Project Management

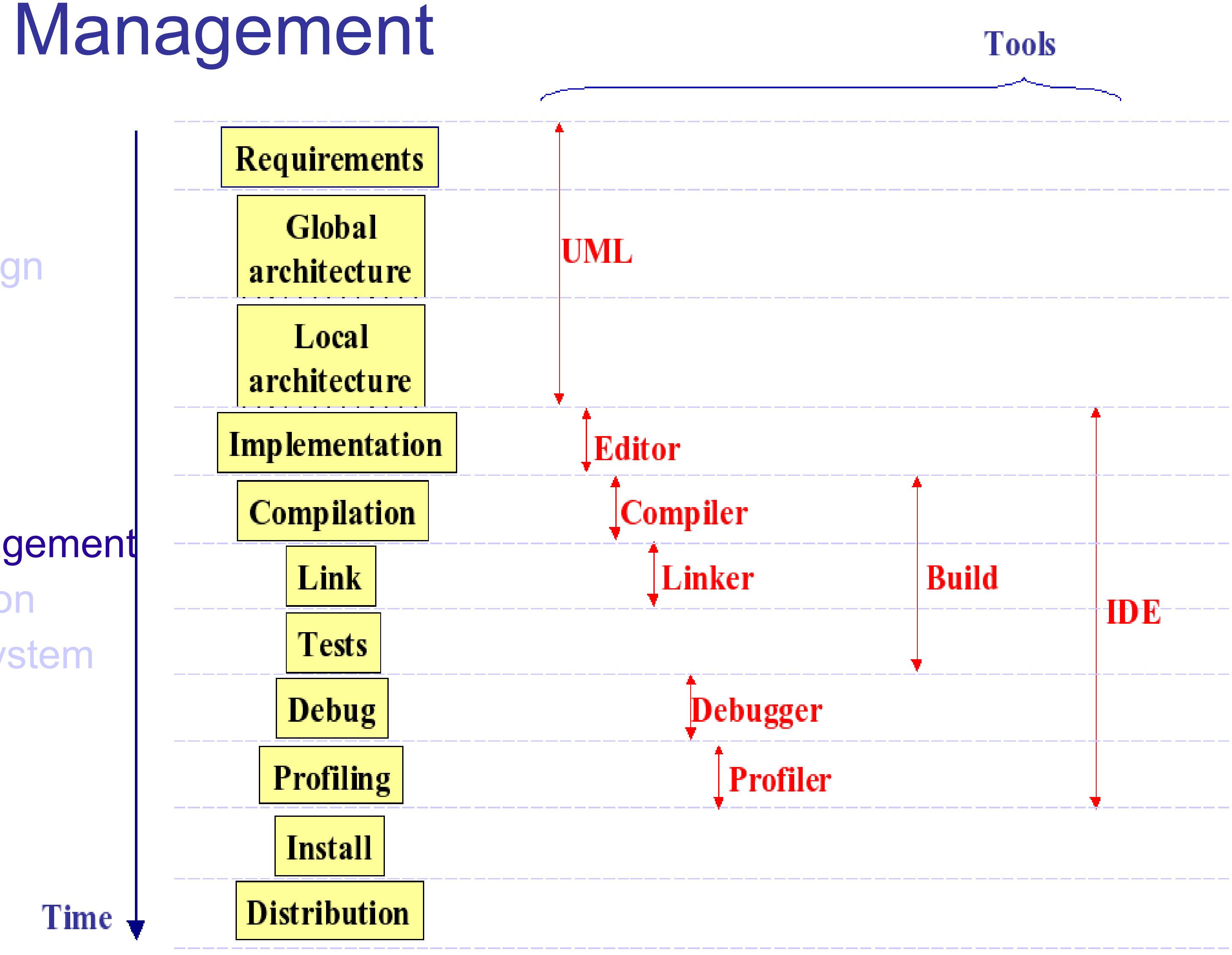
Nicolas Niclausse - DREAM

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
SOPHIA ANTIPOLIS

Project Management

- Introduction
- Analysis/design
- Build
- Tests
- Debug
- Profiling
- Project management
- Documentation
- Versioning system
- IDE
- GForge
- Conclusion



