Project/
Open

Architecture

27.7.2003

Frank Bergmann <fraber@fraber.de>

Content

- The Big Picture
- GUI Structure
- Component Architecture
  - Component Use Overview
  - Component Conventions
- Localization
- Permission Model
- Cusomization Concept
  - Application Lifecycle
  - Customization Options (1)
  - Customization Options (2)

Project/ Open

The Big Picture

- Project/Open is based on the ACS (ArsDigita Community System) platform.
- The base ACS modules are licensed under GPL 2.0, so Project/Open is also GPL.

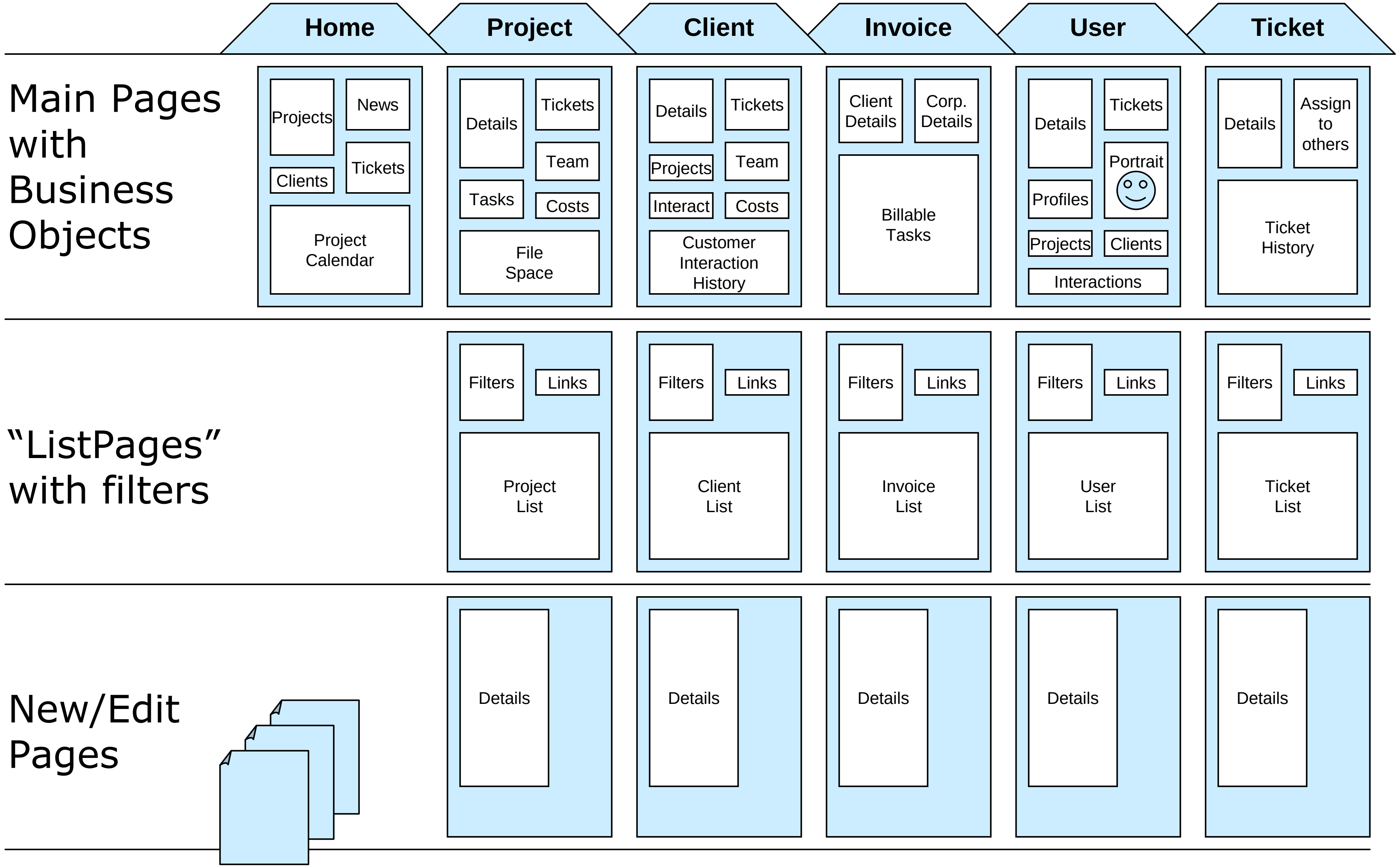
Project/ Open	ACS	Data base
Version 1.0	ACS 3.4.9	Oracle 8i (8.1.7)
Version 2.0	OpenACS 4.0	8i, Postgres, SAP-DB
Version 3.0	OpenACS 4.1	8i, Postgres, SAP-DB

