TIME

Inroduction

- Time is addressed differently by different people and cultures;
- for example, in western culture, time is mainly associated with financial profit, i.e., ''Time is money.''
- Time plays a special role in software engineering: the project schedule should be met, the product should be delivered on time, teammates should complete their tasks on time, and so on.

List of Time-Related Problems in Software Projects

1. Bottlenecks: In software development occur when one or more functions in the process await the output of another function in the process, with teammates having nothing to work on in the meantime.

- Eg. when developers wait for artifacts from system analysts, like the specification of a specific module.

List of Time-Related Problems

2. Project planning and schedule. Two main problems are associated with schedules, which are, in fact, closely connected. The first is the mere construction of a feasible project schedule. The second problem is to meet the schedule that has been set.

List of Time-Related Problems

3. Time estimation. There are different ways to support time estimation

- (Boehm 1981, Boehm et al. 2000, Kemerer 1987, SEI 2001). With respect to the estimation of the development time of a specific module/class/task, it is well known that the greater the module/class/task is, the more difficult it is to estimate its development time.
- Tomayko and Hazzan (2004) present evidence that the smaller the estimated unit is, the more accurate is its time estimation.

List of Time-Related Problems

4. Time pressure. Time pressure is the result of the previous problems. It happens usually toward the end of the development process, when teammates cannot meet the project schedule, either because of poor time estimations or bottlenecks.

- Time pressure usually leads to the skipping of different testing activities, which in turn leads to a decrease in software quality.

List of Time-Related Problems

5. Late delivery. Late deliveries occur as a result of inappropriate project planning, usually due to poor estimations.

- Data indicate that the percentage of software projects that fail to accomplish on-time delivery is quite high.
- See, for example, the data presented by the Standish Group Report (Standish 1994).

Time

- Case Study 4.1. Software Organizational Survey from the Time Perspective

Tightness of Software Development Methods

- Project plan dimension: number of releases and feedback milestones, level of emphasis placed on planning, number of days for which specific planning is made (the smaller the number of days, the tighter the software development method).
- Procedures and standards dimension: level of detail that describes the software development method.
- Responsibility and accountability dimension: level of role performance, level of personal accountability, frequency at which team members are required to report on their progress.
- Time estimation dimension: importance given to time estimation, resolution level of time estimation (hours, days, months)|the smaller the time units, the higher the resolution, and the higher the value of this dimension.
- Individual need satisfaction dimension: mutual dependency of team members, level of planning that inspires the message ‘‘Invest now for the future.’’

Sustainable Pace

- Tightness is one time-related characteristic of agile software development. Sustainable pace is another one.
- Sustainable pace means that the development process is carried out in a reasonable number of hours, which are well planned and enable the team to be productive and to produce quality products

