Progress report

Second implementation phase

PMoC

12.2.2002

Version 1.0

T-76.115 Progress report – Project planning phase PMoC

Version	Date	Author	Description
1.0	10.02.02	JF,JL	

Table of Contents

1. Project status

The second implementation phase was basically a good experience, and went over all quite well. Specially taking into account the nature of this particular project, concerning the big amount of things that *could* be implemented, I think that the group has performed quite well all though maybe not quite all features planned were yet totally implemented during this phase.

1.1 Overview

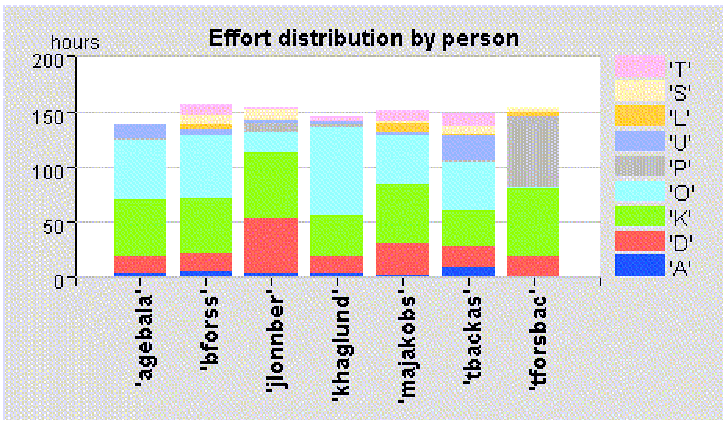

Figure 1

An inspection of the hours spent on the project per person (Figure 1) shows that the workload has been very well shared inside the group. The budgeted 140 hours total after phase 3 (I2-phase) has been slightly overrun by everyone.

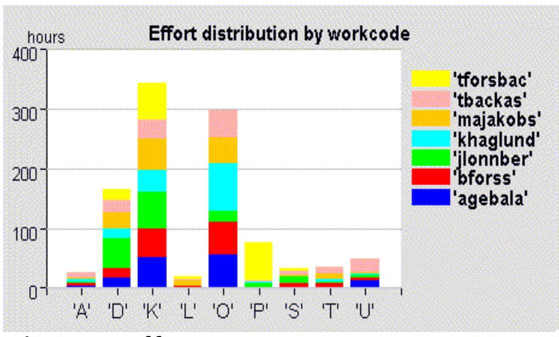

Figure 2 effort entire project

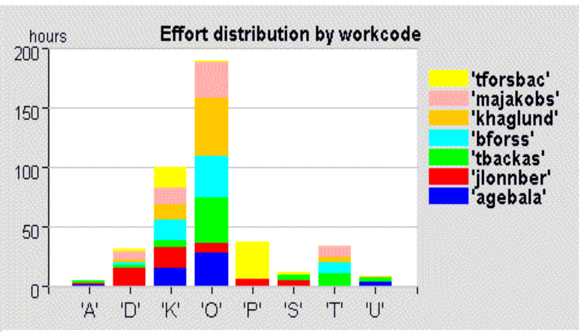

Figure 3 effort phase I2

From these two figures we can se that on the whole project (effort by work code Figure 2) most efforts have been spent on 1) meetings, 2) programming and 3) documentation. Secondly, as seen on the right (effort by work code I2-phase) during this phase most efforts have been spent on programming, then meetings, project management and documentation.

