

2007

Study Project Report

Reliable development of complex enterprise systems with limited resources

This is summary and analysis of the study development project taken from May 3 to June 24, 2007. The project concentrated on delivering complex enterprise-like solution with rich functionality, extensible architecture with limited development resources.

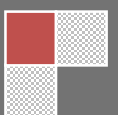


Table of Contents

- Introduction 3
 - Summary 3
 - Plan 3
- Requirements for the Software Development Process 4
 - Software Project Management 4
 - Development Methodology 4
 - Architecture and design 4
 - Automated Development Environment 4
 - Development Implementation 5
- Implementation 6
 - Software project management 6
 - Decisions 6
 - Statistics 6
 - Automated Development Environment 7
 - Decisions 7
 - Statistics 7
 - Images 8
 - Architecture and design 12
 - Decisions 12
 - Images 13
 - Development Implementation 14
 - Statistics 14

Introduction

Summary

This paper provides brief summary of the study development project taken from May 3 to June 24, 2007. The project concentrated on delivering complex enterprise-like solution with rich functionality, extensible architecture with limited development resources available.

Plan

The first part of this paper provides brief outline about integral part of any complex development project. It lists some of the requirements that are to be met in order for it to succeed in ever changing business environment.

The second part gives more detail on the development project undertaken as the reference implementation and the actual software deliverable. Detailed UI screenshots are provided in the ZIP archive along with this document.

Requirements for the Software Development Process

Software Project Management

Software project management sets the cornerstone of the project's success. All the critical decisions and business requirements come down to the development from this level. The success originates here as well. Those are some of the requirements that are to be met by the Software Project Management:

- Continuous improvement is essential.
- Traditional Project Management (as outlined by the *Project Management Body of Knowledge* by PMI) is not inherently applicable to the software projects, due to their specific nature.
- Right decisions in the right time can save a lot of time and resources. And saved resources are no different from the procured resources.
- Managing project is about managing its risks. Some risks are better to be taken care of before they happen.
- Proper standards on all change management, documentation, communication are required. Consistency saves time.
- Waste should be minimized. Depending on the size of the project, writing excessive documentation and sketches (that do not bring business value) could be unnecessary. Prototyping is faster and provides better communication channel between the development and management

Development Methodology

Development methodology continues and extends the principles and practices defined by the management. It brings them to the soil of the specific project. Those questions are to be addressed properly by the development methodology:

- Standards and guidelines
- Issue management practices
- Change management practices
- Information management practices
- Resource versioning
- Testing

Establishing practices that suit the specific project is necessary.

Architecture and design

Great benefits can be achieved if the software architecture supports the project management and adopted methodology and aims to integrate with the existing development environment. Architecture deals with the:

- Addressing cross-cutting concerns
- Extensibility and flexibility of the application
- Ability to break down system into semi-independent blocks, replace, configure and test each one of them individually
- Supporting infrastructure that reduces the maintenance burden

Automated Development Environment

The development environment should be designed to support the adopted practices, routines and standards. It actually implements them and provides systems, services and/or servers for:

- Resource version control
- Automated builds

5 Study Project Report

- Issue management
- Shared information management
- Communication and notification
- Deployment, and distribution of the project deliverables
- Project Management
- Unit testing and functional testing
- Documentation generation
- Automation of any other tasks that may seem necessary

Not every system is needed by every development project, since each one has unique requirements.

Those are some of the common requirements for efficient and productive development environment:

- Iterative Development Cycle
- Full testing of every code change; regular regression tests
- More than 80% of the code should be covered with tests
- Release and deployment automation should be implemented
- Integration with the external information management and delivery systems should be implemented where appropriate
- Project shareholders or responsible parties should be notified of the event as soon as it takes place

Development Implementation

That's the last layer of the responsibility in the application. It actually generates business value behind the specific project. Here are some of the well-known requirements for the success:

- Vigorous code refactoring
- Test driven development
- Proper code documentation
- Following the development standards
- Exercising development practices and utilizing development patterns that bring in code quality, stability and maintainability
- Good development ethics and sharing knowledge

Implementation

This development project concentrated on creation of the prototype of the enterprise information management software solution that supports:

- Feature rich UI
- Client-server architecture with centralized data storages
- Ability to work in offline mode
- Seamless installation and software updates
- Seamless issue reporting
- Essential functionality that includes:
 - Filtering, sorting and rich data navigation
 - Printing and export capabilities
 - Creation of custom data reports against available data
 - Simple integration with other solutions
 - Effortless error reporting
- Essential development functionality that includes
 - Highly configurable modular design
 - Separation of cross-cutting concerns from the business logic
 - Extensive logging capabilities
 - Stable handling of all exceptions and other unexpected situations
 - Ability to monitor and measure performance of the critical components in real-time
 - Simple maintenance and updates of multiple end-user installation
 - Integration of the solution with the existing development environment

Software project management

Decisions

In order to shorten total development cycle, minimize resource investments and ensure success, following decisions were made:

- Extensive period of requirements definition, detalization and high level planning was considered
- All the project initialization information was to be thoroughly documented and revised.
- Series of easy-to-implement prototypes were to be created in order to bring functional requirements close to the development opportunities and leverage risks.
- Time was invested in evaluation of the external tools, packages and libraries that would provide the most benefit to the project with the least investment.
- Effort was made for creation low-friction automated development environment that would enforce code quality and stability
- Development standards, patterns and practices were to be enforced.
- Project management process was implemented as the entity in the development environment (change notifications, documentation versioning, branches and tagging were considered for the implementation)

Statistics

Reference project took 52 days to complete. Of those:

- 42% - planning, architecture and prototyping
- 30% - initial implementation
- 19% - setting proper development environment
- 9% - developing server and server modules

Traditional waterfall methodology was accepted for the planning process and the establishment of the automated development environment. Prototyping and actual development were based on the iterative approach. All documentation was versioned along with codebase.

