MID-SWEDEN UNIVERSITY

Magnus Eriksson  
**Tools for large system development projects**This is a list of tools recommended to Mid-Sweden University students for large system development projects, and often used in the IT business. Students in computer engineering as well as industrial engineering and management, are expected to meet all these categories of tools during their studies. They should use them in group projects and theses for project management, application development, version control, documentation, test, analysis, and more. The list will be used in the SIMS courses [(Student Innovation Mid-Sweden](http://www.miun.se/sims)) and also recommended to some thesis students.

**Contents**

[1. Project management and project documentation](#_eycz4gpnwrps)

[Scrum-board and similar for distributed project teams](#_cwy04npa0hls)

[Issue tracking](#_n67iw3ogqn5q)

[Gantt chart](#_ror7po9u3ngc)

[Flow / blocks / UML / use case / process diagram](#_81hwjpsgsu7p)

[Requirements and Test Management](#_azzyj0mbgro)

[Gui mockups, prototypes and wireframes](#_kpgidwz5dgh8)

[Questionnaires](#_fhxdn6orj9px)

[Collaborative authoring](#_id2iy0sy9in9)

[Video conferences](#_4ykym1yoyuk1)

[2. Software](#_bzorzt2gj2a5)

["Repo" (Source code repository / hosting fascility)](#_l9weqeh1umr4)

[Versioning](#_s48e8fp9756n)

[Development Environments (IDEs)](#_deh1ou29dnjz)

[Advanced text editing appropriate for the program](#_dggdpq9ao4k6)

[Distributed code editing for pair programming / Extreme Programming mm](#_20db40sopvs6)

[Unit tests and automated tests](#_pd0bg7jfpzga)

[3. Analysis and Evaluation](#_xzzvtm6i0lrl)

[Bug Trackers](#_2s36hbx0ilcv)

[Performance Measurement and profiling](#_vrkacbsapmvq)

[Emulation of mobile devices](#_zfq3ssh0bs3d)

[Emulation of varying network performance](#_wdo1e1daq7k9)

# 1. Project management and project documentation

This section describes the tools that can be used both coding and non-coding system developers, for example, problem analysis, design and project management, high-level , close to users and customers, before and after the implementation phase of each partial delivery.

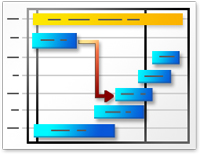
## Scrum-board and similar for distributed project teams

Usually whiteboard or folding cartons and postitlappar to show project status and to-do lists according to various agile system development methods such as Scrum board or Kanban interactive whiteboard. If the project team members working in different physical locations, you can instead use electronic tools:

* Scrum Desk
* Trello
* Microsoft Visual Studio Scrum process template
* [JIRA Agile](https://www.atlassian.com/software/jira/agile) plug - formerly called Green Grasshopper - Scrum Board for case management system Atlassian [JIRA](https://www.atlassian.com/software/jira/)

## Issue tracking

Case Management System (issue tracking) functions as electronic Backlog:

* Code-repositories such as Bitbucket and Github, see Chapter 2.
* Atlassian [JIRA](https://www.atlassian.com/software/jira/)

## Gantt chart

* Spreadsheet, see [projektmallar.se](http://www.projektmallar.se/gantt-schema)
* Microsoft Project
  + MIUN students can install this free of charge by the [Microsoft DreamSpark](http://www.miun.se/student/stod-och-service/it-helpdesk/programvaror/ny-sida/microsoft-dreamspark) offer.

## 

## Flow / blocks / UML / use case / process diagram

* Gnome Dia (open source)
* Violet UMLeditor (open source)
* draw.io
* Umbrello (open
* source),Microsoft Visio (supplement to Microsoft Office)
  + MIUN students can install Visio free of charge as part of [Microsoft's DreamSpark](http://www.miun.se/student/stod-och-service/it-helpdesk/programvaror/ny-sida/microsoft-dreamspark) deal.

