Practical Reasoning Systems

Bricks2CRM White Paper

October 22, 2001

Abstract
The purpose of this paper is to give an overview of Bricks2CRM. Bricks2CRM is a highly flexible multitier J2EE component managing customer, application and partners relations.

Bricks2CRM will provide your company with all the essential business services required by a modern business.

Bricks2CRM supports a multitude of uses subh as time reporting, content and campaign management, portals, car financing systems, booking systems among others. Consequent division into layers and heavy use of well-known design patterns and proven technology ensure a high degree of integratability, reliability and excellent performance. Bricks2CRM will significantly reduce your Time To Market for applications built on Bricks2CRM.

Bricks2CRM is fully UDDI-prepared.

Presys A/S
Herstedoestervej 27-29
DK-2620 Albertslund, Denmark
Phone +45 43 42 74 00
Fax +45 43 42 74 01
http://www.presys.dk

Introduction

Bricks2CRM is a J2EE multitiered business component designed to provide all the important business services required for CRM in a modern organization. An overview of the system is given in figure 1. Bricks2CRM is comprised of a representation tier, a business tier and a data tier. The presentation tier is a Model View Control-Pattern. The business tier is a J2EE component comprised of a number of Enterprise Java Beans. The business tier is capable of communicating with several databases and legacy systems. Furthermore, Bricks2CRM is UDDI-prepared.

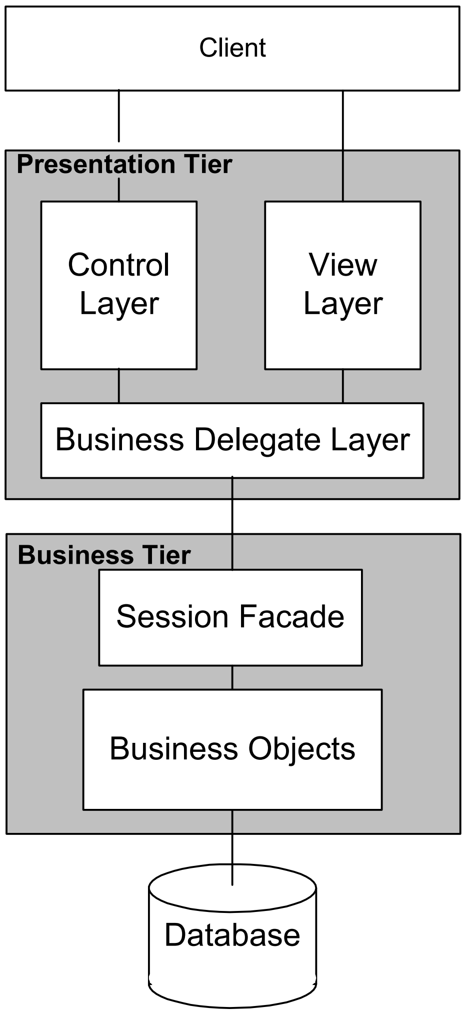

Figure 1: Overall system architecture. *Clients connect to the presentation tier, which is designed as a Model View Control-Pattern built upon the Apache Struts Framework. The presentation tier interacts with the business tier through the business delegate layer on the presentation tier side. The business delegate interacts with the session beans that provide the business services through the session facade. The Entity bean layer handles persistent data and is implemented using container managed persistency.*

Bricks2CRM is designed with the aid of well-known patterns ensuring the following qualities:

- High reliability.
- Rapid integration with existing systems.
- Easy to adapt new technologies, such as WAP or XSLT.
- High degree of reusability.
- Tiers and layers are easily changed or modified without impacting other parts of the system.
- High performance.
- Easy maintenance.
- Communication between designers in standard vocabulary.

The following gives a description of the presentation tier and the business tier with the purpose to give an understanding of Bricks2CRM's reliability, flexibility and ease of use. The paper is concluded with a note on the design process that lead to Bricks2CRM, denoted as *extreme design,* to further illustrate the easiness and adaptability of Bricks2CRM.

