<table>
<thead>
<tr>
<th>TITLE</th>
<th>Compendium: Introduction to course TDT4290 Customer Driven Project, Autumn 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYER</td>
<td>IDI, NTNU</td>
</tr>
<tr>
<td>COURSE</td>
<td>TDT4290 Customer Driven Project</td>
</tr>
<tr>
<td>LAST UPDATE</td>
<td>Tuesday, 26 August 2014</td>
</tr>
<tr>
<td>PAGES</td>
<td>(<a href="http://www.idi.ntnu.no/emner/tdt4290/docs/TDT4290-compendium-2013.pdf">http://www.idi.ntnu.no/emner/tdt4290/docs/TDT4290-compendium-2013.pdf</a>)</td>
</tr>
</tbody>
</table>
Table of Contents

Introduction ........................................................................................................... 5
General information .................................................................................................. 5
Goal and rationale of the course .................................................................................. 8
Required Knowledge .................................................................................................. 9
Motivation on Project Work and Group Dynamics .................................................. 10
About the Course ...................................................................................................... 10
Project work in a didactic perspective ........................................................................ 10
Training in group dynamics ....................................................................................... 11
Administrative information ....................................................................................... 12
Work load .................................................................................................................. 12
Timeline .................................................................................................................... 12
Group assignments ..................................................................................................... 12
Rating of project work ............................................................................................... 13
Supervision and meetings ......................................................................................... 14
Pre-delivery for examiner ......................................................................................... 15
Final presentation and demonstration on November 20 ........................................... 15
Anti-plagiarism .......................................................................................................... 16
Copyright or Intellectual Property Rights (IPR) ....................................................... 16
Course reflection, evaluation and feedback .............................................................. 17
Appendix A – The project plan .................................................................................. 18
A1. Overall project plan ............................................................................................. 18
A2. Concrete project work plan .................................................................................. 19
A3. Project organization ............................................................................................. 20
A4. Templates and standards ...................................................................................... 20
A5. Version control procedures .................................................................................. 20
A6. Documentation of project work .......................................................................... 20
A7. Quality Assurance (QA) ...................................................................................... 21
A8. Test plan ............................................................................................................. 22
Appendix B – Suggestion for appendices in your project plan ........................................ 23
B1. Partners ............................................................................................................. 23
B2. Concrete project plan ............................................................................................ 23
B3. Detailed phase plans ............................................................................................ 23
B4. Table for handling of risks ................................................................................... 23
B5. Table for effort registration .................................................................................. 24
Appendix C – Content of the phase/sprint documents/chapters ................................. 26
C1. Introduction ......................................................................................................... 26
C2. Planning .............................................................................................................. 26
C3. Pre-study of the problem space vs. solution space ............................................ 26
C4. Lifecycle model: waterfall vs. agile? ................................................................. 27
C5. Requirements Specifications ............................................................................... 27
C6. Estimation of realization effort of a use-case model .......................................... 28
C8. Programming ..................................................................................................... 29
C9. Testing ............................................................................................................... 30
C10. Internal and external documentation .............................................................. 31
C11. Evaluation ........................................................................................................ 32
C12. Project presentation and demonstration ......................................................... 32
C13. Appendices ....................................................................................................... 32
APPENDIX D – Administrative and Technical Resources ........................................ 34
D4.1 Office Resources ............................................................................................. 34
D4.2 Technical Resources ......................................................................................... 34
# Table of contents

1 Introduction  
  1.1 General information  
  1.2 Goal and rationale of the course  
  1.3 Required Knowledge  

2 Motivation on Project Work and Group Dynamics  
  2.1 About the Course  
  2.2 Project work in a didactic perspective  
  2.3 Training in group dynamics  

3 Administrative information  
  3.1 Work load  
  3.2 Timeline  
  3.3 Group assignments  
  3.4 Rating of project work  
  3.5 Supervision and meetings  
  3.6 Pre-delivery for examiner and writing course  
  3.7 Final presentation and demonstration on November 21  
  3.8 Anti-plagiarism  
  3.9 Copyright or Intellectual Property Rights (IPR)  
  3.10 Course reflection, evaluation and feedback  

Appendix A – The project plan  
  A1. Overall project plan  
  A2. Concrete project work plan  
  A3. Project organization  
  A4. Templates and standards  
  A5. Version control procedures  
  A6. Documentation of project work  
  A7. Quality Assurance (QA)  
  A8. Test plan  

Appendix B – Suggestion for appendices in your project plan  
  B1. Partners  
  B2. Concrete project plan  
  B3. Detailed phase plans  
  B4. Table for handling of risks  
  B5. Table for effort registration  

Appendix C – Content of the phase/sprint documents/chapters  
  C1. Introduction  
  C2. Planning  
  C3. Pre-study of the problem space vs. solution space  
  C4. Lifecycle model: waterfall vs. agile?  
  C5. Requirements Specifications  
  C6. Estimation of realization effort of a use-case model  
  C8. Programming  
  C9. Testing  
  C10. Internal and external documentation
4 **APPENDIX D – Administrative and Technical Resources**

D4.1 Office Resources
D4.2 Technical Resources
   D4.2.1. Workstations
   D4.2.2. Source Control Management
   D4.2.4. Use of collaboration technology in the project

5 **Appendix E – SCRUM – a popular agile method**

E5.1. Product backlog
E5.2. Sprint planning meeting
   E5.2.1. Daily Scrum status meeting
   E5.2.2. Irregularities
   E5.2.3. Sprint review meeting
   E5.2.4. How do I prepare a project for Scrum (short tutorial)

6 **Appendix F – Use-case based effort estimation**

F6.1. Introduction to use-case estimation
F6.2. More on the estimation method
F6.3. A mini-discussion
F6.4. References
F6.5. Appendix: A small use-case diagram, with extra comments
F6.6. Add-on to use-case example

7 **Appendix G – Project Descriptions**

   7.1 Schlumberger
   7.2 Autronica Fire and Security
   7.3 Gridmedia Technologies
   7.4 Work Research Institute
   7.5 Adresseavisen
   7.6 Sintef Fisheries and Aquaculture
   7.7 Netlight
   7.8 FourC
   7.9 FORCE Technology Norway
   7.10 Altran Norge
   7.11 Trondheim kommuneAcando AS

8 **Appendix H – Student lists and groups**

9 **Appendix I - Kick-off rooms for groups**

10 **Appendix J – Guest Lectures**
Introduction

General information

This master-level course TDT4290 Customer Driven Project deals with a project assignment that is mandatory for all computer science (“Datateknikk”) students in their 4th study year at NTNU/IDI, typically with 50-75 and mostly Norwegian students. In addition comes participants from the two-year, international master program in Information Systems at IDI (with 10-15 students from all over the world), plus Erasmus and other guest students (usually a handful of Europeans).

This compendium contains all the necessary information for this course, the assignments (one for each project-group), and a suggested outline for the final project report. In addition, some examples of what a project plan should contain.

Practical information regarding project-group composition, dates etc. can also be found on the web page of the course (http://www.idi.ntnu.no/emner/tdt4290/). All students should check this web page for updates. In case of mismatch between information in this compendium, information given during lectures, by email, and on the above web page, the last updated information should be regarded as correct. The following IDI people are involved with this course:

**Course responsible:**
Prof. Jon Atle Gulla  
Course responsible  
jag<##>idi.ntnu.no  
tel 735.91847

Özlem Özgöbek  
Practical coordinator  
ozlemo<##>idi.ntnu.no  
tel 735.94427

**Group advisors:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Picture</th>
<th>Email contact</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prof. Jon Atle Gulla</td>
<td><img src="image.png" alt="Picture" /></td>
<td>jag&lt;##&gt;idi.ntnu.no</td>
<td>735.91847</td>
</tr>
<tr>
<td>PhD-stud. Alfredo Perez Fernandez</td>
<td><img src="image.png" alt="Picture" /></td>
<td>perezfer&lt;##&gt;idi.ntnu.no</td>
<td>944.70498</td>
</tr>
</tbody>
</table>
Students will be divided in project groups of 5-6 students (detail in Section 3.3). Each group will be allocated an advisor from IDI - either a faculty member, a postdoc, or a PhD student - plus a main customer representative.

**The mandate of the advisor:**
The group’s advisor serves as a one-person “steering committee” for your project. His/her responsibility is to keep an eye on the main process of the work, and to oversee that sufficient contact with the customer is maintained. The advisor must therefore regularly receive updated status reports, copies of relevant work plans and technical documents from the group, so that all this can be discussed in a weekly **advisor meeting**.

In addition, the group should have several, weekly **internal meetings**, and regular **customer meetings** (detail in Section 3.5),
Goal and rationale of the course

The goal of course TDT4290 is to teach you and your fellow students – by working in groups – software engineering (SE) skills in the context of a development project to make a realistic prototype of an Information System (IS) “on contract” for a real-world customer.

Each project group is initially given a one-page project assignment from an external customer. All the phases of a typical IS/IT project are covered, e.g. project management and planning, pre-study, requirements, design, programming, testing, evaluation and documentation, but no “maintenance”. However, we do not accept customers that just want the group to write a “summary and evaluation” of some hot topic, with no ensuing implementation. And inversely, we do not want customers that come with pre-made requirements, and just want the group to complete a pre-designed system architecture.

Due to resource constraints, the focus should nevertheless be on the “early” lifecycle phases, i.e., project planning, pre-study, requirements specifications and system design – but the exact focus and emphasis will be decided by the group, in dialog with their customer. For instance, groups with focus on the early phases should not omit making a working prototype of some system parts. On the other hand, “programming-eager” groups may try to make a rather complete prototype, e.g. by applying agile iterations in Scrum-style (see C.4 and Appendix F). The important issue is that the group clearly justify their decisions, and that there is a logical flow in the project report from start to end of all the phases, and that all the phases and iterations build naturally on each other.

Each group should write a project report in English, and hold a presentation and demonstration of the final prototyped product for the customer, while an external examiner (censor) is present.

If a fundamental disagreement with the customer arises, the group has, if needed, “the final word” since the group members gets the credit through a final exam (report) worth 15 Sp. But such a deadlock situation has hardly happened in the 40 years that this course has been arranged! However, the group and their advisor should do what is possible to resolve any major disagreements. Conflicts are to be explained, negotiated and resolved (managed), as this is part of the real world work. It is therefore crucial that the group is focused and has a good dialog with the customer.
Required Knowledge

Required theories and methods for making large and long-lived SE/IS systems are mostly covered in previous, bachelor-level courses. This knowledge base is supplemented by a double seminar on group dynamics, and four guest lectures on project management, Scrum, IT architecture, and use-case based effort estimation. In addition, we arrange a course in presentation techniques and a seminar on technical writing.

Since 2008 in this course, the students have been encouraged to use the Agile software development method so-called “Scrum”. Given the time constraints of this student project, there is hardly time for more than 2-3 increments, called “sprints” in Scrum. Many groups have since 2008 chosen an agile development model and had good results. As mentioned above, the structure of the final project report must reflect the choice of overall lifecycle model, i.e. waterfall vs. iterative/agile development methods.
Motivation on Project Work and Group Dynamics
About the Course

The goal of this course is to teach fundamental software engineering skills through realistic training in software development and project management. You will have the opportunity to apply the knowledge you have gained previously in your studies.

During this course, you will experience situations that will require:

- Decision, solving and design and development of a relatively large and complex system.
- Creative and collaborative problem solving. Earlier in your studies, the tasks have been smaller and more well-defined. In this project, there are (conflicting) decisions to be made. You will have to show creativity, be pragmatic and be capable of solving fuzzy tasks under heavy time and resource constraints; i.e. fast decision making under great uncertainty.
- Coordination of efforts and distribution of work and responsibilities.
- Project management, cooperation, decisions, follow-ups, and a dispute resolution.
- Ability to adapt to no-ideal working situations.
- Planning and execution of plans. This involves creation of project plans and registration and monitoring of effort and resource usage.
- You must handle difficult customers. They can be unreliable and/or unavailable. They might change directions, come up with new ideas, and have an unclear picture of what they really want. An important part of this course is to manage the group project, so that the results match the customer’s needs, even though the situation may turn difficult. This requires routines for quality control. Each group should deliver a project plan with resources and milestones, coupled to quantitative measures, and use both “dry” verification (mainly reviews) and “wet” testing.
- Structuring of requirements specifications.
- Documentation. The project documents must be complete, well structured and target the technical knowledge level of the customer.
- Defend decisions that are taken on behalf of the customer. You should document all delays, overruns, and weaknesses, so that they can be explained and argumented. Ideally, all decisions should match conditions coming from the customer (the customer has the right to complain on any aberration that is not his/her fault).
- To present (and sell) the final product for the customer / external examiner. Under the final presentation and demonstration, it is important to give the customer a complete and good impression of the system delivered.

