COMP 3111: Software Engineering

Project Management Using Scrum[1]
Tutorial Notes

Managing software development projects is not easy! The larger a software development project, the harder it is to manage and, thus, the more critical it is to apply a divide and conquer approach to project organization. That is, you need to divide the project into smaller pieces (i.e., tasks) and to assign the most appropriate project members (i.e., individuals or teams) to each task. Furthermore, to ensure smooth running of the project it is vital to share information among the project members through both formal and informal communication channels. Good project organization and communication contribute as much to the success of a software development project as does great design or efficient implementation. Poor project organization or failure to communicate well among the project members can have a detrimental, and sometimes fatal, impact on a project.

PROJECT ORGANIZATION

A software development project consists of four inter-related components.

Participant	A person participating in a project in a certain role (e.g., project member, client, user, etc.).
Task	The work to be performed by a project participant.
Work product	An item (also called an *artifact*) produced by a task (e.g., models, documents, code, etc.).
Schedule	The organization of tasks according to priority and the time when they should start and end. A schedule often also specifies who (i.e., what role) should carry out a task.

Participant

An important part of any project organization is to define the relationships among the participants and between the participants and the tasks, work products and schedule. In a small project, each participant may work on a separate task. In a larger project, the participants may be grouped into teams where all members in a team work on the same task. Unless a team is very large, there is normally no redundancy of effort among team members. Any project depends on the punctual delivery of each participant's contribution to successfully complete a task/work product.

A role defines the set of technical and managerial tasks that a participant or team are expected to carry out. Management roles deal with the organization and execution of the project within constraints (e.g., time and budget). Development roles deal with specifying, designing and constructing subsystems. Cross-functional roles deal with coordination among teams. The cross-functional role is also called *liaison*. The liaison is responsible for exchanging information relevant to other teams and negotiating interface details. A liaison may also be called upon to resolve inter-team issues. Consultant roles deal with providing temporary support in areas where the project participants lack expertise. The client and users act in most projects as consultants on the application domain. Technical consultants may bring expertise on new technologies or development methods. Non-technical consultants can help address legal and marketing issues.

Two important management roles in a project are that of *project manager* and *team leader* or *person in charge (PIC)*[2]. In a team-based organization, project participants report to their PIC who is responsible for tracking the status of the participants and identifying problems. The PICs of each team in turn report the status of their team to the project manager who then evaluates the status of the entire project. Project managers and PICs respond to deviations from the schedule by reallocating tasks to participants or obtaining additional resources. The project manager is also responsible for the interaction with the client.

A project manager/team leader has to be able to combine technical skills with managerial skills, both task- and people-oriented. Some of the responsibilities of a project manager/team leader may include:

- giving vision (i.e., motivation) to the project/team;
- assessing project/team members' strengths/weaknesses;
- planning the project/team work and dividing the work among project/team members;
- setting project/team meeting agendas;
- tracking project/team progress;
- reallocating tasks/resources among project/team members;

[1] Adapted from Chapter 3 of Bruegge, B. and Dutoit, A.H., *Object-Oriented Software Engineering: Using UML, Patterns, and Java, 3rd Edition*. Prentice Hall, 2010.

[2] A project has a project manager; a project team has a team leader or person in charge (PIC).

- resolving both technical conflicts and personality conflicts among project/team members;
- encouraging project/team members (i.e., maintaining project/team spirit).

To facilitate the smooth running and management of a project, there are certain guidelines that *project managers/team leaders* should follow:

1. Avoid compromising the project's/team's objective with political issues.
2. Exhibit personal commitment to the project's/team's goal.
3. Do not dilute the project's/team's efforts with too many priorities.
4. Be fair and impartial toward all project/team members.
5. Be willing to confront and resolve issues associated with inadequate performance by project/team members.
6. Be open to new ideas and information from project/team members.

Similarly, there are certain guidelines that *project/team members* should follow:

1. Demonstrate a realistic understanding of your role and responsibilities.
2. Demonstrate objective and fact-based judgments.
3. Collaborate effectively with other members.
4. Make the project/team goal a higher priority than any personal objective.
5. Demonstrate a willingness to devote whatever effort is necessary to achieve project/team success.
6. Be willing to share information, perceptions and feedback appropriately.
7. Provide help to other members when needed and appropriate.
8. Demonstrate high standards of excellence.
9. Stand behind and support project/team decisions.
10. Demonstrate courage of conviction by directly confronting important issues.
11. Demonstrate leadership in ways that contribute to the project's/team's success.
12. Respond constructively to feedback from others.

Task

A task is a well-defined work assignment. Either the participants select the tasks they work on or the project manager or team leader assigns the tasks to participants. A participant carries out the task, and the project manager or team leader monitors its progress and completion.

