5th International Conference on Mass Customization and Personalization in Central Europe (MCP-CE 2012)

Customer Co-Creation in Central Europe
September 19-21, 2012, Novi Sad, Serbia

CONFIGURE YOUR ENTERPRISE APPLICATION

Ana Marija Ćirić, Ivan Dobrić, Stefan Martinov, Stefan Seiler
Danulabs, Novi Sad, Republic of Serbia

Abstract: *Nowadays, ERP software is necessary for successful business, which puts small and medium enterprises in a difficult position, since these software solutions are expensive. A small business might not need all of the components of such complex software, but has to pay for them, nonetheless. In this paper we present a solution which relies on Software as a Service model and mass customization to offer enterprise resource planning to smaller companies, at a reasonable price. This is attainable through, for example, reusing existing applications and customizing them to serve new purposes, as well as building modular applications on a single database, which enables combining the modules from different applications, thus creating new, customized applications with specific purposes.*
Key Words: ***Mass Customization, Software as a Service, Enterprise Resource Planning, Application Iteroperability***

1. THE PROBLEM

Since the beginning of the 20th century, mass production has been a successful business model in various industries and has been enabling fairly cheap production for many decades. The problem lies within the fact that whenever a production line switches from one product to another, setup costs arise. These costs apply regardless of the type of product we are discussing: cars, clothing, furniture or software. Setup costs include labor costs, tooling costs and time costs. Unlike software products, material goods, especially large ones, induct additional storage costs - storage space is expensive. These additional expenses force production of only popular and widely accepted products [1], which leaves the customers with specific requirements unsatisfied.

2. THE SOLUTION

Mass customization is an innovative and powerful business model, which applies to production in both manufacturing and service industries. Its main objective is to enable mass production, whilst providing the means to customize each product to customers' specific needs.

The pros of mass customization are more than obvious from all stated above. By implementing mass customization in the production process, manufacturers can satisfy the requisites of more customers. Among other benefits [2] of mass customization, we can count in increased market share, increased customer knowledge, reduced order response time, reduced manufacturing cost, and increased profit. Software products, which require maintenance, are more easily managed if mass customization was introduced to the production process - changes made in one place, take effect in all the customized products.

Every concept which possesses such notable merits, must, naturally, have a downside. While the production costs are significantly lower with mass customization, the preparations which are required to enable it, take their toll. Initial investments are commonly much higher than in cases where every product has a unique design and production process. It is very important to define what kind of market is aimed at, and thus determine whether these initial investments [3] are worthwhile.

3. SOFTWARE MASS CUSTOMIZATION

Production of enterprise software solutions is a particularly fertile ground for mass customization. In the era of information technology, even the smallest businesses are virtually forced to use software to run their operations effectively. This phenomenon puts small and medium enterprises (SMEs) in a serious disadvantage - production, installation and maintenance of the ERPs presents a gargantuan expense, in comparison to their budget.

A convenient solution to this problem is for SMEs to use ERPs developed in Software-as-a-Service (SaaS) model. SaaS software solutions have a easily scalable, single-instance, multi-tenant architecture, which, if equipped with customization possibility, bring benefit to both producer and customer sides. Producers save precious time, otherwise consumed by development, installation and maintenance of software for each customer separately, through implementing and maintaining only one application, which is available to users via web. This implies that no installation is needed, as the users will access the application through, for example, a web browser. Saving resources by utilizing this model, enables software producers to offer the application to their customers for a much reasonable price. SMEs can, thus, use powerful ERP software, for a price they can afford.

Although all customers are using essentially the same ERP software, the interface they see may differ. Namely,

a customer may want some adjustments to the look-and-feel of the user interface, such as branding (customers logo), different layout or special labeling. Furthermore, a customer might have special requests regarding the functionality of the application, whether that is a need for additional functions, or a need to hide the redundant functions. Customization can be performed on both usage- and production side. Usage side customization is performed by the customer, using a customization interface, if available. Production side customization is performed by the software producer, according to customer's requirements.