Project work in a didactic perspective
Several evaluations have been carried out of previous versions of TDT4290 (i.e. ”Systemering prosjektarbeid” and “Programmering prosjektarbeid”). These evaluations are generally very positive. See Markus Sorge: ”Evaluering av prosjektundervisningen ved IDI, NTNU”, Program for lærerutdanning, NTNU, spring 2000, 63p, http://www.idi.ntnu.no/undervisning/siving/docs/prosjektevaluering.pdf.

Technology is experience-based knowledge – composed and refined over many years – to be able to satisfy human-societal needs in a cost-effective way. Engineering is the process of combining and applying suitable technologies to construct specific means, such as houses, food, clothing, roads, bridges, vehicles, books, sewing machines – and recently – computer- and information systems. That is, an engineer creates new reality (e.g., kitchen tables) – not only studies the existing one (e.g., a humming bee in a forest).

Engineering requires a domain-specific methodology (a technology itself) for how to describe the actual context – being farming, bridge-building, or banking. An engineer will apply scientific insight
(both technical and social), combined with knowledge and experience from many sources – all representing different technologies. It is often a strong relationship between what is being constructed and the available time, budget, and tools / methods. Because of the substantial complexity and diversity of the engineering work and the characteristics of the processes, it is often necessary for several people to work together. That means that engineering has a social dimension, since it is executed as group work. Cooperative and communicative skills are therefore essential.

Project work in teams is an important part of the engineering discipline. Your study program at NTNU is among the ones with most emphasis on project work. Project work means on the one hand that you need to make an agreement with a customer (customer / organization) about what should be constructed. On the other hand, you have to design and implement technical solutions that satisfy the elicited constraints and requirements. You also have to consider changes over time, as most customers are not sure about what they really want. As a consequence, the proposed unfinished solution must be modified. The project groups must also be well organized and effective, and try to avoid destructive internal conflicts.

All this means that you will get a hectic work situation – sometimes at the edge of chaos. You have to combine your theoretical knowledge from previous courses to solve specific and practical problems. You have to use a considerable amount of effort in cooperation, communication, planning, and improvisation and show capabilities of working under pressure.

Your project will give you essential training to become a professional software engineer. Feedback from industry says, that it is almost impossible to get more done in 3 months than what such a group of students is capable of. Further, software engineers from NTNU are useful from day one: they possess the theoretical knowledge and know how to work efficiently in teams.

So, the expectations are great from all participants: the IDI department, lecturers, advisors, external customers and of course the students themselves. The project report (written in English) should not be more than 200 pages, exclusive appendices and graphics.

If your group experiences that some of the team members are not participating satisfactorily, you should immediately contact your advisor. If you experience other minor problems, the advisor is the one to contact. However, most (minor) problems are to live with; in fact, it is a part of the course to learn to deal with such issues in a project.

So, welcome to an interesting and hectic semester in this course!

**Training in group dynamics**

Good teamwork and group dynamics are essential for the success of any collaborative project. Therefore, ”social” skills are of upmost importance to become a successful project co-worker. A seminar on group dynamics is planned as a part of this course, to support the project groups to learn more about teamwork and group dynamics. In addition to the seminar, the following time slots should be used wisely to create a good team atmosphere among your group, particularly at the beginning of the project.
Administrative information

Work load
NTNU has officially an autumn semester with 19 weeks, and a spring semester with 21 weeks. Of the former 19 weeks, two are spent on continuation exams (and immatriculation etc.), and three are spent on exams in December – which you don't have. This implies 17 "study weeks" in the autumn semester, each of 40 person-hours (work-hours) per student. This again corresponds to 340 (17 * 25% * 40) person-hours per student for a "15-Sp" course (50% of 30-Sp semester total). Note that an hour has 60, not 45 minutes.

Since this project runs in 14 weeks (really 13.6) instead of 17 weeks, the weekly effort per student then be adjusted to 40 * 50% * 17/14 = 24.3 person-hours.

Furthermore, the official web page of the course – http://www.ntnu.edu/studies/courses/TDT4290 – specifies 24 weekly person-hours per student, but this is for 14 weeks. So let us round off to 25 person-hours per week and student, i.e., 350 person-hours per student for totally 14 weeks.

For a project group of 5-6 students, the available effort per group will lie between 1750 and 2100 person-hours, including own reading, meetings, lectures, and seminars. Earlier projects have shown that it is possible to deliver really good results within that timeframe.

It is important that everyone is honest and registers all effort (as person-hours) spent on the project. This means that the project documents must show the real work load. Effort overruns will result in less sparetime for you personally and less time for other courses. Inflated work effort does not affect the grades given in this course!

Timeline
This is the preliminary time plan for the activities of this course. For further details and updates check http://www.idi.ntnu.no/emner/tdt4290/.

Group assignments
We expect about 80 students in total. This gives in total 11 groups with 7-8 students per group. Each group is preallocated to one customer and one group advisor. The groups should therefore have a tight cooperation with their advisor.

Group assignments are essentially made randomly. This is done intentionally to create groups where the members generally do not know each other beforehand. This is a typical situation in a real life, especially when working as a consultant.

Since many of the students in the course are foreign, with limited or no knowledge to Norwegian language, the lectures and seminars will be held in English.
Table 1 – Overview of groups, customers and advisors

<table>
<thead>
<tr>
<th>Group</th>
<th>Customer</th>
<th>Title</th>
<th>Advisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Schlumberger Information Solutions</td>
<td>Spatial Data Indexing and Ranking Scheme</td>
<td>Özm Özgöbek/ Soudabeh Khodambashi</td>
</tr>
<tr>
<td>2</td>
<td>Autronica Fire and Security AS</td>
<td>Analyze and present event log for a fire detection system</td>
<td>Soudabeh Khodambashi</td>
</tr>
<tr>
<td>3</td>
<td>Gridmedia Technologies AS</td>
<td>Shattered</td>
<td>Gleb Sizov</td>
</tr>
<tr>
<td>4</td>
<td>Work Research Institute (AFI), Oslo and Akershus University College of Applied Sciences</td>
<td>Alternative Spaces – the digital prototype</td>
<td>Gleb Sizov</td>
</tr>
<tr>
<td>5</td>
<td>Adresseavisen AS</td>
<td>Generic HTML5 print-ad editor and pdf-generator</td>
<td>Gleb Sizov</td>
</tr>
<tr>
<td>6</td>
<td>SINTEF Fisheries and Aquaculture</td>
<td>Ocean Forecast</td>
<td>Gleb Sizov</td>
</tr>
<tr>
<td>7</td>
<td>Netlight AS</td>
<td>Bitcoin Coupon (BitCoupon)</td>
<td>Alfredo Perez Fernandez</td>
</tr>
<tr>
<td>8</td>
<td>FourC AS</td>
<td>Crowd tracking using wireless signals</td>
<td>Alfredo Perez Fernandez</td>
</tr>
<tr>
<td>9</td>
<td>FORCE Technology Norway AS</td>
<td>CMIS for Wind</td>
<td>Mohsen Anvaari</td>
</tr>
<tr>
<td>10</td>
<td>Altran Norge AS</td>
<td>Internet of Things</td>
<td>Mohsen Anvaari</td>
</tr>
<tr>
<td>11</td>
<td>Using Wearable devices in welfare services</td>
<td>Trondheim kommune</td>
<td>Jon Atle Gulla</td>
</tr>
</tbody>
</table>

Rating of project work

The project work will be evaluated based on the quality of the project report and presentation delivered at the end of the course and the students’ reflections on the project work:

- The project report and presentation will be 95% of the marks.
- The reflection report will contribute to 5% of the marks.

The project report must be written in English, and the presentation must be done in English. Both the project report and the presentation count towards the grade in an integrated way (they are not formally weighted against each other).

How the group actually has worked, technical problems, customer behaviour and availability, etc. are a part of the reflections report. The group is asked to deliver a 1-2 page report, reflecting upon their experiences during the project and reflecting upon what they had learned and how they could have
done things differently. The focus of this part is on the process rather than the product. This report is due at the end of the course and will be evaluated by the group advisor and the course coordinator.

The following criteria are evaluated in an integrated way:

- Whether the group has solved the given assignment, according to the customer’s objectives of the project.
- Reasonable grounds for decisions taken.
- Logical flow in the report.
- Visibility of limitations done.
- Layout and structure readability.
- The students’ ability to reflect on the process during the project.

The criteria are not formally weighted against each other.

Note that since the presentation counts towards the grade, it is important that you maintain a functioning version of your program in case you (the group) appeal the result (grade). If an appeal is made, you will have to make your presentation for the new examiner, including demonstration of the program.

Supervision and meetings

Your very first group-internal meeting is scheduled for the same day as the kick-off day. Each of you should introduce yourself to the others in the group, and try get the group organized for the first customer meeting the following hours. It is up to you!

Furthermore, your group should have a main advisor meeting with your advisor once a week, normally lasting one hour. Such meetings will have a group-specific content, but share a template agenda from Appendix A.7. All written documents for such meetings (agenda, weekly status report, phase-specific documents etc.) must be delivered on paper or by email to the advisors before 14:00 the day before. In Appendix A – Project plan, you will find more information about such meetings. Thus, during the first, pre-planned advisor meeting between your group and your advisor on Wednesday (see room/time in Appendix I), you will have to agree upon when and where the weekly advisor meetings shall take place for the rest of the semester. The group is responsible for booking a meeting room for these meetings (possibly helped by the advisor). During this hour, the advisor will also focus on the teamwork and group dynamics aspects and support you to establish a good group atmosphere.

How to book a room for a meeting:

You can book a room by contacting Ellen Solberg at the IDI information desk. It is recommended that you send an email to Ellen.Solberg<##>idi.ntnu.no or by phone 73 59 34 71.

It is wise to suggest 2-3 alternatives times for the meeting, because many of the rooms may already be booked. Always give Ellen Solberg a note if you not are using a reserved room.

A suggestion of an email:

Regarding the “TDT4290 - Customer Driven Project” we would like to book a room for X persons.
We would like one of the following times (in prioritised order)
1. date, from-to
2.
3.

Best regards,
It is also possible to book some rooms through the room reservation site “Romres” (https://romres.ntnu.no/). This reservation page is only accessible from users on the NTNU-intranet.

Customer meetings are held when needed, starting on the kick-off day (see rooms in Appendix I). The next customer meetings arranged in dialog with the customer, but the group is responsible for booking a room and other logistics. We recommend taking more contact with the customer, before the second advisor meeting in the following week.

You will probably also need several weekly, internal group meetings. So try to book a fixed room once or twice a week during the semester.

Note: Before the first advisor, the group is collectively responsible for making a written resume of the first customer meeting held on the kick-off day. This resume should be sent by email to the persons involved (group members, advisor, customer) later on the same day. So take good notes of this first customer meeting!

Pre-delivery for examiner

You are required to submit a copy of the Abstract, Introduction, the Pre-study and the Choice-of-Lifecycle-model chapters to the external examiner (censor). These chapters are normally not in their final stage and will not be used as part of the final assessment. It is only to let the external examiner be better prepared.

In addition to the chapters mentioned earlier, the groups are required to include the outline of the full report (Table of Contents). The Table of Contents should not be too detailed, but must contain enough detail to understand how each chapter is structured and what the final focus of each section should be.

This delivery should be electronic to the group’s supervisor by the end of October 17th 2014.

Remember that you are the “consultants” that invites to a presentation of your work. You should be proud of your product and give it the publicity it deserves. Create a flyer to catch the attention of the audience, and send this to the customer, the advisors, and others that might be interested - such as local TV/radio, Universitetsavisen, Adresseavisen etc. - well ahead of the presentation.