Case Study 4.2. An Iteration Timetable of an Agile Team

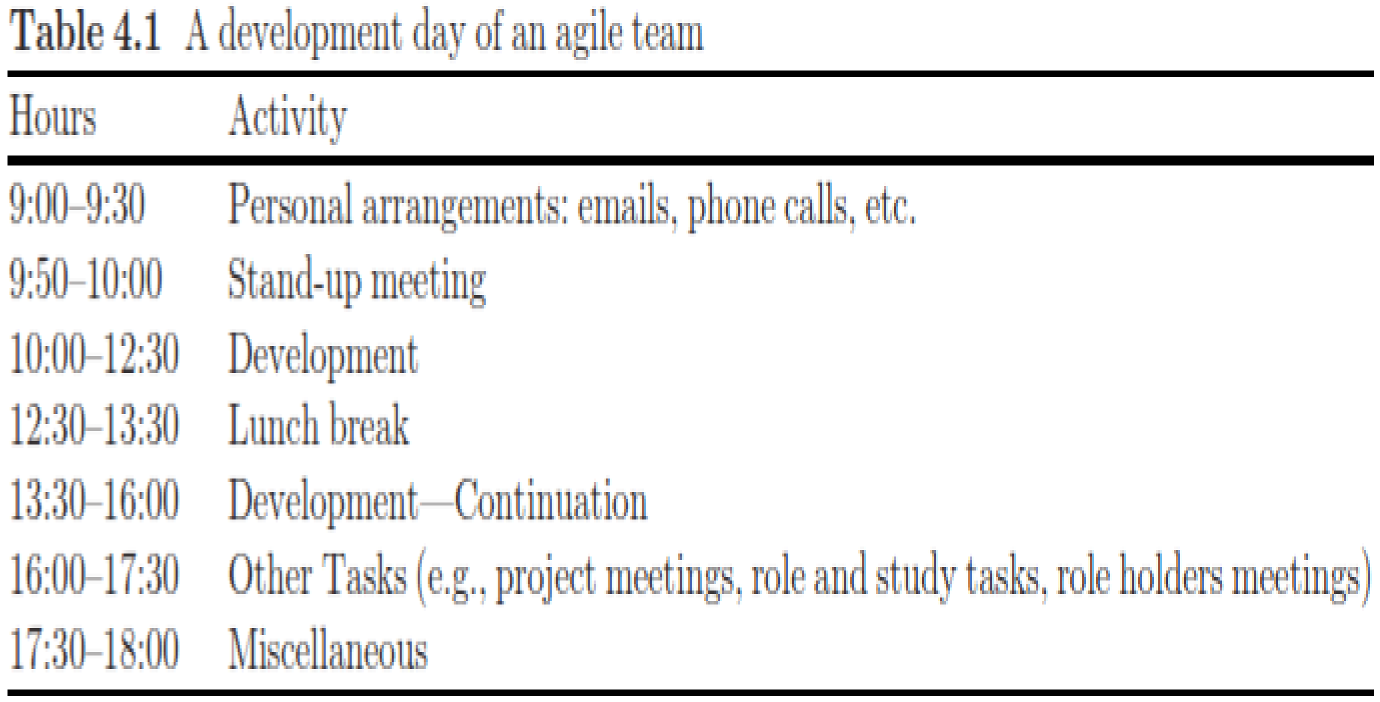

Table 4.1 A development day of an agile team

Hours	Activity
9:00–9:30	Personal arrangements: emails, phone calls, etc.
9:50–10:00	Stand-up meeting
10:00–12:30	Development
12:30–13:30	Lunch break
13:30–16:00	Development—Continuation
16:00–17:30	Other Tasks (e.g., project meetings, role and study tasks, role holders meetings)
17:30–18:00	Miscellaneous