1.2 Accomplished tasks

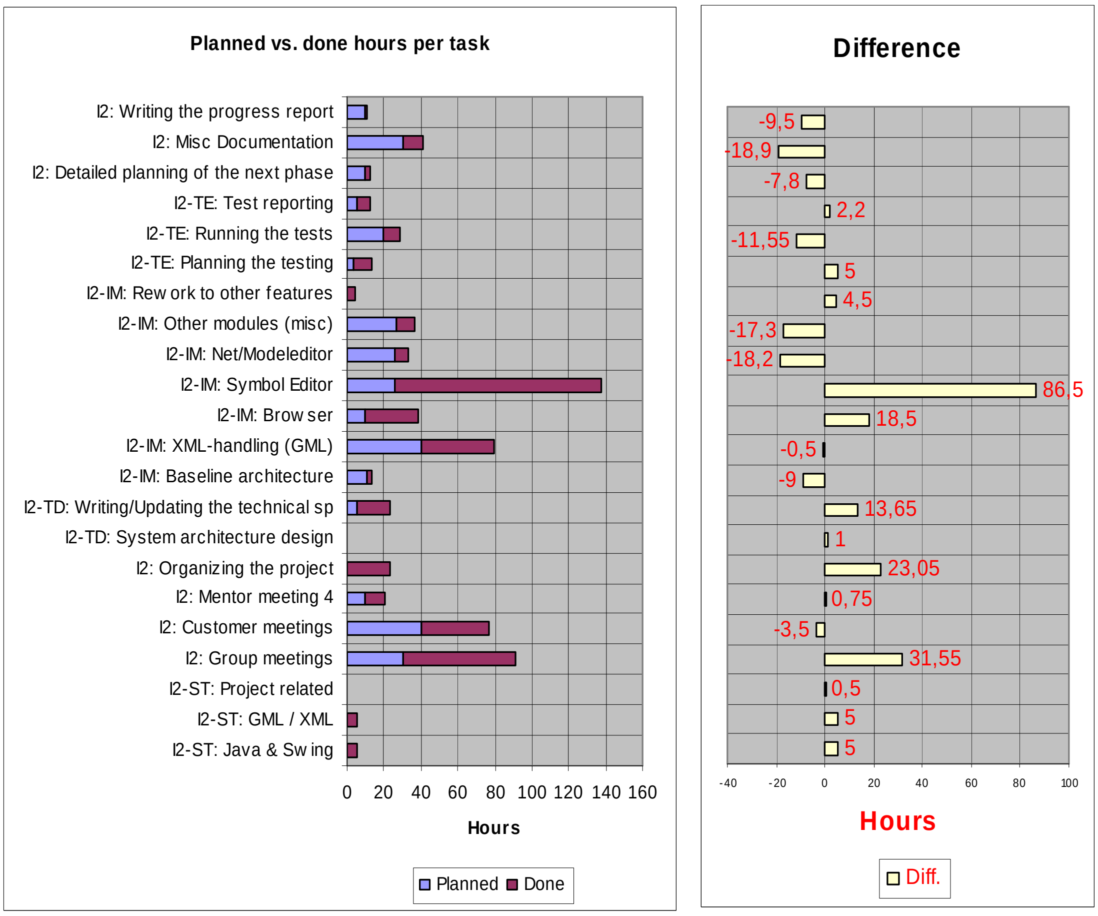

Figure 4 hours spent by task, and difference to the plans

It can easily be established from figure 4 that some tasks had been overestimated in effort, and others underestimated, some of them quite heavily.
The most underestimated tasks were:

- Symbol Editor – as can be seen, in stead the Net/Model editor didn't require as much effort as budgeted, this mostly because the Symbol- and Net-editor had very much in common, therefore also the mayor overrun of hours for the symbol editor.
- The tree browser also has required more effort than planned
- Organizing the project – as of this phase, it has become even more vital that the plans work out, and that everyone knows what to do next. More about this beneath on "Management and organizing".
- Group meetings – it is quite clear that the enormous amount of knowledge and thoughts that the customer has, has taken, and will take very much time. >Fr o m the point of view of our group it's both good and bad. On the other hand, we are mostly implementing a very far-thought-of idea with relatively long traditions (referring to the older software that is in use by the customer) that might be heavy lag, but on the other hand in the scope of such a project, it would have been impossible to reach even this far, if the group should have worked out all details.

2. Work performed

2.1. Organizing and planning

As mentioned above, this phase has required more on-site strategic organizing and planning than the phases before. Partly because this software as a whole is quite complex with many relatively difficult parts that depend on each other. Therefore it has become vitally important to organize very well in what order to implement the functionality agreed on. More details are fund in chapters 3.1 management and 3.2 risk management.