Final presentation and demonstration on November 20

The projects will be presented and demonstrated at NTNU on Thursday, November 20, between 09:15 and 14:00.

All groups have to make their presentations in English.

Room: If the project demonstration requires special facilities (such as virtual reality or cave equipment), the groups can also book and have the presentation in other rooms. If your group needs to have the presentation in a specific room, please notify the course coordinator (Özlem Ö zgöbek). Remember that the room must have space for 10-15 persons. The time and place for each presentation, will be published on the course webpage.

Laptop: Most groups use one of their personal laptops for the demonstration. If your group do not have a suitable laptop for the presentation, please notice drift<#di.idi.ntnu.no two weeks before the presentation. A beamer will be made available in all presentation rooms.
Copying project documents: We want four printed and bound copies of the project report. The costs for copying and binding four complete project reports are covered by IDI. Copying should be carried out, at the latest, one day before the final presentation (e.g. on Nov. 20 or before).

In addition, one electronic (PDF) copy, identical to the printed and bound version, should be e-mailed to ozlemo@idi.ntnu.no.

Delivery of final report: Four bounded paper copies of the final report (with implementations on attached CDs / DVDs) should be delivered on the same day as the presentation. The customer should get one of the copies, and the three others should be delivered at the IDI information desk. If the information desk is closed at the time your presentation is finished, contact the practical course coordinator. The four printed copies of the report will be distributed to:
- Customer
- Examiner
- IDI archive
- Advisor

In addition, the course coordinator should also receive a digital single-file, .pdf-format copy of the entire final report by as an e-mail attachment.

On the CDs / DVDs, you should include the final report as a .pdf-file, with relevant implementations enclosed (source code etc.). All this should be documented by a Readme.txt file.

Anti-plagiarism
The rules for this are very strict, see §36 in "Forskrift om studier ved NTNU" (page 23 in “Studiehåndbok for Sivilingeniørstudiet 2011-12”) regarding cheating and http://www.lovdata.no/all/hl-20050401-015.html#4-7

See also http://www.idi.ntnu.no/grupper/su/publ/ese/plagiarism.html.

Copyright or Intellectual Property Rights (IPR)

For the entire lifetime of this course, it has been "unclear", although rather frictionless, which “legal person” actually owned the IPR for the produced work, typically a project report and associated software. The Norwegian copyright law stands in LOV-1961-05-12-2 (http://www.lovdata.no/all/nl-19610512-002.html), and follows the Berne Convention for the Protection of Literary and Artistic Work from 1886.

Note: In Norway there is no need for a © symbol as in USA. Patents on software does not apply in Europe, but can be awarded in USA. Your employer owns the copyrights for software written as part of your employment contract (EØS rule).

IPR can be dealt with in four ways:
1. Just letting all parties (students, advisors, companies, anybody) do whatever they want with the work and with no explicit rules, just as now and compliant with the principle that scientific results (including implementation) shall be open and free of charge to everybody. However, legally and by default (Berne convention), the copyright (or IPR) belongs to those (the students) that have their names on the front page of the actual work.
2. Modifying NTNU’s new IPR policy, aimed at covering master’s theses - alone or in a group - but not project (i.e. pre-master) reports, see http://www.ntnu.no/studieavd/skjema/standardavtale07e.doc, 2p, dated 2007-08-28.
3. Using a revised, liberal and BSD-inspired IPR policy (about free and open software), formulated and adapted by Reidar Conradi in August 2008:
http://www idi.ntnu.no/grupper/su/publ/ese/new-standardavtale08e.doc. It is a combination of point 1) and 2).

4. Applying NTNU’s new IPR policy (also) for the TDT4290 project course, possibly revising point 2): The original IPR policy proposal was drafted by NTNU’s legal advisor Morten Øien, but withdrawn at its board meeting on June 12, 2008, awaiting vital clarifications, see http://www.ntnu.no/styret/saker_prot/12.06.08web/46.08_vedl.pdf, 12p. NTNU’s IPR policy was finally approved at its board meeting on June 6, 2010; see NTNU’s S-sak 36/10: http://www.ntnu.no/styret/saker_prot/09.06.10web/36.10.pdf, 12p.

NTNU generally tries to coordinate its IPR policy with that of the University of Oslo from October 19, 2010: http://www.uio.no/for-ansatte/abeidsstotte/fa/kontraktingaelse/ipr-politikk-191010.pdf, 16p. See also proposed extensions from the “Sejersted-II committee” at University of Oslo from May 20, 2011: http://www.hf.uio.no/imv/om/dok/2011/instituttstyret/SAK192011Høringsnotat_fra_rektor.pdf, 5p.

Check also what law professor Olav Torvund from University of Oslo has written in his interesting blog on IPRs and other issues: http://blogg.torvund.net/. See finally some overall comments from June 2008 by Reidar Conradi in: http://www.idi.ntnu.no/~conradi/IT-debate/ip-politikk-ntnu-26jun08.html, 6p.

Course reflection, evaluation and feedback

We intend to do a systematic evaluation of this project course. For this purpose, a “student reference group” must be established among the course participants. The course will be evaluated in the following ways:

- **Student surveys**: individual students will be asked to fill in a questionnaire at the beginning and at the end of the course. The questionnaires at the beginning of the course will be used to gather data on the students’ expectations and the questionnaires at the end will be used to gather data on if the students’ expectations have been met and other relevant feedback from the students.

- **Mid-term assessment by advisors**: the advisors will be asked to provide a mid-term assessment on how their project group(s) are progressing and notify if they foresee any challenges with their groups.

- **Customer feedback**: Customers will be asked to fill in a short questionnaire at the beginning and at the end of the course. These will be used to gather data on if and how the customers’ expectations have been met and to gauge customer satisfaction with their project group and the course in general.

The feedback received from the different parties will be used to improve the course for the future students.
Appendix A – The project plan

This section gives an example of how to structure a project plan. The project plan is a dynamic document that will evolve and change throughout the whole project. The project plan regulates the administrative part of the project and guides the project.

Depending on the type of lifecycle model you use you will have to structure the project plan differently.

A1. Overall project plan

Recommended content of the project plan (“project directive”):

- **Project name**
- **Project sponsor** (customer)
- **Partners including responsible third party providers**
- **Background for the project**: software system development
- **Measurement of project effects**, i.e. **goals** like:
  - Reduce the time it takes to create a daily production report with 3 hours…
  - 30% cost reduction of…
  - 40% increase in sales…
  - etc.

The effect measures are typically stated by the project sponsor, but it is likely that the group has to take the initiative to specify these in detail.

- **General terms.** What are your limitations, tool selections, organizational demands from the customer, resources etc.?
- **Based on the planned effort**: How many person-hours are to be used?
- **Schedule of results.** When should deliverables be available as milestones or sprints/iterations?
A2. Concrete project work plan

Recommended content:

- phases/sprints
- activities
- milestones
- person-hours per activity and phase + lectures + project management

The project plan (in form of a Gantt diagram) can be attached as appendices.

It is also recommended to attach the detailed plan of the phases as appendices.

The workload (measured in %) of this course is normally scattered out on different project phases, as shown in table 2. A suggestion of a relative workload is found in column “Norm”, while the experiences from groups 1 to 7 in 1997 are found in the subsequent columns. As you can see from the table, the relative workload varies a lot from group to group. This is a normal variation and is caused by the different assignments, the groups working differently, and that it is not strictly defined what each phase should contain. The suggestions given in “Norm” are a good starting point for the project plan.

For projects based on iterative development like Scrum, you will have to incorporate the phases which are needed into each sprint into the documents for each sprint.

Table 2: Relative workload for waterfall-like project.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Share in %</th>
<th>Norm</th>
<th>Gr.1</th>
<th>Gr.2</th>
<th>Gr.3</th>
<th>Gr.4</th>
<th>Gr.5</th>
<th>Gr.6</th>
<th>Gr.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project management</td>
<td></td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lectures and self study</td>
<td></td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Planning</td>
<td></td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Pre study</td>
<td></td>
<td>15</td>
<td>24</td>
<td>14</td>
<td>26</td>
<td>21</td>
<td>17</td>
<td>22</td>
<td>38</td>
</tr>
<tr>
<td>Requirements specification</td>
<td></td>
<td>20</td>
<td>26</td>
<td>34</td>
<td>25</td>
<td>25</td>
<td>18</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Design</td>
<td></td>
<td>15</td>
<td>27</td>
<td>19</td>
<td>15</td>
<td>18</td>
<td>22</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>Programming and documentation</td>
<td></td>
<td>13</td>
<td>6</td>
<td>25</td>
<td>11</td>
<td>14</td>
<td>31</td>
<td>23</td>
<td>14</td>
</tr>
<tr>
<td>Project evaluation</td>
<td></td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Presentation and demonstration</td>
<td></td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Ideas to the content of the different phases are found in Appendix B. It is also recommended to look at previous project reports, which can be found at the web site of the course.
A3. Project organization

Recommended content:

• An organizational diagram of how the group is organized
• Roles, i.e., project leader, system analysis, system architect, system designer, test leader, customer contact, QA responsible, etc. Try to be inventive in role allocation!
• Responsibilities of the different roles
• Weekly schedule

A4. Templates and standards

The group should create templates for all relevant document types. Even though it will take some time to create these in the beginning, the group will benefit from these in two ways: 1) the layout will be correct when creating project documents and 2) reduction of irritation and stress within the group.

Templates ought to be made for:

• phase documents
• agenda for meetings
• weekly status reports for the advisor meetings
• etc.

The group should also create pragmatic standards for:

• organization of files
• naming of files
• coding style
• etc.

A5. Version control procedures

The group must create a systematic procedure for version control for all textual documents, source code, etc., see Appendix D.4.2.3 on actual tools like CVS, SVN, Make etc.

A6. Documentation of project work

Internal project meetings

Try to have internal meetings at least once per week. In these meetings you should present the status, coordinate activities, divide tasks, and check the “mood” of the project. Set up an agenda and write precise minutes for each meeting.

Internal reports

A typical internal report:

• Person-hours - used and remaining (according to plan)
• Activities - done and remaining
• Achieved and not achieved milestones. These are important progress indicators.

Reporting of person-hours used should be done in written form to a specified time, e.g. as a part of the weekly reports.
A7. Quality Assurance (QA)

QA assumes that the relevant product qualities have been identified, so that the development process can be tailored to achieve these, e.g. reliability, performance, usefulness etc. There exists an ISO-standard for this (ISO 9126). More information can be found on Wikipedia, see http://en.wikipedia.org/wiki/ISO_9126

Time of response
Make agreements with the customer. There should be time of response on:
- Approval of minutes of customer meeting (e.g. 24 hours)
- Feedback on phase documents the customer would like for review (max 48 timer)
- Approval of phase documents (max 48 hours)
- Answer to a question (e.g. 24 hours)
- To get agreed documents etc (e.g. 24 hours)
- Other

Routines for producing high quality internally
This has something to do with how you organize the specification and programming work, e.g. user involvement, “pair programming”, design examination, etc. The number of people involved should be weighed against available resources.

Routines for approval of phase documents
Specify how you are going to approve the phase results (deliverable), which mainly consists of the phase documents. It is natural to involve the customer in the approval of pre-studies and requirements specification (or whatever you might call these documents). You must, as stated earlier, agree upon a time of response with the customer.

Calling for a meeting with the customer
For all the meetings with the customer you should send a call for the meeting, specifying time, place, intention (result), agenda, and background documents. It is vital to specify what preparations you expect of the customer and the group before the meeting.

You have to agree with the customer how long in advance the calling for meeting should be sent, e.g. at 12:00 two working days before the meeting is going to take place.

Minutes of a customer meeting
You must write a summary of the meetings with the customer. It is vital that you write down decisions, actions (what, who, and deadline), clarifications etc. that are important for further work in the project. The customer must approve the minutes of the meeting, to make sure there where no misunderstanding of decisions made etc. The minutes of meetings are part of the “contract” with the customer. In normal working life it is not uncommon that the minutes of meetings are part of the contract document with the customer.

In the project plan you should specify when the summary of the meeting should be done, when it is to be given to the customer for approval, how to distribute (e-mail, fax etc.) and expected time of response from the customer. It ought to be written by 12:00 o’clock the following day and should be distributed as soon as possible when you are done with internal approval within the group. It is vital that you get an approval as soon as possible to avoid misunderstandings.

