Support for

Design Presentation

on a Project’s Home Page

Templates described here can be customized

Gebhard Greiter

June 2011

Contents

Purpose of this Document **4**

Document Status **5**

Reviews so far 5
Distribution List 5
Document Content Approved by 5

Management Summary **6**

Notation and Terminology **7**

Publishing Software Design to the Team Members **8**

Convention on Specification Card Format **9**

C_ Name_of_Component **10**

Requirements on C_… **11**

R_ Name_of_Requirement (e.g. keywords) 11

Data Views for C_… **12**

B_ Name_of_Data (schema or instance) 12
V_ Name_of_Dataview 12

Services offered by C_ **13**

S_ Name of Service 14

Actors (= User Roles) to be supported **15**

Processes to be driven by the Actors **15**

A_ Name of Actor 16
P_ Name of Process 17

Why these Templates are helpful **18**

Why the first part of these Templates is necessary **19**

Content Index based on Semi Titles (and how to Print) **20**

ProjWeb.exe - a Tool for generating Project Webs **21**

Data Views for C_projweb **21**

B_ Project_Web_Configuration 22

Services offered by C_projweb **22**

S_ Create Project Web 23

How to get a Copy of projweb.exe **25**

Annex 1: A Rationale **26**

The Importance of Business Objects **26**

The Importance of View Objects..26

Services versus Service Presentation...26

Components that are Component Interfaces (use _><_) 27

Annex 2: 28

Collaboration Cards (Sample and Template)..28

One specific EAI Workflow...28

Purpose of this Document

This document is to explain patterns and tools that turned out to be helpful for the purpose of

- structuring software design in a way to keep it easy to maintain,
- publishing software design documentation in form of hypertext,
- detecting inconsistent text (or incompleteness), and
- guaranteeing that the team will always see the most up to date version (without the need to know in which documents or files such a version is maintained).

The basic idea is to use patterns – so called ***Specification Cards*** – that guarantee that all hyperlinks can be generated by a tool that should recreate the web site automatically on a regular basis (e.g. once per day).

An non-trivial, though small example for such a presentation can be found here:

www.greiterweb.de/spw/xs_wissDB/1/1.htm

The solution proposed is based on:

- Requirements seen by
  - Gebhard Greiter for development, maintenance, and support projects

Document Status

Revision	1.0
Date	June 2011
Status	
Confidence	Public

Reviews so far

- Review by ... The paper is updated according to the results of this review since ...

Distribution List

…	for …
…	for …

Document Content Approved by

…	for …	………………………………….
…	for …	………………………………….

Management Summary

Each person working in a team needs to read results produced by other persons.

Many such results – e.g. requirements or design – are constantly changing. To find the latest agreed upon version, the version that is to be the basis for the programmer's current work, is not always easy.

Even more: Design is usually distributed across many documents. This implies that to search for a certain detail is often time-consuming. When not finding it, we can not even be sure whether it is hidden somewhere or needs still to be produced.

When reading code (e.g. in the role of a maintenance programmer) we should be able to find therein hyperlinks to the corresponding design.

What would help us is a web page on which all code and documentation a programmer needs to look up frequently in order to know about

- current requirements
- current design
- and existing code

can be found presented in a form that will not require the reader to know file locations or to have an idea in which document exactly a certain detail he/she may be interested in could be found:

When seeing the name of a concept, the corresponding spec should be away only one mouse click.

To create such web pages seems to be rather tedious and unaffordable.

However: This paper here, together with Gebhard Greiter's tool C_projweb, is proof of the fact that creating such a web site – and keeping it up to date nearly automatically – need not be costly at all.

All we need is

- MS Office 2000 (or higher),
- a convention how to form concept identifiers, and
- a convention to guarantee that each occurrence of a concept identifier can automatically be classified as a reference to a spec or as the headline of the spec itself.

Notation and Terminology

The word ***Concept*** – when used in this document – always refers to a typed concept in the following sense:

Each concept specified in a software design paper is given a name starting with a prefix telling you the type of the concept. Prefix semantics for the concepts currently supported by C_projweb are:

–	C_	=	A software component
–	S_	=	A service offered by software component (and callable via at least an API)
–	m_	=	A manual solution
–	$_	=	An installation parameter
–	P_	=	Process (usually a business process or a technical process supporting a business process)
–	A_	=	An actor (in the sense of a user role or a system's role)
–	D_	=	Some logical name of a document (a so-called Semi Title)
–	R_	=	A requirement well distinguishable from other requirements
–	V_	=	A logical data view on top of the physical data stored in some database. Such a view should be definable via SQL (so that standard reporting tools can be applied).
–	B_	=	A business object type (a B_x is a V_x such that all instances of type B_x are owned by a unique Component. Ownership on the logical level – which is the right to define this data structure – may not be the same as ownership on the physical level.)