## 

## Requirements and Test Management

Requirements Capture entails first identifying the user stories and use cases, ie such cases, among project stakeholders. Formulating the requirements specification (or target specification) based on the examples. Test Driven Design (TDD) means that the requirements are verified by tests and measurements before each installment, and the test results influence the design of subsequent installments. Often appoints the project a *test manager* responsible for this. Purchaser is responsible for developing the acceptance tests before delivery. Low-level developing automated unit tests of coding developers, before implementing the code. (See also the section on **Unit Tests** in Chapter 2, program.) Other tests can be done at a high level of non-coding developers, such as usability testing and performance measurements (see Chapter 3, **analysis and**evaluation).Acceptance testing is done by the customer / end user.

* [JIRA with test management](https://marketplace.atlassian.com/plugins/com.thed.zephyr.je)plugin,for example, [Zephyr for JIRA](http://www.getzephyr.com/products/test-management-add-ons-for-atlassian/zephyr-for-jira-server)
* Konsultbolag1 [ReQtest](http://reqtest.se/)

## Gui mockups, prototypes and wireframes

Usability testing can be performed early in the project on simplified mockups, protyper and wireframes of the system's planned user interface. Examples of tools:

* Pencil and Paper
* Presentation tools like Microsoft PowerPoint

Interactive wireframes

* [Balsamiq mockups](http://balsamiq.com/products/mockups/) (commercial product, available in demo version)
* [Marvel App](https://marvelapp.com/)
* [Invision App](http://www.invisionapp.com/)
* [Axure](http://www.axure.com/) (Commercial product)



## Questionnaires

Students often implement requirements capture and evaluation of the usefulness through surveys, such as using:

* [Google Drive](http://drive.google.com) - > Create form

## Collaborative authoring

Reports, manuals and other project documentation can be written by several students together with the help of various tools:

Cloud services

* [Overleaf](https://www.overleaf.com/)
  + For TeX / LaTeX documents, suitable for mathematical text, and research articles
* [Google Drive](http://drive.google.com) (formerly docs.google.com)
* Microsoft Teams/One Drive
  + Students can log in with their student portal ID
* [Dropbox](http://dropbox.com)
* [Owncloud](https://owncloud.com/)

Wikis

Wikis are often used for documentation of the open source project. Examples:

* Wiki hotels such <http://wikia.com>
* Install own wiki server, such as MediaWiki. See different [software](https://www.mediawiki.org/wiki/Software_bundles)bundles.

*Document Template*

* [Mittuniversitetets OpenOffice template](http://www.miun.se/universitetet/organisation/avdelningar/iks/examensarbete) for technical reports and theses reports

## Video conferences

Video Conferencing Tools used for student projects for meetings with business at another location or for meetings with distance students. Example:

* Google Hangout
* Skype
* Zoom
* Microsoft Teams

# 

# 2. Software development

Here are examples of tools that normally require expertise in programming, and mainly used in connection with the implementation phase of each installment.

## "Repo" (Source code repository / hosting fascility)

A "repo" is a website for source code escrow, originally intended for open source, but which can also be used for proprietary code. It can include many other tools, such as versioning, bug tracking, wiki documentation, Gant schedule, and more. Examples of sites:

* [Bitbucket](http://www.bitbucket.com)
* [Github](https://github.com/)
* Gitlab
* [Google Code](https://code.google.com)
* [SourceForge](http://sourceforge.net)

## Versioning

The following version control systems (source control or revision control system) can be installed on your own server, but is also included in some web-based "repositories" above. They are distributed to run on your own computer without the server.

* Git - distributed
* Apache Subversion (SVN) - centralized
* Mercurial - distributed
* TFS

## Development Environments (IDEs)

Many Integrated Development(IDE)Integrated Development Environment also includes tools for example versioning.