Platfrom Roadmap: V1.0 depends on Oracle 8i, but V2.0 will feature alteast one free DB alternative.

- It’s a component-oriented transaction system with no object orientation (http://www.project-open.com/whitepapers/why-java-is-bad.html)

Project/
Open

GUI Structure



	Home	Project	Client	Invoice	User	Ticket
Main Pages with Business Objects	Projects News Tickets Clients Project Calendar	Details Tickets Team Tasks Costs File Space	Details Tickets Projects Team Interact Costs Customer Interaction History	Client Details Corp. Details Billable Tasks	Details Tickets Portrait Profiles Projects Clients Interactions	Details Assign to others Ticket History
“ListPages” with filters		Filters Links Project List	Filters Links Client List	Filters Links Invoice List	Filters Links User List	Filters Links Ticket List
New/Edit Pages		Details	Details	Details	Details	Details

Admin-Pages

Project/
Open

Component Architecture

Component Use Overview

	Home	Project	Client	Invoice	User	Ticket
Project Components	-Project List	-ViewPage -ListPage -New/Edit	-View: Project List (short)	-New: Project List -List: Filter -View: Renderer -Print: Renderer	-Project List	-Project Select -Project Render
Client Components	-Client List	-List: Renderer -List: Filter	-Project List	-Project/Task List -Project Filter	-Project List	-Project Select -Render Comp
Invoice Components	- . . .	- . . .	- View: Invoice List	-View Page -List Page -New/Edit	- . . .	- . . .
User Components	- . . .	- . . .	- . . .	- . . .	-View Page -List Page -New/Edit	- . . .
Ticket Components	- . . .	- . . .	- . . .	- . . .	- . . .	-View Page -List Page -New/Edit
Permissions	- . . .	-View: Perm+Estim	-View: Key Accounts (perm) -View: Customer employees (perm)	- . . .	- . . .	- . . .

Component Architecture

Component Conventions

- (GUI-) Components are used to render lists of objects, to render individual objects (in various variations) and to “filter” objects (restrict the elements of a SQL “where” clause).
- These components have to deal with all important functions such as permissions, localization and design templates.
- HTML pages are used to validate input variables, generate the page layout, and to “wire” the GUI-components together (glue-code).