A ***typed name*** is an identifier following one of the prefixes defined in this list.

The prefix, denoting the type of Concept identified, may or may not be followed by a blank. The blank, if there, will tell the tool that this specific occurrence of the concept name should not become a hot spot.

Hyperlinks created by the tool C_projweb always start

- at an occurrence of a typed name,
- or at a navigational element added by the tool itself in order to provide useful hot spots to the reader of the web page.

To use type concept prefixes is all the author of a design paper needs to do in order to allow the tool to create a presentation in which human readers can navigate to the spec via one mouse click only.

If, on the web page, a typed name is not a hotspot, the corresponding concept is not yet specified (which also is useful information). The tool will mark such occurrences by adding **[?]**. This will help the QMB to see how incomplete the current design may be.

Publishing Software Design to the Team Members

Each person working in a team needs to read results produced by other persons.

Many such results – e.g. requirements or design – are constantly changing. To find the latest agreed upon version, the version that is to be the basis for the programmer's current work, is not always easy.

The solution we propose is:

Create a tool C_projweb for publishing all these results in form of a heavily hyperlinked web page: As soon as such a tool exists, a demon can regenerate this web page on a regular basis, e.g. once per day (or whenever the page configuration file was updated).

Let us call this web page ***the Project Web***. It is to support the following requirements:

- **Portals to he Project Web need to be customizeable**

  Solution: C_projweb will need a configuration file. It has to say which documents we want to publish, and for which user roles specific portals should be generated (each of these specific portals being just a subset of the always existing default portal).

- **All kinds of documents must be capable of being published**

  Solution: In the configuration file we distinguish between enhanceable and unenhanceable documents. A document, or any piece of text, is called ***enhanceable*** if C_projweb is capable of generating hotspots in a copy of this document or text section.

  This precondition is fulfilled as soon as you created – via MS Office 2000 or later – a copy in HTML format.

- **To keep the Project Web up to date must be extremely easy**

  Solution: One person in the project – usually the project leader or the QMB – is to accept the responsibility to keep the configuration file up to date in order to guarantee that

  - at any time all relevant documents are published, and
  - each document published is published in the most relevant version.

- **Documents describing Software Components, Processes, or Actors should be enhanceable to a large degree.**

  Solution: Templates for specifying concepts of type C_, S_, m_, P_, A_, B_, V_ (see our section on Notation above) have to be provided.

  The templates we use are called ***Specification Cards***.

Convention on Specification Card Format

Clicking on a typed name is to bring the reader to the text specifying the corresponding Concept. In order to let the tool C_projweb know how to find the specifying text, each Concept should be described on a so called Specification Card.

A Specification Card is a section with a header of a specific form.

In order to guarantee this form, we define templates. Using the templates will guarantee that the design papers you create are enhanceable. A more detailed rationale for our templates can be found on pages 17 and 18 of this document.

First however let us give the template specification (pages 9 to 15). Text shown in green is sample text, text shown in blue specifies how to use the template, text in black is part of the templates):

C_ Name_of_Component

Component:	**C_ Name_of_Component** part of C_Name_of_another_component
Abstract:	This component offers services for – ...
Has to:	**Own and manage the following data:** – B_... **Implement** the following functions: – S_...
Because of:	

Following this table you may have more text of any length in any format (some kind of Annex to the Card, e.g. pictures).

The ***Because of:*** row in the table may be missing – if it is present, it should contain enhanceable references to sections of a requirements document or text that could be seen as a design rationale.

Typical examples for C_ Specification Cards can be found here:

www.greiterweb.de/spw/xs_wissDB/1.1.htm

Just go there, click on Index and there on Components in the header navigation menu, then choose any entry.

For Components specifying data structures, the C_.. Specification Card is to be followed by sections specifying requirements, data and/or services. They should have the following form:

Requirements on C_…

If empty, this section may be missing. Otherwise it contains a sequence of R_.. Specifications Cards.

This section contains all business object specifications B_.. owned by C_..

Requirements V_.. described here can be used to generate a Compliance Table.