Tools

Requirements

Global architecture

Local architecture

Implementation

Compilation

Link

Tests

Debug

Profiling

Install

Distribution

Time

UML

Editor

Compiler

Linker

Build

IDE

Debugger

Profiler

Outline

- Project management, planning
- Example
- Software development management
- Links

Project management

- What is it ?
  - **Project** : set of actions to do to reach a specific goal within a given time.
  - **Project management**: set of rules to organize resources in such a way that the goals are reached within defined time, cost and quality constraints.
- What is it used for ?
  - To be more efficient

Methods:

- Planning
  - What to be done ? (must split the work in phases and subtasks)
  - By who ? (resources)
  - How much time does each task take ?
  - What priorities and dependencies between the tasks ?
- Accounting
  - Follow the time spent on tasks
- Validation
  - End of phase = deliverable to be validated

Planning

WBS: Work Breakdown Structure

- Based on “Divide and conquer” strategy
- Decompose project in phases, phases in tasks, tasks in sub-tasks ...
- Don’t go too far (in general, task ≠ implementation of a single function)
- Don’t forget or minimize important tasks: e.g. Documentation
- Tasks can be decompose in subtasks later in the project (iterative planning: e.g. after the specifications has been finished)

How to estimate the duration of a task ?

Hofstadter's Law states that:

> *It always takes longer than you expect, even when you take into account Hofstadter's law.*

- Ask “experts”
- Experience helps: feedback based on previous projects
- Analogy
- Use intervals (min/max)
- For development, debugging takes a lot of time !

Example 1/6

Small project :

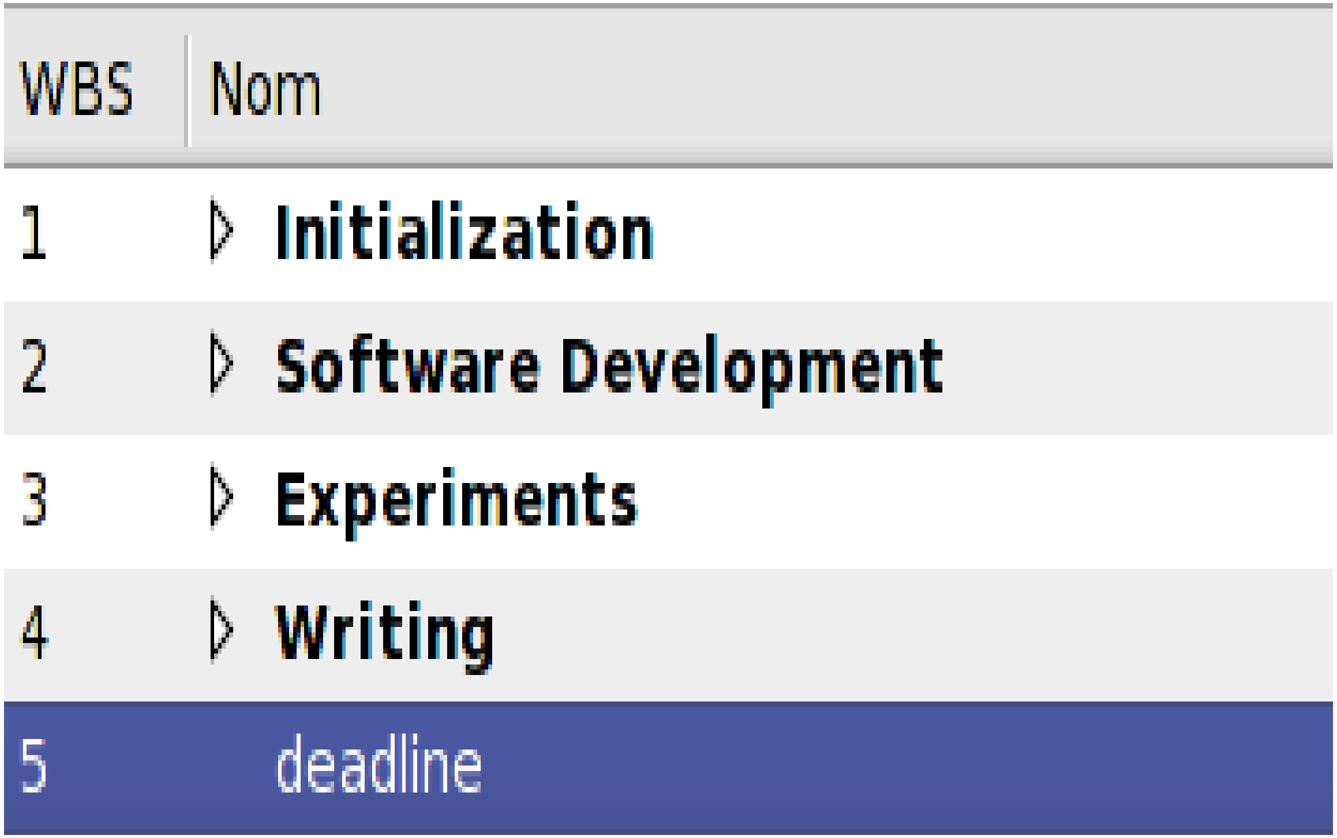
- *write a library for an existing software to test an new algorithm and write a paper for a conference.*