Calling for the weekly advisor meeting between the group and its advisors
12:00 o’clock the day before the meeting, with print out of all needed paper.

Agenda for the weekly meeting with the advisor - a template that is to be followed
1. Approval of agenda
2. Approval of minutes of meeting from last advisor meeting
3. Comments to the minutes from last customer meeting or other meetings
4. Approval of the status report, which may be structured as follows:
   4.1 Summary
   4.2 Work done in this period
       Status of the documents that are being created
       Meetings
       Other activities
   4.3 Problems – what is interfering with the progress or taking resources? Problems are often risks that have taken effect.
   4.4 Planning of work for the next period
       Meetings
       Activities
   4.5 Other
5. Review/approval of attached phase documents
6. Other issues are listed here…
7. Other issues

The status report (see Section 4.1-4.5 above) should be handed in as a separate document.

**Minutes of the weekly meeting with the advisors**
Is attached to the next calling for meeting and is a fixed subject on the agenda.

**A8. Test plan**
The project has to have an overall test plan, which either can be part of the project plan or as a test document (the latter is recommended, see Appendix C6).
Appendix B – Suggestion for appendices in your project plan

B1. Partners
Owners, target audience, customer representative(s), project group, advisors. For each person, record:
• name
• address
• phone number
• e-mail
• etc.

B2. Concrete project plan
The current project plan and old project plans. By also keeping the old plans the group can see how they have evolved and also possible learn from previous experience.

B3. Detailed phase plans
A detailed description of what each phase consists of.

When ending a phase, the next phase is fine planned in detail. The detailed plans for each phase are put here and not in the end of last phase document.

B4. Table for handling of risks

<table>
<thead>
<tr>
<th>Nr</th>
<th>Activity</th>
<th>Risk factor</th>
<th>Consequences</th>
<th>Probability</th>
<th>Strategy and actions</th>
<th>Deadline</th>
<th>Responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All</td>
<td>Hans is involved in UKA</td>
<td>H: The quality of the project results will decrease</td>
<td>M</td>
<td>Reduce Assign delimited tasks to Hans with clear deadlines</td>
<td>Continues</td>
<td>Project leader</td>
</tr>
</tbody>
</table>

L = Low, M = Medium, H = High
B5. Table for effort registration

All projects need to register the effort spent by each project participant on the different activities (e.g. Prestudy, Programming etc.) and in what period (week 1, week 2 etc.). This is needed to ensure that the project is on track according to the project plan. A weekly registration or periodization is common.

So each of you must weekly report - in a so-called time sheet – seven data items per relevant activity and period: project group no, person name, date of registration, period no, activity name or id, your effort spent and given in person-hours (possibly zero).

Make a template time-sheet for this information as soon as possible (a textual email-message format will do), and establish reporting procedures from the very project start. The reported effort data should be delivered ca. two days before the weekly advisor meeting. The project manager (or a delegated person) should be responsible to collect and synthesize the individual effort data into an updated project effort-matrix on a spread-sheet. The matrix data will be used to regularly monitor the planned (or estimated) effort vs. the actual one for the whole project. This matrix has time (period number) as the horizontal dimension, and activity as the vertical dimension. Each matrix cell contains a number measured in person-hours (ph).

So very early in the project, as part of making a Project Plan, you must break down the project’s total available or estimated effort (ca. 1700 person-hours) into a dozen main activities or phases, which again are allocated to periods (week no 1-13), cf. Appendix A2. Naturally, activities belonging to the last part of the project cannot be broken down in detail in the start. Thus the project plan must be adjusted over time.

Example: Assume that we have a software project with three estimated activities (A1-A3) over three time periods (T1-T3):

- A1. Prestudy, whose estimated effort is 40 ph (person-hours).
- A2. Requirements, with 40 ph.
- A3. Implementation, with 20 ph.
- A. Total of 100 ph.

The project has an unspecified number of participants, so our project manager must keep track of the total resource usage (effort, time).

Version 1: Initial effort-matrix with very uneven effort estimates in the three periods:

<table>
<thead>
<tr>
<th>Activity \ Period</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>Activity sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. Pre-study</td>
<td>40</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>A2. Requirement</td>
<td></td>
<td>40</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>A3. Implementation</td>
<td></td>
<td></td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Period sums</td>
<td>40</td>
<td>40</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>
Comment: It makes sense to overlap the three activities a bit, to get a more even effort distribution over the three periods.

**Version 2:** Reconciled matrix version, where the three “diagonal” ph-estimates (40, 40, 20) are spread out to get a more even effort distribution over time - please discuss the revised ph-estimates

<table>
<thead>
<tr>
<th>Group no: ...</th>
<th>Date: ...</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Activity\ Period</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>Activity sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. Pre-study</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>A2. Requirement</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>A3. Implementation</td>
<td>0 (OK)</td>
<td>5</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Period sums</td>
<td>30</td>
<td>35</td>
<td>35</td>
<td>100</td>
</tr>
</tbody>
</table>

**Version 3:** Now introducing estimated (E:) vs. actual (A:) effort per period (T1-T3), both per running period (as above) and accumulated over several periods (see after the “/”-symbol in the below effort-matrix):

<table>
<thead>
<tr>
<th>Group no: ...</th>
<th>Date: ...</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Activity\Period</th>
<th>Start</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>Activity sums</th>
<th>Activity comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. Pre-study</td>
<td>E:20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 20/20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: 13/13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 10/30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:12/25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 10/40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ??/??</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: */40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2. Requirement</td>
<td>E=10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 10/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: 11/11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 20/30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A::19/30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 10/40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ??/??</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: */40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3. Implementation</td>
<td>E=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 0/ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: 2/ 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 5/ 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: 7/ 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: 15/20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:??/??</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: */20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period sums</td>
<td>E=30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E:30/30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:26/26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E:35/65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:38/64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E:35/100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A:??/??</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E: */100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: ..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Period comments | A1 delayed A3 before | A1 delayed A3 before | ... | ...

* means irrelevant

Let us assume that two time periods (T1-T2) have passed, with T3 just about to start.

**Observation:** in activity A1 after time T2 the Estimated running effort is 10 ph and the estimated accumulated effort (i.e. including T1) is 20+10 = 30 ph. However, the Actual effort for A1/T2 is 12 ph, and the accumulated effort is 13+12 = 25 ph. So it seems that A1 is a bit behind the estimated effort (“plan”) – but that can have many valid reasons. We are only measuring resource usage (effort, time), not the actual state of the software under development!
Ex. what advice will you give to all the activities A1-A3 for the last T3 period?

Appendix C – Content of the phase/sprint documents/chapters
This appendix contains information about what the different phase documents (or report chapters) should include. See also former project reports for more details.

That is, it is common to divide a software project into the following 13 (or so) phases, whose documentation then becomes a chapter in your final project report:
C1. Introduction
C2. Planning
C3. Pre-study of the problem space vs. solution space
C4. Choice of lifecycle-model: waterfall vs. agile?
C5. Requirements specifications
C6. Estimation of realization effort for use-case model
C7. Construction/ design
C8. Programming
C9. Testing
C10. Documentation
C11. Evaluation
C12. Presentation and demonstration
C13. Appendices

If an agile lifecycle model is chosen, like in Scrum, we recommend one “phase” (or report chapter) per Scrum-sprint, typically 2-3 such. These “Scrum phases” will then replace the waterfall phases C5-C10.

Below you will find some more information on what each phase document is expected to contain. Note that the group can also select another phasing.

C1. Introduction
Write a good, one-page abstract early, and explain the overall context, motivation, demands and results.

C2. Planning
See appendix A – Project plan.

C3. Pre-study of the problem space vs. solution space
The preliminary studies are vital for the group to obtain a good understanding of the total problem. Here, you will have to describe the problem at hand. You should describe the current system and the
planned solutions (text, workflow, use-case scenarios, information flow, and other graphical presentations you can use). It is all about getting a good understanding of the challenges ahead!

The group should investigate if existing and potentially competing solutions exists on the market. If such solutions exist, they should be described. You should also describe alternative solutions that fully or partially require custom implementations. The group must also set up evaluation criteria that form the basis for choice of a solution. Software by third party software providers (as OSS or COTS) should be actively pursued as candidates for implementation of large parts of your software system, see http://sourceforge.net

In cases where existing components can be applied as modules in the project solution, a simple cost-benefit analysis should be carried out.

Summary:
- Describe the main business requirements, both functional and non-functional, that will constitute the requirements for the final solution and its functionality. These requirements will later form the base for later formalization of requirements. Try also to make use-case diagrams to express the major functional requirements, cf. the simple effort-estimation method in Appendix F.
- Describe the situation and solutions of today (“as-is”)
- Describe the wanted situation and its possible solutions (“to-be”)
- Evaluation criteria
- Market investigations
- Description of alternative solutions
- Evaluation of alternative solutions, including adjusted requirements and potential costs and benefits.
- Choice of solution, in dialog with customer.

To conclude, the pre-study should have two main deliveries:
- A (partly) prioritized set of requirements – cf. the Scrum “backlog”.
- A proposed system architecture to enable a fast break-down (modularization) of the technical work

This is also the time to revise your project plan, possible due to use-case based estimation or clear choices of 3rd party software.

C4. Lifecycle model: waterfall vs. agile?

For the realization of the recommended requirements above, the groups are fairly free to choose between a waterfall or an agile lifecycle model to make a prototype for this project. However, some of the assignments do require use of agile development methods, and the customer will anyhow like to have his/her hand on the steering wheel.

The concept of waterfall development and its properties are considered known from previous courses, and are not covered here. See also phase descriptions in sections C5-C10 below.

Use of the agile methodology called Scrum is covered briefly in Appendix E and by the lecture on September 17th.

C5. Requirements Specifications
In the requirements specification phase, it is important to explicitly state the system requirements and link them to the business requirements from the pre-study phase. Typically, requirements are divided into functional and non-functional requirements. Structure the requirements such that the presentation is well organized.

Some persons like to enumerate requirements (R1, R2 ...), which may create “boring” reading where it is easy to lose track of the content. The advantage with numbering is that it is then easy to separate the requirements from the rest of the text, each becomes explicit, and you achieve traceability and structure.

Use a lot of figures! Good figures say more than a thousand words. We strongly recommend making use-case diagrams here, also because we then can make quick and reliable estimates of the ensuing design, programming and test effort (Appendix F). Before requirements are stabilized, you should give an overview of the software architecture. This means including figures that show how separate modules are related.

Prototype-diagrams of the user interfaces are often very helpful for communicating with the customer.

Suggested outline of a more detailed Requirements specifications
The IEEE (Institute of Electrical and Electronics Engineers) have made a good recommendation paper on recommended practices for writing requirements specifications which can be found on; (http://ieeexplore.ieee.org/iel4/5841/15571/00720574.pdf). If you are using this recommendation, please note that it is designed as a general-purpose software requirement specification plan and that some parts may not be relevant for your project.

The outline of the IEEE software requirement specification is:

Table of Contents
1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Definitions, acronyms, and abbreviations
   1.4 References
   1.5 Overview
2. Overall description
   2.1 Product perspective
   2.2 Product functions
   2.3 User characteristic
   2.4 Constraints
   2.5 Assumptions and dependencies
3. Specific non-functional requirements (e.g. performance requirements, database requirements, security, reliability etc.)
Appendices
Index
Glossary

C6. Estimation of realization effort of a use-case model

For TDT4290 Customer-driven Project, IDI, NTNU - by Reidar Conradi, Aug. 2011. See also Appendix F for an extended presentation; here comes the short one.

Use-case based effort estimation is a very simple and cheap method to estimate the remaining implementation effort of a requirements specification, when the latter is expressed as a set of
“upgraded” use-case diagrams. This means that a textual specification (resembling “pseudo-code”) must be added to the graphical actor and use-case models. The actors and use-cases must then be categorised as Simple, Average, and Complex - by you.

An effort estimate in person-hours can now be calculated, by you or by a spread-sheet tool (www.idi.ntnu.no/grupper/su/publ/reidar/uc-ProjectEstimateMethod-2011-v2.xls). It typically takes 30 minutes for you to execute the entire method, given that "upgraded" use-case diagrams are available. And only six “numbers” need to be given by you!