2.2. Implementation

This phase has definitely been the most implementation intense phase so far. At quite an early stage of this phase (early January) the group really realized how much there was to be done, and how important a good scoping would be. And all though most efforts have been put on implementation during this phase, I would say from a management perspective that there is *very* much yet to be done, for the software to feel complete enough. And the question arises if it at all is possible to implement such software, as defined by the customer, with 1400 man hours? Of course it is always possible to limit the scope even more, but at some point the software might not be considered a complete software product any more.

2.2.1 Implemented parts during this phase.[1]

- Further support for the GML structure
  - categories and departments
  - better module tests
  - range check of values
- The browser tree
  - gml-structures to be used through the tree view
  - handling, creation and removal of different components in the tree
- IO-interface
  - reading and parsing of departments, categories, components and a components several properties
- Graphics & gui
  - Many different tools and supporting functions for drawing, such as arc-tool, tool tips, zoom, grid, choose of properties after objects drawn, freehand tool, depth arrangement of objects, grouping functions, polyline tool
  - drag-and-drop functionality for instantiating components, and converting graphical symbols to component- or terminal symbols.
- SVG-handling
  - Parsing and handling of the SVG-documents as required for this project
- Canvas manager – a module for handling several SVG-documents open at the same time

[1] Many software- and project specific terms are used. For explanations of these are found in following documents 1) User Requirements © PMoC 2) Technical specification © PMoC 3) GML-specifications © VTT. These can be found under the project web-page from the forum (requires login which is freely available) www.yawc.net/pmoc

Additional implementation not originally in the plan

- Coordinator package
  - A new package not originally in the design, was needed for handling events and communication between the individual subparts of the software. This was successfully implemented, as far as possible during this phase.

2.2.2 Half done parts

- Tree browser
  - Many important functions are still lacking for handling the GML-tree in a proper way
  - Drag-and-drop within the tree.
- Coordinator package
  - Several parts must yet be implemented to support the functionality to be implemented during the next phase.
- IO-interface
  - Support for constraint values, vector values and various xml manipulation.
- Net editor
  - Terminals were implemented but connections between terminals didn't fit into the budgeted time limits.
  - Right-click controls are not finished, e.g. getting properties via the SVG-canvas for objects

It is somewhat difficult to exactly define which parts really were to be implemented during this phase, and which of them should be part of the next phase, because all the packages include such an big amount of functionality, and it is even difficult to tell exactly on beforehand what all functions there will exist. (E.g. the coordinator package evolved through some specific communication needs between modules)

But over all it is yet realistic to dear to expect a software as defined through out the customer meetings. And the implementation is very well at schedule apart from the parts mentioned in section 2.2.2.

2.3. Design

The design of the program has changed a lot during the last phase, mostly due to the customer's vision of the program becoming clearer. The most important aspects are:

- The I/O routines have been redesigned to be closer to the customer's existing software.
- The scope has been extended to include category and department objects.
- The user interface has been designed and specified down to the actual operations that it should include.
- The communication between the GML and SVG data models has been completely redesigned to handle the data communication required by the operations available to the user.
- The top-level classes (that instantiate the components for the various models) have been redesigned.

2.4. Documentation

The technical specification and user requirements documents have been updated to reflect the changes in design described above and the changes in required features necessitated by the design changes. In general, changes have been made to the technical specification first and then propagated to the use cases in the requirements document, as this matches the customer's way of expressing himself better.

The documents describing our software development practices have been updated to reflect our experiences from this phase. A test report[2] has been written that describes the results of the testing performed according to the test plan[3], which was revised to match the changes in design made during this phase. A preliminary draft of the user's guide has been written, although it is currently not very informative. As you no doubt have noticed, a progress report summarising the work done this phase has been written. The project plan has been revised to include plans for the next phase and peer testing as well as modifications to the large-scale plan to accommodate deviations from the plan.