Work Product (Artifact)

A work product (also sometimes called an artifact) is a tangible item that results from a task, is subject to a deadline and may be used by other tasks. Any work product to be delivered to the client is called a *deliverable*. The software system and the accompanying documentation usually constitute a set of deliverables. Work products that are not visible to the client are called *internal work products*.

Schedule

A schedule maps tasks onto time; each task is assigned a start and an end time. This allows deadlines for individual deliverables to be planned. Tasks need to be sequenced (i.e., what task must be done first, what task second, etc.). Some tasks may overlap (i.e., they can be carried out in parallel), while other tasks must follow a strict sequence. The schedule for a project should be reviewed periodically (e.g., daily or weekly) and revised as necessary. *It should reflect the actual status of the project.*

PROJECT COMMUNICATION

Software development often requires the collaboration of people with different backgrounds (e.g., domain experts, analysts, designers, programmers, users, etc.) who depend on each other to accomplish their work. Dependencies among the parts of a software development project make it critical that people communicate information in an accurate and timely manner if the requirements are to be correctly represented in the final system. Communication in a software development project can take several forms.

- Models and their corresponding documentation communicate requirements and design alternatives.
- Meetings and meeting minutes communicate the status of people, tasks and work products.
- Reviews communicate project status (to developers or to the client).
- Spontaneous information exchanges, such as phone messages, email and hallway conversations, are often used to handle crises and misunderstandings.

Communication can be divided into *planned* and *unplanned* communication events.

Planned Communication

Planned communication events are scheduled points in time during which participants exchange information on a specific topic or review a work product. Such events are formalized and structured to maximize the amount of information communicated and to minimize the time participants spend on communication.

Problem inspection

Information is gathered from the *problem statement*, the client and the user about the application domain, the desired functionality of the system and any nonfunctional requirements the system must meet.

Client review

The client assesses the progress of the development and the developers confirm or change the requirements of the system. The client review is used to manage expectations on both sides and to increase the shared understanding among participants. The focus of the review is on what the system does and what constraints are relevant to the client. A client review is conducted as a formal presentation during which developers focus on specific functionality with the client. The review is preceded by the release of a work product (e.g., specification document, interface mock-up, prototype system, etc.). At the conclusion of the review, the client provides feedback to the developers.

Project review

During project reviews, the project manager assesses status and teams review their work products. Project reviews can also encourage the exchange of operational knowledge across teams, such as common problems encountered with tools or the system. The focus of the review depends on the deliverable under review. A project review is typically conducted as a formal presentation during which each team presents its work product to the management or to teams that depend on the work product. The review is usually preceded by the release of a document describing the aspects of the system under review. At the conclusion of the review, the developers may negotiate changes in the work product and changes in the schedule.

Peer review

During peer reviews, team members identify defects and find solutions in preliminary work products. Peer reviews normally consist of inspections or walkthroughs of models or the code. A developer presents to the other team members the model or the code he has written. The other team members challenge any suspicious requirements or code and attempt to discover as many errors as possible. During inspections, the team members focus on the compliance of the model or code with a predefined list of criteria.

Status review

During a status review, project teams review their task progress. Status reviews are primarily conducted in scheduled meetings (e.g., daily or weekly). The objective of a status review is to detect deviations from a task's schedule and to correct them. The review of task status encourages the discussion of open issues and unanticipated problems. Often solutions to common issues can be shared and operational knowledge disseminated more effectively when discussed within the scope of a team.

Brainstorming

The goal of brainstorming is to generate a large number of solutions to a problem, regardless of their merit, and then to evaluate them. The fundamental idea behind brainstorming is that ideas, however invalid, proposed by any participant can trigger other ideas and proposals from other participants. Brainstorming encourages thinking "outside the box". Brainstorming also has two beneficial side effects: evaluating proposals within the group will lead to more explicit evaluation criteria, and the brainstorming process itself has the effect of building consensus for the chosen solution.

Releases

A release is a version of the system and its documentation that is made available either to other project participants or to the client and end users. Releases allow a large amount of information to be made available in a controlled manner by batching, documenting and reviewing many changes together. Project and client reviews are typically preceded by a release of one or more deliverables.

Postmortem review

Postmortem reviews focus on extracting lessons from the development team once the software is delivered. Postmortem reviews need to be conducted shortly after the end of a project so that minimal information is lost

or distorted by subsequent experience. The end of a project is usually a good time to assess which techniques, methods and tools have worked and have been critical to the success (or failure) of the project.

Unplanned Communication

In an ideal project, all communication takes place during planned communication events. In practice, however, it is difficult to anticipate all information needs and plan all communication. Thus, unplanned communication needs to be dealt with.