R_ Name_of_Requirement (e.g. keywords)

supported by <a comma separated list if typed concept names> is the following requirement:

Please insert here in all detail this requirement.

Additional Comments:

- ...

To use R_.. Specification Cards following this template will have the advantage that C_projweb will be able to generate an always up to date compliance list. On product delivery, the compliance list can be used to explain to the customer why we believe that all requirements on the product are actually implemented.

The compliance list generated will, in the Project Web's presentation, take the red part above (i.e. this part is not meant to come from the author of the R_.. specification. It comes from Because: sections found in A_, P_, C_, S_ specification cards.

Data Views for C_…

If empty, this section may be missing. Otherwise it contains a sequence of B_.. Specifications Cards followed by a sequence of V_.. Specification Cards.

This section contains all business object specifications B_.. owned by C_..

Data objects V_.. described here are auxiliary data structures (often views). Each of them should, if possible, be given in form of an SQL SELECT statement based on only B_.. data structures.

B_ Name_of_Data (schema or instance)

Owner of this data structure is C_Name_of_component

Please insert here a description of the structure of the business object to be specified, usually a list of attributes as shown in this example:

- ShortName (primary key)
- Is_NetworkOperator
- Is_ServiceProvider
- ...

Attribute Value Specification:

- **ShortName** (up to 20 char) and **Name_1** (up to 80 char) must not be empty.
- **Is_NetworkOperator** is either empty or an ID of the form Dxxx (defined by RegTP)
- **...**

Additional Comments:

- ...

V_ Name_of_Dataview

Owner of this data structure is C_Name_of_component

Please insert here a description of the data in question (preferrably in form of an SQL SELECT statement mapping this view V_.. to business objects types B_..)

If the Component is to implement Services, the section ”Data Views for C_...” is to be followed by a section containing S_.. Specification Cards:

Services offered by C_…

Please insert here a piece of text specifying which form the implementation of the services in question will have to take. The following text, and also the next page, is an example borrowed from the BOSS project:

If not stipulated otherwise, all the services (= functions) specified in the following have to be available, to the end-user, via a graphical, web-based user interface, i.e. via a web browser such as MS Internet Explorer.

The browser is to map user requests to so-called Java Server Pages. These are dynamically generated Java Servlets that access the application programming interface of BOSS, i.e. the API for the here specified BOSS component.

A suitable command line interface (CLI) must allow the user to invoke each service S_… also from the NT command window or out of a batch file.

The implementation of all these interfaces has to comply with the rules specified in C_BOSS_API, C_BOSS_CLI, and C_BOSS_GUI, the most important rules being:

- Both, CLI and GUI, must map themselves onto the API (and must not implement any functionality that is not implemented by the code below the API).
- The API of each service S_… is a resource in the sense of C_BOSS_Access_Control an can therefore only be invoked or activated by a user working in a well specified role.

Please insert here a sequence of services (= API specifications), i.e. a sequence of S_.. Specification Cards in the form shown on the following page.

The ***Because of:*** row in S_.. Specification Cards may be missing – if it is present, it should contain enhanceable references to sections of a requirements document or a short design rationale.

An enhancable reference to a requirement has the form ***R_(short title)***.

The ***Abstract:*** row is to describe type and semantics of all parameters of the service in question. Do not forget to classify them as **In**, **Out**, or **InOut** parameters:

S_ Name of Service

Component:	**S_ Name_of_Service** part of C_Name_of_component
Abstract:	This is the function to specify from when on number ranges owned by FMC should be available for allocation to customers. Parameters: – **In: NumberRange** A number 49x contained (as a set of number resources) in a union of number resource ranges owned by Dxxx – **In: Dxxx** A porting ID of a German carrier (seen as the ID of the network that is to terminate traffic for the numbers in question). – **In: ActivationDate**
Has to:	Ensure that the numbers in question have attribute values stating Activation_Date = the given date However: The requested update is to be rejected – If not all of the numbers prefixed by the given number are owned by the carrier owning the given network Dxxx, – or if they form a range containing more than 10.000 numbers of length 10 (no number resource range bought from RegTP contains more than 10.000 numbers if we see them as numbers of length 10).
Because of:	

Following this table you may have more text of any length in any format (some kind of Annex to the Card, e.g. pictures).

Business processes and Actors may be described in a separate document.

In order to be enhanceable, the last part of such a document should take the form of two chapters bringing the reader to Actor (A_..) respectively Process (P_..) Specification Cards:

Actors (= User Roles) to be supported

- A_Name_of_Actor
- A_Name_of_Actor
- A_...

