

Projects' proposals:

1.	C++ Memory Leak Detector / Analyzer	1
2.	GdbLogReplay	2
3.	Usage statistics	3
4.	Java Grid Form Factory	3
5.	Hardware Viruses.....	5
6.	Hardware Viruses Design Analyser	5
7.	Radio-on-Chip	5
8.	TC Transceiver.....	6
9.	CLB Transceiver	6
10.	Dynamic Logic Analyzer	6
11.	ERDB.....	7
12.	Hardware Performance Tuner	7
13.	Evaluation of GIS sources for use in desktop application	8
14.	Large XML viewer.....	8
15.	Programming Contest.....	8
16.	Department equipment inventory	9
17.	Scrum Planner	10
18.	Wireshark plugin for LTE messages.....	10
19.	Robot Framework K3 plugin	10
20.	LTE Performance Counters Manager.....	10
21.	License Manager	11
22.	License Reporter	11
23.	Eclipse remote C/C++ debugging.....	11

1. C++ Memory Leak Detector / Analyzer

Scope: Tool for memory leaks detection and memory usage analysis in C/C++ applications

Duration: 1-4 people / 1-2 semesters

Description:

The goal of this project is extension of existing Visual Leak Detector library which is free software allowing memory leaks detection in C/C++ applications written using Visual Studio. VLD is, however, quite hard to use – it has no graphical user interface and currently cannot be used for tracking of memory usage in different parts of the software and functionalities it offers are in general quite basic. Created VLD extension would have to have following functionalities:

Requirements:

- Easy and intuitive graphical user interface
- Reports creation for detected memory leaks
- Easy tracking of memory usage in different parts of the software (filtering per module, function, selected class, etc.)
- Smooth integration with Visual Studio (plug-in?)
- Good performance allowing analysis of applications of different size

Restrictions:

- Support for C/C++ based applications
- Support for visual managed classes not required
- Support for Microsoft Visual Studio 2005 and higher
- Support for other C/C++ based libraries (STL, boost, QT)
- (optional) integration with Visual Studio – could be just separate tool allowing analysis of VLD output

Additional Information: <http://sites.google.com/site/dmoulding/vld>

2. GdbLogReplay

Scope: Log parser simulating breakpoints for debugger

Duration: 1-4 people / 1-2 semesters

Description:

Complex bug reports may contain megabytes of logs. Solving this kind of problems requires from developer to manually match logs with program execution flow. The idea of this project is to automatically parse print logs and simulate them as breakpoints for debugger. At least two applications are needed to be written. First: application for matching print logs with places in program (using ELF/DWARF information). Second: server for gdb (similar to gdbreplay) for emitting breakpoints.

Requirements:

- provide configurable log<->breakpoint matching application (with manual matching possibility)
- provide gdb server application for emitting breakpoints (server should be able to dynamically filter breakpoints)
- requirements for log matching should be provided
- at least command line interface for all programs
- functionality should be implemented as libraries with C/C++ interface

Sample use case:

Input:

- * A.exe application (ELF executable) with DWARF debugging information included
- * a.log application execution log

How gdblogreplay could be used:

```
# find breakpoints in executable A.exe for a.log logs and write them to  
a.breakpoints file (log2breakpoint needs to be written)  
log2breakpoint A.exe a.log -o a.breakpoints
```

```
# start gdb server on :1234 port (gdblogreplay needs to be written)  
gdblogreplay A.exe a.breakpoints :1234
```

on second terminal:

```
#start debugger (standard gdb)  
gdb A.exe >#in debugger, select remote target, connect to gdblogreplay  
>target remote :1234  
>#after continue gdb will stop, pointing the place in source code where first  
print log is placed  
>continue
```

3. Usage statistics

Scope: Solution for gathering usage statistics for Java tools

Duration: 2-4 people / 1-2 semesters

Description:

Different standalone java tools need to gather statistics of their usage (for example that given feature was used, that user clicked on something, saved some file, etc.). Eclipse Usage Data Collector (UDC) provides such functionality but has no official API and it is tightly connected with Eclipse RCP, so the more general Java framework for usage statistic collection is needed.