Time Management of Agile Projects

1. Time Measurements

Time Management of Agile Projects

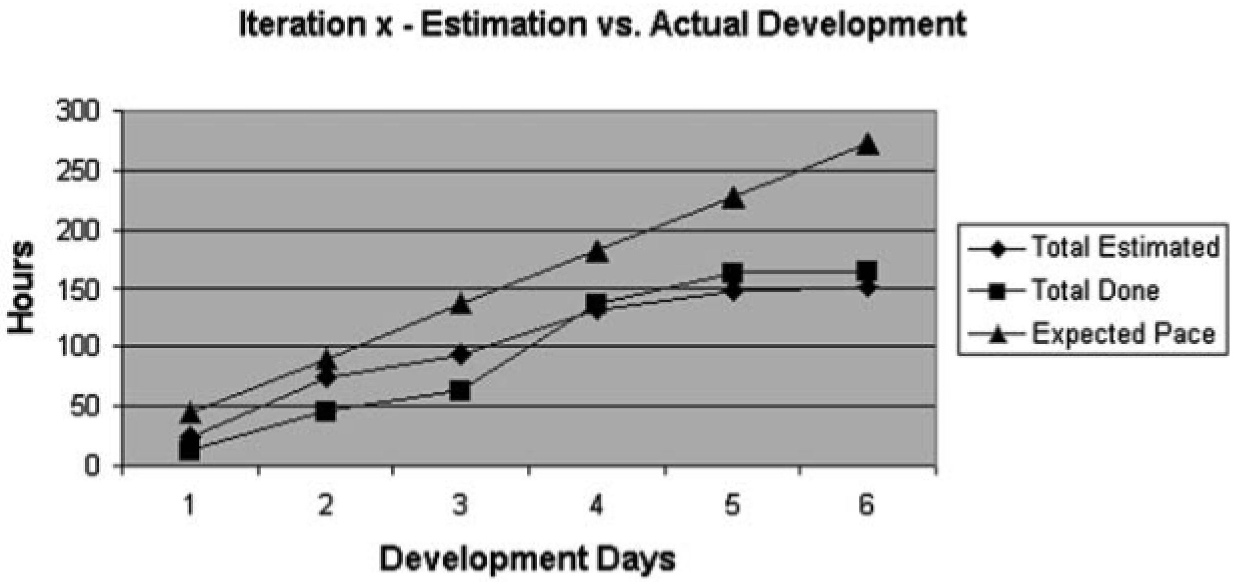

Figure 4.1 *Estimation versus actual development time: example 1.*

Time Measurements

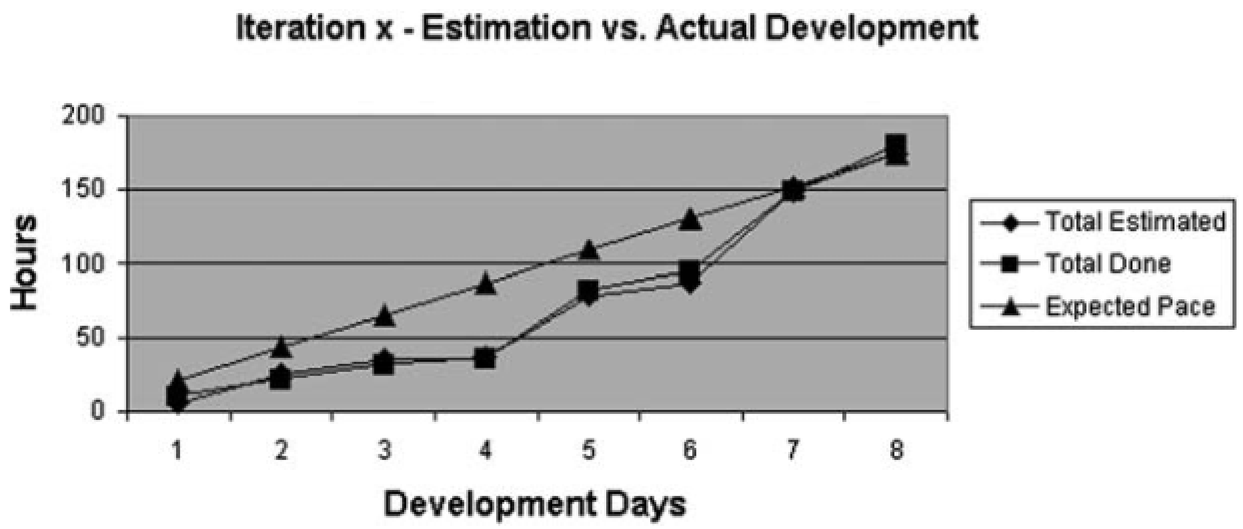

Figure 4.2 *Estimation vs. actual development time: example 2.*

2. Prioritizing Development Tasks

Case Study 4.4. First Things First

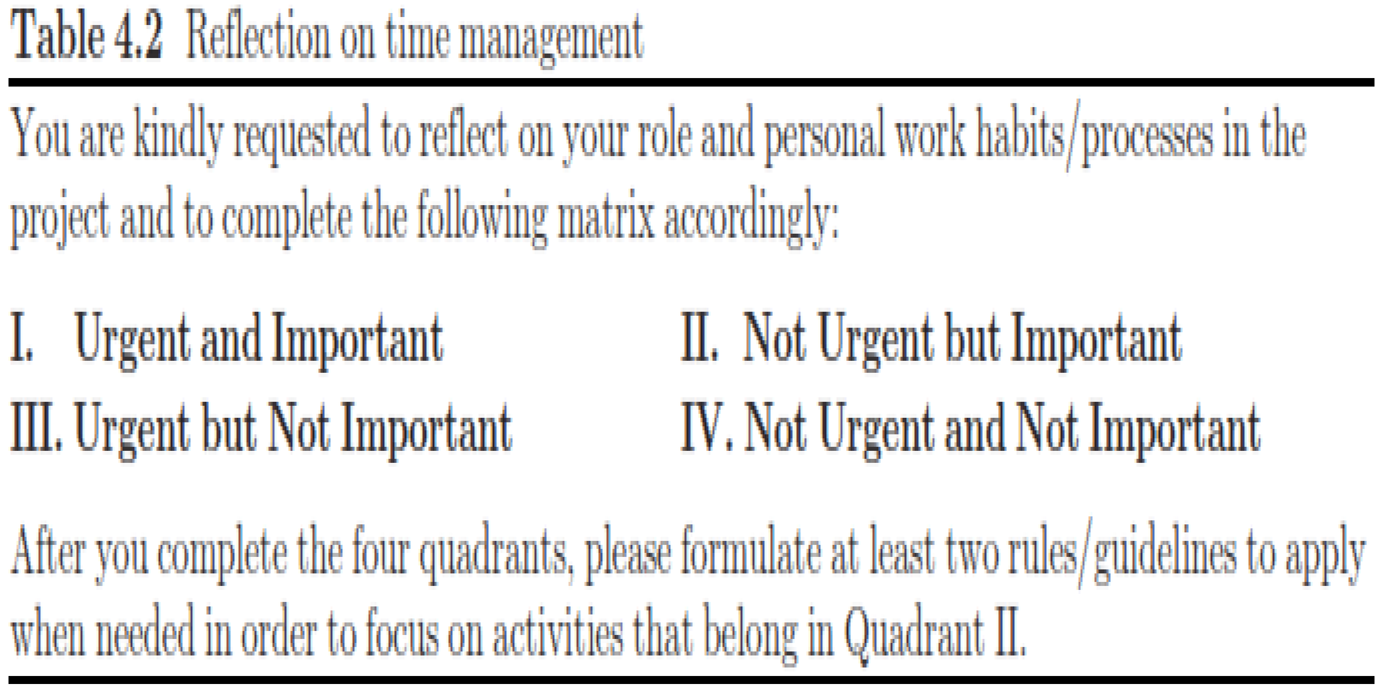

Table 4.2 Reflection on time management

You are kindly requested to reflect on your role and personal work habits/processes in the project and to complete the following matrix accordingly:

I. Urgent and Important	II. Not Urgent but Important
III. Urgent but Not Important	IV. Not Urgent and Not Important

After you complete the four quadrants, please formulate at least two rules/guidelines to apply when needed in order to focus on activities that belong in Quadrant II.