Processes to be driven by the Actors

- P_Name_of_Process
- P_Name_of_Process
- P_...

For each Actor or Process listed in these sections there has to be a corresponding A_.. respectively P_.. Specification Card. Hence:

These two sections have to be followed by a sequence of A_.. cards followed by a sequence of P_.. cards (actor and process specifications).

Here are the corresponding templates supported by C_projweb:

A_ Name of Actor

Actor:	**A_ Name_of_Actor** Subclass of A_Name_of_some_other_Actor By an Actor we understand a Role in which humans or systems may work.
Definition:	Any person or system seeing BOSS as an application which can be used as a tool to do business in a way more effective and more efficient than it would be possible without such help.
Has to:	Please insert here a more detailed description of what an Actor (usually a user) of this type is to do. This description should contain references to all processes P_ this kind of Actor is to drive.
Has to access:	Actors in this role will need access to the following Services and/or Business Objects: ▪ S_Name_of_Service ▪ S_Name_of_another_Service Note: Because data should always be implemented as Abstract Data, only Services and Actors (Roles) need to be seen as resources that are to be protected by access permissions.
May work in Role:	Special groups of end users are: ▪ A_Sales_Person ▪ A_Network_Planner ▪ A_Network_Operator

Following this table you may have more text of any length in any format (some kind of Annex to the Card, e.g. pictures).

P_ Name of Process

Process:	**P_ Name_of_Process** part of P_Name_of_another_process
Abstract:	Here a short characterization of this process.
Has to:	Her a description of pre- and post conditions (especially of results the process is to produce). As far as results are data, such objects have to be specified in B_.. or V_.. Cards. There names must occur in this Card here. If the process has subprocesses, their name also must occur in this description so that only one mouse click is enough to retrieve the Specification Card of any such subprocess.
Because of:	

Following this table you may have more text of any length in any format (some kind of Annex to the Card, e.g. pictures).

The ***Because of:*** row in the table may be missing – if it is present, it should contain enhanceable references to sections of a requirements document or a short design rationale.

Why these Templates are helpful

Templates as above for specifying Components, Business Objects, Data Views, Services, Actors, and Processes are helpful because they guarantee that design papers become easy to read and easy to maintain:

- Each Specification Card can be read and understood in isolation.
- Specification Cards may be read in any order.
- Making the specifications they contain more detailed is quite easy (much more easy as when using design documents in any traditional form):

Usually, when you design software, you create a document containing high level design. Later on other documents containing low level design are created. Because high level and low level design come in form of different documents, they may contradict each other. Design delivered in form of a set of Specification Cards is much less redundant (and much of the redundany left can be controlled by a tool such as C_projweb).

Each Specification Cards is only a short piece of text (usually not more than one page long). But it can grow by adding details and annexes.

One reason why most Specification Cards do never grow beyound one or two pages in length is that all the text needed to understand them is either in the card, or is reachable via hyperlinks (each typed name is the source of such a link, and the specification of the corresponding concept is only one mouse click away).

To ***sort out and archive reusable parts*** of system design is now easy: Simply because each Card can be read and understood (to a very high degree) without any specific context, or can be fully understood in the context of of a sequence of easy to find and easy to extract additional Cards, all on separate pages.

In Softlab, the technique of Specification Cards is used by Gebhard Greiter since 1998. It generalizes the well known concept of CRC Cards and was found to be much more helpful than any other form of design documentation.

This document itself is a good example for how flexible the technique can be applied: Created in only three days, it is explaining and motivating the design of a Component C_projweb given on page 19 to 21 in the form of Specification Cards.

Why the first part of these Templates is necessary

Creating hyperlinks manually would be tedious, costly, boring, and error-prone. So we need to create them automatically – our tool C_projweb can do it.

The tool is capable of creating them because our templates guarantee that:

- Begin and end of each Specification Card are clearly marked.
- The text phrase defining a concept name can be identified automatically, and
- all names that should become hot spots are typed.

For the tool to work it would be enough if the text tables in the templates above would contain only their first row (this is possible because all further description could go in the text that may follow the table). However, we do NOT recommend such a solution because it would give up the slot raster making our Specification Cards so easy to read and so extremely easy to create – the raster will not let us forget to add any a certain minimum of information.

For example: The S_ template says that we have to describe

- all parameters for the function
- the effect of a call of the corresponding API
- and also how the effect depends on the values given for **in** or **inout** parameters.