Automated Development Environment

Decisions

It was decided that the automated development environment should support all the necessary services outlined above with the exception of:

- Project Management
- Issue Management
- Documentation generation

This decision was made due to the following specifics of the software development project undertaken:

- Straightforward project schedule that could be managed in Excel (no benefit of setting separate Project on the server)
- Single development resource, and thus no need for the management of the resource pool
- Absence of the external project stakeholders – no need to deliver extensive project status information via Microsoft Project Web Access or other similar means.

Statistics

It took 10 days to establish proper environment. Some parts of the existing development infrastructure were reused.

- Source Control repository went through 312 revisions.
- Every code change was automatically followed by the notification in the internal newsgroup, integration testing and the success/failure notification.
- There were 37 releases.
- Every release deliverable was archived and made accessible to the development environment via FTP; notifications on these were delivered via email.
- Simple versioning policy was enforced for every release, every version was tagged in the repository
- Different stages of the project build and delivery process were clearly separated between different automation instances. This simplified management and the maintenance of the environment.

Some aspects of the environment are shown in the images section below.

Images

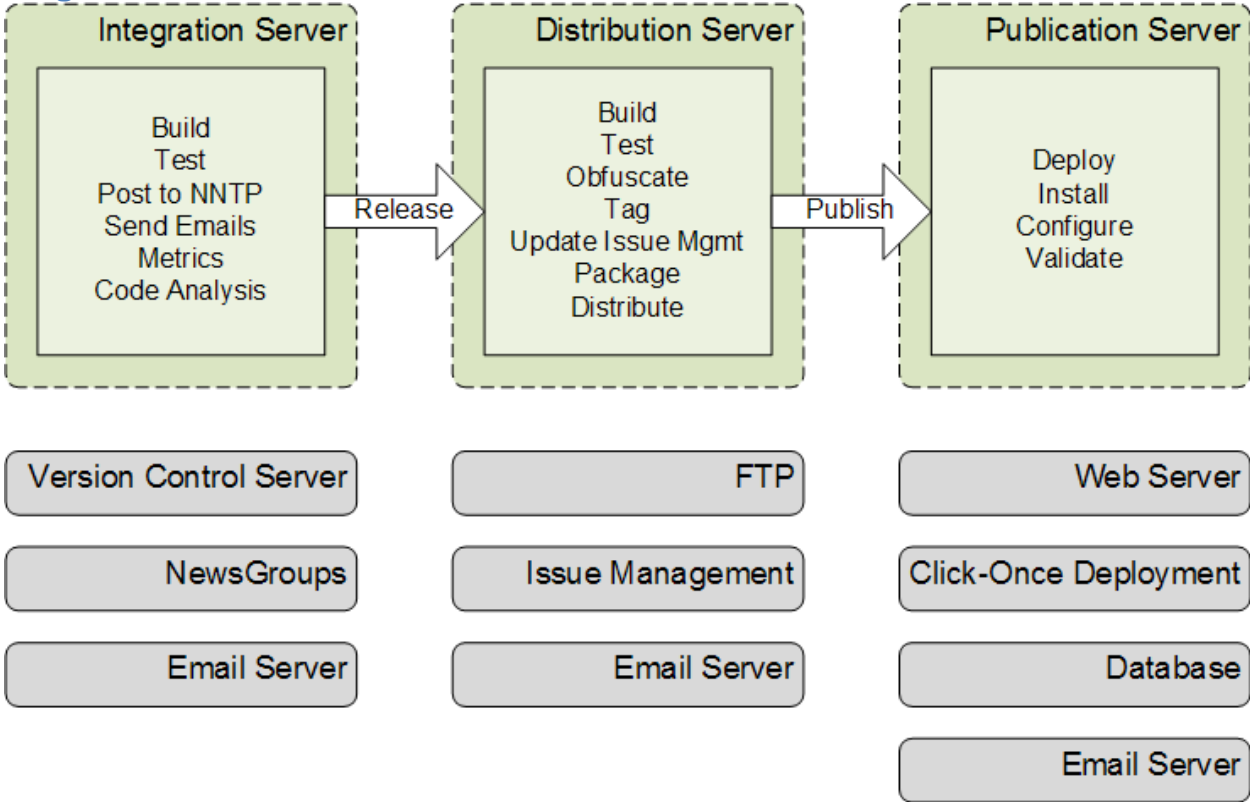


Figure 1 - Automated Development Environment

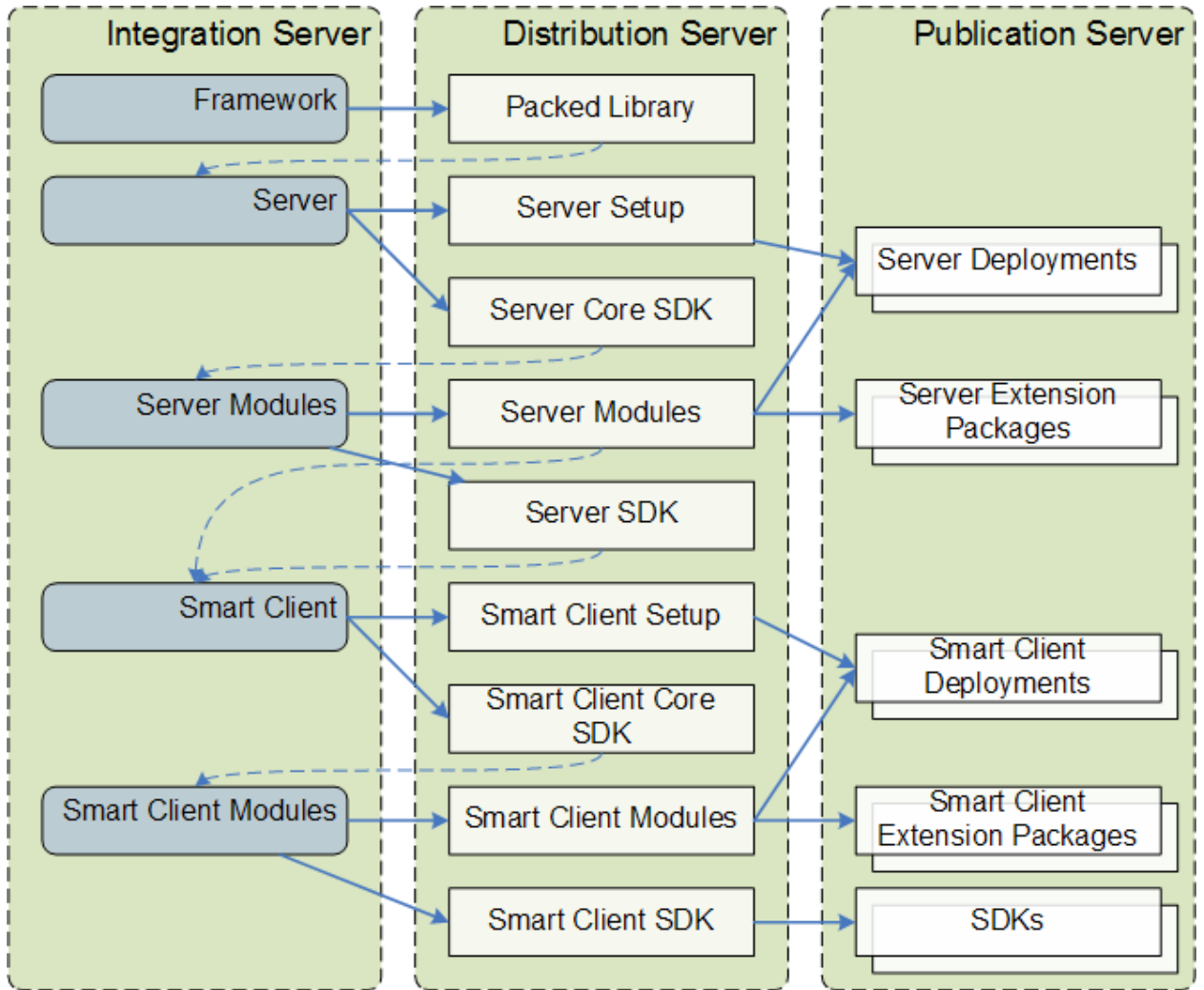


Figure 2 - Build Process

Project Name	Last Build Status	Last Build Time	Last Build Label	CCNet Status	Activity	Messages	Admin
EnterpriseLibrary	Success	2007-06-19 17:28:28	4	Running	Sleeping		
Lim.Client	Success	2007-06-24 13:53:00	26	Running	Sleeping		
Lim.Extensions	Success	2007-06-14 17:17:26	3	Running	Sleeping		
Lim.Server	Success	2007-06-24 12:16:17	16	Running	Sleeping		
Lim.Shared	Success	2007-06-19 18:13:51	7	Running	Sleeping		
ObjectBuilder	Success	2007-06-14 16:31:31	2	Running	Sleeping		
Release EnterpriseLibrary	Success	2007-06-19 17:36:04	2007.6.19.2	Running	Sleeping		
Release Lim.Client	Success	2007-06-23 17:11:53	2007.6.23.2	Running	Sleeping		
Release Lim.Extensions	Success	2007-06-14 17:19:10	2007.6.14.1	Running	Sleeping		
Release Lim.Server	Success	2007-06-24 12:18:00	2007.6.24.1	Running	Sleeping		
Release Lim.Shared	Success	2007-06-19 17:52:58	2007.6.19.1	Running	Sleeping		
Release ObjectBuilder	Success	2007-06-25 15:55:55	2007.6.25.1	Running	Sleeping		