Localization

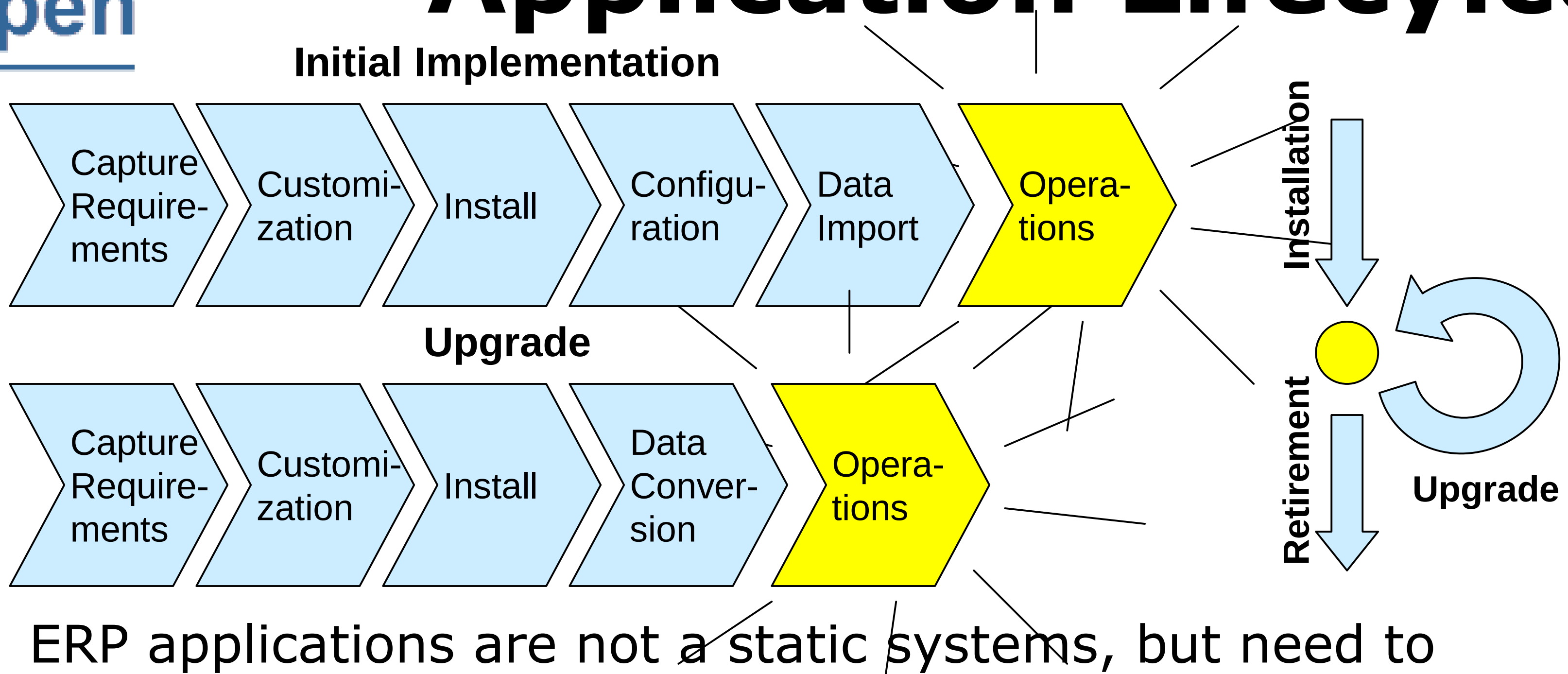
- All numbers, currency amounts, dates, times and physical locations are rendered as a function of the chosen user locale
- All currency amounts are preceeded by a currency identifyier. A system currency determines the default.
- All static text used in pages and components is translated before rendering the HTML page. An English text in the target location is used as a default value in case of incomplete translations. The translation interface also specifies the target location to provide a context for translators.
- Dynamic text (names and descriptions ob business objects) are not translated.
- Future versions of Project/Open may contain accounting modules specific for individual countries.

Permission Model

- Permissions are divided into profile-based permissions and role-based permissions
- **Profile-based permissions**
  - Restrict the visibility of pages and modules to a user
  - Are modified as part of system administration
  - API: **`im_permission(user_id, token)`**: Validates that a user has access rights to the permission token.
- **Role-based permissions**
  - Restrict the access to business objects (currently “Project” and “Client”)
  - Permissions can be modified by the creators of the respective objects, to configure project/client team.
  - API: **`im_has_role(user_id, object_id, role)`**: Validates that a user has been assigned the specifc role.
  - Roles are ordered hierarchically (administrator -> member -> specific_roles).

