	None	None
Sun WebSynergy	Individual	Explicit Application Upload
iGoogle	Community Supported	None
Open Mashups Studio	Individual	Explicit Application Upload
Mashup Demo	Community Supported	Integrated Copy with Data
Karma	Community Supported	Explicit Application Upload

WSO2 Mashup Server	Individual	As Templates
Serena Business Mashup	Community Supported	As Templates

Table 1 - User Support Dimensions in Mashup Makers

Potluck [6] does not provide for resource discovery or reuse.

Sun WebSynergy² provides the ability for the individual to search for resources in its environment. Resources are created through the NetBeans creation of Portlets to JSR specifications. Mashups are deployed from the Sun Java IDE through to the Sun Application Server/Glassfish through Explicit Application Upload.

iGoogle³ provides Community Support for resource (gadget) discovery. It does not provide for mashup Reuse (though there is a comprehensive discovery facility for adding gadgets to existing personalised views).

Open Mashups Studio⁴ provides Discovery on an Individual basis. Mashups reuse is enabled through Explicit Application Upload.

[7] created a mashup maker which has been given the name "Mashup Demo" for this description. It uses standard Java portal technology to implement the graphical interface. In particular, resources are implemented as portlets and the mashup maker provides Community Supported Discovery. Reuse is supported through the provision of mechanisms for users to share Integrated Copy with Data.

Karma [8] learns from others' choices and preferences to provide suggestions during the discovery process from a central repository of resources, based on profiles and history. Created mashups are explicitly uploaded for subsequent execution by other users.

WSO2⁵ provides Discovery for the Individual. Mashups are reused as templates, particularly where user-based input is required for the execution of the mashups. In this way WSO2 looks more like traditional system integration but is included here as a mashup due to its user focus.

Serena Business Mashup⁶ enables Community Support Discovery through user ratings. Reuse is provided for through the use of Templates.

4.2 Security

Resource Access and Resource Use are grouped in Table 2 as elements of security support in mashup makers.

Mashup Maker	Resource Access	Resource Use
Aptar Data Mashup Integration	Anonymous	None
Dabble DB	Login	Access Control

Table 2 - Security Dimensions in Mashup Makers

² blogs.sun.com/theaquarium/entry/project_websynergy_liferay_and_glassfish

³ www.igoogle.com

⁴ www.open-mashups.com

⁵ wso2.com/products/mashup-server

⁶ www.serena.com/products/sbm

Apatar⁷ provides for accessing local and remote open services but does not entail the securing of resultant mashups that might be created using secure/sensitive data. Given that there is no user identity, authorisation is not in-built in the mashup maker.

Dabble DB⁸ provides an authentication feature through logins, thereby identifying users. It also provides for feature and user driven access control.

4.3 Flexibility

Flexibility is a measure of how a mashup maker contends with heterogeneity in resource integration. Table 3 groups resource combination logic, resource interaction, resource communication mode, and extensibility together as reflections on flexibility in mashup makers.

Mashup Maker	Resource Combination Logic	Resource Interaction	Communication	Extensibility
RatchetSoft Ratchet-X	Code	API	Synchronous	Fixed
Karma	Data Federation	Configuration	Synchronous	Fixed
LiquidApps	Pipes & Filters	API	Synchronous	Fixed
mashArt	Pipes & Filters	Event	Combination	None
IBM Mashup Center	Code	Event	Asynchronous	Customizable
JackBe	Pipes & Filters	API	Combination	Customizable

Table 3 – Flexibility Dimensions in Mashup Makers

RatchetSoft's Ratchet-X⁹ is a pure client installation but reuses plug-ins and app spaces for connectivity to other resources. Commander acts as the communication point on the desktop between web services and other remote resources and the local applications. C-like code is written by the user to integrate resources and from this the Resource Interaction is through Code. Resource Communication is synchronous and Extensibility is Fixed through the provision of plug-ins and app spaces which the user cannot develop for his own use.

Karma [8] uses a spreadsheet model for integration and federates the data into these spreadsheets from a central repository and query-type Resource Interaction which is configured by the user. Data is fetched synchronously and Extensibility is Fixed and provided for by pre-developed Wrappers with connectivity to other systems such as OpenKapow and Fetch.

