Contract Wizard II

PROJECT PLAN

Diploma project	
Project period	2003.06.06 – 2003.10.06
Student name	Dominik Wotruba
Status	9. semester
Email address	wotrubad@student.ethz.ch
Supervisor name	Karine Arnout

1. PROJECT DESCRIPTION

Overview

Building reliable software is the aim of software engineering. Assertions, which are part of the Eiffel programming language, provide the programmer with essential tools for expressing and validating correctness arguments. The key concept is Design by Contract. Viewing the relationship between a class and its clients as a formal agreement, expressing party's right and obligations, is the only way one can hope to attain a significant degree of trust in large software systems. [1]

The current implementation of .NET does not support Design by Contract yet. The goal of this project is to develop a tool, which provides the ability to add contracts like preconditions, post conditions and invariants to an arbitrary .NET assembly, which will be a big enrichment for many programming languages based on .NET.

Scope of the work

The main work of my diploma thesis is to implement a tool, called Contract Wizard II. There already exists a tool called Contract Wizard developed by Karine Arnout. The main difference between Contract Wizard II and Contract Wizard will be the implementation.

I will write Contract Wizard II in Eiffel for .NET and this way take advantage of the underlying .NET-technology. This will enable turning Contract Wizard II to a Web service later.

I will implement the tool considering future Eiffel language extensions, like overloading. Furthermore, all inserted contracts will be stored in XML. This has the advantage that for example, tools for automatic contract extraction can be easily incorporated.

The work consists of the following parts (See Figure 1 below):

Parser	Implement a parser which makes use of the reflection mechanism of the .NET technology
Name Resolver	Implement a Name Resolver based on the core algorithm for the Eiffel.NET language [7] to translate .NET names into Eiffel names.
Contracts	Definition of a DTD (document type definition) to store contracts in XML (one XML contract file per class).
Basic GUI	Implement a basic GUI (Graphical User Interface) which enables adding contracts

Contract Reader Implement a Contract Reader which reads contracts defined by the mentioned DTD

Basic GUI and Contract Handler Contract Handler providing the ability to add contracts to the Symbol DS (A data structure containing information about types and features). The Control Handler will be used by the Basic GUI

Code Generator Implement a Code Generator consisting of the following parts:

- **XML Writer**: The XML Writer stores contracts in XML files based on the DTD (one XML file per class)
- **ACE File Producer:** The ACE File Producer generates an ACE (Assembly of Classes in Eiffel) file needed by the Eiffel compiler
- **Proxy Class Generator:** Generates proxy classes in Eiffel containing the contracts inserted by the Contract Handler

Eiffel Compiler A class which calls the Eiffel compiler to compile the contracted classes