Request for clarification

A participant may request specific information from other participants (teams, client, etc.) about any aspect of the system, the application domain or the project that seems ambiguous. Situations in which most information needs are handled through requests for clarification are symptoms of a defective communication infrastructure. Such projects usually face serious failures resulting from misunderstandings, and missing or incomplete information.

Request for change

A participant reports problems encountered in the system or makes a request for new features that the system should support. For better project management, requests for change are often handled in a formal manner when the number of participants and the system size is substantial.

Issue resolution

A conflict between different stakeholders is identified, solutions explored and negotiated, and a common resolution agreed upon. The outcome needs to be documented and communicated to the relevant participants. Documentation of the resolution allows participants to refer back to the decision later in the project, in case of a misunderstanding. Effective communication of the decision enables participants to remain synchronized.

Meetings

The bulk of the planned communication events in a project consist of face-to-face meetings of various kinds. Face-to-face meetings enable a number of participants to share, review and negotiate issues and solutions[3]. In order to increase the information transfer and the number of decisions made during a meeting, certain roles are assigned to selected participants.

- The **meeting chair** or **facilitator** is responsible for organizing the meeting and guiding its execution.
- The **minute recorder** is responsible for recording the meeting minutes. The minute recorder takes notes that capture as much information about the meeting (mainly status and decisions) as possible, organizes them after the meeting and releases them shortly after the meeting for review by the meeting participants. This enables the participants to reiterate their commitment to the outcome of the meeting and for team members who missed the meeting to catch up with team status.

Meeting minutes should record the following.

1. The meeting location, date, time, and present and absent participants.
2. For each participant:
   a. What he/she did since the last meeting;
   b. What impediments hindered progress, if any;
   c. What he/she plans to do from now until the next meeting;
2. The date, time and place of the next meeting.

[3] Serious technical or personality conflicts should be resolved outside of the meeting time.

PROJECT MANAGEMENT USING SCRUM[4]

Scrum is an agile process for software development. It is ideally suited for projects with rapidly changing or highly emergent requirements. With Scrum, projects progress via a series of iterations called *sprints* each of which is typically 2-4 weeks long.

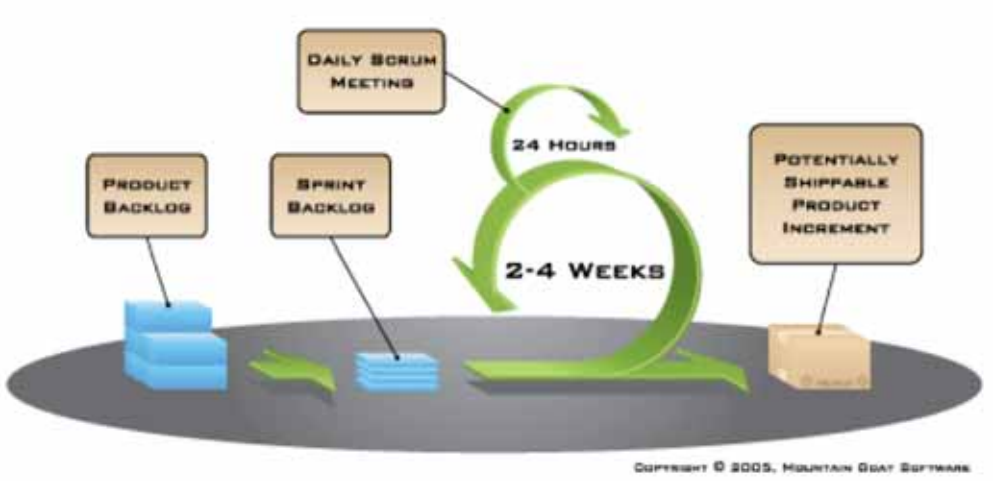

Figure 1: A graphical view of Scrum software development.

Figure 1 shows the essential parts involved in Scrum software development. On the left, is the product backlog, which is prioritized by the product owner and contains everything currently wanted in the product. The larger green circle shows the 2-4 week sprints. At the start of each sprint, the team selects some amount of work from the product backlog and commits to completing that work during the sprint. Part of figuring out how much they can commit to is creating the sprint backlog, which is the list of tasks (and an estimate of how long each will take) needed to deliver the selected set of product backlog items to be completed in the sprint. At the end of each sprint, the team produces a potentially shippable product increment (i.e., working, high-quality software). Each day during the sprint, team members meet to discuss their progress and any impediments to completing the work for that sprint. This is known as the daily scrum, and is shown as the smaller green circle on top of the larger one in Figure 1. The rest of these notes describe each of these parts in more detail as well as the roles involved in, and other aspects of, Scrum software development.