But why, you may ask, is the first row in the template table necessary?

This question may be provoked by the fact that the pattern for B_ and V_ cards (i.e. the two data description patterns) do not contain such a row.

Our answer is: The first row, i.e. the row containing the ***part of*** or ***subclass of*** clause is needed where you specify a concept that may have a hierarchical structure (this is actually quite frequent: Components may have subcomponents, processes may have subprocesses, services are contained in components, and actors may have specializations).

The ***part of*** and the ***subclass of*** clause enables C_projweb to see this tree-like structure so that it can add – in the web presentation of all this design – local tables of content: a set of hotspots the reader will usually find helpful.

Such a table of content is not restricted to one document only. Using a Project Web, documents only act as containers in which design is delivered. Human readers often do not care in which document exactly a spec they are interested in is actually maintained: Hyperlinks help them to find the spec and see the context in which it is to make sense. This is one of the main advantages of a Project Web created by C_projweb.

Coming back to the templates we see that, if need arises, they can easily be tailored to specific needs of your project: Without the need to change the tool projweb, you may redefine the templates as long as you keep

- the first two lines of our B_ and V_ templates
- as well as the first row of our C_, S_, A_, and P_ templates.

Content Index based on Semi Titles (and how to Print)

The structure of a Project Web is, first of all, content-oriented (so that, when we try to find information, we do not need to know in which physical document, or in which file, it is maintained).

On the other hand, if we want to print any document, we need access to a printable version.

Furthermore, content has to be indexed in a suitable way.

In order to solve these problems, the Project Web should be given an always present header menu containing links to various indexes. One of these is Topics.

The Topics index is a set of so called Semi Titles: A ***semi title*** is a short phrase characterizing an important aspect of a document's content.

Each document must have at least one, but may have more than only one semi title. One of these might be identical to or might be an abbreviation of the documents actual title. Semi titles are to be defined in the Project Web's configuration file.

Text in documents that are meant to be enhanceable should refer to documents only via semi titles. Each such reference is to take the form ***D_(a semi title)***.

The document you are currently reading, e.g. may have the following semi titles:

- How to Create a Project Web
- Use Specification Cards to create Software Design
- Specification Card Patterns

As soon as they are listed in the web site configuration file, the tool creating the web site will enhance to hotspots document occurrences such as

- D_(How to Create a Project Web)
- D_(Use Specification Cards to create software Design)
- D_(Specification Card Patterns)

Clicking on any of them in the Project Web will open a printable version of this document. Whether you use capital letters or not, or additional blanks between the brackets, should not matter. To implement some more robustness against typos would be possible.

One more advantage of semi titles is that they can be used where otherwise you would have to insert more complicated, or much more cryptical references.

Even more: Using ***D_(a semi title)*** references has the advantage that you do not need to update them when documents are split or merged, or when content is moved from one document to another one. In such a case it would be enough to update the Project Web's configuration file.

ProjWeb.exe - a Tool for generating Project Webs

Component:	**C_projweb** part of C_Productivity_Tools
Abstract:	This is Gebhard Greiter’s tool for creating Project Webs to support the publishing of project results (especially software design and Java code) to the members of a development, maintenance, or support team. The concrete specification card templates supported are specified in D_(Use Specification Cards to create Software Design).
Has to:	**Own** (i.e. specify the format of) the following data: – B_Project_Web_Configuration **Implement** the following functions: – S_Create_Project_Web
Because of:	Existing design and code should be as easy to browse as possible (and without the need to know in which file exactly text is maintained.

Data Views for C_projweb

In order to specify the content to be published in a Project Web (and to be re-published again and again in updated versions), we need a web configuration file. C_projweb requires that it has the format B_Project_Web_Configuration:

B_ Project_Web_Configuration

Owner of this data structure is C_projweb.