Resources:

- PhD student
- Intern
- Research advisor

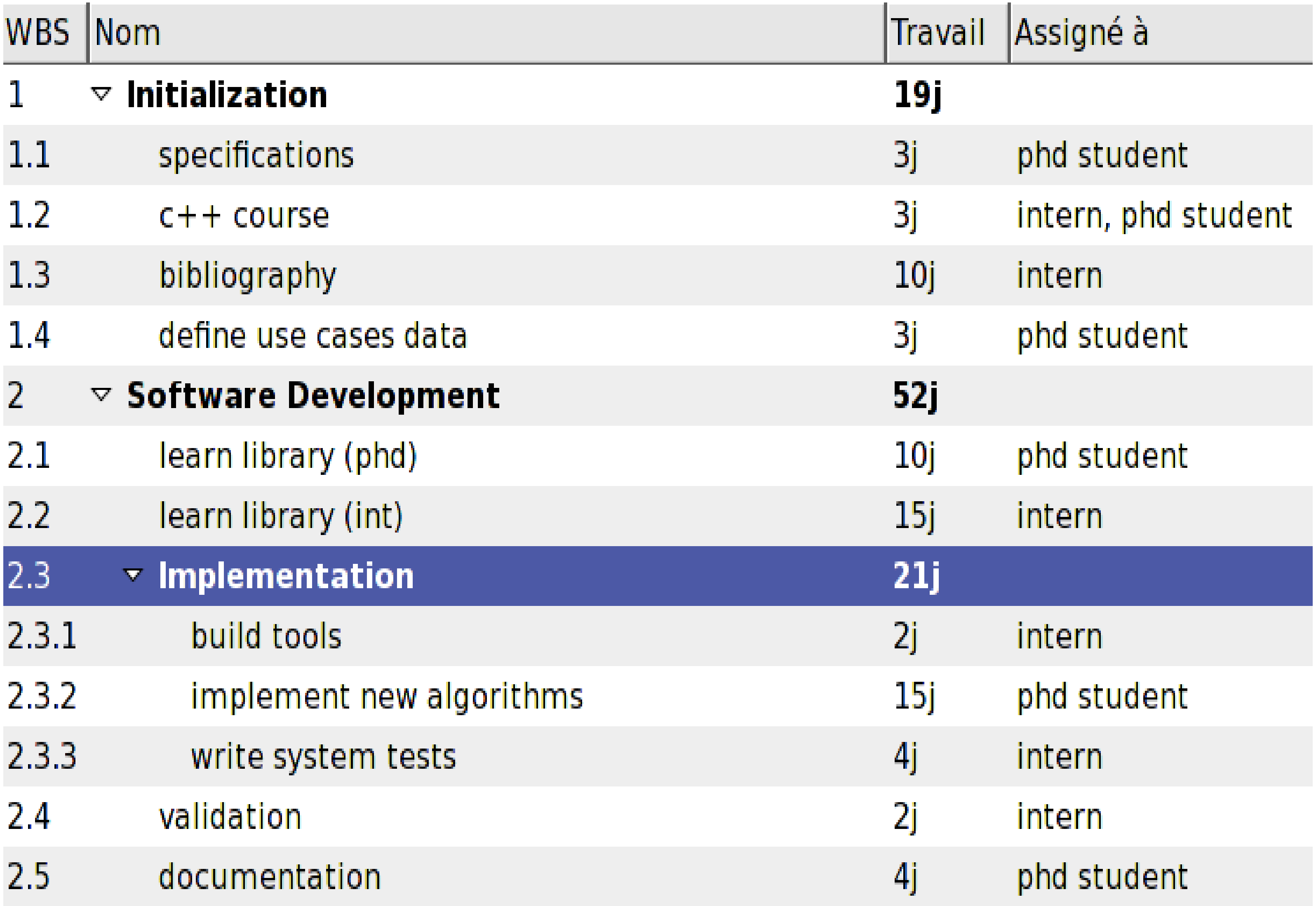
Example 2/6

WBS: define phases and deadlines:



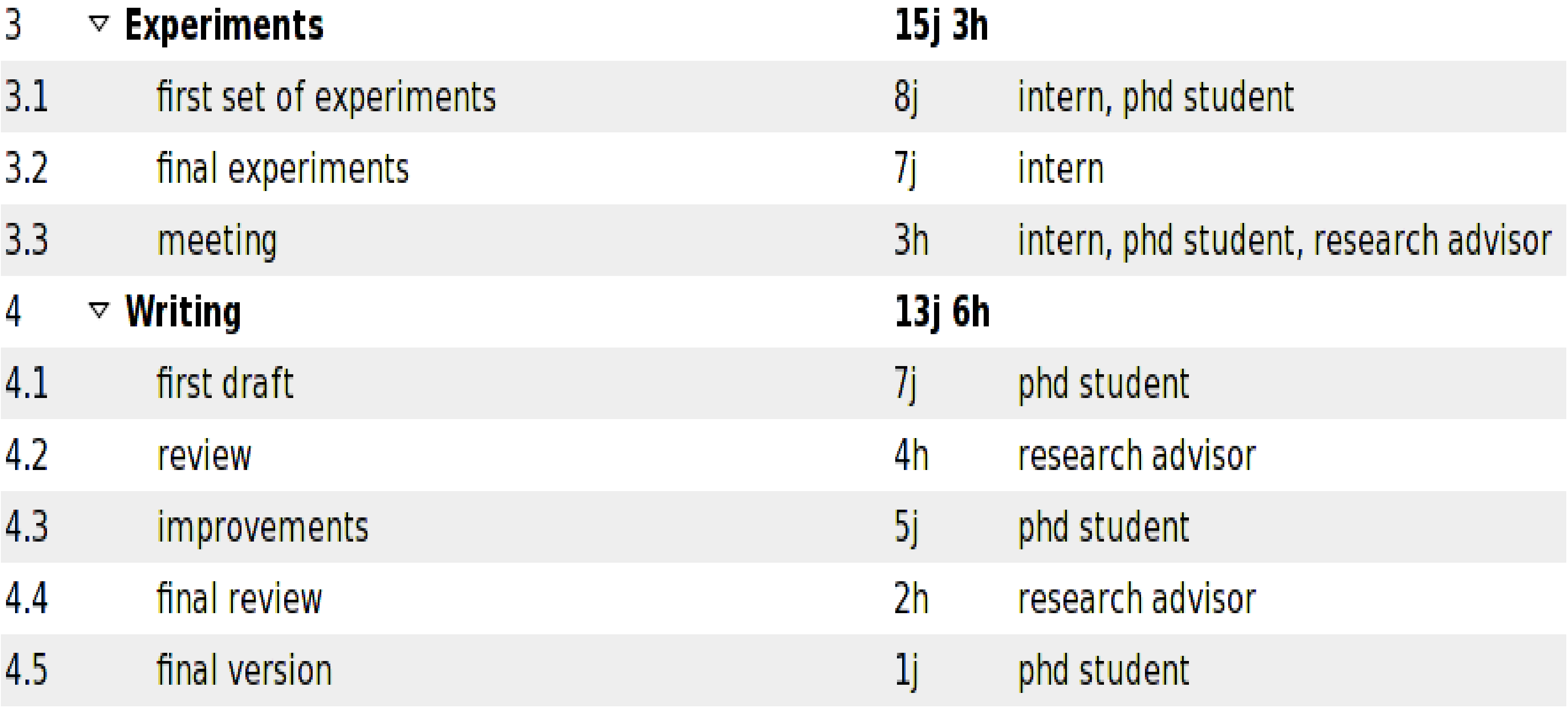
WBS	Nom
1	**Initialization**
2	**Software Development**
3	**Experiments**
4	**Writing**
5	deadline