LiquidApps¹⁰ uses a Pipes and Filters approach to mashup creation. Even though the UI looks like it offers Configuration capabilities for Resource Interaction, on further inspection it becomes apparent that the configuration is very code-like.

⁷ www.apatar.com

⁸ www.dabbledb.com

⁹ www.ratchetsoft.com

¹⁰ www.liquidappsworld.com

mashArt [9] proposes the merging of UI integration into Universal integration – enabling the integration of data, services, processes, and UIs. As such Data Flow Modelling is used as a Pipes and Filters implementation using events to kick off the integration process during the runtime. Events are fired and data is fetched/updated synchronously.

IBM Mashup Center¹¹ enables Resource Interaction through Events. A code engine runs in the server to react to the events. Resource Interaction is asynchronous and Extensibility is provided for by Connectors which can be developed to allow the inclusion of other resources.

In JackBe¹² Pipes and Filters are used for Resource Combination Logic. Resource Interaction is implemented through an Eclipse plug in that allows full access to resources and uses EMMML (an XML variant) to describe resources and their interactions. Combinations of Resource Interaction mechanisms are enabled through connectors which allow for data fetch and events. Extensibility is provided for through the use of Presto Connectors to databases, spreadsheets, etc.

4.4 Infrastructure

Many factors influence the infrastructure required to support a mashup maker during the design time and run time. Factors such as processing and location transparency, integration level, connectivity required of the mashup, and the portability provided for mashups are shown in Table 4.

Mashup Maker	Processing Transparency	Location Transparency	Integration Level	Connectivity	Portability
SAP Research EMAP	Server; Server; Combination	Both	Data	Immobile; Continuous	None
SnapLogic	Server; Server; None	Remote	Data	Immobile; Continuous	None
Extensio	Server; Server; Combination	Both	Data	Partly; Occasional	None
JackBe	Server; Server; None	Both	Data	Immobile; Continuous	None
Smashup	Server; Server; None	Remote	Code	Immobile; Continuous	None
Denodo Platform	Client; Server; App Add-on	Both	Combination	Immobile; Continuous	Standards

Table 4 - Infrastructure Dimensions in Mashup Makers

SAP Research EMAP [10] mashups are built and executed on the Bijoux web server. Mashups can be run on the server using a browser, but also comes with a client installation which enables the integration of both Local and Remote resources.

¹¹ www-01.ibm.com/software/info/mashup-center/?ca=fv1001&me=feature3&re=lm11

¹² www.jackbe.com

Integration is carried out at the Data level. Mashup users are assumed to be at a computer and continually connected to the mashup maker environment. Portability is not supported.

SnapLogic¹³ is entirely server based with no client installation required for mashup installation, mashup creation, or execution. Given that environment it will only work with Remote resources. As before, mashup users are assumed to be at a computer and continually connected to the mashup maker environment. Portability of mashups is not enabled.

Creation of mashups in Extensio¹⁴ is carried out using a browser which accesses Extensio Studio. Execution is on the server and interaction with the mashups is carried out through the browser or through application add-ons to desktop applications such as Microsoft Outlook and Excel. Based on the application add-ons, resources can be both Local and Remote, integrating at the Data level. Also, through the use of channels the mashup execution can be mobile or immobile and the connection can be Occasional. Portability is not supported.

JackBe mashups are created and executed on the Server. Presto Wires in the browser platforms is provided for creation of mashups but this can be extended through the use of an Eclipse plug-in for direct coding access. Local and Remote resources can be accessed. Mashups are assumed to be Immobile and their connection to the server is expected to be Continuous. Portability is not enabled.

Smashup mashups are created and executed on the server with no client installation required. Remote resources are integrated using a Code Engine which uses XSLT transformations which lift and lower RDFa annotated REST services to be included in the mashups [11]. Mashups are assumed to be Immobile and their connection to the server is expected to be Continuous. Portability is not enabled.

Denodo mashups are created on the client through incremental features which are implemented in a browser add-on. Local and Remote Resources are accessible, and integration is enabled through a Combination of data integration and web page clipping. Mashups are assumed to be Immobile and their connection to the server is expected to be Continuous. Widgets can be exported as a JSR-168/JSR-286 portlets, Microsoft WebPart, and through OpenAjax.