4. THE REAL-LIFE EXAMPLE OF SOFTWARE MASS CUSTOMIZATION

To describe the possibilities of mass customization in the field of SaaS better, we will first present the concept of B-op platform. B-op platform is a cluster of SaaS model business applications, gathered around the omni-database - a large central database. These applications, called B-apps, can be developed by various parties, provided that certain B-op requirements are met.

In order to use any of the B-apps, a company or a person (which we will address as customer) has to register with the B-op platform. Upon registration, the customer can access their corporate dashboard, which offers user administration, account settings, payment overview and so on. Once the customer has registered with B-op, they can browse the app market and subscribe to B-apps. These apps will also appear on the corporate dashboard, as well as on personal dashboards of all users, to whom the customer has granted app usage. The corporate dashboard itself, is the first customization point. The customer can upload their logo, which appears next to the B-op logo. Thus, the customer has a co-branded dashboard, which all their employees (users) will see when logged in.

As it has been stated before, B-apps, although running within the B-op platform, are developed by various parties. Nevertheless B-apps are, essentially, SaaS model business applications which belong to certain categories (e. g. CRM, accounting, logistics) and can be combined to application chains ranging from simple business processes to a custom-tailored ERP for each customer. This combination of apps is enabled through subscription packaging - subscription models for multiple apps are bundled into one special subscription model. Subscribing to this model grants the customer usage of all applications included, which can, from this point of view, be considered modules of one complex ERP.

B-op offers a *Package Assistant* to all of its customers - a module which helps choosing suitable subscription packages, in accordance to customer's needs. Through this wizard-like program, a customer fills out a questionnaire about what kind of business they run, how many users they have, whether they need human resource management, project management, accounting, etc. After that, the package assistant offers the customer a small number of subscription packages, which are the most suitable to help them conduct their operations most efficiently.

The possibility to use apps from different vendors as a package comes from the app compatibility, which is based upon the fact that all apps use the omni-database as data model.

4.1. The Compatibility Layer

Other important topic of the mass customization problem references to the compatibility layer customization which is in praxis most commonly represented as a huge multidimensional database shared amongst other applications.

The compatibility layer enables us to reuse some parts of the database to build different applications. As the number of applications that use the database grows, the database itself advances. The reason for this is that the given applications in most cases do not have completely similar structure. Therefore each application can contribute to the database with its own diversity.

Enabling development of different applications on a common database is a process that requires constant attention and support. Our common database would provide existing data model if there is a table structure which can support a certain application, extended data model with certain modifications, or completely new part which has nothing to do with the data constructs that already lie somewhere within the database.

In order to enable application development over the same database, the database has to be very intuitive and easy to comprehend, since the number of tables in it can grow to the extreme dimensions where, without well-planned organization, orientation would be impossible. One way for resolving this kind of issue is using table namespaces with a strict standard in the table nomenclature.

Huge databases tend to be slow, non-scalable and prone to errors. Problems like these are not unusual and there is a well-known cure for them: database replication. Similar to the one seen in the online multiplayer game databases, data replication enables clients to read data from any database replica while updates have to be executed at all available replicas. Thus, reads can be distributed among the replicas leading to reduced response time and scalability. Furthermore, the system is fault-tolerant as long as a replica is available [4].

4.2. Application Faceting

Application facets are another aspect of B-op customizing. This feature enables us to offer multiple B-apps, which are, from the functionality point of view, one and the same, but adapted to different groups of users. This is achieved by using the B-op ontology which describes entities kept in the omni-database and their relationships to each other. As the database grows more complex, so does the ontology. This ontology is, other than for application faceting, useful for describing the database to parties which develop B-apps, as it can easily scale up to more than a thousand tables and become too complex to comprehend to anyone but the architect himself.

Naturally, every B-app should have one or more configuration files which contain textual content of the labels from the user interface. Extracting label contents outside of the source code is common practice as it, aside

from keeping the code clean, enables software internationalization. Although B-app producers have the freedom of choosing the format of their label content files, they also have the Nomenclature framework at their disposal.