Example: Appendix F contains a small use-case diagram with 3 actors and 10 use-cases. All 3 actors are termed Complex. Further, 2 of the use-cases are termed Simple, 3 are Average, and 5 are Complex. We then multiply the actor numbers with the “cost points” 1,2,3 for respectively Simple, Average, Complex actors, i.e. 1*0 + 2*0 + 3*3 - or totally 9 actor points. We similarly multiply the use-case numbers with the cost points 5,10 and 15 for respectively Simple, Average, Complex use-cases, i.e. 5*2 +10*3+15*5 or totally 115 cost points. The sum is 9+115 = 124 cost points, which should be multiplied by a Productivity factor to get real person-hours. For small systems like yours, this factor can be set to 10. Thus, the estimated implementation effort becomes124*10 = ca. 1250 person-hours (+- 20%)!

C7. Construction / Design

Construction is all about getting a vague system description, to a specific and detailed description that can be implemented and realized.

A client-server software system typically includes three tiers:
1. User interface
2. Business or application logic
3. Database or file system

Pseudo-code and UML are useful for describing solutions in the construction phase. Regardless of which type of development strategy is chosen, (waterfall, incremental) most software implementation projects start with system architecture and a sketch of the desired design, in order to ease later division into parts. The project most likely will be further developed later, so a “modular” design is to prefer. Designing a modular system also makes testing much more easily, since defects can be tracked down to individual modules.

C8. Programming

In the very beginning of the project a lot of important technical decisions have to be made. This is vital for the remainder of the project, but there are certain practices that make it all easier.

General knowledge about programming is expected to be covered in previous courses. Depending on the actual project commitment, this project might require that the team members learn new programming language(s), new concepts of programming, various technical skills etc. The group have to plan how to obtain this knowledge, maybe in cooperation with the customer and the advisor.

Best practices
The code you write might be used as a base for further development, and may also be used by the other team members.
Inside the group, it will also be practical to have common design and code conventions that all group members understand and practice. If the customer has a coding convention, they probably would like you to use it also.

Knowledge to and use of patterns is also useful when it comes to programming. This is briefly covered in previous courses, but examples of patterns from architecture and coding are found practically anywhere on the internet. A good book resource on patterns is “Design Patterns: Elements of Reusable Object-Oriented Software”, see also: http://en.wikipedia.org/wiki/Gang_of_Four_(software)

Legal issues
Please observe that some freeware or trialware licenses of code editors etc. states that it is prohibited to use them to write code for commercial use. Check the license of the software that you decide to use in the project, and discuss it with the customer if there are such clauses that you might be in conflict with.

Technical support tools
Versioning control and backup CVS / SVN tools should, as mentioned in Appendix A5, be used for all technical artefacts (UML-diagrams, code, test data, etc.) and documents in the project. When you set up a development environment, make sure to set up a functional versioning system with backup as well. You never know when you might need it! It is possible to use CVS/SVN on IDI, for more info, contact the Gurutjenesten in the P-15 building.

Coding style
How to write source code should also be specified. Such documentation should typically contain:

● Programming conventions, e.g. in use by the customer
● Standards for commenting source code.
● Show examples of source for how the programming conventions look like in practice.

The source code should be commented and documented so well, that the customer easily can make modifications and build on your work after the project is finished.

At the end of your project, the source code and necessary resources should be included on the CD / DVD attached to the project report, supplemented with a Readme.txt file.

C9. Testing

Testing is usually planned and carried out in five parts:

1. Overall test plan – This should be created as the last part of the requirement specification phase.
2. A plan for each test that need to be carried out. – This should be done in the end of the construction phase.
3. Creation of detailed test specifications or checklists for each test. – This should be done in the end of the construction phase.
4. Execution of tests, including correction of defects, re-testing and documentation of test results.
5. Approval of test results

When you create a test plan it is important to specify:

● Which tests should be carried out?
Which tests should contain checklists? (Checklists are most common for entity and module testing)
Which tests should contain detailed specifications? (Detailed test specifications are common for testing systems, integrations, usability and acceptance)
Who are the test persons? (Project, customer, others, ...)
When should the tests be carried out?
Who are responsible for carrying out detailed test specifications and check lists?

The level of detail should fit the nature of your project.

The detailed test specifications should contain:
- Test descriptions (the operations that should be carried out)
- Data that will be tested (input and expected output)

<table>
<thead>
<tr>
<th>Tests carried out</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit test (programming phase)</td>
<td>Testing of the smallest units in the project, i.e., user interface, methods, stored procedures, objects, classes, etc.</td>
</tr>
<tr>
<td>Modul test (design phase)</td>
<td>Entities integrated into bigger software components. Modules are tested to assure that the coordination and communication between the entities are as expected.</td>
</tr>
<tr>
<td>System test (requirement phase)</td>
<td>All modules that together form a complete version of the system should be tested. The systems are tested to assure that the coordination and communication between models are as expected.</td>
</tr>
<tr>
<td>Integration test (design phase)</td>
<td>This is a complete test of the system and its interfaces to the world around. The last defects should be found and it should be verified that the system behaves well according to the requirement specifications. In some projects integration and system tests are merged.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tests carried out by the end users</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usability tests (non functional)</td>
<td>These are tests that assure that the interaction between users and the system is as expected. The goal is to get user friendly applications.</td>
</tr>
<tr>
<td>Acceptance tests (non functional)</td>
<td>Here, the end users should test if the system and its user interface to its environment are as expected. Based on this acceptance test, the management or customer make decisions on whether the product should be used or not.</td>
</tr>
</tbody>
</table>

When testing, you should perform the test after completing a phase/design/component etc. This is very close to the V-model used in software development. For more information about this, the Wikipedia article gives a good introduction (http://en.wikipedia.org/wiki/V-Model_(software_development)).

C10. Internal and external documentation

A user- and installation-guide for the final product must also be created. It should include an installation guide, which describe the installation process step by step.
Note, that your system and its installation will be tested by your advisor.

Hint: start on the documentation as soon as possible, as it describes the current state of the project for the project team.

C11. **Evaluation**

The groups decide themselves what to include in the project evaluation, but we recommend including the following elements:

1. The internal process and results: How have you worked together as a team? What have you done well? What have you not done so well? What would you have done differently? Conflicts that arose and how these were handled? Did you reach the project goals? What did you learn?
2. The customer and the project task: How was the communication with the customer? How did you experience the project assignment?
3. The advisors: How was communication with the advisors? Was the supervision good enough? How could the course be improved to next year?
4. Further work: Give an estimate for how much effort that is necessary to complete the product / project.
5. Suggestions for improvement. What is missing to make this course better for both students, customers and advisors.

It is important to describe hidden problems that can have affected the work but is not shown in the project report. Make sure that you also describe any additional work that is not shown in the project report.

C12. **Project presentation and demonstration**

To be used in Section 3.7.

The start-up is divided into three parts:

1. **Presentation**
   - Explain the project assignment and goals
   - Problems and priorities
   - Which solution did you choose? What were the alternatives? Why did you choose the final solution?
   - A description of the final solution
   - Some final reflections

2. **Demo**
   - Show your implemented prototype and its main functionalities.

3. **Questions:** max 5 minutes
   - **Total:** 45 minutes

Remember to give a hand-out of the presentation to your advisor, customer and external examiner (censor) at the presentation.

C13. **Appendices**

These may - for instance – include the following, but see also section C10 above:

1. User and installation guides (cf. external documentation?)
2. Technical/internal documents (cf. internal documentation?)
3. Other, e.g. special material provided by the customer.
4. Possible contracts and non-disclosure agreements.
APPENDIX D – Administrative and Technical Resources
This chapter presents an overview over the resources available during the project work.

D4.1 Office Resources

<table>
<thead>
<tr>
<th>Printouts</th>
<th>The group has a quota of 500 pages each.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo Copying</td>
<td>All the documents you deliver during the project period can be copied for free at the main copier in IT-154. The secretaries at the information desk will give you the information to operate the copier.</td>
</tr>
<tr>
<td>Telephone</td>
<td>If you need to use a phone as a part of your project work, please contact the course coordinator or your advisor.</td>
</tr>
<tr>
<td>Telefax</td>
<td>You have access to a fax machine (73 59 44 66) at IDI (ITV-254). Ask at the information desk.</td>
</tr>
<tr>
<td>Homepage</td>
<td>The course webpage contains a lot of important information. It should therefore be checked frequently, see <a href="http://www.idi.ntnu.no/emner/tdt4290/">www.idi.ntnu.no/emner/tdt4290/</a>.</td>
</tr>
</tbody>
</table>

D4.2 Technical Resources

D4.2.1. Workstations
Fourth year computer science students have access to the 5th floor in the P15 building. Additionally IDI technical group offers virtual servers for all the groups to host/run any special software necessary that is not available on the normal servers available to students.

If technical problems occur or you have particular needs regarding the computers, contact the IDI technical group, drift<##>idi.ntnu.no

D4.2.2. Source Control Management
It is usually essential for groups to maintain some sort of source control management. We will not advise on a specific tool, but not that groups have had success with CVS, SVN, Git, Mercurial, Arch and Bazaar in the past.

It is worth to keep in mind that CVS and SVN both adopts a centralized repository design, a lot of developers are shifting towards distributed repositories due to weaknesses in a centralized design. Distributed alternatives to look at would be such as Git (and github), Mercurial, Arch and Bazaar (all open source).

D4.2.4. Use of collaboration technology in the project
The groups are encouraged to use some kind of collaboration technology (file sharing, defect registration etc.) to coordinate the different tasks among group the members. There exist several
tools for this. The following is a list with some of the more well-known software packages for collaboration and information gathering:

**Gathering of static information on the web:**
Mediawiki: Open-source program for gathering static information on the web.  
Example of use: wikipedia.org

**Management software for defects etc:**
Trac: Great for registration of defects, things to be done etc. Open-source.  
Example of use: http://trac.edgewall.org/report

**Sharing of files:**
Microsoft Sharepoint: For sharing of files, documents etc. Commercial, but is available to NTNU students for free, under special licensing rules.  
Dropbox is another popular example, http://www.dropbox.com
Additionally Stud.ntnu.no offers group areas: http://www.stud.ntnu.no/kundesenter/
Appendix E – SCRUM – a popular agile method

Written by Torgeir Dingsøyr, PhD, SINTEF/IDI; Aug. 2008.

Scrum is an iterative planning and execution method for software development, is based on the “Agile Manifesto” (http://agilemanifesto.org/).

Scrum is based a series of increments of Sprints, being focused efforts for a limited period toward concrete goals. The concept originates from the mechanical engineering industry. Due to limitations in time in this project, there are not room for more than just a few sprints. Most textbooks and tutorials state that the sprints should take approximately 30 days, sprints of 14 days are becoming more widespread. For a small enough set of tasks, sprints of shorter duration also goes well.

![The basic workflow in Scrum development](http://www.mountaingoatsoftware.com/Scrum)

**E5.1. Product backlog**

The other key concept of Scrum is the product backlog, or just the backlog. The product backlog is the list containing all functionality in the product. This should be made in the initial phase of the project, but it can be altered and changed throughout the project as the requirements evolve. The backlog must not be mistaken with the requirements specifications. Consider the backlog as a more informal document.

**E5.2. Sprint planning meeting**

Before beginning a new sprint you should have a sprint planning meeting. This meeting is attended by the product owner (customer), the Scrum master and the rest of the Scrum team. The meeting could also be open for any interested.

During the meeting the Scrum team and the product owner should come to agreement over which features and functions that have the highest priority. Based on this, the Scrum team should be able to determine which task they will move from the product backlog to the sprint backlog. The Scrum team and the product owner should collectively define a sprint goal, a short description of what the sprint will achieve. This goal will later be discussed during the sprint review meeting.
After the sprint planning meeting, the Scrum team will have to discuss how much they are able to commit during the sprint. This might lead to renegotiation with the product owner, but it will always be up to the team how much they can achieve during the sprint.

The Scrum team organization is well suited for a group of students, because none of the traditional software engineering roles like programmer, architect, designer or tester exist. Instead the Scrum team is focusing on collectively complete the tasks within the sprint.

**E5.2.1. Daily Scrum status meeting**
The daily Scrum(-meeting) should be held every day during a sprint. Usually these meetings are held in the morning, so that the team members can plan the rest of the day. Anyone can attend these meetings, but others than the team members are only allowed to listen.