5 RELATED WORK

In [1] the authors group 4 mashup tools in terms of 4 large-grained characteristics with associated lower level properties: Component Model (type, interface, and extensibility), Composition Model (output type, orchestration style, data passing style, instance-based or continuous, exceptions and transactions), Development Environment (interface paradigm, target users, system requirements), and Runtime Environment (deployment style, runtime location, system requirements, scalability). The lower level properties

¹³ www.snaplogic.com

¹⁴ www.extensio.com

provided input to the dimensions used in our design space as can be seen with similar naming. We further extended the dimensions to have specific classes, and also extended our review to 45 mashup makers.

In [2] the authors carried out a market overview of mashup tools and classify mashup makers according to two main axes: Functionality/Property and Target Group. Within the Functionality axis they further classify according to Catalogue (with Adapter and repository as sub classifications), Editor (with transformation/aggregation and presentation layer as sub classifications). Within the Target Group they further classify with respect to Consumer and Enterprise Mashups). This classification also finds its way into our dimensions both directly and indirectly, though with a lower level of granularity (for example, Consumer vs. Enterprise is determined by such dimensions as Authentication, Authorisation, and Software Engineering Support). It is our opinion that such features will become part of the consumer mashup tool environment making them indistinguishable from enterprise mashups as requirements extend from the enterprise into the open internet.

In [12] the authors identify 7 patterns of mashups by reviewing mashups created and made available on the web. Elements of these patterns have found their way into our dimensions (Processing and Integration Level). We consider both the mashup and the mashup maker as direct contributors to the design space.

We took inspiration from [3] as the authors define a design space for wireless sensor networks. The criteria used in creating our design space share a grounding with those defined in [3].

A summary of this work is to be found in [13] currently under submission.

6 DISCUSSION

In this report the important dimensions that bound the design space of mashup makers are described. Core decisions must be made around the mashup maker provisions, but also more subtle influences must be taken into account relating to the use and execution of mashups and the environment into which the mashup maker and the resultant mashups will be deployed.

These identified dimensions are mapped to existing mashup makers, showing examples of how those dimensions are implemented by mashup makers in the provision of services to their user groups.

Analysing those dimensions within themes allows the identification of challenges within mashup maker research, of areas not addressed by current mashup makers, and illustrations of how people developing mashup makers are addressing the concerns common in system integration, regardless of the target user group. Issues such as resource heterogeneity, sensitive data and processes, and avoiding proliferation of badly developed integration are just some of the objectives common between mashup makers and traditional system integration.

7 REFERENCES

1. Yu, J., et al., Understanding Mashup Development. IEEE Internet Computing, 2008. **12**(5): p. 44-52.
2. Hoyer, V. and Fischer, M., Market Overview of Enterprise Mashup Tools, in Proceedings of the 6th International Conference on Service-Oriented Computing. 2008, Springer-Verlag: Sydney, Australia.
3. Romer, K. and Mattern, F., The Design Space of Wireless Sensor Networks. IEEE Wireless Communications, 2004.: p. pp. 54-61.
4. Zang, N., Rosson, M.B., and Nasser, V., Mashups: who? what? why? in CHI '08 extended abstracts on Human factors in computing systems. 2008, ACM: Florence, Italy.
5. Ko, A.J., Myers, B.A., and Aung, H.H., Six Learning Barriers in End-User Programming Systems, in Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing. 2004, IEEE Computer Society.
6. Huynh, D.F., Miller, R.C., and Karger, D.R., Potluck: Data mash-up tool for casual users. Web Semant., 2008. **6**(4): p. 274-282.
7. López, J., et al., A Component-Based Approach for Engineering Enterprise Mashups, in Proceedings of the 9th International Conference on Web Engineering. 2009, Springer-Verlag: San Sebastian, Spain.
8. Tuchinda, R., Szekely, P., and Knoblock, C.A., Building Mashups by example, in Proceedings of the 13th international conference on Intelligent user interfaces. 2008, ACM: Gran Canaria, Spain.
9. Daniel, F. and Matera, M., Turning Web Applications into Mashup Components: Issues, Models, and Solutions, in Proceedings of the 9th International Conference on Web Engineering. 2009, Springer-Verlag: San Sebastian, Spain.
10. Gurrum, R., Mo, B., and Gueldemeister, R., A Web Based Mashup Platform for Enterprise 2.0, in Proceedings of the 2008 international workshops on Web Information Systems Engineering. 2008, Springer-Verlag: Auckland, New Zealand.
11. Lathem, J., Gomadam, K., and Sheth, A.P., SA-REST and (S)mashups: Adding Semantics to RESTful Services, in Proceedings of the International Conference on Semantic Computing. 2007, IEEE Computer Society.
12. Lee, C.-J., et al., Toward a new paradigm: Mashup patterns in web 2.0. WSEAS Trans. Info. Sci. and App., 2009. **6**(10): p. 1675-1686.
13. Fox, R. and Hauswirth, M., A Unifying Model for Mashups. 2010, Digital Enterprise Research Institute, National University of Ireland, Galway.