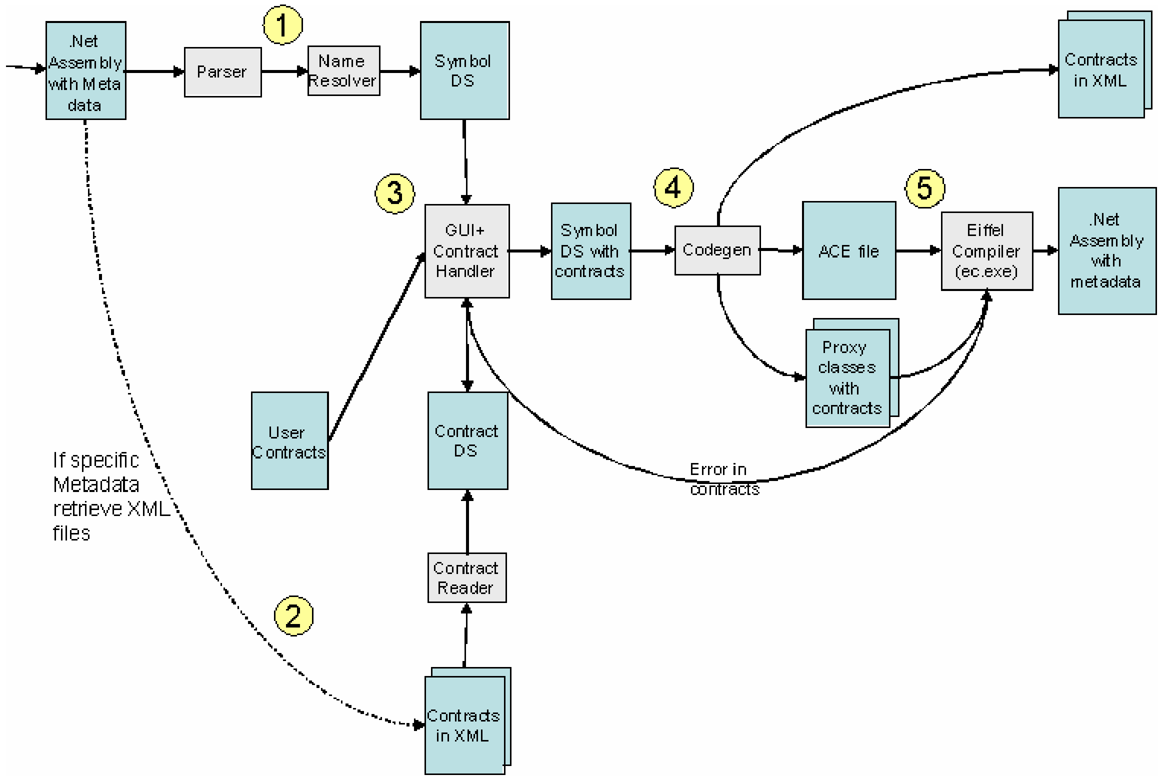


Figure 1: Software Architecture

(1) If there is no specific Metadata the parser parses the .NET assembly and produces a Symbol DS. The Name Resolver keeps track of renaming .NET names to Eiffel names

(2) If there is specific Metadata (e.g. a path to the XML contract files) in the .NET assembly expressing that there already are contracts for it, the Contract Reader reads the contracts from XML contract files and produces a Contract DS. Furthermore like in point (1) a Symbol DS is produced.

(3) Both the Contract DS and the Symbol DS are handled by the Contract Handler and the basic GUI. They produce a Symbol DS containing information about the .NET assembly and the contracts. The Contract Handler and the basic GUI are also responsible for user contracts. User contracts are append to the Contract DS

(4) The Code Generator traverses the Symbol DS with contracts and produces proxy classes with contracts. The proxy classes will contain metadata expressing where the Contract Files are located. Another task of the Code generator is to generate XML files storing all information about contracts and to produce the ACE files needed by the Eiffel compiler

(5) The Eiffel compiler is called by the GUI and compiles the generated proxy classes using the ACE files. If the contracts are not correct an error message is displayed and the contracts can be corrected taking the error messages into account

Intended Results

Implementation:

Parser	Implementation of a parser, which parses a .NET assembly and produces a Symbol DS. **Scope:** • The Parser keeps track of .NET types and .NET methods
Name Resolver	Implementation of a Name Resolver. **Scope:** • The Name Resolver keeps track of .NET type names and .NET method names. It uses the Eiffel [7] core algorithm for renaming
Contracts	Implementation of a DTD to store contracts in XML
Contract Reader	Implementation of a Contract Reader based on the mentioned DTD
Basic GUI	Implementation of a basic GUI for demonstration purposes to enable adding contracts (preconditions, post conditions and invariants)
Contract Handler	Implementation of the Contract Handler which is used by the basic GUI
Interfaces	Specification and implementation of an interface towards the

	GUI
Code Generator	Implementation of a code generator consisting of the following parts: • **XML Writer:** Implementation of an XML writer which enables storing contracts in an XML file based on the specified DTD. (One XML contract file per class). • **ACE File Producer**: Implementation of an ACE File Producer, which keeps track of the .NET assembly and the Eiffel proxy classes • **Proxy Class Generator**: Implementation of a Proxy Class Generator which produces Eiffel classes with contracts. **Scope:** Generation of a proxy class for every class (subclass) in the .NET assembly • **Assembly Updater:** This part updates the proxy classes with Metadata (e.g. path to the XML contract files)
Demo application	Implementation of a demo demonstrating the functionality and scope of Contract Wizard II
Test cases	Test cases that will be used for the demo

Documentation:

Developer manual	Writing a developer manual (see documentation in chapter project management, page 6)
User manual	Writing a user manual (see documentation in chapter project management, page 6)
Intermediary report	Writing an intermediary report (see documentation in chapter project management, page 6)
Thesis report	Writing a thesis report (see documentation in chapter project management, page 6)

2. BACKGROUND MATERIAL

Reading List

- Chapters in OOSC2 [3] in particular:
    - 1 Software quality
    - 11 Design by Contract: building reliable software
    - 23 Principles of class design
    - 26 A sense of style
    - 28 The software construction process
- NET Training Course [4]
- Rapport de stage “Jeune Ingénieur [2]

Software

- Source code of Contract Wizard

3. PROJECT MANAGEMENT

Objectives and priorities

Objective	**Priority ***
Software architecture	1
Parser	1
Contract Reader	1
Contract Handler	1
Code Generator	1
Demo application	2
Basic GUI	3
Test cases	1
Optimization	3
User documentation	2
Developer documentation	1
Intermediary report	3
Thesis report	1

* 1 means highest priority, 3 means lowest priority

Criteria for success

While the focus on this project is quality the result may be a partial implementation of the intended results and objectives without implying any penalty on the success of the project.

Quality of software:

- Use of Design by Contract
    - Routine pre- and post conditions
    - Class invariants
    - Loop variants and invariants
- Careful design
    - Design patterns
    - Extendibility
    - Reusability
    - Careful abstraction
- Core principles of OOSC2 [3]
    - Command/query separation
    - Simple interfaces
    - Uniform access
    - Information hiding
    - etc.
- Style guidelines
- Correct and robust code
- Readability of source code
- Ease to use

Quality of documentation:

- Completeness
- Understandable documentation
- Usefulness
- Structure

Method of work

The technologies involved are:

- The System. Reflection library from the .NET Framework [7]
- The System.XML library from the .NET Framework [7]
- Programming language: Eiffel for .NET [7]

- ISE Eiffel compiler beta version 5.4 for test purposes [7]
- Eiffel Build for the basic GUI [7]
- The development tool is Eiffel Studio 5.3 [7]

I will develop the software using the Eiffel style of design in close cooperation with the supervisor.

Quality Management

Quality will be ensured by:

- **Weekly progress reports:** Short weekly progress reports sent to the supervisor
- **Milestone progress reports:** Detailed reports for each milestone (see plan with milestones below)
- **Review and Validation:** Review of each milestone by the supervisor concluded by a meeting
- **Validation:** Validation of each milestone after review (see validation steps below)
- **Testing:** Testing of the software by application of different test cases

Documentation

- **Progress reports:**
    - Short weekly progress reports consisting of the main tasks completed
    - Detailed reports for each milestone consisting of:
        - The main tasks
        - Eventual encountered difficulties
        - Implementation, scope of the implementation
- **Developer report**: This manual documents the software architecture and its limitations, describes the difficulties encountered during the implementation, explains how the software could be extended and contains a section discussing the test cases.
- **Intermediary report**: The intermediary report consists of the intermediary developer report
- **User manual:** The user manual describes the usage of the tool Contract Wizard II
- **Thesis report:** The thesis report consists of the final user manual, the final developer report and a theoretical part discussing the possibility to store the contracts in the assembly metadata

Validation Steps

The validation for each milestone comprises:

- **Report:** Sending detailed report and the relevant parts of the work to the supervisor for review
- **Meeting:** Organizing a meeting with the supervisor or presentation and discussion of the conducted work

- **Revision:** Revision of parts of the work or all of the work for this milestone, depending on the conclusion of the supervisor