Project Web configuration objects are text files formed as shown in the following example:

```
   The <ProjectID> Web Configuration File

   C_projweb (= projweb.exe) is to understand commands -?, -e, -n, -j.

   Each flag is to start in the first column of a line following an empty line.
   Text following an empty line and not starting with '-' in the first column is
   seen as comment and will be ignored.


   Abbreviation for folder locations:
   ---------------------------------

   The Change Log for the content of the Project Web will be generated in form of
   a tree 1/docProfiles (i.e. it will be placed into the first folder given):

-? 1=k:/sampleDocuments


   Enhanceable (-e) and other (-n) documents to be published:
   ---------------------------------------------------------

   Enhanceable document files are assumed to be *.htm files created via MS Office
   2000 or later (earlier MS Word versions do not generate good enough HTML code).
   The HTML version of each such file is assumed to be found in subfolder publish
   beneath its *.rtf or *.doc version.

   For each -e or -n document at least one Semi Title has to be given:

-e 1/WebSiteGenerator.rtf
   How to create a Project Web Site
   Use Specification Cards to create Software Design
   Specification Card Patterns

-n 1/NotEnhanceable.doc
   A file assumed not to be enhanceable (any type of file)


   Also publish all *.java files found in or under the following folders:
   ---------------------------------------------------------------------

   Java code will be parsed for occurrences of typed names. They will become
   Hotspots linking code to design documentation (the effort to find and open
   such documentation is then reduced to only one mouse click).

-j 1/src
```

Services offered by C_projweb

There is only one such service:

S_ Create Project Web

Component:	**S_Create_Project_Web** part of C_projweb
Abstract:	This service is provided in form of an executable **projweb.exe** capable of running on MS Windows NT or later. Parameters: – **In: Configuration** A value of type B_Project_Web_Configuration. – **Out: ourWeb** A folder created (or recreated) in the current directory, i.e. in the folder from which you call the executable. – **Out: trace to stdout** so that you have an idea where to start when the tools does not work as expected.
Has to:	Given a B_Project_Web_Configuration, this tool has to create the corresponding Project Web (= **ourWeb**). Sections to be added automatically to **ourWeb** are: ▪ A Change Log for documents and document sections, and ▪ An updated version of a Compliance List for the complete product shown distributed to the R_.. specification cards. Activating **projweb.exe** without parameters will show online help.

Currently a Prototype Implementation exists:

Gebhard Greiter's prototype implementation is fully functional. However, because it was created first of all to be a prove of concept, it has certain minor restrictions. These are:

- There is a unique portal page (called ALL): the page where you always can see in which project you are and where you can see what type of content is actually published.

  This page is mandatory and is identical to the first document listed in the Project Web Configuration file (so that you will be able to customize it).

  In a large project however, this page is not the most convenient one to navigate into the Project Web because it cannot be customized to the role in which you work. As soon as a large set of documents is published, the user will need more selective portal pages.

  One solution to this problem could be to create a portal page showing only user roles (or tasks mapped to persons already). Clicking on such a role or task could then produce a page that is more dedicated than the page showing all documents published. It should contain a news ticker tailored to this role or task: When, e.g. a document was replaced by a more up to date version, the user should be alerted here.

However: As long as we have only the ALL portal page, the user can use the header menu to reach index pages. There is one index page for each of the following set of objects:

– Topics
– Actors
– Processes
– Business Objects
– Data Views
– Components
– Services

- Here the Topics index is the alphabetically ordered set of all semi titles defined for documents published in the Project Web. This set can be seen as summarizing the web's content. The Topics index will be especially useful if you create semi titles in the form

  <**concept A**> **...** <**concept B**>

  such as

  — **Requirements on ...**
  — **Support for ...**
  — **Test Design for ...**
  — **Test Data for ...**
  — **How to ...**

  This form is useful because such titles tend to be short, and will show up in the Topics index also in the form

  <**concept B**>, <**concept A**> **...**

  To create this second form, the tool assumes that <**concept B**> is – not counting commas and the word "and" – the longest sequence of capitalized words found at the end of the semi title.

- For MS Word files published as enhanceable, each section will become a separate HTML page. This may not always be desirable. So, if somewhere a hidden text line **[seeX]**, X any non-negative integer, is found, all text following this flag and the next X sections will be shown together on only one page (for an example, please see the web presentation of this document here: it has such a flag immediately preceeding the note **Business processes and Actors should always be described in a separate document**.

- Change Logs added will describe how the current content of ourWeb differs from versions created on any day **YYYY.MM.DD** found given in form of a file **1/ ourWebVersions/ pu_YYYY.MM.DD** (here **1/** is the first folder specified in ourWeb's configuration file).

  These logs are shown to you after clicking on CLog in the upper left corner of pages reachable via links named DOCs or Documents.

How to get a Copy of projweb.exe

Persons interested in my implementation of **C_projweb** are promised a free copy if they contact me by e-mail via my address ggreiter@gmx.de or visit

http://greiterweb.de/spw/meinprojweb.htm

A non-trivial but nevertheless small example for software design published in form of a Project Web can be found here:

http://greiterweb.de/spw/xs_wissDB/1/1.htm

C_projweb is also good for creating so called **DocNets** (also called Lexicons)**.** A DocNet in this sense is a well indexed HTML presentation of one or more documents that can be see as both

- One Lexicon or
- A sequence of documents (each of them printable as a whole).