APPENDIX A - TECHNICAL DIMENSIONS WITHIN THE MASHUP DESIGN SPACE

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
Apatar Data Mashup Integration	None	Explicit Application Upload	Anon	None	Pipes & Filters	Synch	Fixed	Client; Client; Client Installation	Both	Config	Data	Immobile; Occasional	None
Bungee Labs Bungee Connect	None	Integrated Copy with Data	Access Control ¹⁵	None	Code	Synch	Fixed	Client; Server; None	Both	API	Data	Immobile; Continuous	None
C3W	None	None	Anon	None	Pipes & Filters	Synch	None	Client; Client; Client Installation	Remote	Config	Custom/ Clipping	Immobile; Continuous	None
Chickenfoot	None	Explicit Application Upload ¹⁶	Anon	None	Code	Synch	Custom ¹⁷	Client; Client; Application Add-on ¹⁸	Remote	API	Custom/ Clipping	Immobile; Continuous	None
CHIP	None	None	Anon	None	Code ¹⁹	Synch	None	Client; Client; Client Installation	Remote	API	Combinations (Data, Clipping)	Immobile; Continuous	None
d.mix	Community ²⁰	Integrated Copy with Data	Anon	None	Code	Synch	None	Server; Server; None	Remote	Config	Custom/ Clipping	Immobile; Continuous	None
Dabble DB	None	Integrated Copy with Data	Login	Access Control	Data Federation	Synch	None	Server; Server; None	Both	Config	Data	Immobile; Continuous	None
Dapper (Dapp Factory)	None	Integrated Copy with Data	Login	None	Pipes & Filters	Synch	None	Server; Server; Application add-on ¹⁸	Remote	Config	Custom/ Clipping	Immobile; Continuous	Exports ²¹

¹⁵ This appears to be overlaid on the product and, in some cases, delegated to the hosting company rather than integrated

¹⁶ To a scripts wiki

¹⁷ Through the use of Includes

¹⁸ To the browser

¹⁹ Using a set of three API commands automatically generated as a result of drag and drop

²⁰ Supported by example

²¹ Using Transformers

DERI Technical Report 2010-06-02

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
Denodo Platform	None	Integrated Copy with Data	Login	Access Control	Data Federation	Asynch	Custom	Client; Server; Application add-on ¹⁸	Both	Config	Combinations (Data, Clipping)	Immobile; Continuous	Exports ²²
DERI Pipes	Individual	Integrated Copy with data ²³	Anon	None	Pipes & Filters	Synch	None	Server; Server; None	Remote	API	Data	Immobile; Continuous	None
Extensio	None	Integrated Copy with Data	Login	Access Control	Pipes & Filters	Asynch	Custom	Client; Server; Application Add-on ²⁴	Both	API	Data	Partly ²⁵ ; Occasional	None
Ezweb	Community Supported ²⁶	As Templates	Login	None	Pipes & Filters	Synch	None	Server; Server; None	Remote	Config	Combinations (Gadget creation: Data, Wiring: Functionality)	Immobile	None
Hunter Gatherer	None	None	Anon	None	None	Synch	None	Server; Server; None	Remote	None	None	Immobile; Continuous	None
IBM DAMIA	Individual	Integrated Copy with Data	Login	Access Control	Pipes & Filters	Synch	Fixed	Server; Server, None	Both ²⁷	API	Data	Immobile; Continuous	None
IBM Mashup Centre	Individual	Integrated Copy with Data	Login	Access Control	Code	Asynch ²⁸	Custom ²⁹	Server; Server; None	Both	Event	Data	Immobile; Continuous	None
IBM MARIO	Community ³⁰	Integrated Copy with Data	None	None	Pipes & Filters	Synch	None	Server; Server; None	Remote	Config	Data	Immobile; Continuous	None

