Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Marcel Kyas, Gunnar Schaefer Martin Steffen


# P-I-T-M: Coma

WS 2004/05

# Handout 5

16. November 2004 (prerelease 15.11)

**Handout5: Group architecture + next steps**

**Ausgabetermin: 16. November 2004 (prerelease 15.11)**

**Abstract**

This handout serve to prepare the decision finding, in particular concering the "group-architecture" for the rest of the semester. We propose a certain group-structure as base for discussion, and give arguments and motivations about that structure. Consider the structure and think whether it makes sense from your perspective. If not, give arguments/alternatives.

## Where are we?

What's needed now is to keep on rolling (or for some to get rolling ...). The *first phase* is *finished* which gave some clarification about the following points

- how many people still on board (21?)
- what tools we choose.
- some more insight on the specification, in particular the data model.

As a further comment on what happened so far is, that the spec-groups had the tougher task as compared to the tools groups.

### Status (Monday evening)

**org:** Snert is up and running (apache, tomcat, buzilla, email-exploder, svn, passwords are handed out); some proposal for the next steps is worked out

**tools:** we have statements about tools etc from the tools groups (in written form only by Alexander Derenbach). So far (Monday evening), the tools groups have not yet tested their proposals on snert; they also don't have logins.

**spec:** we have one (preliminary) specs by today.

**test:** we probably got one proposal from the test-groups

### Evaluation

As a *critical review* I'd say:

- The *specifications* are helpful and necessary (as far as we have seen so far). The fact that we have *two* is helpful insofar as we could see ideas and get inspiration and discussion from both. As we have seen, the work that pushed the semester ahead so far was the spec work, so having only one spec group would have left more people "idle". Furthermore, it forced the developers to *think* and *discuss* the tool.
- Concerning the tools-specification, the success and the usefulness (in the context of the project) is less transparent. This has various reasons
  - the task was perhaps more fuzzy
  - it's difficult to *see* the success. If the thinking about the platform s allowed us to avoid a definite wrong choice, even then the success is not visible (because we never know what would have been).
  - Another important goal in this phase was, that the people proposing the tools *are* or *become* experts in using/installing/maintaining the tools. From the organizer's perspective, we don't know to which extent this has happened. Also this depends on how good the "deliverable" of "Getting started with Tool xxx" etc will be.

  On the other hand: it's easy to say: "why haven't we decided this at the beginning" because in hindsight each decision could have taken immediatly (if one had known in advance what it was...). For the future: this phase should be faster, nonetheless.

## Group architecture

To keep rolling it is important to get something productive (and useful) to do for everyone for the weeks to come (in particular already the next week). Things to be *decided* now is the group architecture. In Figure 1, we make some proposal. The key points are

- 3 local modules, which work in isolation i.e., in parallel. Each group consists of *6* persons. Considering the general preferences, perhaps 2 "PHP" groups and one "Java" group will be formed.
- 3 global modules, which are relevant for all particpants. Those are
  - testing: *3 persons*
  - data base:
  - requirements/specification

Local means that one needs the code of one of the local groups for the full tool. For instance, the configuration indicated in Figure 1 by the dotted blob consists of the local implementation of *group 1* together with the 3 global (or shared) modules.

Before we mention concrete tasks within this proposal in a bit more detail, we first clarify the (partially conflicting) design goals of our proposal.

**programming in the (not too) many:** We are, according to our statistics, currently *21 people*. All programming in the same group will lead to chaos, in particular it will conflict with the goal that (more or less) all participants will contribute in the next phase. Furthermore, the task at hand is too small for task, so in order to make this work and avoid too many complicated interfaces, there would be a few "real groups" and the rest would do the doc, the manual, etc., which would be boring.

We propose therefore *3 parallel, local groups*. They are small enough that internal communication is still possible and that one can split the task in interesting enough subparts without creating subtasks just because we need to have a task for everyone. Furthermore, the parallel work might inspire competition

**preferences** We have seen, that the course is split into "PHP vs. Java" fractions. The split allows that (more or less) everyone gets what he likes.

**one spec.** We propose, as the course is intended as a *common* programming project, that we do not split it cleanly into completely independent working groups. Three things should remain common and shared, which are

- testing
- requirements/docu, and related to that,
- the concrete data representation in SQL

**constant progress** Having finished (more or less) the work on spec/tools by *today,* it is not adivsable to make further specification week(s) concerning the data-model, because this

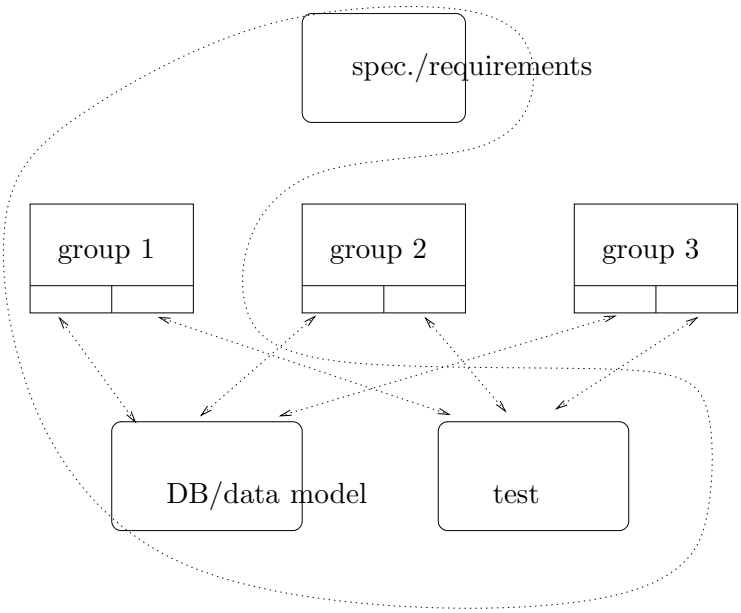


Figure 1: Groups

The goals are conflicting, and the architecture is a compromise between goals mentioned above. A critical point are (as always) the shared modules, and in our case the data base. The

reason is that everything else depends on that.[1] Since a common tool seems a worthwile and interesting goal, we want to stick to that even if it will cost time and effort.[2] It's less critical for testing and the requirements, obviously. Nevertheless, built on the work of the spec-group, we need to hammer out a unified data model concretely (down to SQL-expressions) *soon-ish.* This should doable quickly, and not much discussion should be needed there.

### Structure of each group

We propose therefore the following structure and responsibilities for each group. So currently this still some sort of "special assignment", since we are not yet coding the final product.

**DB:** **task** *one person* of group 1 – 3 does a realization of the (agreed/common) data model in SQL merge them into one specific data-model.

**deadline:** 21?

**deliverable:** specification + SQL-code

**profile:** preferably an SQL-specialist (or someone who wants to learn something new *very fast*...)

**data model:**

**profile** that's us, the authors of the handout, we might need some clarification from you spec people, however

**task** merge the spec. delivs into one common and approved specification

**deadline** 18.11 (that's damn soon ...)

**deliverable:** some textual format (or graphical, let's see.) It should be precise enough that the SQL-person can do their work.

**architecture:** the rest of the persons of groups 1 –3

**task:** depending on your choice of tools: work out a modularization of the tools into submodules

**deliverable:** presentation + written description

**deadline:** 23.11

**test:** So far, haven't found a task for those. Lucky again ...

[1] Note that the picture is not intended to mean, that all tools *run* concurrently on the same instance of the data base.

[2] Some software people claim also that interfaces tend to get better by the pure fact that there's more than one client ...