4. PLAN WITH MILESTONES

Project Steps

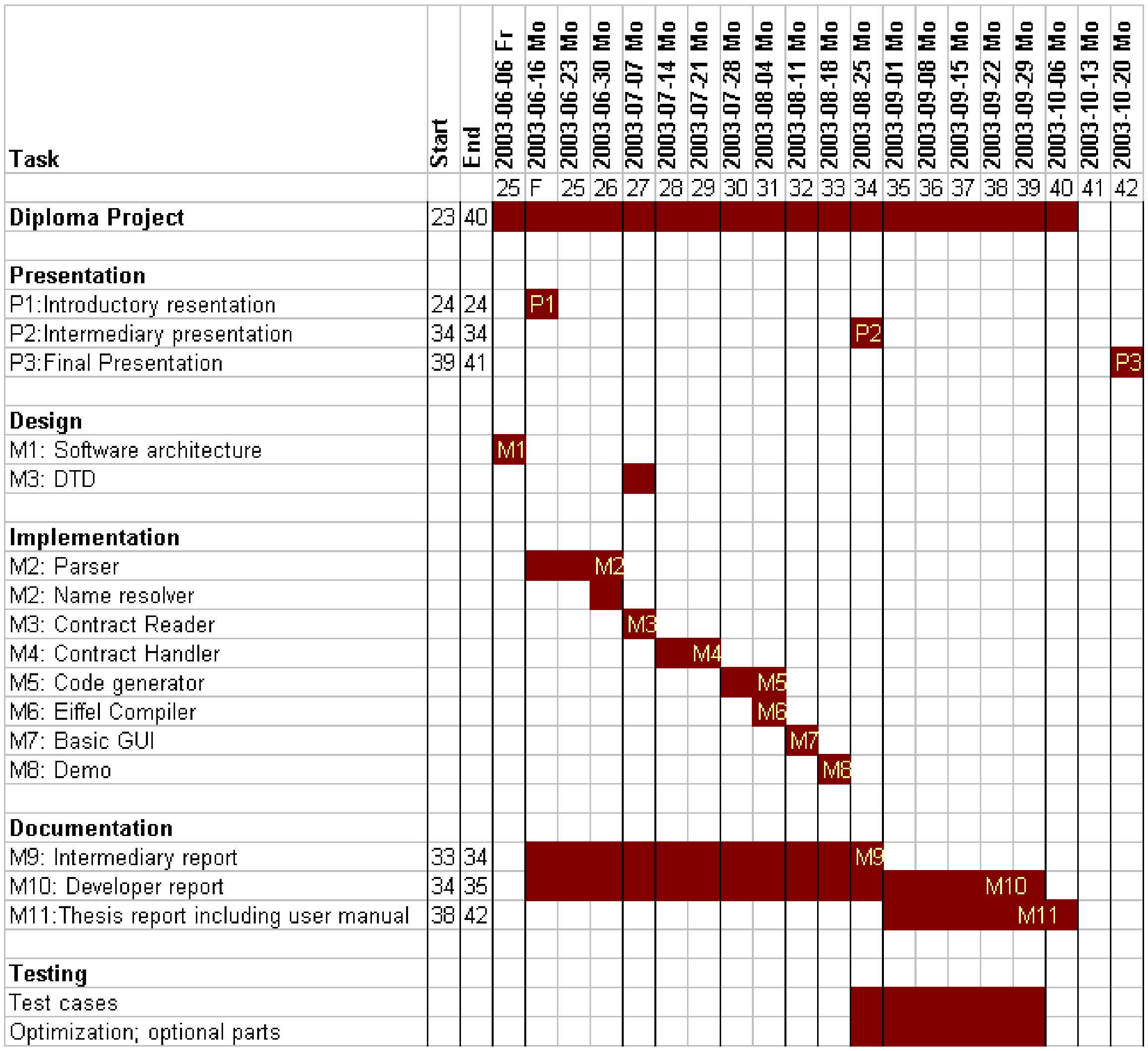
Milestones	Objectives
M1: Software architecture	Design of the entire Software Architecture (This includes basic interfaces and contracts for the Parser, the Name Resolver, the Contract Reader, the basic GUI, the Contract Handler, the Code Generator, the Symbol DS and the Symbol DS with contracts). The Software Architecture will be refined in every programming milestone.
M2: Parser	Implementation of the Parser, Name Resolver and the Symbol DS
M3: Contract Reader	Designing the DTD and implement the Contract Reader based on it
M4: Contract Handler	Implement the Contract Handler
M5: Code Generator	Implement the Code Generator consisting of three parts: • XML-Writer • Ace-File producer • Proxy Class generator
M6: Eiffel Compiler	Implement a class which calls the Eiffel Compiler
M7: Basic GUI	Basic GUI (only for demonstration purposes)
M8: Demo	Implement a demo which illustrates the capabilities of the software
M9: Intermediary Report	Write the intermediary report. Presentation of the intermediary results in a group meeting
M10: Developer Report	Write the developer report
M11: Thesis Report	Write the thesis report, including the user documentation, the developer documentation, test case documentation, a theoretical part discussing the possibility to store the contracts in the assembly Metadata and a project review. Presentation of the project results in a group meeting