Project/ Open

Application Lifecylce

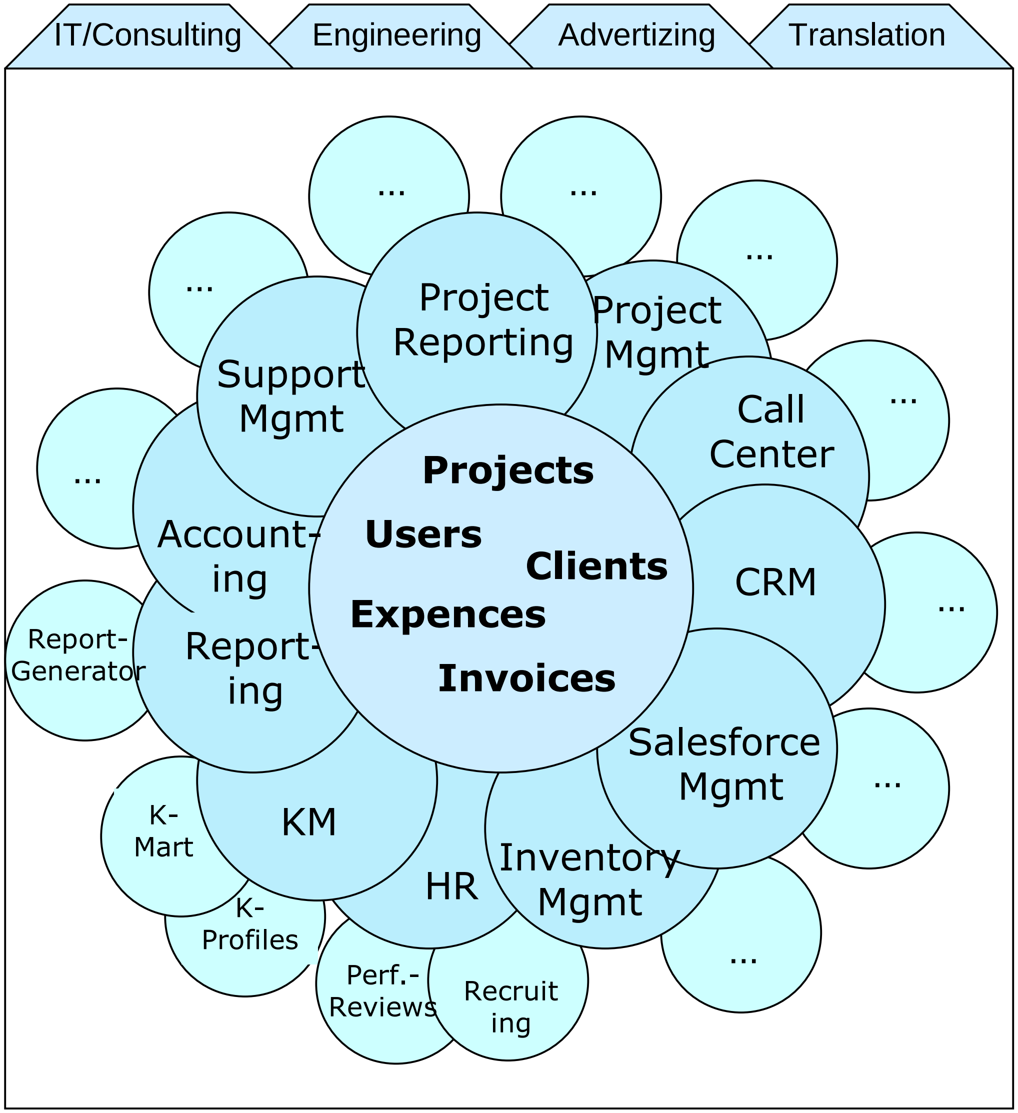


- ERP applications are not a static systems, but need to evolve continually with changes in the business processes of the client (customizations)
- ERP applications need to integrate/communicate with other corporate applications
- ERP applications have life-cycles of 5-20 years

➔ **The challenge is to maintain customizations when upgrading to the next release.**

Extensible Architecture

- A central core allows for extensions with different application modules
- Different industry sectors need different configurations and modules
- Customizations should be portable to the next version as easily as possible.