The *Nomenclature Framework* offers developers a mechanism to specify which text corresponds to which ontological concept within the B-op ontology, for a language specified by an ISO language code. These ontological concepts represent database tables, their columns and relationships with one another. Upon entering text for each desired concept of the ontology (naturally, not all of them have to be used for every application), the user gets a generated XML nomenclature file for their application. The framework provides the possibility to read label contents from the generated file for specified ontology concept and specified language code. If the application is developed in a programming language which is not supported by the framework, developers must implement label content acquisition by themselves. This is the reason why XML format is used for nomenclature files - it is a well known standard. Using Nomenclature framework enables developers to easily replace label content on the user interface, by simply switching or modifying nomenclature files, without changes to the source code. This brings us to clarification of application faceting - one application can pose as another functionally same application, with different nomenclature, adapted to different groups of users, by using different nomenclature files.

We will illustrate application facets in the following scenario: Within the database, there is a group of tables designed for project management. Naming only a few, in order to keep this example simple, we have:

- *Project* (with columns *Name*, *Description*, *Status*, etc.)
- *BusinessTask* (with columns *Name*, *Description*, *Identification Number*, *Estimeted Completion Time*, etc.)
- *Feature* (with columns with columns *Name*, *Description*, *Identification Number*, *Deadline*, etc.)
- Developer Task (with columns *Name*, *Description*, *Identification Number*, *Completion Time*, *Deadline*, etc.)

The part of B-op ontology representing these tables is shown in Figure 1.

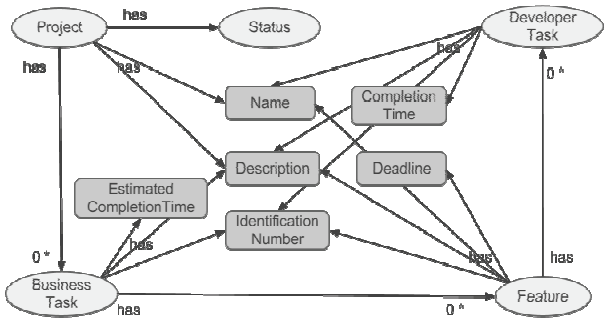

Figure 1. A part of the B-op ontology, project management domain

The first part of this scenario is B-app *DanuTask* - a project management application. The names in the data model suit the semantics of this application: A product manager creates a new project and adds business tasks, which represent non-technical task specifications, corresponding to user requirements. These tasks are later reviewed by a technical manager and from them, features which need to be implemented, are produced. Features are devised into developer tasks, very specific, technically expressed tasks, which are later assigned to developers. Please note that this is a simplified explanation of the data model and business processes within the application. After entering nomenclature content, we get an XML file, such as the following one:

```
<?xml version="1.0" encoding="UTF-8"?>
<Nomenclature app="DanuTask">
 <Entry onto="Project">
  <Content lang="en">Project</Content>
 </Entry>
 <Entry onto="BusinessTask">
  <Content lang="en">Business task</Content>
 </Entry>
 <Entry onto="Feature">
  <Content lang="en">Feature</Content>
 </Entry>
 <Entry onto="DevelopmentTask">
  <Content lang="en">Development task</Content>
 </Entry>
</Nomenclature>
```

The second part of the scenario is a conceptual B-app for production tracking in a furniture workshop. We will call it *FurniShop.* Sales person of a furniture workshop collects requirements from their customers and uses *FurniShop* to specify what needs to be made, e. g. a set of furniture or a single piece. We will call that a product. To create that product, a plan needs to be made. Plan items represent roughly specified steps which need to be taken in the process of product creation, like: design and carpentry. These are, then, devised into task bundles. For example, carpentry can be devised into cutting and molding, component fabrication, assembly, etc. After that, specific tasks are derived from the bundles, for example, component fabrication involves machine processing and shaping of the timbre, as well as sanding and smoothing the surfaces.