3. Applied process

3.1. Project management

As mentioned in chapter 2.1 coordination of the things to be done has required more focus than before. Especially as not the entire group has attended all customer meetings – or group internal meetings, there has been a need of new ways to communicate what to do. Everything agreed upon during the meetings has been very well documented mostly in the Technical specifications document, but the threshold was at times rather high for all implementers to put the effort to read and get a clear understanding of what should be done, and in what order. One thing that probably made this worse was that documentation is in English only, but all communication is done in Swedish within the group, and because of that English documents might not very strongly be considered as part of the communication.
Therefore we began writing short weekly documents in Swedish (!!) of what *exactly* should be done, who should do it, and in what order – putting most important functions on the top of the list.
In this way it was very much easier for all implementers just to take a look at their own parts and start writing code. Then of course the technical specification was used as reference when more details were needed! See appendix A for an example of a weekly todo (Swedish).
The experience from this was very good, and implementers were happy to have a more concrete description of what to do. This practice had been used within the graphics group in some level earlier already.

3.2. Risk management

To some extent risk management has been done on a day to day basis, handling problems as needed. But definitely risk monitoring should be done more frequently than so far. The risk management plan has been put up to date, taking into account all

[2] Markus Jakobsson/PMoC: Test Report for I2
[3] Markus Jakobsson/PMoC: Test Plan

possible risks to date. Starting from next phase a more regular risk monitoring will be done, going through (and documenting when needed) current risks and their effect on the project. *Reference: Risk Management document.*

3.3. Use-cases

Reference: Use Cases document

3.4. Meeting practices

Reference: Communication and meeting practices document

3.5. Customer and mentor relations

Both the contact to the customer and the mentor has worked very well just as before. The fact that the picture of the software broadens every customer meeting has lead to a constant need of new meetings, but a good sign has been that the need of meetings has been less frequent during the second half of this phase.
The mentor has fulfilled his task to our full satisfaction, and has been willing to give helping advice always when needed.

4. Improvement

Some points that has become evident still to need improvement

- Even more clear structuring and organizing of what to do and when – we will develop the weekly todos.
- There has been more need of access to what has been decided on during meetings, as not all group members have been participating to all meetings, possibly PMs from the meetings will be put on the web frequently.
- Still very much time has been put on meetings. Through better notes and distribution of them optimizing of members needed to participate in meetings will be done. In the time planning the time for meetings has been cut to half. This shouldn’t be too unrealistic.

Appendix A – Example of Todo

```
ToDo till tisdag 4.2.2003!!! -- Don't choke ;-)
Author: Johan (---.win.hut.fi)
Date:   01-28-03 19:09

Todo - Absolut DL tisdag 4.2

Innehålls förteckning:
- Browser
- Interaktion
*Interaktion till SVG-canvas (initierat av trädet) (janne speksar ti torsdag)
*Interaktion från SVG-canvas (janne speksar till torsdag 30.1)
*Interaktion från flikhanterare - Tab manager (coordinator paketet)
*Implementationsfas 3
- Janne att speksa
- Att implementera på grafiksidan
*SymbolEditor
*NetEditor
- IO

Browser
=======
Läs TechSpec 3.6.2

Absolut måste före DL 4.2
------------------------------------
* Inladdning och visande i trädet av en avdelning (department) och dess innehåll
- Avdelningar, kategorier, komponenter, komponenttyper är de nödvändigaste att kunna
visas!
- Editering av komponentens barn (dvs properties) kan vid behov skjutas upp en aning
till början av nästa fas.
- Men komponentens barn-komponenter bör naturligtvis kunna browsas!
* Skapandet av nya kategorier
- genom högerklick på kategori eller avdelning (enligt TechSpec 3.6.2)
* Instantiering ("skapande") av komponenter (internt på gml-sidan)
- Enligt drag-and-drop, så att man draggar komponenttypen till den kategori eller
komponent som den skall bli ett barnelement av
* Import av komponenttyper
- Måste trots allt implementeras för att användaren måste kunna bestämma VART (till
vilken kategori) komponenettyperna skall

Ej absolut nödvändigt för denna fas - men för nästa :-)
-----------------------------------------------------------------------------
* Editering av properties
- tills vidare enbart i trädet (bra att kunna ändras på för syns skull, fast det inte
ännu skulle sparas konkret till HD:n)
* Borttagning av element från trädet, som är tillåtna att ta bort... (=-]
* Editering av editeringsmöjliga attribut i trädet (Specs 3.6.2)


Interaktion
================
Läs TechSpec kap 4.4

Interaktion till SVG-canvas (initierat av trädet) (janne speksar till torsdag 30.1)
------------------------------------------
* Drag-and-drop instantiering av komponenter från trädet till SVG-canvasen
* Öppna nät eller symbol på canvasen genom dubbelklick av passligt element
(komponenter iaf.) i trädet
* Välja komponent i trädet => väljs också på canvasen (genom högerklick/select-on-net)
* Delete av komponent informeras till canvasen så att den tas bort därifrån också

Interaktion från SVG-canvas (janne speksar till torsdag 30.1)
---------------------------------
* När connection görs/editeras i neteditorn kallas på lämpliga metoder för att
registrera detta i gml-trädet
* Emottagligehet i komponenttypsnoden för en symbol, som då blir denna komponenttyps
symbol
- Dvs drag-and-drop från symbolens editors TAB (flik) till en komponenttypsnod
* Delete av komponenter eller connections återspeglas i gml-trädet.
* Välja något på canvasen => väljs också i trädet (genom högerklick/select-in-tree)
```