A Sample DocNet can be found here:

http://greiterweb.de/spw/xs_Agile/1.htm

Annex 1: A Rationale

This annex is both, a kind of rationale for our meta design but also help on how we should use it:

The Importance of Business Objects

Why do we distinguish two kinds of data structures (B_ and V_)? The reason is:

- ***Business object specifications*** provide common terminology, and they discuss ideas that can be shared across an organization for both technical and non-technical people alike. They describe concepts that should make sense to the entire organization. If multiple applications from the same business domain exist, it's very likely that the same business objects exist across the application boundaries. Such reuse of information and behavior allows for faster application development and reduces the redundancy.
- ***Business objects*** have the ability to evolve with the organization through modifications to the original object, or through proper specialization. This is very important because as an organization changes, the information and behavior must adapt and change with it.

The Importance of View Objects

View objects are auxiliary data structures. They are needed to describe what kind of data components send to or receive from each other.

Create a V_ specification for data that cannot be seen as a Business Object in the sense above (e.g. because it is not seen by the end users of your application).

Both, B_ or V_ objects can be schema specification, or can be the specification of specific instances of such data.

Services versus Service Presentation

A Service in the sense of an S_.. card is an algorithm that can be started in possibly more than only one way. Interfaces to start it may be an API, a GUI, or a command script.

If you want to specify characteristics of such interfaces, please do so by creating C_.. specification cards such as C_API, C_GUI, C_CLI_for_A_Support, or C_Scripts.

Components that are Component Interfaces (use _><_)

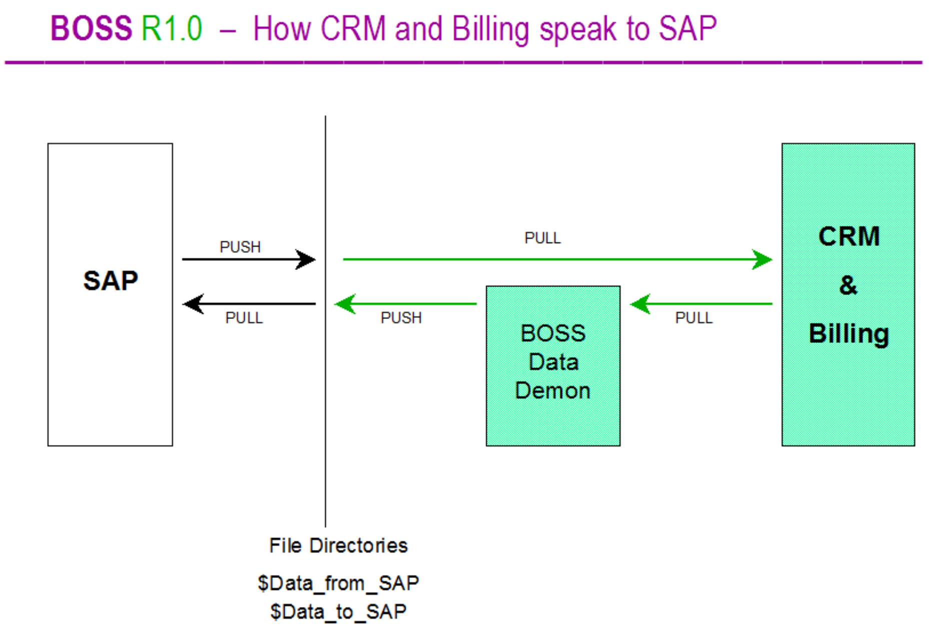
What we see as a ***component C_*** can be defined to some extent in your project. The general rule however is: Each specific type of spec or docu not yet typed in the Notation Semantics chapter of this document should be classified as a component.

Component interfaces, e.g. are important – so we need to specify them. If such an interface is not simply an API, we will have to specify data and data flow:

- Data is specified in V_.. specification cards,
- Data flow should be specified at least via a picture with PUSH and/or PULL arrows.

Here is an example for such a picture, the architecture diagram for the interface between a Customer Relationship Management System (CRM), a Billing System, and an SAP System.