²² JSR-168/JSR-286 portlets, WebPart, OpenAjax

²³ Can then be modified

²⁴ Depending on level of interaction required provides None, Excel add-in, Outlook add-in

²⁵ Via mobile extension via SMS

²⁶ Using tagclouds

²⁷ Local files uploaded to server

²⁸ Using AJAX

²⁹ Using a plug-in architecture

³⁰ Tag based

DERI Technical Report 2010-06-02

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
iGoogle	Community ³¹	None	Login	None	None	Asynch	None	Server; Server; None	Remote	None	None	Immobile; Continuous	None
InfoBeans	None	None	Anon	None	Code ³²	Synch	None	Client; Client; Client Installation	Remote	API	Data	Immobile; Continuous	None
Intel Mash Maker	Individual	Integrated Copy with Data	Login	None	Code ³³	Synch	None	Client; Client; Application Add-on ¹⁸	Remote	Event	Combinations (Data, Clipping)	Immobile; Continuous	None
JackBe	Community ³⁰	Integrated Copy with Data	Login	Access Control	Pipes & Filters	Synch	Custom ³⁴	Server ³⁵ ; Server; None ³⁶	Both	API	Data	Immobile; Continuous	None
KapowTech	None	Integrated Copy with Data ³⁷	Login	Access Control	Code	Synch	Fixed	Client; Server; None	Remote	API	Data	Partly; Continuous	None
Karma	None ³⁸	Explicit Application Upload ³⁹	Anon	None	Data Federation	Synch	Fixed ⁴⁰	Client; Client; Client Installation	Both	Config	Data	Immobile; Continuous	None
LiquidApps	Individual	Explicit Application Upload	Anon	None	Pipes & Filters	Synch	Fixed	Client; Server; None	Both	API	Data	Partly; Continuous	Exports ⁴¹

³¹ Through ratings and other users' deployments

³² HyQL

³³ Using a Widget API

³⁴ Using Presto Connect APIs

³⁵ Although an Eclipse plug-in is provided for direct code access

³⁶ Presto Wires provided through the Browser

³⁷ Uploaded as a web service

³⁸ But provides suggestions of integration methods from others use

³⁹ Of generated HTML files

⁴⁰ Using Wrappers for different sources

⁴¹ Through code generation

DERI Technical Report 2010-06-02

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
MARGMASH	None	None	Anon	None	Pipes & Filters	Synch	None	Server; Server; None	Remote	Code	Data	Immobile; Continuous	None
Marmite	None	None	Anon	None	Pipes & Filters	Synch	None	Client; Client; Application Add-on ¹⁸	Remote	API	Combinations (Data, Clipping)	Immobile; Continuous	None
mashArt	None	None	Anon	None	Pipes & Filters	Synch	Fixed ⁴²	Server; Server; None	Remote	Event	Application	Immobile; Continuous	None
Mashup Demo	Community ⁴³	Integrated Copy with Data	Login	Access Control	Data Federation	Synch	Fixed ⁴⁴	Server; Server; None	Remote	Event	Combinations (Data, Clipping)	Immobile; Continuous	None
MIXUP	None	None	None	None	Code	Synch	None	Server; Client; Client	Both	Event	Data	Immobile; Continuous	None
NetVibes	Community ⁴⁵	Integrated Copy with Data	Login	None	None	None	None	Server; Server; None	Remote	None	None	Immobile; Continuous	Abstract.
Open Mashups Studio	Individual	Explicit Application Upload	Anon	None	Pipes & Filters	Synch	Fixed ⁴⁶	Client; Client; Application Add-on ¹⁸	Remote	API	Data	Partly; Continuous	Exports ⁴¹
PageFlakes	Individual	None	Login	None	None	Asynch	None	Server; Server; None	Remote	None	None	Immobile; Continuous	None
PotLuck	None	None	Anon	None	Data Federation ⁴⁷	Synch	None	Server; Server; None	Remote	API	Data	Immobile; Continuous	None