The presentation tier

The presentation tier is designed as a Model View Control Model 2, also known as MVC2-pattern. The pattern prescribes a controller that receives requests from clients, validates these, and begins the appropriate interactions with the model, which in this case is the business delegate. The business delegate handles communication with the business tier. The View presents the resulting data provided by the business tier. Figure 2 provides an overview of the MVC2-pattern.

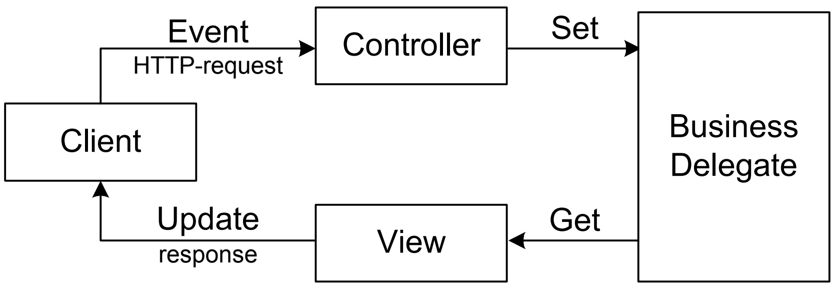

Figure 2: Bricks2CRM MVC Model 2.

In Bricks2CRM, the implementation of MVC2 is based on the Apache Struts Framework (http://jakarta.apache.org/struts), thereby ensuring high degree of reliability. Bricks2CRM's MVC2 is a set of cooperating classes, servlets and JSP tags designed to control requests, validate input data and represent resulting views (a detailed depiction of our implementation is given in appendix A).

Control layer

The control layer determines the rules of processing. Requests from either client or the model are processed with respect to the general application scheme as well as the basic validation rules implemented in each single control unit, called an Action. Up front validation ensures minimal network traffic and fast and friendly dialogue with users. All Actions are tied to a single servlet, called ActionServlet, which is the core of the control-layer. The ActionServlet lives throughout the entire application cycle, and is responsible for calling the correct Actions and passing responses to the view-layer.

View layer
The view layer renders the actual response according to the control layer's decisions. Each type of response is encapsulated in a unit, called an ActionForm. An ActionForm provides access to the underlying business tier through the business tier's business delegate layer, and with the aid of value object. The business delegate layer will be further explained below.

The output from ActionForms can be presented in multiple formats and ways, depending on the implementation of the view. Traditionally, the view layer generate HTML-Pages by the use of TagLibs, JSP templates and predefined resources, which makes it easy to implement for example internationalisation. A large number of technologies such as WAP or XSLT are easily implemented, since MVC2-transparency ensures that no other layer takes part of the view. So when cellular phones change their presentation technology again, only a new view must be added to support it.

New views can be manufactured without any Java expertise, due to the fact that the view layer can produce among other things, XML, HTML, WML (Wireless Mark-up Language), WSDL (Web Services Description Language), Windows API and so on, but no Java code. Addition of new views and screen layout are thus turned into technologically trivial tasks. This enables you to use technologies such as cellular phones, browsers, web services, Windows, 3270 displays, card readers, ATMs, etc.. Bricks2CRM will handle all the necessary details behind the curtains and will allow you to fully concentrate on your business.

The Business Delegate Layer
The business delegate layer is responsible for the integration between the presentation tier and the business tier. Business services are accessed from the presentation tier via a number of business delegates in the business delegate layer, which is designed as an application of the Business Delegate Pattern and the Value Object Pattern (Alur et. al. 2001). Two significant advantages are obtained with this design. First, access to business services is vastly simplified, and second, changes to the presentation tier caused by possible changes to the business tier are reduced to a minimum. Bricks2CRM will therefore be highly customisable and it is an easy task to run Bricks2CRM against new systems.

The business delegate layer will also reduce network traffic. The presentation tier and the business tier will typically be physically separated and connected via some network, so data requests should preferably be bundled. This is accomplished with data objects that, when transmitted between presentation tier and business tier, data are automatically bundled together, according to the frequency of use. In order to connect from presentation tier to business tier, knowledge of the interfaces of the business delegate classes and the value object classes is the only prerequisite (Appendix B provides an example of the company business delegate to illustrate the ease of use).