Functional Requirements:

- Different tools will gather different statistics - so the format of statistics is tool dependent.
- The tools should be able to easily send the statistics every hour/day etc...
- There should be implemented a server for storing these statistics - in DB.
- This server should be accessible for tools using RESTful approach (PUT "statistics")
- This server should also have web view for browsing the statistics (login/password required) - the statistics should be presented in tool dependent format.
- There should be implemented a library to be used by the tools to send statistics - it should hide the details of communication (REST, XML format of message?) and allow given tool to use some OO API to create statistics.

Because every tool has different format of statistics, so it can have a little different OO API on client side and different web view of gathered statistics. So some mechanism for defining statistics format for given tool is needed - the solution should allow to plug-in different formats.

The implementation should provide 1 plug-in with some exemplary statistics format.

Technical Requirements:

- MySql data base
- Hibernate for DB access
- Spring for web service implementation (REST access method)
- Spring MVC for web application
- Spring based client application

Development Requirements

- Tool should be distributed as Open Source
- Continuous Integration (Hudson)
- Source Versioning System

4. Java Grid Form Factory

Scope: API for creating GUI forms / editors in Java using Swing & SWT

Duration: 2-4 people / 1 semester

Description:

The purpose of the project is to create a common Java API for grid forms creation for both Swing & SWT. The API interfaces should represent definitions of forms elements like Text or Combo controls. Example:

```

interface IText extends IControl {
    void setText(String text);
    String getText();
    void setModifyListener(IModifyListener listener);
}

interface ICombo extends IControl {
    void setSelection(Integer index);
    Integer getSelection();
    void setSelectionListener(ISelectionListener listener);
    void setListValuesProvider(ICollectionProvider provider);
}

```

All controls should be placed into the grid layout of given form section. The sections, layouts etc. should also be represented by API Interfaces e.g. ISection, ILayout, IRow, IColumn, ICell. Each cell would contain exactly one control (IControl, however it should be possible to span certain columns or rows).

Using those interfaces there should be a possibility to create form definition (like place all controls into the given layout) which can be further used to create the real form instance in Swing or SWT. Example:

```

IFormFactory factory = new FormFactory();

//First create form, section and layout
IForm form = factory.createForm();
ISection section = factory.createSection();
ILayout layout = factory.createLayout(2, 2);

//Setup section and add it to form
section.setHeader("My New Section");
section.setLayout(layout);
form.addSection(section);

//Fill section layout with controls
ILabel label1 = factory.createLabel();
label1.setText("My Custom Control 1: ");
layout.getCell(0,0).setControl(label1);

IText text1 = factory.createText();
text1.setText("Custom Text1");
text1.setModifyListener(new MyModifyListener());
layout.getCell(0,1).setControl(text1);

ILabel label2 = factory.createLabel();
label2.setText("My Custom Control 2: ");
layout.getCell(1,0).setControl(label2);

ICombo combo2 = factory.createCombo();
combo2.setListValuesProvider(new MyListValuesProvider());
combo2.setSelection(1);
layout.getCell(1,1).setControl(combo2);

//Now having logical definition of the form we can build real GUI form
//for any of supported GUI systems

if (useSWT) {
    SWTFormBuilder.getInstance().build((Composite)parent, form);
} else if (useSwing) {
    SwingFormBuilder.getInstance().build((JComponent)parent, form);
}

```

Requirements

- Should support at least Swing and SWT form builders
 - Should be released as Open Source Project
 - Source Versioning (SVN) & Continuous Integration (Hudson) should be used
-

5. Hardware Viruses

Duration: 1 semester / 1-2 people

Description:

This project investigates feasibility of malicious digital designs (viruses) which could be incorporated into genuine designs. The project outcome should include: possible implementations, digital design obfuscation techniques, security risk analysis etc.