First Things First

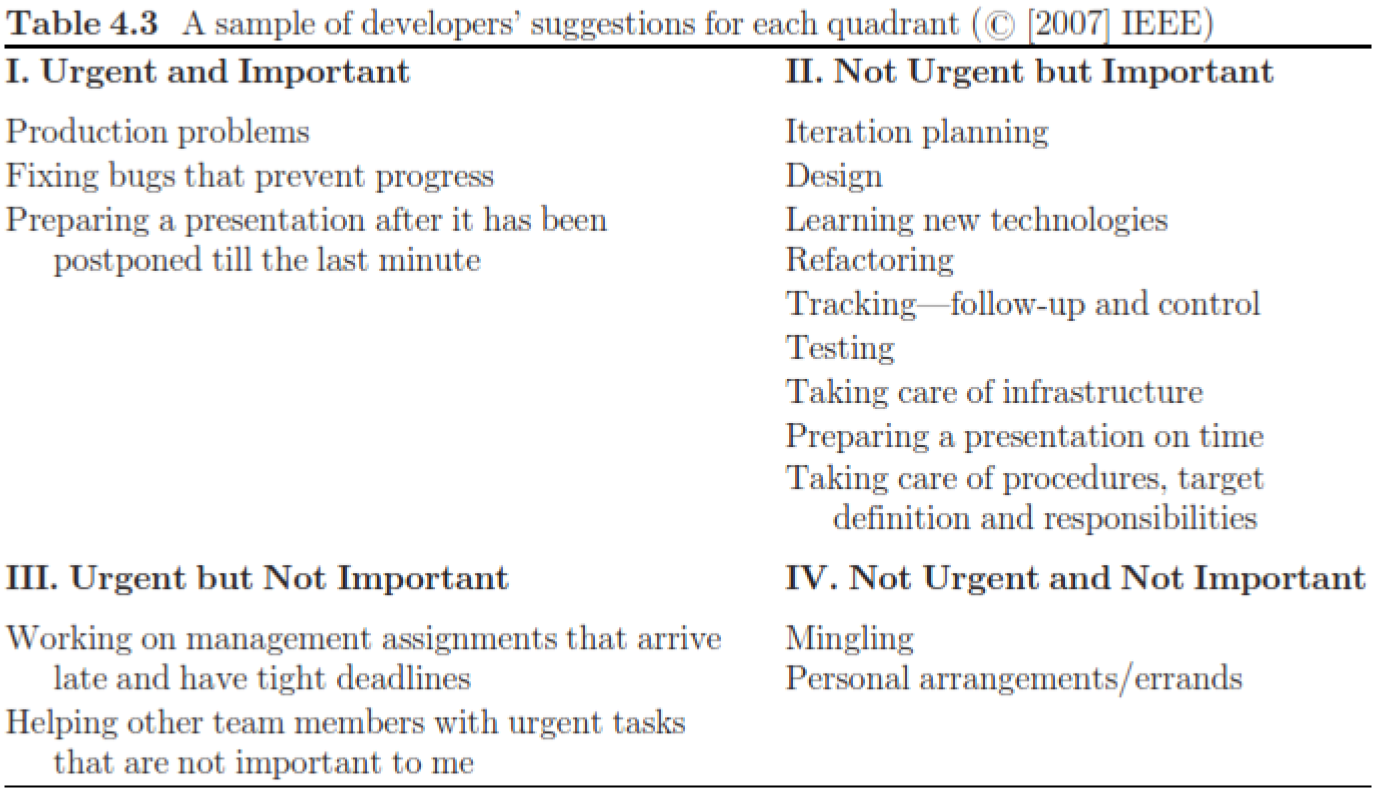

Table 4.3 A sample of developers' suggestions for each quadrant (© [2007] IEEE)

I. Urgent and Important	II. Not Urgent but Important
Production problems	Iteration planning
Fixing bugs that prevent progress	Design
Preparing a presentation after it has been postponed till the last minute	Learning new technologies
	Refactoring
	Tracking—follow-up and control
	Testing
	Taking care of infrastructure
	Preparing a presentation on time
	Taking care of procedures, target definition and responsibilities
III. Urgent but Not Important	**IV. Not Urgent and Not Important**
Working on management assignments that arrive late and have tight deadlines	Mingling
Helping other team members with urgent tasks that are not important to me	Personal arrangements/errands