If we pay closer attention, we can see the similarity between these two applications. The data model is the same. To be precise, whole applications are practically the same, where the latter just needs to be adapted for a specific field of production. This adjustment can just be on the user interface side. In the following XML snippet, we can see the nomenclature file, which can be applied to the first application, thus, producing a whole new application for a specific purpose:

```
<?xml version="1.0" encoding="UTF-8"?>
<Nomenclature app="DanuTask">
 <Entry onto="Project">
  <Content lang="en">Product</Content>
 </Entry>
 <Entry onto="BusinessTask">
```

```
  <Content lang="en">Plan item</Content>
 </Entry>
 <Entry onto="Feature">
  <Content lang="en">Task bundle</Content>
 </Entry>
 <Entry onto="DevelopmentTask">
  <Content lang="en">Task</Content>
 </Entry>
</Nomenclature>
```

4.3. Application Fragmentation and Recombination

There are also cases where nomenclatures cannot offer the customizations that the customer needs. We introduce the concept of *application fragmentation* to overcome some of the issues that might occur when a customer needs a specific part of an application to be changed, added or upgraded.

Application fragmentation is a process of separating aspects or domains of a single large integral application, where every application fragment corresponds a single problem domain that the integral application solves. *Application fragmentation* allows the integral application to be much more customizable and extensible making modifications to the integral application faster and less expensive.

We have already developed a fragmented application *DanuTask* that is an extensible project management tool consisting of 4 application fragments: administrator fragment, business manager fragment, technical manager fragment and a developer fragment. Any of these fragments can be replaced by a custom application fragment that can interact with other fragments through the omni-database; thus conforming to the original integral application. This also allows the customer to select not only whole applications, but a set of application fragments that is a subset of the integral application that are most beneficial to his needs.

We can illustrate the practical use of the application fragmenting with the *DanuTask* application, where the customer requires a new custom-tailored module that manages the domain of catering named *catering officer*. We can reuse the underlying architecture of *DanuTask* with the required *nomenclature* changes for the administrator, business manager and technical manager fragment and replacing the developer module with the catering officer fragment. The interoperability is maintained on the omni-database level to ensure fragment integration. This is portrayed in Figure 2.

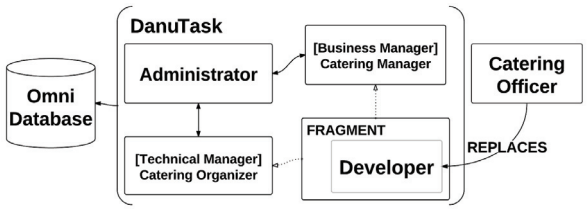

Figure 2. Application fragmentation and recombination

5. CONCLUSION

In order to explain the possibilities of mass customization in software production, we have made an example of how SaaS-based business applications, which are aimed at thousands of users, can be customized to suit each individual customer, better. The customization possibilities range from aesthetical intra-application customization, over functional customization to inter-application customization, enabled through compatibility layer. The B-op platform, which is exemplified in this paper, is a work in progress and these customization possibilities will be refined further, in the future.

6. REFERENCES

[1] C. Ardito, P. Buono, M. F. Costabile, R. Lanzilotti, A. Piccinno, B. R. Barricelli, S. Valtolina, "An Ontology-based Approach to Product Customization", *Third International Symposium on End-User Development 2011*, Torre Canne (Italy), June 2011

[2] R. Selladurai, "Mass Customization Strategy in Management and its Applications to Small Business", *Small Business Institute Directors' Association,* 2004.

[3] C. W. Krueger, *"Software Mass Customization",* BigLever Software Inc., March 2006

[4] Y. Lin, B. Kemme, M. Patino-Martinez, R. Jimenez-Peris, "Applying database replication to multi-player online games", *Fifth Workshop on Network and System Support for Games*, Singapore, October 2006.

CORRESPONDENCE

Ana Marija Ćirić, CBDO
Danulabs
Kraljevića Marka 11/18
21000 Novi Sad, Serbia
ana.marija.ciric@danulabs.com

Ivan Dobrić, CEO
Danulabs
Kraljevića Marka 11/18
21000 Novi Sad, Serbia
ivan.dobric@danulabs.com

Stefan Martinov, CRO
Danulabs
Kraljevića Marka 11/18
21000 Novi Sad, Serbia
stefan.martinov@danulabs.com

Stefan Seiler, Founding Partner
Danulabs
Kraljevića Marka 11/18
21000 Novi Sad, Serbia
stefan.seiler@danulabs.com