Example 3/6: then tasks+duration+resources



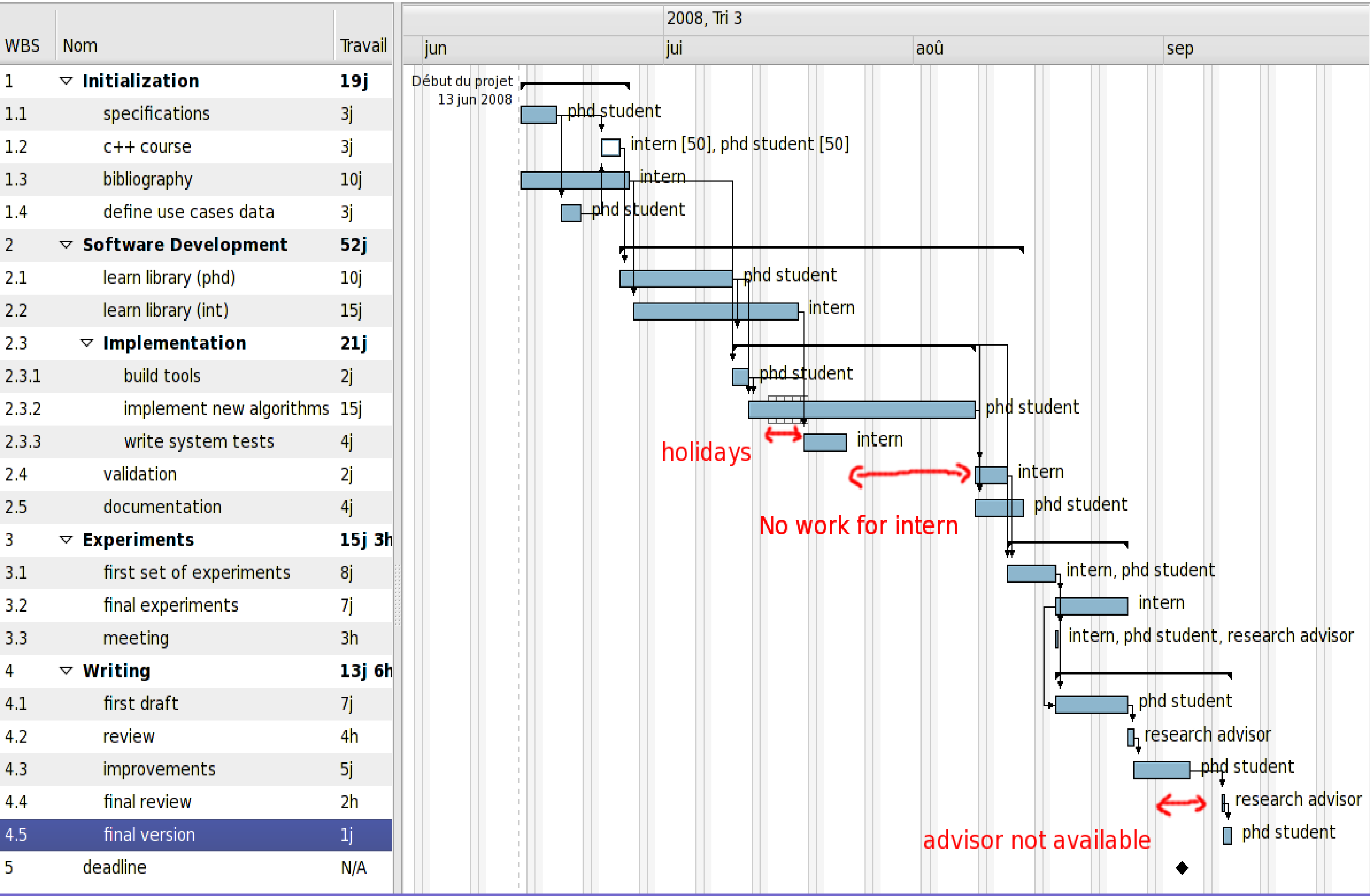
WBS	Nom	Travail	Assigné à
1	▽ **Initialization**	**19j**	
1.1	specifications	3j	phd student
1.2	c++ course	3j	intern, phd student
1.3	bibliography	10j	intern
1.4	define use cases data	3j	phd student
2	▽ **Software Development**	**52j**	
2.1	learn library (phd)	10j	phd student
2.2	learn library (int)	15j	intern
2.3	▽ **Implementation**	**21j**	
2.3.1	build tools	2j	intern
2.3.2	implement new algorithms	15j	phd student
2.3.3	write system tests	4j	intern
2.4	validation	2j	intern
2.5	documentation	4j	phd student

Example 4/6: tasks + duration + resources



3	▽ **Experiments**	**15j 3h**	
3.1	first set of experiments	8j	intern, phd student
3.2	final experiments	7j	intern
3.3	meeting	3h	intern, phd student, research advisor
4	▽ **Writing**	**13j 6h**	
4.1	first draft	7j	phd student
4.2	review	4h	research advisor
4.3	improvements	5j	phd student
4.4	final review	2h	research advisor
4.5	final version	1j	phd student