Time in Learning Environments

1. The Planning Activity

The Planning Activity

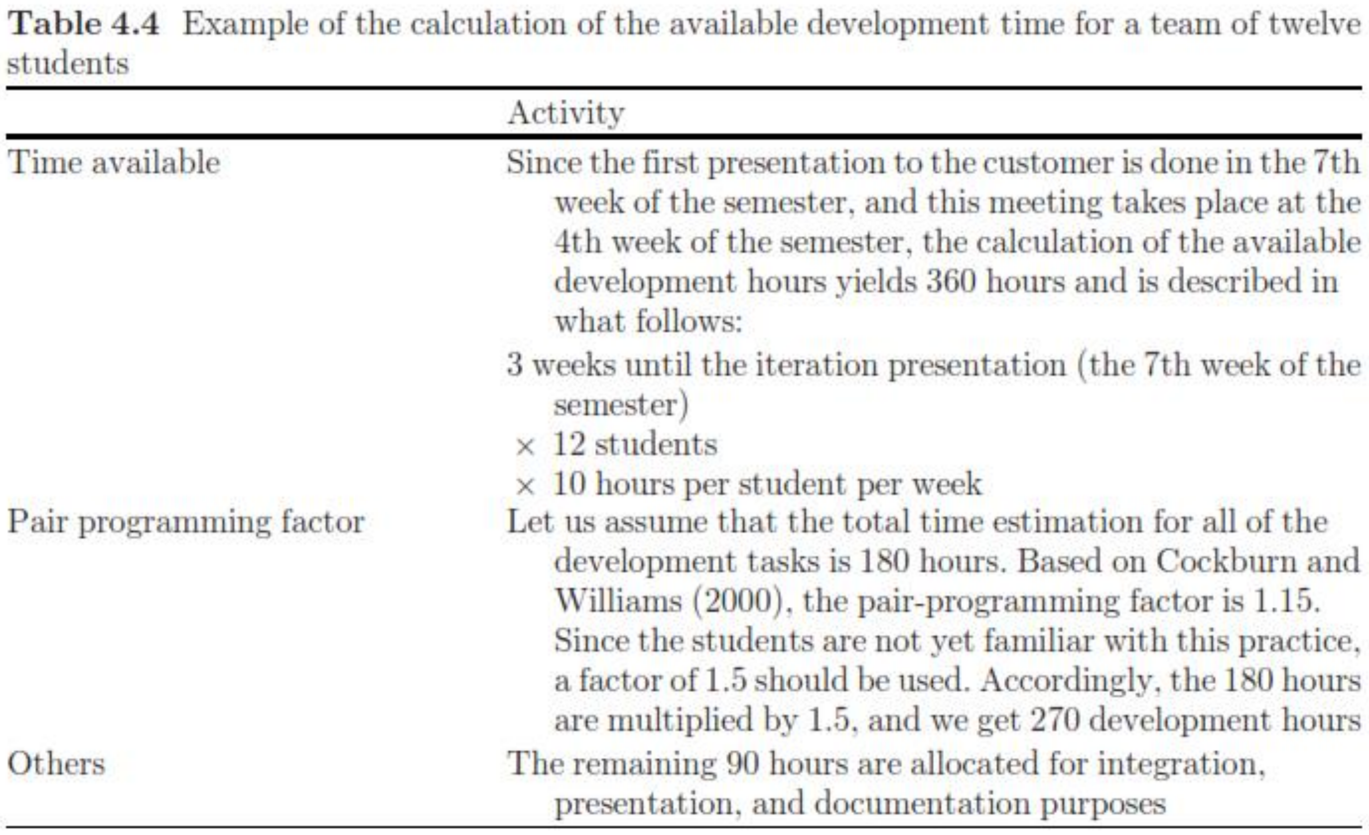

Table 4.4 Example of the calculation of the available development time for a team of twelve students

	Activity
Time available	Since the first presentation to the customer is done in the 7th week of the semester, and this meeting takes place at the 4th week of the semester, the calculation of the available development hours yields 360 hours and is described in what follows: 3 weeks until the iteration presentation (the 7th week of the semester) × 12 students × 10 hours per student per week
Pair programming factor	Let us assume that the total time estimation for all of the development tasks is 180 hours. Based on Cockburn and Williams (2000), the pair-programming factor is 1.15. Since the students are not yet familiar with this practice, a factor of 1.5 should be used. Accordingly, the 180 hours are multiplied by 1.5, and we get 270 development hours
Others	The remaining 90 hours are allocated for integration, presentation, and documentation purposes

2.Teaching and Learning Principles

- Teaching and Learning Principle 3: Explain

While Doing

- Teaching and Learning Principle 8: Manage Time

3. Students' Reflections on Time-Related Issues

- ''MUCH more time should be dedicated to the planning of the structure of the program, both interfaces and implementation, before anyone starts writing any code.''
- ''Things usually take more time than expected, especially because of integration and misconceptions |this must be taken into account when estimating times.''
- ''As for time evaluation, we met our estimations almost exactly, and in some cases even finished tasks sooner than we had expected.''
- ''In iterations 2 and 3 the times were better defined, because of the experience we had gained.''

4. The Academic Coach’s Perspective

The planning session gives the academic coach an excellent viewpoint on the students’ work with respect to the project’s planning, the design of its parts, and the development management. During the planning activity, the academic coach becomes extensively familiar with the project details and with each student’s part in the development.