Figure 3 - Web dashboard

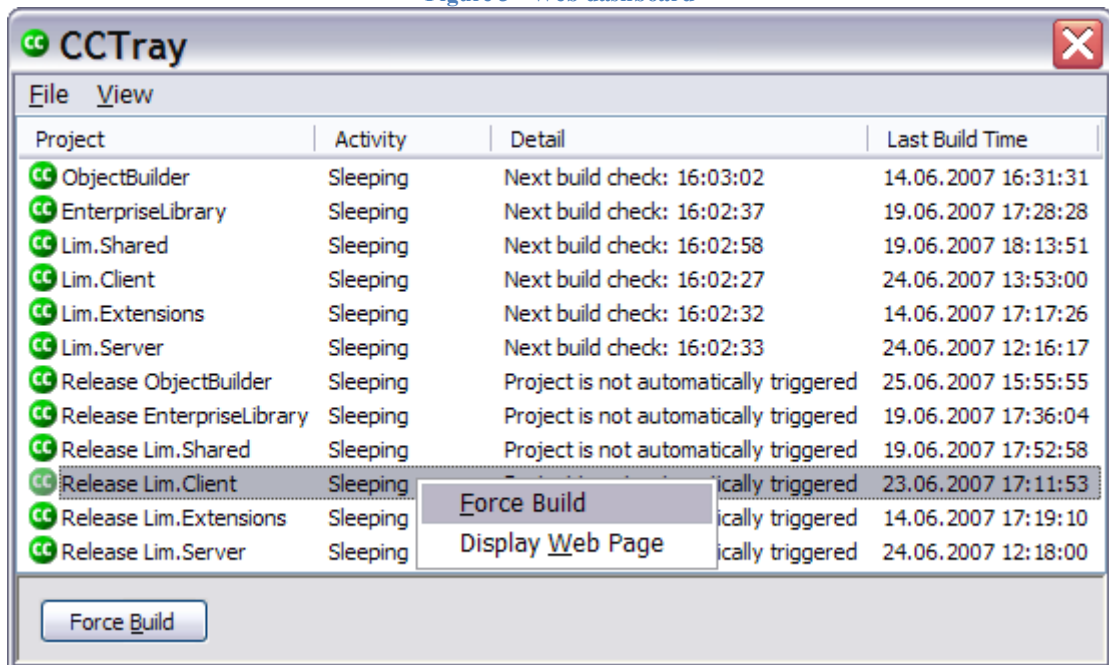


Figure 4 - Tray notification tool

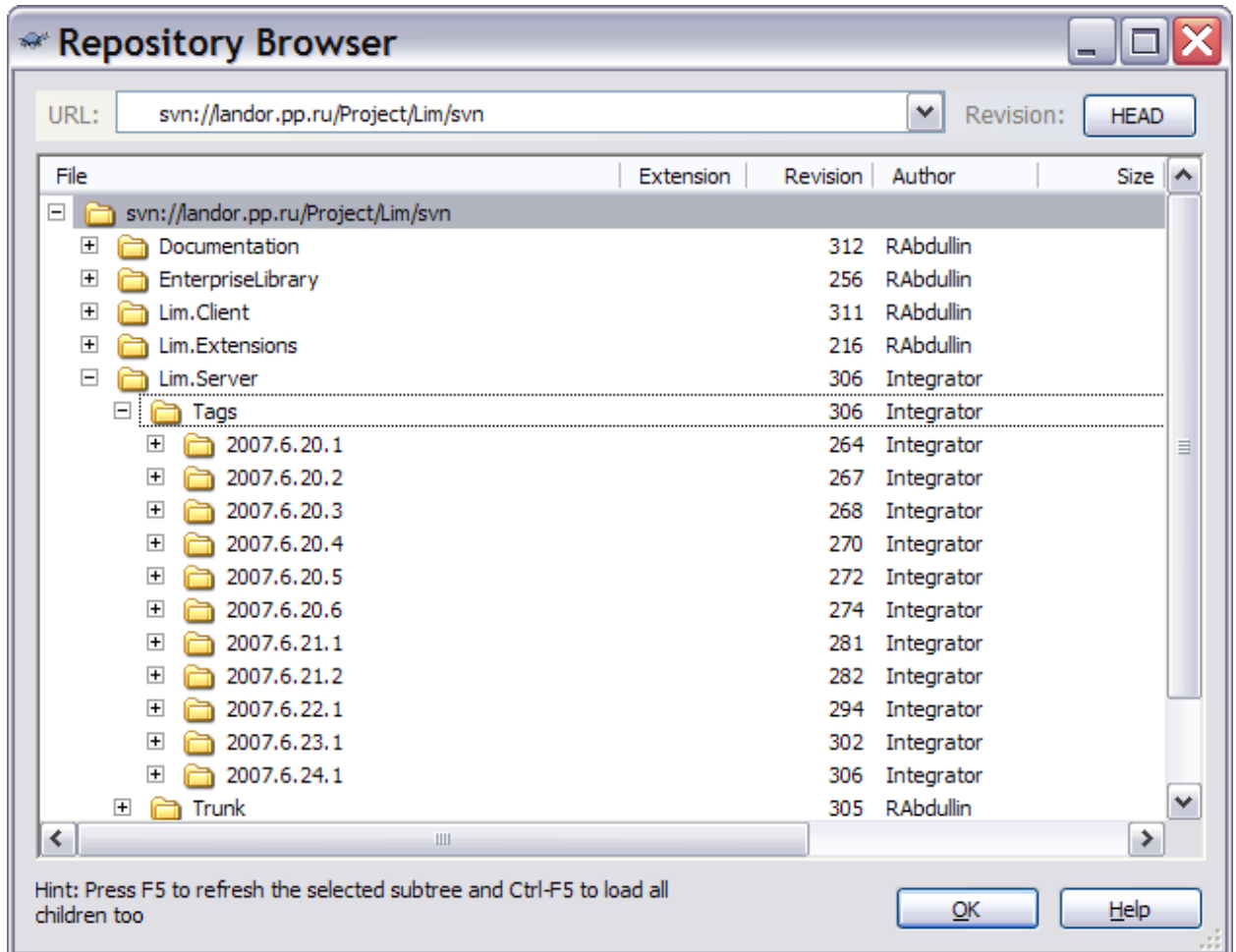


Figure 5 - Version control repository and tags created by the Integrator Service

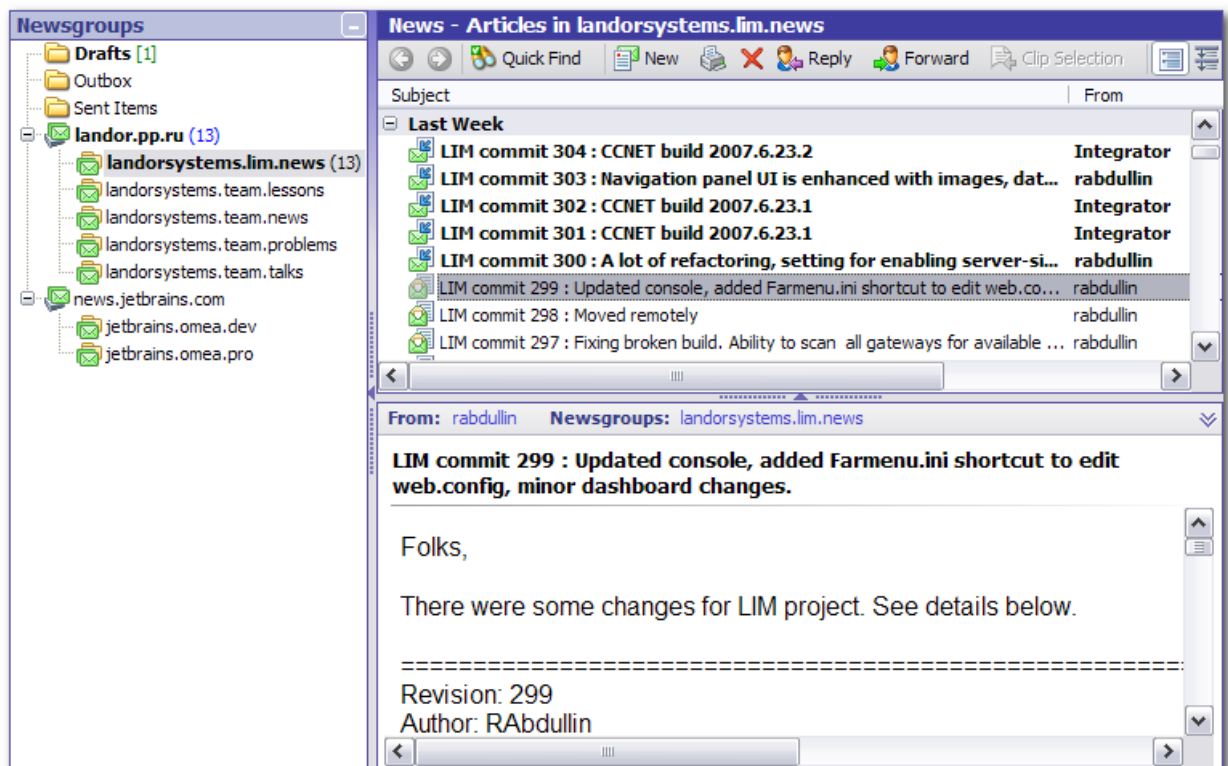


Figure 6 - Change and build notifications in the internal local newsgroup

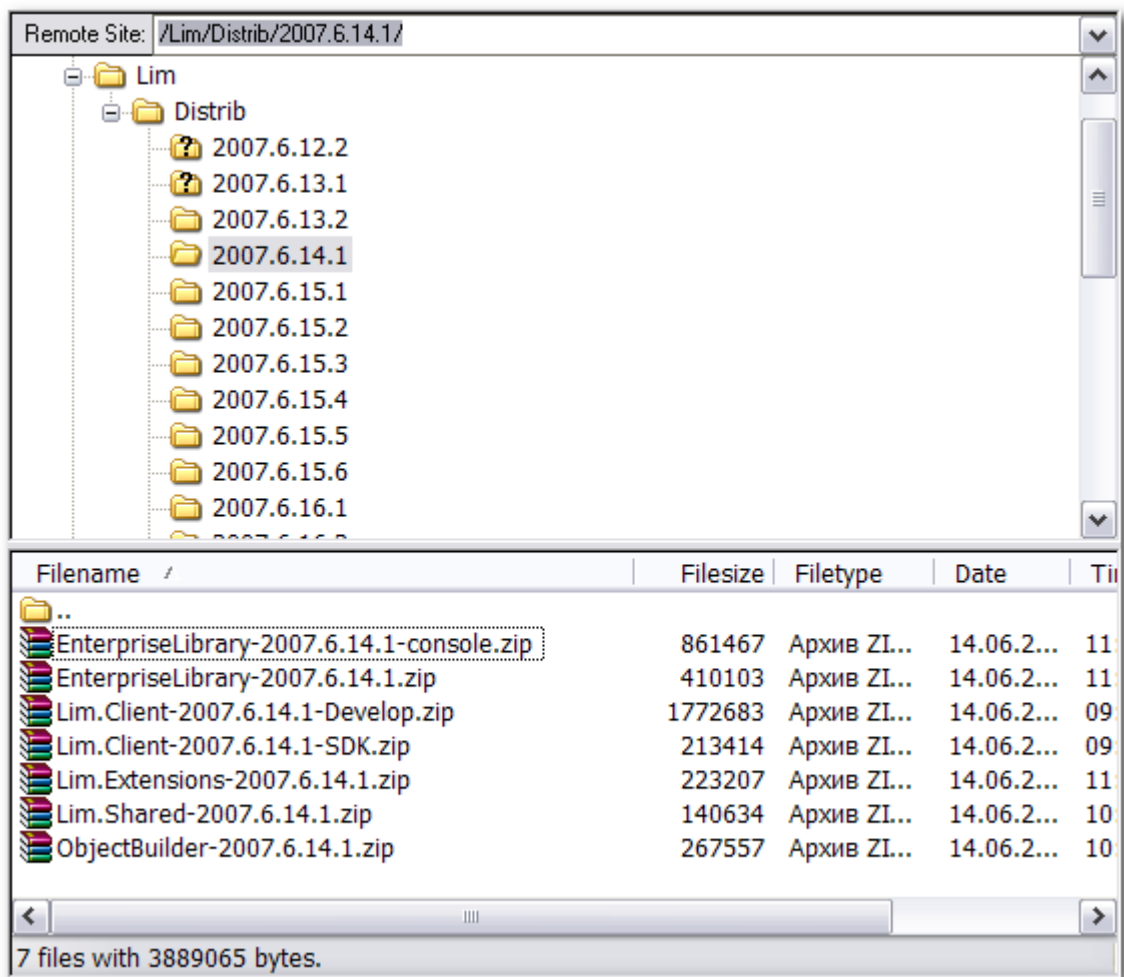


Figure 7 - Distribution packages created by the environment

Architecture and design

Decisions

Following decisions were made during the development process:

- Architecture based on the Inversion of Control and Dependency Injection was implemented as the part of the modular and extensible design
- External components were integrated into the software solution where appropriate.
- Composite UI concept was selected for the Smart Client UI
- Overall design was implemented with the testability in mind
- High-level architecture was created up-front. As the development process went on, vigorous refactoring and efficiency-oriented redesign took place.
- Issue management was considered to be integrated into the project deliverable.

Some architecture-related diagrams are shown in the images section below.