The daily Scrum should not be used as a meeting for problem solving; this should rather be discussed after the meeting only by the involved team members.

During the daily Scrum every team member should answer the three following questions:

1. What did you do yesterday?
2. What will you do today?
3. Are there any impediments in your way?

The daily Scrum is not a status update, it is more like a commitment to the other team members of what you will do till the next day, and what you have done since last meeting. The good thing about having a daily Scrum meeting is that it helps the team to see how important these commitments are to themselves and the team.

Since this is a student project, and you all have other courses to attend (and students are not known to work from 08-16 either) it is not always possible to have a daily Scrum meeting every day or in the morning. The team have to work out a solution that every team member feel comfortably and in the same time makes it possible to monitor the progress of each team member.

**E5.2.2. Irregularities**
Most likely there will sometimes be impediments between the scheduled plans and what is actually committed from the team members. Impediments could have many and various causes but it is the Scrum masters responsibility to resolve them as soon as possible. In cases where the impediments regard the Scrum master, he or she takes responsibility so that someone else is solving them. In more extreme cases (one on the team is not doing his workload, serious illness etc.) the team should contact the advisors of the course.

**E5.2.3. Sprint review meeting**
After each sprint a sprint review meeting is held. This is an informal meeting, which typically can consist of a demo of the new features made during the sprint. As for the other meetings this one is also open for everyone interested, but the Scrum team, Scrum master and the customer should normally participate this meeting.

During the sprint review meeting the project is assed against the sprint goal.

**E5.2.4. How do I prepare a project for Scrum (short tutorial)**
First you transcribe the requirements of the requirements specification to a list called the Product Backlog
Then, for each sprint, you make a prioritized list called a **Sprint Backlog**.

**E5.2.4.1. The team**
The product owner is usually the contact person with the customer. The product owner administers the product backlog
The Scrum master is similar to a project manager, but is not the same as a project manager.
E5.2.4.2. During sprints
The first day of a new sprint is used to create and analyze the sprint backlog. This is revised with the product owner before the work begins.
Every day during the sprint, the group have a meeting called the daily sprint, or sometimes the daily stand-up. This meeting should not take long, and should for all members of the group answer the following three questions:

1. What have you done since the last meeting?
2. What will you do between now and the next meeting?
3. Are there any threats, or anything preventing you from doing what you have planned?

E5.2.4.3. Recommended online resources about Scrum
Tutorial on agile development by Geir Ketil Hansen, SINTEF IKT
Software Process Improvement Conference (EuroSPI 2006), October 2006
http://www.idi.ntnu.no/grupper/su/publ/geirkjetil/eurospi06-slides-agile-hanssen.pdf

Mountain Goat Software is a training and consultant company that provide several free articles about Scrum:
http://www.mountaingoatsoftware.com/Scrum_articles

A brief tutorial on how to prepare a project for Scrum:
http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf

Wikipedia article with several links and citations:
Appendix F – Use-case based effort estimation

Written for NTNU course TDT4290 Project-Driven Project.
by Bente C. D. Anda, UiO and adapted by Reidar Conradi, IDI in 2001-2011.
www.idi.ntnu.no/grupper/su/publ/reidar/tdt4290-usecase-estim-final-23aug11-rc.doc

F6.1. Introduction to use-case estimation

It is quite hard to get a useful (i.e. reliable) cost or effort estimate (measured in system size or work effort) to implement a set of high-level requirements, when almost no design decisions have been made. IBM et al. developed in the 1970s a Function Point (FP) method (http://www.functionpoint.com/) to convert ER- and flow-diagrams into Function Points (FPs), which can be converted into Lines-Of-Code (LOC), and then person-hours.

The FP method has recently been adapted for use-cases by Univ. of Linköping (Karner 1993), Univ. of Oslo (Anda et al., 2001) and NTNU (Mohagheghi et al., 2005). Use-case based effort estimation aims to predict the implementation effort - in person-hours - of a requirements specification, given as a set of textually “upgraded” (also called “structured”) use-case diagrams. Such an estimate covers all remaining design and programming work, plus unit and module testing.

Our industrial results have so far demonstrated satisfactory precision (+-20 %) for systems of moderate size, i.e. about 3000 person-hours or 20,000 Java-LOC.

We need your help: to try out and improve this estimation method, which only takes 30 minutes per “upgraded” use-case-diagram. The added text (“pseudo code”) in the diagrams will anyhow be useful during remaining implementation work!

Some background: other estimation methods include the Delphi approach, i.e. letting 3-5 “human experts” try to agree upon a common prediction - or simply use the median. There is also the Work Breakdown into Subsystems, which at least assumes a high-level design specification. Furthermore, estimates done by human experts often fares better than estimates built on formal models. Lastly, Prof. Magne Jørgensen at the Simula Research Laboratory outside Oslo claims that the industrial overrun of cost- or schedule estimates in software projects is about 30% - and has remained so for 30 years (Jørgensen 2005)!

Findings from the literature: An industrial software developer spends 40% of his time on Requirements and Design, 20% on Programming, and 40% on Testing and documentation. Given a work year of 1500 person-hours, he/she will annually produce 4 lines of C-code (C-LOC) per person-hour for mega-LOC systems. This means 6000 C-LOC or 3500 equivalent Java-LOC, or 60 Function Points per year per person. The number of defects “injected” by programmers in this software is about 4 per Function Point, or 40 defects per 1000 C-LOC, or 240 (6*40) defects totally per year. Of these defects, 204 (85 %) are pre-release ones, each costing 1-3 person-hours to correct. However, 36 (15%) of the defects will survive into the first release, and each will later cost 10-30 person-hours to correct (Jones 2007). This means that almost half the programming effort goes to defect correction!

F6.2. More on the estimation method

The idea behind use-case based estimation is to add textual specifications, written in "pseudo-code", to document and thus upgrade the graphical actor and use-case models. This extra textual information is used to classify each upgraded actor and use-case model as either Simple, Average, or Complex - respectively with "cost points" 1, 2, 3 for actors and 10, 15, 20 cost points for use cases.
From the aggregated and weighted cost points, a total effort estimate in real person-hours can be produced.

The estimation method takes only 30 minutes to perform, based upon upgraded use-case diagrams. The estimation method may also use a spreadsheet (www idi.ntnu.no/grupper/su/publ/reidar/uc-ProjectEstimateMethod-2011-v2.xls), or you can do the conversion yourself with a pen and pencil in the same time.

You should later compare (“cross-check”) the effort estimate against the person-hours actually consumed in the total design, programming and testing of each use-case, or accumulated for all the use-cases.

F6.3. A mini-discussion

Challenges: How to achieve incremental maintenance - i.e., not only for the first release? How to handle reuse from 3rd party software providers, such as Open Source Software (OSS) and Commercial-Off-The-shelves Software (COTS)? And how to scale up to larger systems: - student software has Productivity of 10-12, the smallish “office” systems in the Anda paper have a Productivity of 25-30, while the Ericsson telecom systems in the Mohagheghi paper have a Productivity of 60-70.

F6.4. References


A more detailed explanation of some points in the estimation method stands below

1. Make first a well-structured, textual requirement specification, with requirements numbered e.g. as Ra-b.y (a,b:letters, y:digit). This is assumed made on beforehand.

2. Make a graphical use-case diagram in the usual way. That is, an actor is depicted as a “match-stick” figure, and a use-case as an “oval”. Lines or arrows between these symbols show relations between the underlying entities.

3. Upgrade the graphical diagrams with textual specification for the actors and users. The actual use-case diagram has 10 use-cases and 3 actors (see example later). You must therefore make textual descriptions for totally 13 entities, see below.

4. For each participating actor, categorize him/her as easy, average, or complex:
   a. A simple actor relates to a single system with a defined programming interface (API).
   b. An average actor relates to either another system through a communication protocol, e.g. TCP / IP, or communicates to another actor via a textual interface.
   c. A complex actor communicates via a graphical interface.
   d. Count up the number of actors in each category.

   Ex. Here, the 3 actors are all persons who shall communicate with the system through a graphical interface. Thus, we have 3 complex actors (all are complex, so we do not need textual actor descriptions to clarify this).

5. Similarly, for each use-case, categorize it as simple, average, or complex, depending on the number of major or alternative transactions in the event flow. A transaction is defined as an event, that occurs between an actor and a system:
   a. A simple use-case has 3 or fewer transactions.
   b. An average use-case has 4 to 7 transactions.
   c. A complex use-case has 8 or more transactions.
   d. Count up the number of use-cases in each category.

   Ex. The use-case "Transfer application to another clerk" is one of 10 such, and is marked in the diagram with **. It contains totally 10 transactions -- 7 in normal event flow and 3 in its only variation flow (5a1-5a3). This use-case must therefore be categorized as complex. The other 9 use-cases are up to you to categorize. The spread-sheet has tentatively been filled up with 5 complex use-cases, 3 average and 2 simple.

6. A total of 21 (!!) context factors to express the effect of technical and environmental issues. These must be provided by you in the general case. In the spread-sheet, all these are just preset to 3, i.e. “neutral”, giving an overall multiplicative context factor of 1. So just drop these!

7. Summing up: We first multiply the actor numbers (0,0,3) with “cost points” 1,2,3 for respectively Simple,Average,Complex actors, i.e. 1*0 + 2*0 + 3*3 - or totally 9 such points. We then multiply the use-case numbers (2,3,5) with cost points 5,10,15 for respectively Simple,Average,Complex use-cases, i.e. 5*2 + 10*3 + 15*5 or totally 115 points. The total is 9+115 = 124 so-called Use-case-Cost-Points (UCP). Congratulations, you are now done!! That is, only six “numbers” on actors and use-cases must be provided by you!

8. Productivity per UCP: At last, a Productivity factor must be multiplied with UCP above to get an effort estimate in person-hours. With a Productivity of 10, we will get 10*124 = ca. 1250 person-hours!!
9. Apply the calculated effort estimate to either:
   a. *Down-scale the requirements:* Your group project will have a maximum budget of about 1800 person-hours, with perhaps 1/3 or 600 person-hours for implementation. Thus, your requirements must be trimmed to almost half. And this is definitely useful to know beforehand!
   b. *Start implementing, record the actual effort – and compare with the estimate!*

**F6.6. Add-on to use-case example**

Graphical use-case diagram, with text specification added for 3 Actors and one Use-case (**).
**Example of added textual specification for actor A1.Clerk:**

<table>
<thead>
<tr>
<th>Actor ID:</th>
<th>A1.Clerk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>&lt;A short text to characterize the actor&gt;. Decides who will have a bank loan, specified by a loan Application.</td>
</tr>
<tr>
<td><strong>Examples of actions:</strong></td>
<td>&lt;…&gt; . Has a full graphical interface, to help with getting all the details of a loan Application in place.</td>
</tr>
</tbody>
</table>

**Example of added textual specification for actor A2.Provider:**

<table>
<thead>
<tr>
<th>Actor ID:</th>
<th>A2.Provider (of capital)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>&lt;A short text to characterize the actor&gt;. The capital part of a bank that actually pays out accepted loans.</td>
</tr>
<tr>
<td><strong>Examples of actions:</strong></td>
<td>&lt;…&gt; . Has a full graphical interface, to help with the legal parts of an accepted loan Application.</td>
</tr>
</tbody>
</table>

**Example of added textual specification for actor A3.Customer:**

<table>
<thead>
<tr>
<th>Actor ID:</th>
<th>A3.Customer (in bank)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>&lt;A short text to characterize the actor&gt;. Wants a bank loan, as requested by an emitted loan Application.</td>
</tr>
<tr>
<td><strong>Examples of actions:</strong></td>
<td>&lt;…&gt; . Has a full graphical interface, to help an actual Customer to fill in the application.</td>
</tr>
</tbody>
</table>
Example of added textual specification for a simple use-case:

<table>
<thead>
<tr>
<th>Use-case name:</th>
<th>Transfer loan Application to another clerk. **</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors related:</td>
<td>Clerk (only one actor).</td>
</tr>
<tr>
<td>Trigger:</td>
<td>&lt;Event that starts this uses-case&gt;.</td>
</tr>
<tr>
<td></td>
<td>Some parts of an Application need to be discussed by other clerks.</td>
</tr>
<tr>
<td>Pre-conditions:</td>
<td>&lt;that must be satisfied before the use-case can start execution&gt;.</td>
</tr>
<tr>
<td></td>
<td>The actual clerk must be logged on the system.</td>
</tr>
<tr>
<td>Post-conditions:</td>
<td>&lt;that must be satisfied before the use-case can finish execution&gt;.</td>
</tr>
<tr>
<td></td>
<td>The Application must be stored in the database with a valid and consistent state, and has been assigned to a specific clerk or a group of similar clerks.</td>
</tr>
<tr>
<td>Normal event flow:</td>
<td>&lt;A list of transactions that will be executed in a normal event flow of the use-case&gt;.</td>
</tr>
<tr>
<td></td>
<td>1. The clerk announces that he/she will transfer an Application under treatment to another clerk.</td>
</tr>
<tr>
<td></td>
<td>2. The system presents the name of the applicant and the reference number of his/her Application.</td>
</tr>
<tr>
<td></td>
<td>3. The clerk checks all this information.</td>
</tr>
<tr>
<td></td>
<td>4. The system presents a list over suitable groups of clerks and the customers now assigned to each group.</td>
</tr>
<tr>
<td></td>
<td>5. The clerk proposes another clerk, and possibly a second one just in case.</td>
</tr>
<tr>
<td></td>
<td>6. The (previous) clerk asks the system to transfer the Application to the proposed clerk.</td>
</tr>
<tr>
<td></td>
<td>7. The system transfers the Application to the proposed (new) clerk.</td>
</tr>
<tr>
<td>Variations in the event flow:</td>
<td>&lt;A list with descriptions of possible transaction variations (not necessarily errors) in the normal flow of the use-case&gt;.</td>
</tr>
<tr>
<td></td>
<td>5a. The previous clerk may order the system to generate an email message to notify the new clerk about the transferred Application.</td>
</tr>
<tr>
<td></td>
<td>5a1. The previous clerk asks for such notification.</td>
</tr>
<tr>
<td></td>
<td>5a2. The system generates an email message to the new clerk.</td>
</tr>
<tr>
<td></td>
<td>5a3. The use-case continues from point 7.</td>
</tr>
<tr>
<td>Related information</td>
<td></td>
</tr>
</tbody>
</table>

Comments on the description of event flow:

- The event flow shall be described as a list of transactions on the form:
  <no> <event description>.
- An <event description> shall be on the form:
  <actor> <verb> <direct object> <prepositional sentence>
- Such an <event description> may be one or more of the following:
  - An actor sends a request or data to the system.
- The system validates the received data.
- The system changes its internal state.
- The system responds to the actor with the result.
Appendix G – Project Descriptions

G.1. Spatial Data Indexing and Ranking Scheme

Title: Spatial Data Indexing and Ranking Scheme
Customer (Company): Schlumberger Information Solutions
Address: Risabergvegen 3, 4010 Stavanger

Assignment text:

Background
Hydrocarbon exploration in mature petroleum provinces is often just as much an information exploration as a geological exploration. Changes in energy markets, legislation framework, exploration strategy or technology breakthrough requires that the basin is re-examined in the light of these changes.

Geospatial search tools, like Schlumberger’s Studio Find, assists geoscientists in browsing, searching and analyzing information relevant for such a re-examining.

Spatial Data Indexing and Ranking
We need to be able to search 100 million items and interactively browse a map from global scale to the area of a single oil field in 3D.
Most spatial searching engines use paging or clustering as a scaling mechanism for large result sets. Paging gives false spatial negatives, while clustering gives false negative positives. These techniques are undesired in hydrocarbon exploration, since seeing data in their true spatial shape in the context of the geology is important. The spatial extent and resolution of the data items varies from regional geophysical measurements that span 100 of kilometers to single point well measurements that span centimeters. This proposes a challenge for ranking search results. When zooming into an area to look at many small data items the users do not want a spaghetti of regional seismic lines to hide the smaller items. At the same time, we do not want the users to believe that there is no data in a small area when there actually are a couple of regional seismic lines that goes across them. This means that
the search result ranking needs to take both span of the area of interest, the span of the data item and the density of data items within the area of interest.

Project
Describe spatial indexing strategies and their tradeoffs, performance and scalability consideration. Propose a strategy that fits with the hydrocarbon exploration usecase.

Contact details:
Name: Trond Benum  
Mobile: 917 96 770  
E-mail: benum@slb.com  

Name: Hallgrim Ludvigsen  
Mobile: 97500579  
E-mail: hallgrim.ludvigsen@slb.com

G.2. Analyze and present event log for a fire detection system

Title: Analyze and present event log for a fire detection system  
Customer (Company): Autronica Fire and Security AS  
Address: Haakon VII gt 4  Trondheim

Assignment text:  
The fire detection system writes every event, like alarm, prewarning and fault into a log file. The same for user commands as disablements, silence bells and reset system. This event log can have many thousands events.

The project is to present this information in a user friendly way that is easy to understand. It should be possibly to see an outline of the whole log in one display and then zoom into a period of special interest for more details. The system should run on a Windows PC and preferably also on Linux systems.

We have no such system to day so the project is free to come up with solution.

The project shall detail the user requirements, define the design and GUI and implement and test the Log inspection application.

Contact details: 
Name: Morten Solli  
Mobile: 402 44 859  
E-mail: morten.solli@autronicafire.no

G.3. Shattered

Title: Shattered  
Customer (Company): Gridmedia Technologies AS  
Address: Bakklandet, 7130 Brekstad
Assignment text:
We wish to develop a computer game using C# and our GameEngineComponents. The game will run on Microsoft Windows platforms. We have previously developed a component-based game engine for C#. The project consists of: developing the software application in C# that shows the 3D scene, responds to interactivity, etc. The GameEngineComponents provides all 3D rendering, physics simulation, input handling, audio, etc., that is needed. We will send all the 3D models, designs and audio that are needed. Our Geelix GameDevOS will be used to manage the project, share knowledge and provides educational courses on game development. We can set up virtual machines/servers that the students can use and provide all needed tools.

The goal with this project is to develop an early prototype for the game that features only one level where a ball rolls down the path of a mountain (see picture 1). The user controls the ball and must steer it on the path, avoid obstacles and collect coins. The students will not need to design 3D models, but they have to place 3D models out in the scene. The physics simulation (e.g. ball/ground contact, gravity/forces, etc.) is handled by the GameEngineComponents, but the students must set up the physics simulation in the scene (contacts, joints, friction, etc.). An easy to use component-based object model must be developed (in C#) that handles all of this.

Figure 1. Picture of the game.

Contact details:
Name: Ole-Ivar Holthe
Tlf: +47 47703253
Mobile: +47 47703253
Fax: +1 661-952-1182
E-mail: ole@gridmedia.com
G.4. Alternative Spaces – the digital prototype

Title: Alternative Spaces – the digital prototype

Customer (Company): Work Research Institute (AFI), Oslo and Akershus University College of Applied Sciences

Address: Boks 6954, Stensberggt. 25, 0130 Oslo

Assignment text:

With the increasing commercialization of city spaces, youth citizens have less room to express themselves, whether it is through visual, physical or mediated expressions like movement, dance, street art, sports activities or media outlets. The research project “Alternative Spaces” will explore the possibilities of creating alternative city spaces (both material and virtual) together with groups of youth, dancers, artists, landscape architects, journalists and scientists. Visual expression, digital media and film will make up both the methodological and empirical part of the project.

This assignment is to design a new platform experience that can facilitate digital sharing of such expressions across urban communities globally. As many of the project activities will be happening “on the ground” there is an acute need for a commercial-free digital hub where youth can upload, share and create content that strengthens their local engagement, expression of ideas and experimental thinking when it comes to participation and political influence on the development of their city spaces.

The project is in the early phase of development, where idea generation and the creating of connections and network of researchers and practitioners are integral. There will thus be plenty of opportunities for the students to influence the overall project design. During the fall of 2014 the project will arrange a workshop where all relevant actors will be invited to work together on related issues. This will be a venue for the students in the “customer driven project” to engage directly with the potential users of their platform. The case study of the first phase of the project will be Oslo, but Trondheim may be included as a complementary case if the student group finds the project attractive.

The project assignment will consist of the following activities:
1. Produce a user-oriented case, in dialogue with the research group
2. Participate and present pilot idea in “Alternative Spaces” workshop
3. Produce design suggestions, map needs and find the relevant technology for the pilot
4. Develop the web interface for analysis and representation
5. Evaluate design and technical solution and recommend further development

The assignment will be conducted within the research project “Alternative Spaces”, consisting of partners from Work Research Institute, Storbyprogrammet, Department of Journalism and Media Studies (HiOA), Snøhetta. More partners are coming! The project leader is Senior Researcher Aina Landsverk Hagen (AFI/WRI).

The student project group will be integral in making this project viral and global in outreach. We sincerely hope you find the assignment attractive. Attached you will find a preliminary sketch of the web interface - we are looking forward to discussions on the pilot idea, design and technical solutions.

Contact details:

Name: Aina Landsverk Hagen, Senior Researcher AFI (WRI)
Mobile: +47 93882168
E-mail: aina.hagen@afi.no

Name: Arne Bygdås, Senior Researcher AFI (WRI)
G.5. Generic HTML5 print-ad editor and pdf-generator

Title: Generic HTML5 print-ad editor and pdf-generator
Customer (Company): Adresseavisen AS
Address: Industriveien 13, 7080 Heimdal

Assignment text:
Adresseavisen has an in-house self-service ad generation system written in .NET, called Webassistenten. Traditionally, this system allows a user to generate print-ads (.pdf) based on rules and templates configured for Adobe InDesign server. Webassistenten is essential to facilitate the customers to easily order and produce their own ads for Adresseavisen and several different partners. Last year, Webassistenten processed ads worth over 130MNOK in revenue money. Webassistenten allows Adresseavisen to better control several marketing segments, such as real estate, used cars, death notices, job and classified ads.

We are now facing a crossroad in print ad-production, and want to simplify the process. The current system is very complex, and is very time consuming when it comes to extending and developing new templates.

Current activity workflow:
1. The user either imports an ad to Webassistenten or manually enters ad-data such as text and images (if the ad does not already exist).
2. The user selects which ad-template to use, and presses a button which sends the template (.indt) and an xml with ad-data to Adobe InDesign Server.
3. Adobe InDesign Server generates the ad in .pdf-format and saves a high-res copy to disk, and a low-res .jpg is returned in a callback.

We want a new system, which replaces the Adobe InDesign Server with a generic .NET HTML to PDF generator. This will also enforce the need for a new front-end ad generator, written in HTML5 and Javascript.

Proposed activity workflow:
• The user either imports an ad to Webassistenten or manually enters ad-data such as text and images (if the ad does not already exist).
• The user selects a publication date and which html ad-template to use.
• Javascript on the client-side generates an “iframe” (or similar) next to the ad-data form with a preview of the ad.
  a. When the user changes the text-fields in the ad-data form, Javascript-calls automatically updates the preview.
• The user presses an “Approve”-button, which sends a server-call to generate the ad server-side.

It is required that the HTML5-editor is generic, in such a way that the html-templates contain the logic for generating the ad. The template must contain ID-s for text-, and image fields.

It is also required that the HTML5-editor is generic for different ad-types. For this project, we would like the editor to work with death notice and car ads, which are two different, but somewhat similar, setups.
We have proof-of-concepts available for death notice and real estate ads, which can be used as a reference.

Further requirements, including sketches, will be made available in a simplified requirements document upon acceptance.

**Contact details:**

**Name**: Asle Dragsten Liebech  
**Mobile**: 483 00 525  
**E-mail**: asle.dragsten.liebech@adresseavisen.no

**Name**: Hans Kristian Ormberg  
**Mobile**: 959 07 262  
**E-mail**: hans.kristian.ormberg@adresseavisen.no

---

**G.6. Ocean Forecast**

**Title**: Ocean Forecast  
**Customer (Company)**: SINTEF Fisheries and Aquaculture  
**Address**: SINTEF Sealab, Brattørkaia 17C, Trondheim

**Assignment text:**

In oceanographic simulation as in meteorological simulation, the goal is to predict future environmental conditions. In addition to currents and temperature, oceanographic simulation often contains biological and chemical models such as salinity, nutrient salts and algae growth. This is useful for a number of coastal industries such as fisheries, aquaculture, transportation etc.