Project/Open

Extensible Architecture

- Not all modules are useful for all sectors
- Some modules are different for every sector (for example invoicing)
- Some of these modules may exist in several variants specific for each sector.

	IT	Consulting	Engineering	Advertizing	Translation	Intl. IT Dept.	Intl. Product Dev.	
Presales	*	*	*	*	*	(*)	(*)	
Project Collaboration	*	*	*	*	*	*	*	
Project File Sharing	*	*	*	*	*	*	*	
Customer Management	*	*	*	*	*	*	*	
Emplyee Management	*	*	*	*	*	*	*	
Timesheet	*	*	*	*	*	*	*	
Travel Costs	*	*	*	*	*	*	*	
Project Reporting	*	*	*	*	*	*	*	
Support Management	*	*	*	*	*	*	*	
Recruiting	*	*	*	*	*	*		
Invoicing	+	+	+	+	+			
CRM light'	*	*	*	*	*			

*****=applicable, **(*)**=maybe applicable, **+**=customized modules

→ The Module architecture has to deal with the interdepencencies between each other and over time.

Project/
Open

Customizable Items

How can additional module change the “core”?

- **GUI**
  - Home-Page (module-specific status boxes)
  - The main menu
  - Submenus of business objects
  - Views of biz objects (Project, Client, User, …)
- **Categories** (=workflow states and types)
  - Object status categories
  - Object type categories
- **Data Model**
  - Add columns to existing BizObj tables (system modules)
  - BizObj extension tables (sectorial configuration modules)
- **TCL Libraries**
- **Permission Model**
  - Add new permission tokens
  - New system roles
  - New Object/Group specific roles

Customization (1)

1. **Configurable Customization (100% preservation)**
   The system has been designed for the particular customizations. The customization involves changes in the configuration file and/or database contents.

→ Future version of Project/Open maintain compatibility with these configurable options, possibly extending them.

2. **Business Object Extensions (90% preservation)**
   Special database tables are reserved to hold customer extensions to business objects. These tables are not replaced during a system upgrade. Rendering of these objects is handeled by (modified) components in a new library.

→ Future version of Project/Open must not overwrite the extension tables. The client will have to integrate the modified components into the new View/List/Edit pages.

Project/Open

Customization (2)

3. **New Custom Modules (80% preservation)**
   The user may add new modules to the system, including their own data model and components.

→ Future version of Project/Open will not take into account the compatibility with custom modules.
However, it is likely that the integration of the new modules with the new version will be easy to manage.

4. **Custom Extensions (50% preservation)**
   In some cases the client wants to modify the core business objects or component libraries.

→ Future version of Project/Open will not take into account the compatibility with custom extensions.
That means that the client will have to port the extensions to the new version.

Project/ Open

Module Status

Module Name	Ver	Status	Leader	Comment
Core	0.8	Partially productive	Frank Bergmann	
Project Reporting		Wemo	Frank Bergmann	To be integrated with core?
Knowledge Mgmt		Concept	<please apply>	
CRM		Concept	<please apply>	
Salesforce Mgmt		Concept	<please apply>	
Project Mgmt		Concept	<please apply>	
Support Mgmt		Concept	<please apply>	
Accounting		Concept	<please apply>	
Reporting		Concept	<please apply>	
Human Resources		Concept	<please apply>	
Inventory Mgmt		Concept	<please apply>	

Participate!

- Build _useful_ software
- We teach you how to build real software
  - Learn about high-performance server applications
  - See why Java doesn’t work for _real_ software
  - Learn basics of SQL and Database Management
  - We have prepared some exercises that guide you through the process
  - Learn how to keep a web application running
- Choose a module to concentrate
- Study some background material and become an expert
- Design a user-friendly GUIs and a solid data model

- Being a leading developer with your module, clients will ask for your experience