* Eclipse
* Android
* Studio,Microsoft Visual Studio
  + MIUN students can install it free as part of [Microsoft's DreamSpark](http://www.miun.se/student/stod-och-service/it-helpdesk/programvaror/ny-sida/microsoft-dreamspark)deal.
* Oracle Netbeans
* IntelliJ IDEA
* Matlab
* Anaconda (for Python and R in machine learning)

## Advanced text editing appropriate for the program

* Notepad ++
* Sublime Text
* Vim (can also be used as IDE)
* Emacs (can also be used as IDE)

## 

## Distributed code editing for pair programming / Extreme Programming mm

* Floobits
* [Saros](http://www.saros-project.org/)

## Unit tests and automated tests

development of automated unit tests (unit tests) are part of the test-driven design (TDD, see **Requirements and testhanterning** in chapter 1), but performed just before the implementation phase of coding developers. Based on requirements formulated automated tests for each feature or application component. Only then one begins to develop the production code, ie implement the features. When the production code has passed all test cases are implementation phase completed. With each subsequent modification, check that unit tests still passed. Complete unit testing systematically checks all the conditions which the function can be, including any fault conditions, such as identified based on state diagrams. Example-based tests instead means that only control some typical cases (use cases) of parameter values ​​that are so simple that you yourself can figure out funktionsanropets results, and often gives the example of the function's documentation. The end customer can design the automated acceptance tests by installments at a higher level. Examples of tools:

* JUnit (unit tests for Java)
* Cunit (unit testing for C / C ++)
* Jasmine (unit tests for JavaScript)
* Mocha (unit tests for JavaScript and NodeJS)
* QUnit (unit tests for JavaScript, xUnit compatible)
* Doctest (for Python, Matlab, with and more. Provides automatic testing based on the examples given in the documentation of the function.)
* [Robot Framework](http://robotframework.org/) to create automated acceptance tests, written in the scripting language Python.

# 

# 3. Analysis and Evaluation

After the implementation phase, an evaluation of each part when testing and verifying the project requirementsspecification and user needs. Especially in theses it is important to not only develop a system, but also evaluate the results of development in comparison with current research, existing solution or other solutions. Test method and tools are documented so that others can repeat the survey and check the results, and so the reader can assess how reliable and generalizable results. Test results should be presented in the form of charts, tables and lists, such as checklists that show what requirements are verified, and tables that compare the advantages and disadvantages of various options. The evaluation results are analyzed critically, trying to explain them, do impact analysis and provides a recommended solution. Based on the evaluation results suggest how the operation should be changed and what should be developed and explored in future installments or future projects. Here are examples of tools to produce evaluation results: (See also headings **requirement and test management** in Chapter 1 and **Unit Tests** in Chapter 2)

**Usability Testing**

There are a number of tools to record user operations and count the number of clicks and keystrokes, or take time on various operations, and heuristics (checklists, evaluation form) and then assess the user interface usability. Examples:

* Jira Capture - formerly called Bonfire - plugin CMS Jira

## Bug Trackers

Bug trackers are used to collect bug reports and suggestions for new functionality from test users, especially in open source projects, and acts as case management systems for following up the reports addressed . Examples:

* Bugzilla,
* Trac
* Redmine

## Performance Measurement and profiling

in order to optimize the application's response time and resource consumption and analyzed its scalability can use profiling tools to measure programkomponenters CPU time, memory usage and network utilization. Examples of tools:

* [Google Page Speed ​​tools](https://developers.google.com/speed/pagespeed/) can analyze web pages response
* time,Google Chrome can profile client side script via the menu "tools for programmers"
* Profiling tools included in Visual Studio Matlab development environments
* Plugins for profiling can be installed Netbean and Eclipse

## Emulation of mobile devices

Several tools can emulate different types of displays and devices:

* Google Chrome via the menu "tools for programmers"
* Android Studio Virtual Device Emulator

## Emulation of varying network performance

when analyzing the response time and sensitivity to packet loss when client-server applications, it is important to be able to simulate different nätverksaccesser (wifi / 3G / 4G / transatlantic etc.) with various round-trip-time, throughput, bit error rate, network dropouts and application dropouts. Examples of tools:

* Google Chrome via the menu "tools for programmers"
* Android Studio Virtual Device Emulator

**document contains contributions by Magnus Eriksson, Felix Dobslaw, Joel Halvarsson, Mehdi Tirgar Michael Hassel Ore, Daniel Bosk, Fredrik Håkansson and Johan**Moritz. **The document is available under the freedom of the** [**Creative Commons Attribution-ShareAlike**](http://creativecommons.org/licenses/by-sa/3.0/deed.en)**3.0.**