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Xilinx FPGA, VHDL, C, Python
 - Required knowledge: experience in digital system design, knowledge of the FPGA design flow and analog issues in digital design would be an advantage.
-

6. Hardware Viruses Design Analyzer

Scope: Design analyzer for hardware viruses detection

Duration: 1 semester / 2-4 people

Description:

This project implements the tool for detecting malicious digital circuits within medium/large VLSI designs, based on design netlist analysis. The project outcome should include: implementation of the design netlist (EDIF) parser and number of design pattern detection algorithms supporting detection of digital circuits.

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Python, knowledge of the EDIF(netlist) syntax
 - required knowledge: experience in digital circuits design, basic knowledge of the FPGA design flow,
-

7. Radio-on-Chip

Duration: 1-2 semester / 1-2 people

Description:

This proof-of-concept project implements pair of radio transceivers which communicate inside the VLSI chip (e.g. FPGA) using only simple radio interface.

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Xilinx FPGA, VHDL
 - required knowledge: experience in RF and digital circuits design
-

8. TC Transceiver

Scope: Transceiver for temperature-channel communication

Duration: 1 semester / 1-4 people

Description:

This proof-of-concept project implements pair of transceivers which communicate inside the VLSI chip (e.g. FPGA) using only chip temperature as the communication channel.

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Xilinx FPGA, VHDL
 - required knowledge: experience in digital circuits design, knowledge of the FPGA architecture and design flow
-

9. CLB Transceiver

Scope: Transceiver for clock-line-based communication

Duration: 1-2 semester / 1-2 people

Description:

This project investigates the feasibility of inter-module communication using the global clock-tree inside a VLSI chip (e.g. FPGA)

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Xilinx FPGA, VHDL
 - required knowledge: experience in digital circuits design, knowledge of the FPGA architecture and design flow,
-

10. Dynamic Logic Analyzer

Scope: Dynamically inserted logic analyzer for reconfigurable systems

Duration: 1-3 semesters / 2-6 people

Description:

This project implements simple logic analyzer which can be dynamically inserted into the active FPGA system (using FPGA Partial Reconfiguration). The outcome of this project should include: design and implementation of the simple logic analyzer and tool for inserting the logic analyzer into the deployed design (netlist or FPGA bitstream)

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: Xilinx FPGA, VHDL, C, Python,
 - required knowledge: experience with FPGA design flow
-

11. ERDB

Scope: ERDB - Embedded Routing Database

Duration: 1 semester / 1-2 people

Description:

This project provides the embedded routing database (and API) which describes architecture of the reconfigurable hardware (FPGA). The outcome of this project should be ERDB source C files which can be included in the embedded systems (e.g. SoC on the FPGA) firmware

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: C, Python, knowledge of the FPGA architecture
 - required knowledge: experience in embedded software development, database design and optimization
-

12. Hardware Performance Tuner

Scope: A performance-tuner for reconfigurable hardware

Duration: 1-3 semester / 1-6 people

Description: This project provides the tool for FPGA device performance characterization, e.g. measurement of the internal delays inside the FPGA fabric. The resulting tool should enable improvement of the FPGA design beyond the vendor's device specification (max frequency)

Restrictions (e.g. suggested technology, required interfaces, etc.):

- suggested technology: FPGA, VHDL, C, Python
 - required knowledge: experience in digital systems design, knowledge of the FPGA architecture and design flow, experience in embedded software development
-

13. Evaluation of GIS sources for use in desktop application

Scope: GIS (geographic information system) / C++ / Qt

Duration: 1 semester / 2-3 people

Description / Requirements:

- Evaluation of available GIS sources, in the context of using them in in-house desktop applications (open-source systems are preferred, e.g. OpenStreetMaps)
 - Once appropriate candidate is found, a solution should be provided presenting possibility to use the system in desktop application.
 - The aim is to present a demo Qt application that connects to appropriate GIS source and is able to locate given long&lat.
 - An easy to use API should be proposed (for C++), that would make it possible to use the solution in other desktop applications as well.
-

14. Large XML viewer

Scope: GUI Programming / XML

Duration: 1 semester / 1-2 people

Description / Requirements:

- The aim is to develop an application being able to parse large (>5MB) XML files and present their content in structured, hierarchical way.
 - Key functionalities are: folding/unfolding elements in document structure, fast document searching for keywords, content editing.
-