The component representing this interface could be called C_SAP_><_CRM and could be part of any Business Operation Support System (BOSS):



Annex 2:

Collaboration Cards (Sample and Template)

Special cases of Specification Cards are so-called Collaboration Cards. They are a nice template for specifying how two or more human and/or non-human Actors are to interact in order to support some specific workflow. Here is an example taken from the system stack of a telecommunications provider:

One specific EAI Workflow

Where Enterprise Application Integration (EAI) is based on a simple Message Broker, e.g. a JMS Server or a TUXEDO Bus, Adapters are needed to implement Use Case specific work.

In the following Sample Specification this Use Case is about

- A_ **OrderMS** : an Order Management System,
- A_ **OProc** : an Order Processing System,
- A_ **RMS** : a Resource Management System,
- A_ **GIS** : a Geographic Information Management System
- and Helper Components such as
  - C_ **OrderMS_GUI** : a Dialog Interface to the Order Management System
  - C_ **Poller** : a TUXEDO process
  - C_ **Manager**
  - C_ **Worker**
  - C_ **Adapters_and_Extractors**

EAI Workflow is meant to transport an Order from **OrderMS** to **OProc** thereby contacting – for either validation or completion of the order – additional systems such as, e.g. **RMS**:

Despite the fact that this workflow involves no less than 9 components, it was easy to describe in form of the following Collaboration Card which is certainly easy enough to understand:

C_OrderMS_GUI allows an **OrderMS Dialog** supporting the user to enter a new order. As soon as the user finishes entering enough order details, OrderMS will store the new **Business Event** in the OrderMS Database and will mark it as being in status **committed**. Via a DB trigger, a record will be added to the	

OrderMS BUSINESS_EVENT table to indicate that a new event is ready to be transported to OProc.

Remark: Committing an event that is an order is implemented by sending an ADDORDER message to the Workflow Manager. Orders in this sense are represented by records in the SW_OPPORTUNITY table. (ADDORDER is to be understood as “add order to OProc”).

A **Poller** process (TxS_VA_BEVEN_POLLER)

is to poll the Transaction Database, is to detect the new record, and is then to feed it to a TUXEDO queue named BEVENT.

A **Manager** process (TxS_VA_MANAGER)

is to read the queue, is to find the order there, and is then to update the **Workflow Table**, i.e.

- A serial number (= SR Number) will be created to have a key for identifying this Business Event instance,
- the workflow necessary to process the order will be broken down into so-called Work Items (each of them being a sequence of TUXEDO Services),
- and the Manager is to create records each of them describing one such service activation and also to which Work Item it belongs.

A Work Item is a complete description of the – fully automated – activity bringing the order from one status to a following one (see the document “Order Statuses” for a spec of all possible status values).

The purpose is to collect data from TP systems – such as GIS and RMS – to do more validation. Some Work Items even have to feed data to RMS.

As soon as the Manger updates the Workflow Table the Manager will send a message (the SR Number) to one of many Worker Processes – all of them are based on the same executable.

The **Worker** process receiving the message will know there is a new job to process:

The Worker will read the Workflow Table to see which steps exactly shall be performed.

As long as there is at least one step not yet marked finished, there will be exactly one step that can be started now. The Worker will start it i.e. the Worker will work to finish the step. Work in this sense is calling TUXEDO services.

Whenever a service returns, the worker will update the Workflow Table (by changing the status flags therein indicating how far the service succeeded: value 5 is for indicating failure)

NOTE: Even for calls that succeed, the records will NOT be removed.

OrderMS is informed about success or failure because the Worker will call

- service VA_WRKFLWSTA in case of success,
- service VA_WRKFLWSTF in case of failure.

As far as a service to be called is to rely, e.g. on RMS services, it is implemented by the **RMS Adapter**.

Other **Adapters** support data retrieval from the GIS (to see whether there is Home Zone coverage) and for customer credit checking.

There is a particular Business Event named ADDORDER. It contains an item activating – via TxS_VA_WRKFLWMGR – the so called Extractor:

An **Extractor** process (TxS_VA_EXTRACTOR) is to

- extract the order from the OrderMS Database,
- and transform its presentation to have it available in form of a so-called TLV Tree.

A later item in the same ADDORDER event then feeds the Extractor-created TLV Tree to the OProc Adapter:

The **OProc Adapter**

- is to transform the content of the buffer back to a TLV Tree,
- and is then to feed this tree node by node to OProc via this systems C++ API.

OProc is to store the order – in status **committed** – in the OProc database, is to initiate provisioning, and is also to generate data for Billing.