

# Project Description

## Contents

- Topics
  - Query architecture design
  - Vision 4 Mac project
- Deliverables
- Deadline
- Presentation

## Topics

There are two project options, query architecture design and Vision 4 Mac.

### Query architecture design

#### Description

The aim of the project is to design a query architecture which can retrieve or derive information from some data in particular format, or data model. This kind of query is ubiquitous in our life. For instance, when you are searching emails from a certain sender, or when you are locating a building by its street and door number. Here are two more detailed examples:

#### Yahoo Pipes

Yahoo Pipes (<http://pipes.yahoo.com/pipes>) is a customizable RSS reader. You can specify different media as RSS source, such as subscription XML file, HTML page or pure text. Several sources can be combined to form a larger source. And then search criteria can be used to filter out some items, leaving only interesting ones. Searching criteria are often of the form “contains keyword Mac”, “publish date is before 2.2008”.

Here, the data model is RSS XML format, the query architecture works by applying predicates to select satisfied items, and the result format is also a RSS XML format.

### EiffelStudio Metrics tool

EiffelStudio contains a tool that implements a general query architecture for an Eiffel project.

Given a Eiffel project, you can ask for information such as:

- Classes in the project which are deferred
- Features in a given set of libraries which are called by another feature “LINKED\_LIST.extend”

Here, the data model is a set of queryable elements, such as a set of libraries, a set of classes, or a set of features. Given a set of elements as source, the query architecture first transforms the source when necessary, and then applies predicates to the source to select interesting elements. Transformation of the source is needed when the wanted result is of different type from the source.

For example, for the query: “deferred features in some classes”, source is a set of classes, and the wanted result is a set of features, they are of different types. So first, the set of classes need to be transformed into a set of features by collecting every feature in every class, and then the predicate “a feature is deferred” is applied to that set of features to give the correct result.

More information for EiffelStudio developing can be found at:

[http://eiffelsoftware.origo.ethz.ch/Main\\_Page](http://eiffelsoftware.origo.ethz.ch/Main_Page)

### **Objective**

You are asked to devise some data model and design the query architecture based on that model using principles, design patterns taught in the lecture.

The architecture should be extendable. In the Yahoo Pipes example above, good extendibility will enable you to add new types of search criteria without massive code change, while in the Metrics tool example, it will allow you to extend new queryable element such as AST nodes.

The goal of this project is to design the architecture right, so efforts should be focus on good software engineering practice and the usage of patterns. This project can be conducted by a group with at most 4 people.

### **Development environment**

The project has to be developed using a particular version of EiffelStudio downloadable from <http://dev.eiffel.com/CddBranch>. This version (based on EiffelStudio 6.1) includes the extension for Contract Driven Development (CDD). CDD is an extension for EiffelStudio that supports you in your testing activities. The CDDBranch website has more information on this.

### **Regular Subversion Commits**

It is important that you commit your project on a regular basis (e.g. once a week). In particular please make sure you commit the directory named “**cdd\_tests**”. This directory will be in the directory of your programs **.ecf** file. Also make sure that all files in the “**cdd\_tests**” directory are always added to the svn repository. To check if all files are indeed added, go to the “**cdd\_tests**” directory and run “**svn stat**”. There should be no files marked with a questionmark (?). If there are any make sure you add them via “**svn add my\_new\_file.ext**”.

The information contained in the “**cdd\_tests**” directory contains the test cases of the program, information about the outcomes of the test cases and various timing related information. This data will be used for an empirical scientific evaluation. For the purpose of the study the data will be treated anonymously and the scientific evaluation has no effect on your grade. Please contact your assistant right away if you have objections to this.

## **Vision 4 Mac Project**

### **Description**

The goal of the Vision4Mac Project is to provide a Mac OS X backend for the ISE Eiffel’s Vision2 library. Currently there are implementations for Windows and GTK (Linux). The project has already been started two years ago, so many features are already implemented while others are still missing. It was decided to use Mac OS X’s Carbon API to implement a native Mac OS X GUI. The vision library employs the Bridge pattern to make different implementations accessible through a common interface. The idea of the project is that one person gets to work on one or more quite specific parts of the port (see possible projects); for bigger tasks groups of up to four people are also possible. For more information on the Vision4Mac project see the wiki page: <http://eiffelsoftware.origo.ethz.ch/Vision4Mac>

### **Why you should pick a Vision4Mac project**

- Two assistants with extensive knowledge about the project are here to help you with problems that might come up.
- You will learn about Mac OS X’s Carbon API and gain intimate knowledge of the vision library
- You will work on something that will eventually be used.

- You will see a non-artificial application of patterns.

### **Possible Projects**

- **Menu**

We need to deal with the problem that in Vision2 every window contains its own menu bar, while in a Mac OS application only the menu of the active window is available.

- **EV\_TABLE**

This is the only containers for which doesn't have a Carbon implementation, yet. It organizes widgets in a table.

- **EV\_DRAWABLE\_AREA, EV\_PIXMAP**

These are the basic widgets for customized drawing. The only difference between the drawable area and the pixmap is that the pixmap uses a buffer, while the drawable area draws directly on the screen. This is a good project for a larger group (around four) of students interested in drawing.

- **EV\_FONTABLE**

In Vision2 there are always lots of widgets which share some properties. Almost any Widget that contains text can display this text in different fonts. The project is to implement the representation of the Mac OS fonts in Vision2 and the features to set and get a font.

- **Dialogs**

There are a lot of standard dialogs like "File Open" which need to be implemented.

- **Events**

There are a lot of events that should be implemented in Carbon. Implementing an event can be challenging, because often the event we need in Vision is not available in Carbon and therefore one has to come up with a smart idea.

- **Pick and Dropable**

Design and implement Pick and Drop for the Mac.

- **Dockable**

Dockable is the technique in Vision2 that makes it possible to reorganize the different tool windows in Eiffel Studio with drag and drop.

- **ToolBar**

In a Mac OS application the toolbar has to be on top or on the right side of a window. Unfortunately a Vision2 a toolbar can be everywhere. The goal of this project is to design and implement a toolbar that uses Vision2 components.

### **Development environment**

Since the Carbon API is only available on Mac OS X, you need a computer running Mac OS X 10.4 or newer.

You can get EiffelStudio on Mac at: <http://eiffelstudio.origo.ethz.ch/download>  
Both 6.1 and 6.2 are OK for this project.

## **Deliverables**

Every project will have an own page on Origo (see <http://origo.ethz.ch>). This page must contain:

- Motivation of the project, why it is interesting and useful
- An overall description of the problem and the solution
- A design document or wiki pages, emphasizing concepts from the lectures, such as how design patterns are used. A design document should describe the major architectural decisions, explained through any appropriate means such as textual descriptions, BON/UML diagrams etc.
- A description of how you tested the project, the tests themselves, and the results
- The source code of the project in the subversion repository
- Executables of the project
- Any other complementary element that you find useful

## **Deadline**

Final submission deadline is **May 14<sup>th</sup>**.

## Presentation

Some projects will be presented in the lecture on **May 20<sup>th</sup>**. Please contact your assistant for detailed presentation schedules.

If you have any questions, please don't hesitate to ask your assistants. See course webpage for email addresses.