Course Syllabus

Project course: Development studio, Spring 2011, Block 3 and 4 – DRAFT

Instructors: Klaus Marius Hansen (KMH), Sune Lomholt (SL), Erik Frøkjær (EF)

January 7, 2011

Responsible	Klaus Marius Hansen
Office	DIKU-SC, Njalsgade 128, 24-5-50
Mobile phone	6371 2721
Email	`klausmh@diku.dk`
Webpage	`http://www.diku.dk/~klausmh`
Course web	See Absalon
Lecture hours	Mondays 13:00 - 15:00, location TBD
Exercise hours	Wednesdays 9:00 - 16:00, location TBD

Description

Development studio is a place where software development is taught and where developers practise. In this course the focus is on software development in a broad sense covering all aspects of modern development processes. This includes requirements development, system and software design, software construction, testing, and integration of the system solution in its use context.

Preceding Courses / Prerequisites

Good programming skills (comparable to a computer science bachelor).

Organization

The course activities are anchored around project work which expands over two quarters. In the projects the students are supposed to solve demanding real-life problems whose specification and solution requires active end-user participation. There can be two types of project providers: 1) Partners from industry. 2) Researchers or research labs from our university. Project providers, who wish to offer development tasks, should get their project idea approved in advance by the course team. Moreover, project providers must be willing to participate actively in the development process as domain experts and reviewers.

The groups choose their own project guided by the instructors with the constraints that it must i) develop a software product, ii) be implemented to run on mobile devices (specifically a Google Android device), and iii) be of sufficient scope. Furthermore, the groups are required to follow an agile development methodology (specifically the Scrum methodology) and focus on incremental delivery of quality software.

It is strongly recommend that you start thinking about projects as soon as possible.

Lectures and exercises are designed to support the development project and the course plan will thus be adjusted during the course depending on what projects require. A part of Scrum is regular, short status meetings; these will form the base of exercise classes in the course.

Tentative course topics

The students will be divided into teams of 3-4 people that will work together on a software development project. The course contains material both on the development of useful software and of reflection on the development of such software. The course topics include:

- requirements analysis
- software design and software architecture
- software development methods and processes (including agile software development)
- user involvement
- testing (including test-driven development)
- review and inspection
- documentation and reverse engineering
- software quality
- process improvement

The concrete set of topics is dependent on the needs of the software development projects undertaken by the groups.

Course material

There is no required book for the course. Notes and papers will be made available on Absalon.

Students are expected to find material on Android development themselves; however, if you need additional material, it is suggested that you buy [1]. Android equipment (HTC Desire phones) will be made available for students.

Grading

In order to be allowed to take the final exam, a number of obligatory exercises must be handed in and approved. These mostly consist of *project deliverables*. Project deliverables are handed in as group assignments in the form of a short report and program code (for Scrum deliverables).

The *final exam* is an individual, written take-home exam based on your project (due 2011-06-22). Grading is done on the 7-point grading scale, with 02 needed to pass.

Draft Course Plan

Teaching starts Monday 2011-01-31 and ends Wednesday 2011-06-15.

In general, Monday will be used for lectures and Wednesdays will be used for exercise classes including weekly Scrum meetings and possibly including practical workshops. Groups are intended to also use Wednesdays for joint work.

Deliverables have to be handed in on Fridays in the week they are scheduled for.

Week	**Monday**	**Wednesday**	**Deliverable**
05	Course introduction. Project management. Development processes [14, 13] (SL, KMH)	Android introduction and tutorial. Group formation (KMH)	
06	Scrum [11] (SL)	Android architecture and development (KMH)	1. Android exercise
07	Requirements [9, 15] (SL)	Version control. Product and process brief presentation (KMH)	2. Sprint #0 (Product and process brief)
08	User involvement [5] (EF)	Test-Driven Development. Scrum meeting (KMH)	
09	*(Not scheduled)*	*(Not scheduled)*	
10		Scrum demo. Scrum planning (KMH)	3. Sprint #1 (Scrum)
11	Software design and patterns [6] (KMH)	Build management. Scrum meeting (KMH)	
12	Software architecture [4] (KMH)	Guest lecture on Android in practive (Silverbullet). Scrum meeting (KMH)	
13		Scrum demo. Scrum planning (KMH)	4. Sprint #2 (Design)
14	*(Protected week)*	*(Protected week)*	
15	Documentation. Reverse engineering [7] (KMH)	Scrum meeting (KMH)	
16	*(Easter)*	*(Easter)*	
17	*(Easter)*	Scrum demo. Scrum planning (KMH)	5. Sprint #3 (Documentation)
18	Inspection and review [8]. Guest lecture on operation (KMH)	Scrum meeting (KMH)	
19	Quality and metrics [3, 2] (KMH)	Scrum meeting. Inspection workshop (KMH)	
20		Scrum demo. Scrum planning (KMH)	6. Sprint #4 (Quality)
21	Process improvement [12] (SL)	Scrum meeting (KMH)	
22	Reflective systems development [10] (KMH)	Scrum meeting (KMH)	
23		Scrum demo (KMH)	7. Sprint #5 (Process)
24	*(Whitsun)*	On exam (KMH)	
25		*Final exam hand-in*	

References

[1] W. Frank Ableson, Charlie Collins, and Robi Sen. *Unlocking Android. A Developer's Guide*. Manning, second edition, 2010.

[2] L. Bass, P. Clements, and R. Kazman. Achieving qualities. In *Software Architecture in Practice*, chapter 5, pages 99–128. Addison-Wesley, second edition, 2003.

[3] L. Bass, P. Clements, and R. Kazman. Understanding quality attributes. In *Software Architecture in Practice*, chapter 4, pages 71–98. Addison-Wesley, second edition, 2003.

[4] L. Bass, P. Clements, and R. Kazman. What is software architecture? In *Software Architecture in Practice*, chapter 2, pages 19–46. Addison-Wesley, second edition, 2003.

[5] K. Bødker, F. Kensing, and J. Simonsen. Tools and techniques. In *Participatory IT design: designing for business and workplace realities*, chapter 9. The MIT Press, 2004.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: Elements of reusable object-oriented software*, pages 107–116, 151–161, 207–217, 257–271, 315–323. Addison Wesley, 1995.

[7] P. Grubb and A.A. Takang. *Software maintenance: concepts and practice*, chapter 7, 11.5. World Scientific Pub Co Inc, 2003.

[8] IEEE. IEEE Standard for Software Reviews. Technical Report IEEE Std 1028-1997, IEEE Computer Society, 1997.

[9] Dean Leffingwell. A user story primer. In *Agile Software Requirements*, chapter 6. Addison Wesley, 2010.

[10] Lars Mathiassen. Reflective systems development, 1998. Dr. Techn. Dissertation. Aalborg University.

[11] Ken Schwaber and Jeff Sutherland. Scrum guide. Scrum.org, 2009.

[12] Ian Sommerville. Process improvement. In *Software Engineering*, chapter 28. Addison-Wesley, 8th edition, 2007.

[13] Ian Sommerville. Project management. In *Software Engineering*, chapter 5. Addison-Wesley, 8th edition, 2007.

[14] Ian Sommerville. Software processes. In *Software Engineering*, chapter 4. Addison-Wesley, 8th edition, 2007.

[15] Karl E. Wiegers. The essential software requirement. In *Software Requirements*, chapter 1. Microsoft Press, second edition, 2003.