⁴² Using Adaptors

⁴³ From user tagging, rating

⁴⁴ Using Source Adaptors

⁴⁵ Through user ratings

⁴⁶ Library Imports

⁴⁷ Query unions used to fetch data

DERI Technical Report 2010-06-02

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
RachetSoft Rachet-X	None	As Templates ⁴⁸	Anon	None	Code ⁴⁹	Synch	Fixed ⁵⁰	Client; Client; Client Installation	Both	API	Data	Immobile; Continuous	None
RSSBus	None	Integrated Copy with Data	Login	Access Control	Pipes & Filters	Synch	Custom	Server; Server; Application Add-on ⁵¹	Both	Config	Data	Immobile; Continuous	Exports ⁵²
SAP Research EMAP	Community ⁵³	As Templates	Login	None	Code ⁵⁴	Asynch	Custom ⁵⁵	Server; Server; Client Installation	Both	Event	Data	Immobile; Continuous	None
SAP Research Rooftop	Community ⁵³	None	Anon	None	Pipes & Filters	Synch	None	Server; Server; None	Remote	Config	Data	Immobile; Continuous	None
Serena Business Mashup	None	As Templates	Login	Access Control	Code	Synch	Custom	Client; Server; None	Remote	Event	Combinations (Data, Functionality)	Partly; Continuous	None
Smashup	None	None	Anon	None	Code ⁵⁶	Synch	None	Server; Server; None	Remote	Config	Data	Immobile; Continuous	None
SnapLogic	Individual	Integrated Copy with Data	Login	Access Control	Pipes & Filters	Synch	Custom	Server; Server; None	Remote	Config	Data	Immobile; Continuous	None
Sun WebSynergy	Individual	Explicit Application Upload	Login	Access Control	Code ⁵⁷	Synch	Custom ⁵⁸	Server; Server; None	Both	Event	Data	Immobile; Continuous	None

⁴⁸ For use with different instantiations of the same application set

⁴⁹ App spaces coded using regwin

⁵⁰ Through plug ins to Commander

⁵¹ Browser and Excel

⁵² Using Output formatters

⁵³ Using ratings

⁵⁴ Code generated when user defines widget interactions in wiring

⁵⁵ By Importing and creating widgets

⁵⁶ XSLT transformations for lifting and lowering

⁵⁷ Java NetBeans

DERI Technical Report 2010-06-02

Mashup Maker	Discovery	Reuse	Resource Access	Resource Use	Resource Combi. Logic	Comm.	Extens.	Processing	Location	Resource Interaction	Level	Connect.	Portability
Ubiquity	Individual	Explicit Application Upload	Anon	None	Code ⁵⁹	Synch	Custom ⁶⁰	Client; Client; Application Add-on ¹⁸	Both	API	Data	Immobile; Continuous	None
WebViews	None	As Templates ⁶¹	Login	None	Code ⁶²	Synch	None	Server; Server; None	Remote	API	Custom/ Clipping	Partly; Continuous	None
WorkLight	Individual	As Templates	Login	Access Control	Code ⁶³	Synch	Custom	Client; Server; Client Installation	Both	API	Data	Partly; Continuous	None
WSO2 Mashup Server	Individual	As Templates	Login	Access Control	Code	Synch	Custom ⁶⁴	Server; Server; None	Remote	API	Combinations (Data, Functionality, Clipping)	Immobile; Continuous	None
Yahoo Pipes	Individual	Integrated Copy with data	Anon	None	Pipes & Filters	Synch	Fixed ⁶⁵	Server; Server; None	Remote	Config	Data	Immobile; Continuous	None

⁵⁸ By creating new portlets

⁵⁹ XUL

⁶⁰ By adding new scripts

⁶¹ Allowing for parameterisation

⁶² We page interaction stored and replayed

⁶³ Worklight SDK

⁶⁴ Using Javascript Host Objects

⁶⁵ Possibly customizable through reusing pipes as subpipes