* Dubbel-klick av component på canvasen som har subkomponenter öppnar ifrågavarande komponents canvas.
- går till flikhanteraren som skall skapa en ny flik för nya "nätet" eller symbolen att öppna

Interaktion från flikhanterare - Tab manager (coordinator paketet)
----
* Öppning av symbol då dubbelklick i trädet
- öppnar ny flik "till höger"
- flikhanteraren instantierar grafikeditor
* Koppling av fri symbol till komponenttyp (TechSpech 3.2 & 4.4)
- genom drag-and-drop från symboleditorns TAB (flik) till den komponenttyp som den skall kopplas till

Implementationsfas 3
----
* Utplacering av plain symbol från trädet till canvasen
* Radering av symbol från trädet då det görs på canvasen och vice versa.

Janne att speksa
=============
* Symbol & komponenttyps attachment
* Drag-and-drop instantiering av komponenttyper till svg-canvasen
* Specifiera metoder och funktioner så att all nödvändig interaktion mellan delar (Tree, SymbolEditor, NetEditor) fungerar

Att implementera på grafiksidan
====================

SymbolEditor
==========
* Konvertering av s.k. fria symboler till terminalsymboler (TechSpec 3.2)
* Utplacering av terminaler på komponenttyps symbol

NetEditor
=======
* Connections
* Utplacering av fria- / komponentsymboler
- dvs. emottagligehet för drag-and-drop från trädet

IO
==
Väsentliga delar i TechSpecen som måste läsas för att förstås: 4.1, 4.7
Till detta kommer ännu lite speks som ja bett om att få av Jyrki P. Skickar det åt Tomas snarast möjligt.

Absolut måste före DL 4.2
----
* Komponenttyps laddningen slutförs. TechSpec (kap 4.1 & 4.7 LÄS!)
- Laddas in från spikad katalog, enligt uuid-namn.gml
* Komponentladdning TechSpec (kap 4.1 & 4.7 LÄS!)
- Laddas från samma katalog om komponenttyperna
* Kontrol av vilka barn ett element (komponent, kategori eller avdelning) har (TechSpec 4.7)
- Dvs. getTree-metoden i IO-handler
* Inladdning och sparning av SVG-filer
- getElement och setElement metoderna.
* Laddning av kategorier och avdelningar
- kategorierna och avdelningarna läses från filer liknande som man gör med komponenterna

Ej absolut nödvändigt för denna fas
----
* Sparning av komponenter TechSpec (kap 4.7 LÄS!)
- Sparas till samma spikade katalog
* Sparninga av kategorier och avdelningar
* Bortagning av element (komponenter, kategorier och komponenttyper)
- removeElement-metoden i IO-handler