Example 5/6: add dependencies =Gantt chart



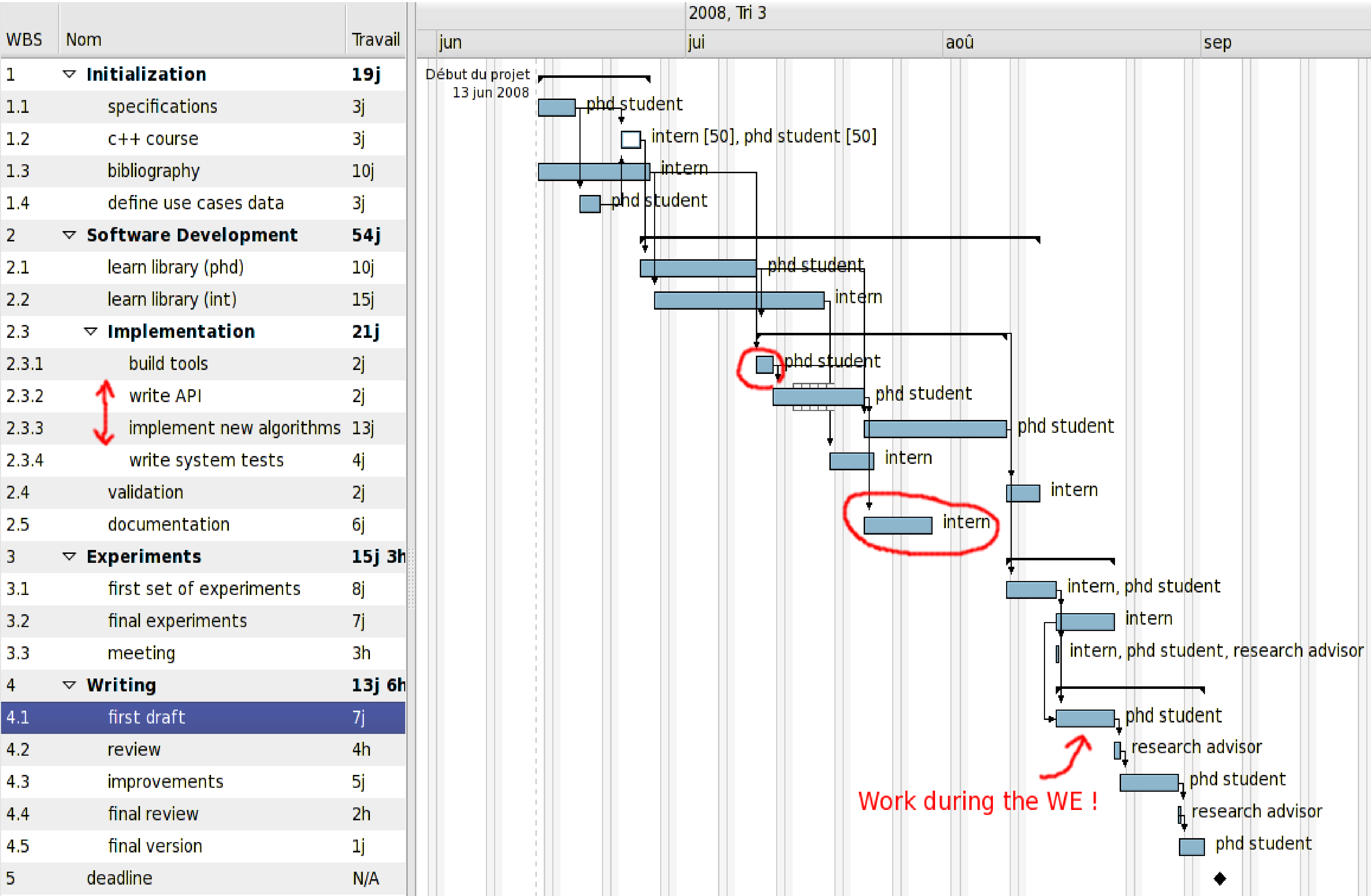
WBS	Nom	Travail
1	**Initialization**	**19j**
1.1	specifications	3j
1.2	c++ course	3j
1.3	bibliography	10j
1.4	define use cases data	3j
2	**Software Development**	**52j**
2.1	learn library (phd)	10j
2.2	learn library (int)	15j
2.3	**Implementation**	**21j**
2.3.1	build tools	2j
2.3.2	implement new algorithms	15j
2.3.3	write system tests	4j
2.4	validation	2j
2.5	documentation	4j
3	**Experiments**	**15j 3h**
3.1	first set of experiments	8j
3.2	final experiments	7j
3.3	meeting	3h
4	**Writing**	**13j 6h**
4.1	first draft	7j
4.2	review	4h
4.3	improvements	5j
4.4	final review	2h
4.5	final version	1j
5	deadline	N/A