A normal simulation run by SINTEF Fisheries and aquaculture, contains a grid solution of 160x160 metres and covers a timespan of 8-12 months. The raw data for such a simulation is roughly 5TB of data stored in a NetCDF file. In total our simulated data is in the Petabyte range. Presentation of data today is handled by matlab scripts that generate videos and pdf's of specific views of the data. This approach is however limited since it is time consuming to generate files that will potentially never be shown and it only shows a limited view of the full data set.

In the prestudy part of the assignment, we would like to explore different possibilities and strategies for indexing large amount of data for fast retrieval. We would then like to explore possible overall architectures for a future system that can handle indexing and reading of large NetCDF files, dynamic creation of different views of the data, and interactive presentation to users. Of particular interest here is finding appropriate open source software that can handle the different parts of the system, as well as identifying which parts have to be implemented to combine the different systems. We are also interested in what kind of services this kind of architecture can provide a base for, so different future scenarios based on the architecture should be sketched out.

Once the prestudy is completed we see two possible routes the project could take: An architect heavy route with focus on the backend and method development with a narrow frontend prototype to demonstrate some functionality, or a scenario heavy route where the focus is on demonstrating different future applications with limited or dummy backend functionality. The choice will be made in cooperation with the students based on their recommendations from the prestudy.

**Contact details:**

**Name**: Finn Olav Bjørnson  
**Mobile**: 97726490

51
G.7. Bitcoin Coupon (BitCoupon)

Title: Bitcoin Coupon (BitCoupon)
Customer (Company): Netlight AS
Address: Karl Johans gate 23B, 4. etasje, 0159 Oslo

Assignment text:
Plan, develop and deploy a system to replace paper coupons, using the bitcoin protocol for authentication and utilizing the mobile platform for mobility.

The assignment is aimed at creating a fully functional prototype of a system that can replace the physical paper coupons common at stores and social events (such as beer coupons and lottery stubs).

The students are limited to choosing an agile project methodology (like Scrum or Kanban).

The project plan should be finalized within the first week of the project.

The scope of the project is limited to using the bitcoin protocol for authentication and validation of the coupon. The scope does not include implementing a payment service/solution.

The students are free to choose one of the following client implementations:
A. Mobile-Web-application, supporting at least iOS 6.0+ and Android 4.0+
B. Windows Mobile application (OS version 6.0+)
C. Android 4.0+ application.
D. iOS 6.0+ application.

The solution must support:
  1. Creation of coupons.
  2. Distribution of coupons.
  3. Redeeming of coupons.

In choosing where to place the group’s effort, the following prioritized list must be adhered to:
  1. Stability of system as a service.
  2. Admin functionality/tracking.
  3. Usability.
  4. Scalability.

The purpose for choosing the bitcoin protocol is to evaluate its suitability and maturity as a lightweight system for transaction authentication and validity. A section of the final report must include the group’s findings, experiences and evaluation of a live testing session.

Contact details:
Name: Sebastian Schöld
Tlf: +47 978 37 093
Mobile: +47 978 37 093
G.8. Crowd tracking using wireless signals

Title: Crowd tracking using wireless signals  
Customer (Company): FourC AS  
Address: Vestre Rosten 81, Trekanten, 7075 Tiller

Assignment text:
Understanding movement patterns of people is important to make business decisions in many industries. One example is public transport where route planning is based on this information. While it is fairly easy to track people entering a vehicle based on ticketing data, knowing when this person leaves the vehicle is much harder. Also, as more ticketing happen through mobile devices the tracking becomes even harder.

As more and more people carry smartphones a lot of wireless signals are produced by each individual as the phones search for wireless networks, etc. These signals can be collected using a lowend computing device with a wireless card and a GPS receiver.

The goal:
- Use signals emitted by personal mobile devices to determine movement patterns over a wide area in a public transport (PT) system.
- Ensure the data is collected anonymously and document privacy concerns and how this is affected by current regulations.
- Create a data model to exclude signals from devices not travelling on the PT system and to estimate the actual number of passengers based on the data collected, optionally also including data from ticketing and other systems.
- Visualize the result on a map showing passengers entering and departing the PT system at the different stops, as well as passengers transferring between routes, in a given period of time.

Contact details:
Name: Pål Løberg  
Mobile: +47 92099440  
Email: pallo@fourc.eu

Name: Tor Rune Skoglund  
Mobile: +47 907 43 563  
Email: trs@fourc.eu
G.9. CMIS for Wind
Title: CMIS for Wind
Customer (Company): FORCE Technology Norway AS
Address: Hornebergveien 1, 7038 Trondheim

Assignment text:

We have an existing application called Corrosion Management Information System (CMIS). This is in principle a web-information portal based on Microsoft Sharepoint technology. We now see a possible new market for an application used for offshore wind farms.

Offshore wind farm owners need to make sure that the static and sub-sea parts of the wind turbines are regularly inspected and that the results of the inspections are properly reviewed and necessary actions and revised inspection plans are made. Typically three or more parties are involved in this work:

- The owner or operator of the wind farm – checking the status of the inspections
- The party (-ies) performing the inspection – submitting inspection data.
- The party doing the review of the performed inspections and planning next year’s inspection (Typically us – FORCE) – reviewing the inspection data.

The application needs to be able to store a number data files, report types, i.e. be a repository for documents. In addition the application need to manage structured data to manage the planning of the inspections and have a database holding the anomalies/findings from the inspections.

The parties doing the inspection shall report the inspection status and findings using standard Microsoft Excel spreadsheet templates. The application shall be able to extract information from these spreadsheets and store them in the database linked up to the individual inspections performed.

Challenges in the project will be:
- Provide a user interface which simplifies how inspections are reported. Inspection reporting will be one infrequently, hence the user interface must be very simple for the inspector-reporters.
- Provide an interface allowing the reviewers to efficiently analyse and group the findings in the review work.

Contact details:
Name: Tore Landmark
Tlf: 64003721
Mobile: 98448569
E-mail: trl@force.no

G.10. Internet of Things
Title: Internet of Things
Customer (Company): Altran Norge AS
Address:

Assignment text:
Internet of Things is moving from instrumenting objects (RFID, Mobile Phone) to instrumenting the environment observing objects (sensors).
A room with hundreds of interconnected wireless sensors can monitor and register changes such as vibration, heat, audio and more, and provide interactive feedback based on what happens, so that people in the room may feel that they can manipulate the audio/video presentation with their bodies. The task is to create an Internet of Things showroom. To control a number of sensor devices in a wireless ad-hoc mesh network that collects sensor data and sends it to a central unit. The central device uses sensor data to control parameters and produces the an audio-visual presentation shown on a set of monitors, speakers, LEDs and more in the room.

The system should be able to be reused in other places, where the layout and configuration is different, so registering the placement of the sensors, over time, in 3D space is essential to support installment and relocation of sensors, as well as handling loss and instability of sensors. Students must expect that the sensing method, characteristics, protocol and quality will vary. In previous customer-driven projects, we have focused on multimedia-focused tasks and have much basis for further development, and the assignment scope will be up for discussion.

It is desirable that students provide an evaluation of the experience with sensors, protocols, problems with the use of devices from different vendors and NoSQL persistence technology for sensor data. Scope of the study will be addressed in the discussion between the client and the student group. Altran is an Engineering consultancy, focusing on IT development and embedded devices. We are inspired by the possibilities of the Internet of Things and wish to pursue a project that shows the possibilities of applying technology in an audio-visual way.

We provide employees with extensive experience in IT development, hardware and customer-driven project management. Altran will also provide sensors.

**Contact details:**
**Name:** Stig Lau  
**Tlf:** 
**Mobile:** 
**E-mail:** Stig.Lau@altran.com

### G.11. Using wearable devices in welfare services

**Title:** Using wearable devices in welfare services  
**Customer (Company):** Trondheim Kommune  
**Address:**

**Assignment text:**
Norway is investing in advanced welfare technologies as a tool to improve the quality and scalability of welfare services for seniors. Low maintenance cost for welfare technology is a decisive success factor. Low maintenance cost can be achieved by using standard and off-the-shelf components as part of future welfare service. In this task we will use off-the-shelf wearable devices to support common welfare services. Example devices are Fitbit and various smart watches.

The project will consist of the following tasks:
- Conduct a study of existing off-the-shelf devices, their capabilities and their APIs and tools for integration with open platforms.
- Create a set of scenarios for future services using these devices. The scenarios will include all stakeholders common in welfare services, such as seniors and municipality personnel. One of these scenarios will be selected for implementation.
- Design and implement one example service. The implementation will be end-to-end and will simulate a real service provided by the municipality.
The project will be run in close cooperation with user groups. The requirements will come from users. The development process will be agile and lean, with weekly or bi-weekly iterations where the project group will demonstrate new functionality. Tools such as paper prototyping and low/high fidelity prototypes will be used. The project will produce open source code.

**Contact details:**
Name: Kirsti Fossland Brørs
Tlf: 
Mobile: 
E-mail: fb9@trondheim.kommune.no
Appendix H – Student lists and groups

Table 3 – Group list. See Table 1 for assigned customer and advisor.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix I - Kick-off rooms for groups
Detailed group allocation will be done in the information meeting on the kick-off day and documented on the course webpage.

Table 4 – Kick-off rooms for groups and customers, 28.08.2014:

<table>
<thead>
<tr>
<th>Group</th>
<th>Room</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDI Rom 242</td>
<td>16:15</td>
</tr>
<tr>
<td>2</td>
<td>IDI Rom 354</td>
<td>16:15</td>
</tr>
<tr>
<td>3</td>
<td>IDI Rom 054</td>
<td>16:15</td>
</tr>
<tr>
<td>4</td>
<td>IDI Rom 454</td>
<td>16:15</td>
</tr>
<tr>
<td>5</td>
<td>IDI Rom 157</td>
<td>16:15</td>
</tr>
<tr>
<td>6</td>
<td>IDI Rom 260</td>
<td>16:15</td>
</tr>
<tr>
<td>7</td>
<td>Gamle fysikk nr 313</td>
<td>16:15</td>
</tr>
<tr>
<td>8</td>
<td>IT-bygg syd nr 147</td>
<td>16:15</td>
</tr>
<tr>
<td>9</td>
<td>Elektrobygg nr G144</td>
<td>16:15</td>
</tr>
<tr>
<td>10</td>
<td>Elektrobygg nr B418A</td>
<td>16:15</td>
</tr>
<tr>
<td>11</td>
<td>Elektrobygg nr C334</td>
<td>16:15</td>
</tr>
</tbody>
</table>

Room search:
http://www.ntnu.no/studieinformasjon/rom/
http://www.ntnu.no/kart/gloeshaugen/it-bygget/
http://www.ntnu.no/kart/no_cache/soek/
http://use.mazemap.com/
Appendix J – Guest Lectures

01.09.2014 Monday
"How to sell in large application projects", Thomas B. Pettersen - Computas,
14.15-16.00, Room KJL5

02.09.2014 Tuesday
"Scrum, agile development method", Torgeir Dingsoyr - SINTEF,
12.15-14.00, Room S3
Please read the following short book prior to the lecture:
http://www.infoq.com/minibooks/scrum-xp-from-the-trenches
(short booklet, available for free to download)

"Estimation, agile/practical project work", Fredrik Bach - BEKK,
14.15-16.00, Room R9

08.09.2014 Monday
"Group Dynamics" - Trude Selfors, Trude Selfors Kurs og Coaching,
14.15 - 18.00, Room S8
(Obligatory for students.)

15.09.2014 Monday
"Project management", Stian Mikelsen - Bearingpoint,
14.15-16.00, Room S3

24.09.2014 Wednesday
"Technical Writing in English", Stewart Clark - NTNU,
14.15-16.00, Room R5

As preparation, the instructor suggests to the students to study three things:
1. Viko which has some useful links on the left of the ‘writing papers’ link. These cover issues such as referencing systems and plagiarism, see http://www.ntnu.no/viko/english

01.10.2014 Wednesday
"Sales techniques with exercises in groups", Morten Selven - Mikos,
12.15-16.00, Room S3