The business delegate layer allows for integration with legacy systems. Each business delegate bean can be modified to communicate with existing systems; they can be replaced by alternative delegate beans, or be combined with new beans. Customized, vertical solutions based on Bricks2CRM are therefore easily developed by either your programmers or Presys' consultants. Such solutions could include time reporting, content and campaign management, portals, car financing systems, booking systems among others. Integration through the business delegate layer requires J2EE and especially EJB expertise.

The business tier

The business tier is comprised of two layers, the Session Facade layer and the Business Objects layer. Each layer is accessed through a standardized interface, making reuse or replacement of either of the layers easy.

Session Facade
The session facade provides a simple and standardized interface to the business services. The layer implements the Java 2EE pattern Session facade (Alur et. al. 2001). Each Session bean provides access to one or more data access objects, and the related business logic.

The session facade contains among others the following facade objects:

Client	Access to client business code. Client is a special case of company containing additional information.
Company	Access to company business code. Handles changes to the company, company structure and employee information. Company is based on NodeStructure allowing for very complex company structures. A company can be both a supplier and a customer, or even your own company.
ContactPerson	Access to Contact Persons business code. This object handles changes to contact persons, and notes linked to the contact person. A Contact Person is a special case of a Person with a relationship to a client, company or product.
Employee	Access to Employee business code. An Employee is a special case of a Person, and this person can in the system also be a contact person to one or more companies or products.
Order	Access to Order business code. Create, update, validate or otherwise work with orders thought this module. Orders can run through several states in a pipeline (from prospect to the final accepted order). Orders can be used in the traditional sense, or can be used by consultant companies to manage their time-reports.
NodeStructure	Access to a grid structure with employees (used for project management). The employees can be placed in a multi-layered manager – employee relationship.
Price	Access to Price business code: Handles special prices and volume discounts.
Product	Access to Product business code: Handles access to product information.

Table 1: The session facade.

The Business Object layer
The Business Object layer is responsible for data handling. The layer consists of a number of Java Enterprise Entity Beans. The layer is implemented with CMP, meaning that persistency of data is handled by the container according to the EJB 1.0, 1.1 and 2.0 specifications.

The entity bean layer ensures easy replacement of database system. Bricks2CRM is fully portable and supports Oracle and DB2 and others, and can furthermore run across many different databases at the same time. It is fully Integratable with legacy systems and ERP-systems such as SAP and People Soft through the J2EE Connector Architecture.

Extreme design --- A new Paradigm

When the earth was still the centre of the universe, the munks, the experts of that time, were able to calculate the motions of the celestial bodies and thus produce reasonable accurate calendars. But adjusting calculations caused by the inherited flawed presumption let to a highly complex situation. This let the Polish astronomer Kopernicus to see things in a totally new way by placing the sun as the centre of the universe. This not only let to more precise calculations, but also, of magnitude practical importance, to a significantly reduced complexity. A new paradigm had emerged (Kuhn 1962), a whole new way to view the world.

Back in the seventies there was a lot of talk of software crisis. Many attempts to build new software systems failed because the developers were unable to cope with the significantly complex nature of the development process . The answer to this challenge was to formalize the development process through development methods such as Boehm's spiral model or the waterfall model. All this enabled developers to cope with the complexity of their work. But not to reduce it.

In 1995, Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides (usually denoted the Gang of Four) published their widely reputed book on design patterns in objectoriented design (Gamma et. al. 1995). By abstracting the structure of simple and elegant solutions to a large number of software problems, the Gang of Four provides a design language constituted of design patterns that enables software developers to focus on system design without untimely delving into the complexities of their work (The essense of design patterns can be traced back to Constantine's ground laying work with structured design (Yordon & Constantine 1975)). Furthermore, design patterns constitute a common language that allows designers to engage in design language games inside design teams as well as across teams and organizations, very much in the nature of Wittgensteinian language games (Wittgenstein 1973). Through patterns, a whole new way to view the world has emerged, a new paradigm has risen.