15. Programming Contest

Scope: Website based and command line tools for running a programming contest

Duration: To be defined based on functionality range

Description (in Polish):

Do zaprojektowania oraz implementacji jest aplikacja wspomagająca przeprowadzenie zawodów w programowaniu. Zadanie składa się z dwóch części. Pierwszej - programu służącego jako pomoc przy kompilacji, ocenie oraz sprawdzaniu poprawności kodu źródłowego zadań (kod źródłowy jest dostarczany na jego wejście, a program zwraca informację czy kod się kompiluje/spełnia wymogi zdefiniowanych testów itp.). Drugiej - interfejsu pozwalającego na przeprowadzenie zawodów.

Wymagania w stosunku do pierwszej części programu:

- a) Działa z linii komend.
- b) Ma możliwość skonfigurowania zawodów w programowaniu (drużyn, zawodników, kapitanów, zadań, testów).
- c) Umożliwia ustalenie liczby zadań do rozwiązania.
- d) Liczba testów do każdego zadania jest konfigurowalna (wejście i odpowiednie wyjście dla każdego przypadku testowego). Liczba testów może być specyficzna dla każdego zadania.
- e) Umożliwia dodawanie/edytowanie/usuwanie treści zadań.
- f) Daje możliwość ustawienia dowolnego kompilatora per zadanie - i co za tym idzie wybór języka programowania dla każdego z zadań (zbiór brany pod uwagę: C, C++, Java).
- g) Umożliwia zdefiniowanie wymogów wydajnościowych/ogólnych dotyczących rozwiązania:

- odnośnie limitu czasu wykonania programu (np <1s).
- odnośnie dostępnej pamięci dla wykonania programu (np. 1MB).
- odnośnie wielkości kodu źródłowego (np. 500 znaków).
- użycie niedozwolonych słów kluczowych (definiowalny zbiór), zewnętrznych bibliotek.

h) Po otrzymaniu zgłoszenia rozwiązania zadania, program sprawdza wymogi stawiane dla kodu źródłowego w regulaminie plus ograniczenia wynikające z treści zadania, przeprowadza kompilację i zachowuje dokładne wyniki.

i) Program akceptuje tylko rozwiązanie zadania, które jest zawarte w jednym pliku źródłowym.

j) Program musi umożliwiać ustawienie górnego limitu liczby zgłaszanych rozwiązań per zadanie.

k) Program sumuje punkty zdobyte przez zadania i potrafi wyłonić zwycięzcę.

l) Podczas oceny zgłoszonego zadania program bierze pod uwagę poprawność odpowiedzi i szybkość działania (oraz spełnianie ograniczeń) - nie jest brany pod uwagę sposób uzyskania poprawnej odpowiedzi.

m) Aplikacja powinna umożliwiać logowanie zdarzeń zchodzących podczas zawodów

Wymagania wobec interfejsu:

n) Jest interfejsem graficznym akcji udostępnionych przez core aplikacji (wymagania powyżej).

o) Pozwala na zarządzanie zawodami (zawodnikami, zadaniami, testami, rozwiązaniami, parametrami zawodów).

p) Generuje raporty zawodów (przebieg zawodów, dyplomy, statystyka zgłoszeń każdej drużyny, statystyki zgłoszeń każdego zadania, itp...).

q) Umożliwia dostęp zarówno dla osób administrujących zawodami, jak i dla uczestników (z różnymi zestawami uprawnień).

r) Musi być niezależny od systemu operacyjnego.

Całość wymagań dostępna będzie po przeprowadzeniu wywiadów z osobami, które docelowo będą korzystały z narzędzia. Ich zebranie, dyskusja oraz zaproponowanie projektu są także elementem realizacji zadania.