5. DEADLINES

Milestone	Deadline
M1	2003-06-13 Fr
M2	2003-07-04 Fr
M3	2003-07-13 Fr
M4	2003-07-25 Fr
M5	2003-08-08 Fr

M6	2003-08-08 Fr
M7	2003-08-15 Fr
M8	2003-08-22 Fr
M9	2003-08-29 Fr
M10	2003-10-03 Fr
M11	2003-10-11 Fr

6. TENTATIVE SCHEDULE



Task	Start	End	2003-06-06 Fr	2003-06-16 Mo	2003-06-23 Mo	2003-06-30 Mo	2003-07-07 Mo	2003-07-14 Mo	2003-07-21 Mo	2003-07-28 Mo	2003-08-04 Mo	2003-08-11 Mo	2003-08-18 Mo	2003-08-25 Mo	2003-09-01 Mo	2003-09-08 Mo	2003-09-15 Mo	2003-09-22 Mo	2003-09-29 Mo	2003-10-06 Mo	2003-10-13 Mo	2003-10-20 Mo
			25	F	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Diploma Project	23	40	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥		
Presentation																						
P1:Introductory resentation	24	24		P1																		
P2:Intermediary presentation	34	34												P2								
P3:Final Presentation	39	41																				P3
Design																						
M1: Software architecture			M1																			
M3: DTD							🟥															
Implementation																						
M2: Parser				🟥	🟥	M2																
M2: Name resolver						🟥																
M3: Contract Reader							M3															
M4: Contract Handler								🟥	M4													
M5: Code generator										🟥	M5											
M6: Eiffel Compiler											M6											
M7: Basic GUI												M7										
M8: Demo													M8									
Documentation																						
M9: Intermediary report	33	34		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	M9								
M10: Developer report	34	35		🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	M10	🟥			
M11:Thesis report including user manual	38	42													🟥	🟥	🟥	🟥	M11	🟥		
Testing																						
Test cases														🟥	🟥	🟥	🟥	🟥	🟥			
Optimization; optional parts														🟥	🟥	🟥	🟥	🟥	🟥			

Figure 2: Gantt chart

7. ACRONYMS

Acronym	**Description**
OOSC2	Object Oriented Software Construction Second Edition, the book by Bertrand Meyer, is used as the primary programming and language reference for implementing the software of this project in Eiffel [3].

XML	Extended Metal Language 8
.NET	*"The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet"* [7] *The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework"* [7].
DTD	Document Type Definition 8
ACE	Assembly of Classes in Eiffel
DS	Data structure
GUI	Graphical User Interface

8. REFERENCES

[1] Karine Arnout, and Raphaël Simon. "The .NET Contract Wizard: Adding Design by Contract to languages other than Eiffel", IEEE Computer Society, TOOLS 39, Santa Barbara, USA - July 2001, p: 14-23.

[2] Karine Arnout: Rapport de stage "Jeune Ingénieur: Développement de la technologie Eiffel dans l'environnement .NET de Microsoft", 2001

[3] Bertrand Meyer: Object-Oriented Software Construction, 2nd edition, Prentice Hall, 1997.

[4] Bertrand Meyer: .NET Training Course, Prentice Hall, 2001.

[5] Acronymfinder.com: http://www.acronymfinder.com, consulted in June 2003

[6] Chair of Software Engineering: Semester-/Diplomarbeiten; Online at: http://se.inf.ethz.ch/projects/index.html, consulted in Mai 2003.

[7] Eiffel Software, http://www.eiffel.com, consulted in June 2003.

[8] The Microsoft .NET library API: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netstart/html/cpframeworkref_start.asp?frame=true, consulted in June 2003.