The development of a common design language for software design through the use of design patterns is carried further among others by Martin Fowler (Fowler) and in Azur et. al.'s book on designpatterns in J2EE (Azur et. al. 2001). In the development of Bricks2CRM we leant heavily on design patterns. Appart from the advantages mentioned above provided by patterns, we have also experienced the actual design process as a design language game inside the designteam, in which we very much were able to focus on the relevant design issues rather than on the underlying technological complexity. By the help of design patterns, we have thus been able to execute the development process as a true case of/in the true spirit of extreme programming (Beck & Fowler 2001). And we denote this *Extreme Design.*

Bibliography

Deepak Alur, John Crupi & Dan Malks (2001): *Core J2EE Patterns: Best Practices and Design Strategies.* Sun Microsystems Press.

Kent Beck & Martin Fowler (2001): *Planning Extreme Programming.* Addison-Wesley.

Martin Fowler (www.martinfowler.com/isa): *Information System Architecture.*

http://jakarta.apache.org/struts: *The Apache Struts Framework*

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1995): *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Thomas Kuhn (1962): *The Structure of Scientific Revolution.* Second Edition, Enlarged. The University of Chicago Press.
Wittgenstein, Ludwig (1973): *Philosophical Investigations.* Basil Blackwell.

Edward Yordon & Larry Constantine (1975): *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice Hall.

Appendix A

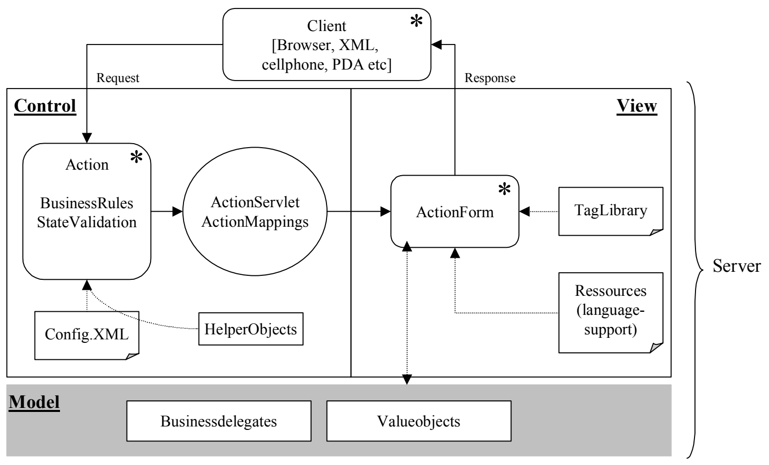

Figure 3: The MVC2-implementation in the presentation tier

Appendix B: The Company Business Delegate

The design of the interface of the business tier is here illustrated by the CompanyDelegate Class. The presentation tier is required to know the interface for the CompanyDelegate Class. The interface is shown in Table 2.

CompanyValueObject GetCompany(long id)	Will retrieve a value object representing information of the company specified by the parameter.
void setCompany(CompanyValueObject val)	Will update information of a given company according to the values specified by the parameter.
Vector getChildCompanies()	Will retrieve a vector of the child companies one level below.
Vector getParentCompanies()	Will retrieve a vector of the parent companies one level above.
Vector getCompanyList(String search)	Will retrieve a vector of companies containing the search string as a sub string of the company name.
void deleteCompany(CompanyValueObject val)	Deletes a company.

Table 2: The CompanyDelegate Class

Since the business delegate will send and receive data as value objects, this also requires knowledge of value objects. Value objects are objects with fields representing the most commonly used values for a company and appropriate accessor methods. The CompanyValueObject is shown in Table 3. It is not expected that information regarding parent- and child companies are requested on a regular basis, for which reason the value objects do not contain this information. Distinct functions are instead provided for these. Note how the system is UDDI-prepared.

long id	
String name	
AddressValueObject address	
String authorizedName	According to the UDDI-specification
String businessCode	According to the UDDI-specification
String businessType	
string operator	

Table 3: The CompanyValueObject Class will have accessor methods for the above fields.