16. Department equipment inventory

Scope: Tool for managing all material resources of department, its placement and ownership

Duration: To be defined based on functionality range

Description (in Polish):

Aplikacja ma za zadanie wspomóc administrowanie wszelkimi zasobami, jakie są niezbędne do poprawnego funkcjonowania firmy/działu. W szczególności:

- stanowiskami pracy (biurko, krzesło, szafki itp.)
- sprzętem komputerowym (monitor, komputer, telefon, stacja dokująca itp.)
- sprzętem multimedialnym (telewizory, projektory, switche)
- akcesoriami (przewody, linki zabezpieczające)

Ponadto powinna umożliwiać:

- eksport/import (także przyrostowy) zasobów z poszczególnych kategorii
 - definiowanie przypomnień o konieczności zamówienia poszczególnych obiektów
 - zdefiniowanie przynależności zasobów do pracowników
 - generowanie raportów (konfigurowalny eksport)
-

17. Scrum Planner

Scope: Tool for managing teams and work in SCRUM

Duration: To be defined based on functionality range

Description (in Polish):

Aplikacja powinna pomóc zarządzać zespołami ludzi (zgodnie z opisem ich ról wg. metodyki SCRUM a także z uwzględnieniem planowania sprintów, kontyngentów, itp.), planować realizację funkcjonalności oraz pracę nad naprawą dotychczas zgłoszonych błędów. Istotnym elementem programu powinno być sprawdzanie ograniczeń np.: aplikacja powinna zwrócić uwagę, że przypisanie do poprawienia błędu w komponencie A, osoby, która ma wiedzę tylko i wyłącznie na temat komponentu B i E może skutkować wydłużeniem czasu realizacji zadania. Dodatkowo powinny być sprawdzane ograniczenia wynikające z 'mocy przerobowych zespołu' i estymowanego czasu trwania realizacji funkcjonalności.

Ponadto, jedną z głównych części aplikacji będzie moduł odpowiedzialny za raporty. Ich generacja powinna być w pełni konfigurowalna. Musi bazować zarówno na danych aplikacji jak i na zewnętrznych źródłach (głównie pliki wyeksportowane z innych aplikacji, których możliwość importu jest niezbędna).

Całość wymagań dostępna będzie po przeprowadzeniu wywiadów z osobami, które docelowo będą korzystały z narzędzia. Ich zebranie, dyskusja oraz zaproponowanie projektu są także elementem realizacji zadania.

18. Wireshark plug-in for LTE messages

Scope: Wireshark plug-in which enables tracing inter-component messages

Duration: To be defined based on functionality range

Description / Requirements: Detailed specification will be provided soon...

19. Robot Framework K3 plug-in

Scope: Robot Framework plug-in for acceptance testing using K3 TTCN-3 compiler

Duration: To be defined based on functionality range

Description / Requirements: Detailed specification will be provided soon...

20. LTE Performance Counters Manager

Scope: Application should ease the process of exporting performance counters from DB to C/C++ code

Duration: To be defined based on functionality range

Description / Requirements: Detailed specification will be provided soon...

21. License Manager

Scope: Small application for license generation and management with DB included

Duration: 1 semester / 2-6 people

Description:

Creating licensing database along with an application to manage existing and generate new licenses.

Requirements:

- system should have restricted access for few operators
 - chosen DBMS should be free of charge
 - GUI should be ease and intuitive
 - there should be automatic notification about expiring licenses
 - there should be well defined SW interfaces which allow to plug-in custom ciphering method
-

22. License Reporter

Scope: Reporting tool for License Manager system

Duration: 1 semester / 2-4 people

Description:

Creating reporting application which using data stored in licensing DB (from License Manager system) will automatically generate following reports:

- active licenses per tool (per entity)
 - recently expired licenses (and not renewed)
 - new users
 - etc.
-

23. Eclipse remote C/C++ debugging

Scope: Eclipse plug-in for remote C/C++ code debugging

Duration: 1 semester / 2-4 people

Description:

Eclipse plug-in (CDT, rconsole etc.) enhancement to enable compilation and debugging of C/C++ code on remote Linux machines.

Requirements:

Local machine:

- os: Windows or Linux
- software: Eclipse, ssh client

Remote machine:

- os: Linux
- software: gcc (g++), gdb, ssh server (smb or nfs server if needed)

Remote development (coding) is already available based on eclipse plug-ins and ssh protocol.