holidays

No work for intern

advisor not available

Example 6/6: planning update



WBS	Nom	Travail
1	**Initialization**	**19j**
1.1	specifications	3j
1.2	c++ course	3j
1.3	bibliography	10j
1.4	define use cases data	3j
2	**Software Development**	**54j**
2.1	learn library (phd)	10j
2.2	learn library (int)	15j
2.3	**Implementation**	**21j**
2.3.1	build tools	2j
2.3.2	write API	2j
2.3.3	implement new algorithms	13j
2.3.4	write system tests	4j
2.4	validation	2j
2.5	documentation	6j
3	**Experiments**	**15j 3h**
3.1	first set of experiments	8j
3.2	final experiments	7j
3.3	meeting	3h
4	**Writing**	**13j 6h**
4.1	first draft	7j
4.2	review	4h
4.3	improvements	5j
4.4	final review	2h
4.5	final version	1j
5	deadline	N/A

Work during the WE !

Tips:

- Do not hesitate to update the planning during the project
  - Planning is a tool, not a constraint
  - after each phase at least
- At the end of the project: debriefing
  - Compare the first planning to the effective planning
  - If it’s very different, try to understand why (bad estimation ? Forgotten tasks ? Availability of a resource ?)

Tools

Project management software:

- Manage projects with tasks: each task has a duration, use some resources and can depend on other tasks.
- Resource usage during the project, gantt chart

Examples :

- MS Project (Windows, cost $$). GUI, feature-rich.
- Open Workbench (Windows, opensource). GUI, feature-rich (server cost $ $)
- Planner (Linux-GTK, free software). GUI, simple (no resource leveling).
- Ganttproject (java, free software). GUI, simple (no resource leveling)
- Taskjuggler (Linux, free software). Text, rich (res. Leveling ...).
- OpenProj (MultiOS, opensource).

Software development management

For development projects involving several person, e.g. :

- Team of several developers
- Single developer, but long lived software (several generations of coders)

Planning is useful but, you also need methods specific to software development management

Good practices

- Specifications:
  - Simple design (iterative design)
  - Use Cases
- Coding standards
- System Testing
  - A test for each Use Case
- Unit testing (“test-driven development”)
- Iterative design + unit tests → Refactoring
- Iterative schedule
- Code review
- Source Code Management, automatic builds

Software development project management

Several methods exist:

- RUP (**U**nified **P**rocess)
  - Related to UML
- Agiles methods (e.g. XP)
  - Pragmatic
  - Reactive (regular adaptation to changing circumstances)
  - Emphasis on software that works rather than complete documentation
  - Client feedback during the development
  - Emphasis on team rather than tools
- Merise
- ...

XP = e**X**treme **P**rogramming

- Frequent releases
- Iterative schedule
- Client on site (he helps to define use cases)
- Sustainable pace
- Simple design
- Refactoring
- Functional Testing
- Unit testing (“test-driven development”)
- Collective code ownership
- Pair programming (code review)
- Coding standards
- System metaphor
- Continuous integration (automatic builds ...)

Links

Évaluation comparative de solutions opensource de gestion de projet:

- http://www-sop.inria.fr/dream/rapports/eval-infoglobe.pdf

Un processus de développement logiciel pour l'INRIA, section Gestion de projet

- http://www-sop.inria.fr/dream/rapports/devprocess/main006.html

Agile software development methods

- http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf

K. Beck. Extreme programming explained: embrace changes.

M.Fowler. *Refactoring. Improving the Design of Existing Code.*

Conclusion

Methodology:

- Planning
- Write the specifications (with several iterations if necessary)
- *Use Cases* and Functional tests to validate them
- Coding standards, Source Code Management, ...

Benefits:

- Better management of deadlines (the code must work before a conference...)
- More efficient team workings (several people at once or reuse of your work after you've left)
- You know where you are, what has to be done, and what has already been done
- Increased quality