Images

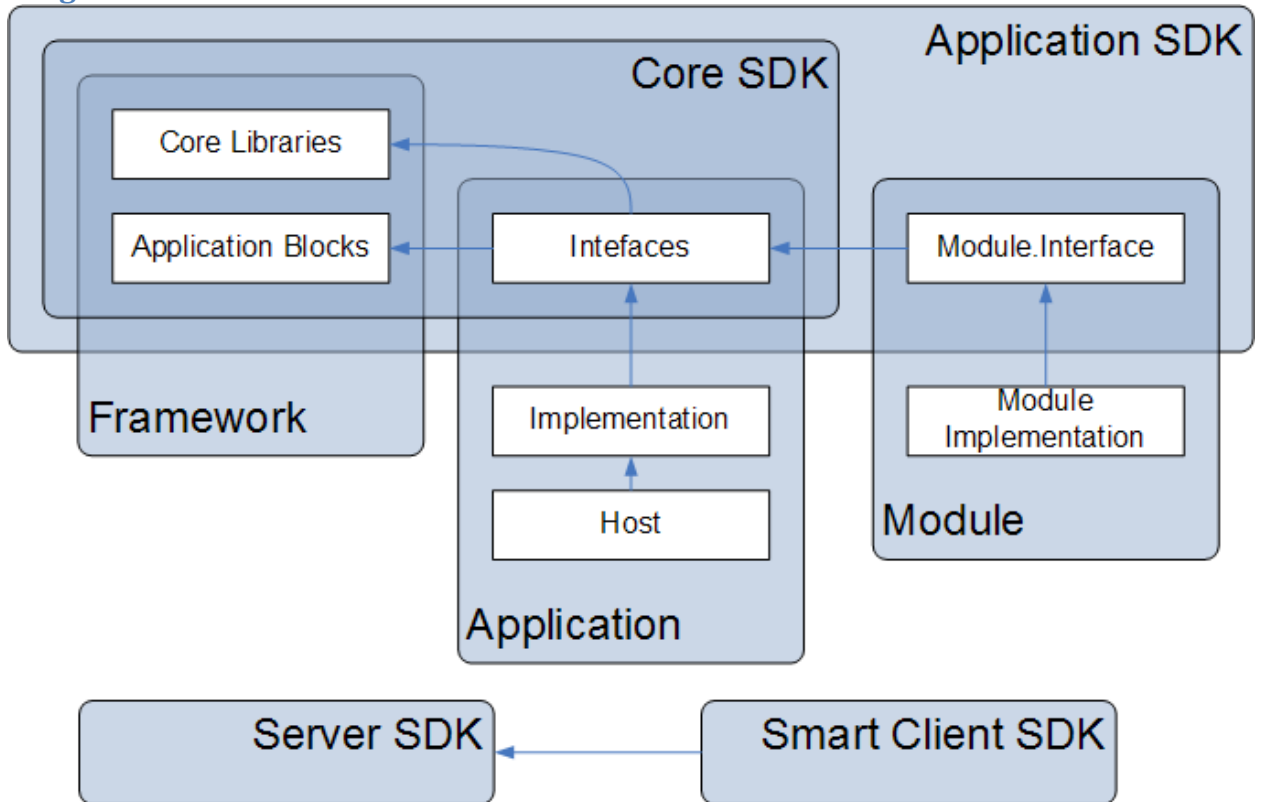


Figure 8 - Package Composition

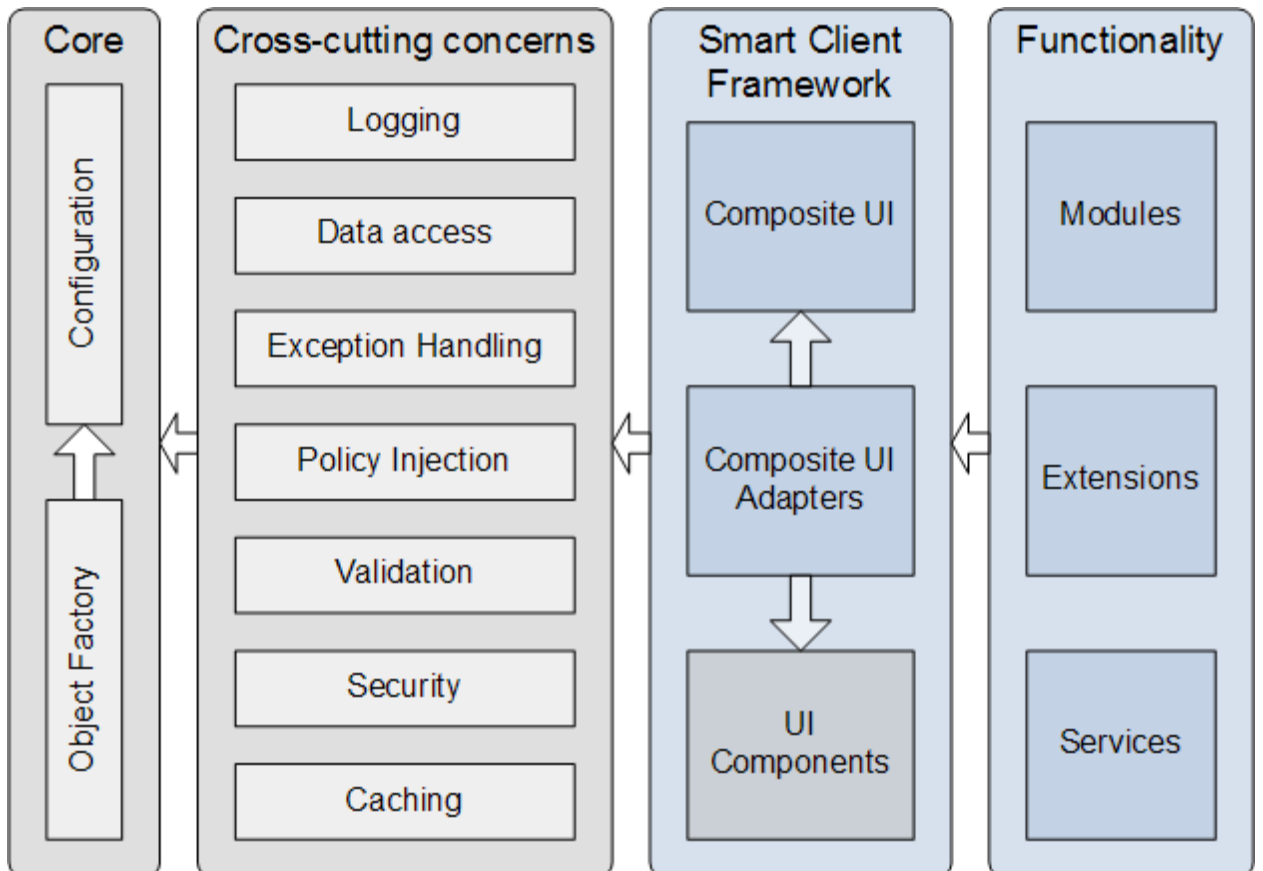


Figure 9 - Architecture overview of the Smart Client

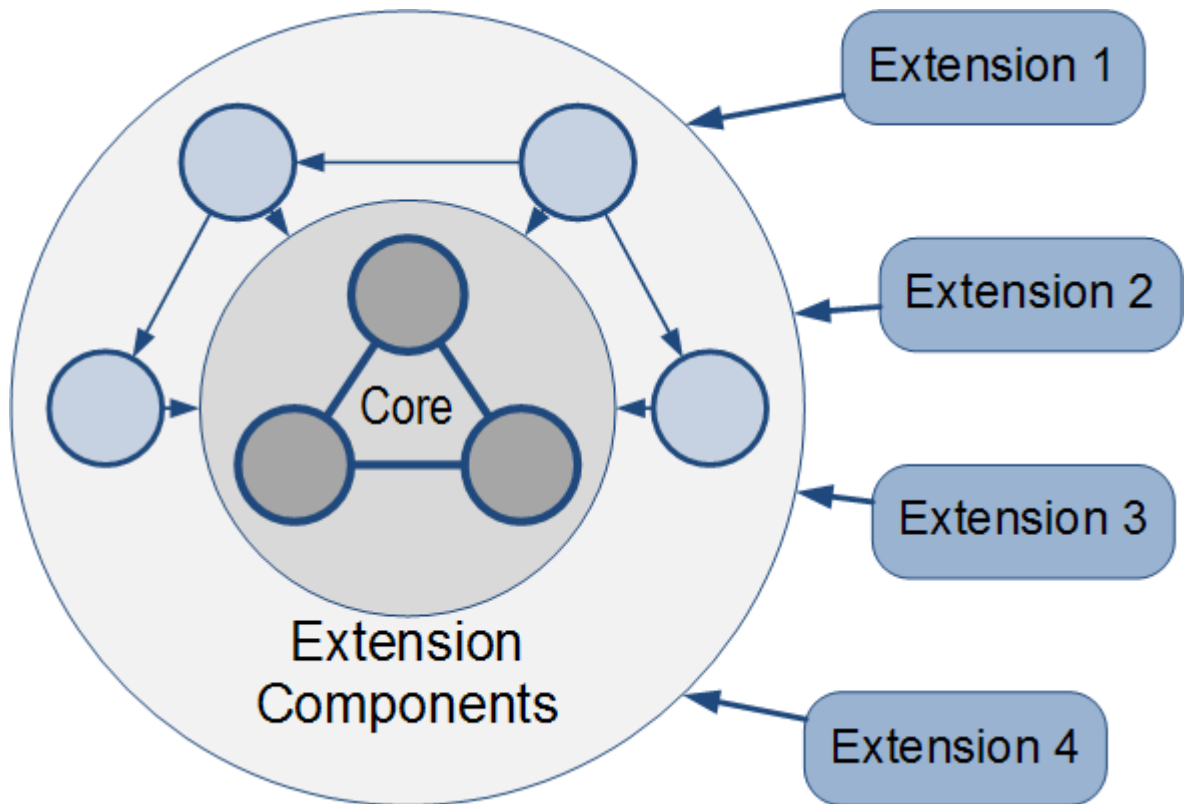


Figure 10 - Modular composition

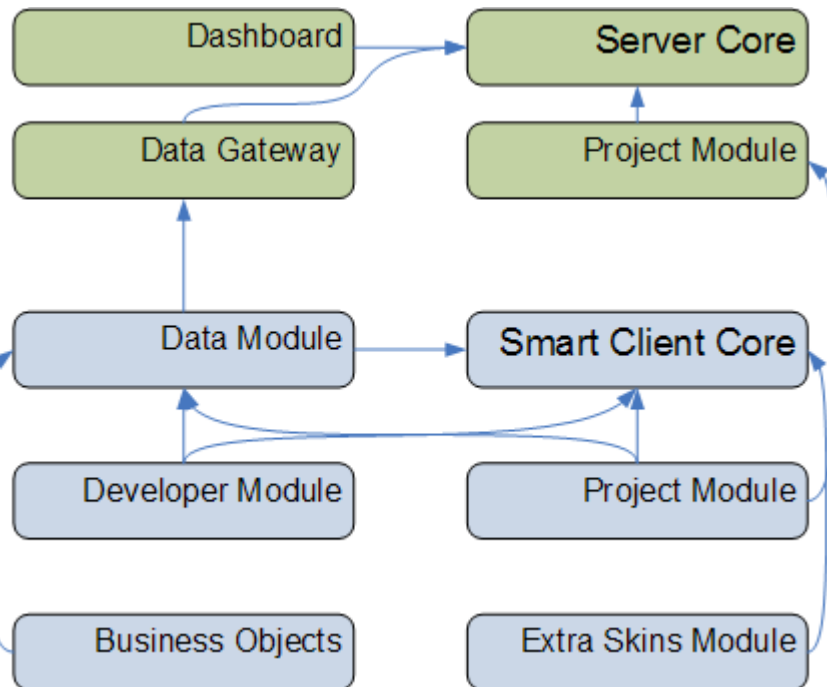


Figure 11 - Implemented modules and their relations

Development Implementation

Statistics

Total codebase under version control was 3178 of C# files (348405 lines of code). 565 C# files (70081 lines of code) constituted active codebase (heavily modified external libraries or new code). Remaining codebase is formed by the external libraries with minor modifications.

Large number of open-source libraries and development tools was used. Only one commercial component suite was chosen for this project after evaluation and prototyping.

Screenshots of the implementation can be found in Finals.zip file along with this report.