Roles

Product Owner

The product owner is the project's key stakeholder and represents users, customers and others in the development process. The product owner defines the requirements of the product and prioritizes the product backlog. The scrum team looks at the prioritized product backlog, selects the top priority items and commits to completing them during a sprint. These items become the sprint backlog. In return for their commitment to completing the selected tasks (which, by definition, are the most important to the product owner), the product owner commits that he or she will not change any requirements or ask the team to work on any new requirements during the sprint. Requirements are allowed to change (and change is encouraged), but only outside the sprint. Once the team starts on a sprint it remains completely focused on the goal of that sprint.

Scrum Master

The scrum master, who is the team leader, is responsible for making sure the scrum team is as productive as possible. He or she does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside interference, etc. In addition, the scrum master

- is responsible for making sure a Scrum team lives by the values and practices of Scrum.
- protects the team by making sure they do not over-commit themselves to what they can achieve during a sprint.
- facilitates the daily scrum and becomes responsible for removing any obstacles that are brought up by the team during those meetings.

The scrum master role is typically filled by a project manager or a technical team leader, but can be anyone.

[4] Adapted from http://www.mountaingoatsoftware.com/topics/scrum © 1998-2010 Mountain Goat Software.

Scrum Team

A scrum team is typically made up of between five and nine people, but Scrum projects can easily scale into the hundreds. The team does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Instead, everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that “we’re all in this together.”

Meetings

Sprint Planning Meeting

At the start of each sprint, a sprint-planning meeting is held during which the product owner prioritizes the product backlog and the scrum team selects the work they can complete during the coming sprint. The product owner, scrum master, the entire scrum team, and any interested and appropriate management or customer representatives attend the sprint planning meeting.

During the sprint planning meeting the product owner describes the highest priority features to the team. The team asks enough questions during this meeting so that they can go off after the meeting and determine which tasks they will move from the product backlog to the sprint backlog. Collectively, the scrum team and the product owner define a sprint goal, which is a short description of what the sprint will attempt to achieve. The success of the sprint will later be assessed during the sprint review meeting against the sprint goal, rather than against each specific item selected from the product backlog.

After the sprint planning meeting, the scrum team meets separately to discuss what they heard and decide how much they can commit to during the coming sprint. In some cases there will be negotiation with the product owner, but it will always be up to the team to determine how much they can commit to completing.

Daily Scrum Meeting

Each day during the sprint, a brief meeting, called the daily scrum, is conducted. The daily scrum is typically held in the same location and at the same time each day. Ideally the daily scrums are held in the morning as they help set the context for the coming day's work and help the team stay on track. All team members as well as the product owner are required to attend the daily scrum and participate. Anyone else (e.g., a departmental VP, a salesperson, or a developer from another project) is allowed to attend, but is there only to listen. This makes the daily scrums an excellent way for a scrum team to disseminate status information—if you are interested in hearing where things are at, attend that day's meeting.

The daily scrum is not used as a problem solving or issue resolution meeting. Issues that are raised are taken offline and usually dealt with by the relevant sub-group immediately after the daily scrum. During the daily scrum each team member provides answers to the following three questions:

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

By focusing on what each person accomplished yesterday and will accomplish today, the team gains an excellent understanding of what work has been done and what work remains. The daily scrum is not a status update meeting in which a boss is collecting information about who is behind schedule. Rather, it is a meeting in which team members make commitments to each other. If a programmer stands up and says, "Today I will finish the data storage module" everyone knows that in tomorrow's meeting he will say whether or not he did finish. This has the effect of helping a team realize the significance of these commitments and that their commitments are to each other, not to some far-off customer or salesman.

Any impediments that are raised become the scrum master's responsibility to resolve as quickly as possible. Typical impediments are:

- My ____ broke and I need a new one today.
- I still haven't got the software I ordered a month ago.
- I need help debugging a problem with _____.
- I'm struggling to learn _____ and would like to pair with someone on it.
- I can't get the vendor's tech support group to call me back.
- Our new contractor can't start because no one is here to sign her contract.
- I can't get the ____ group to give me any time and I need to meet with them.
- The department VP has asked me to work on something else "for a day or two."

In cases where the scrum master cannot remove these impediments directly himself (e.g., usually the more technical issues) he still takes responsibility for making sure someone on the team does quickly resolve the issue.

Sprint Review Meeting

At the end of each sprint a sprint review meeting is held. Participants in the sprint review typically include the product owner, the scrum team, the scrum master, management, customers and developers from other projects. During the sprint review the project is assessed against the sprint goal determined during the sprint planning meeting. Ideally the team has completed each product backlog item brought into the sprint, but it is more important that they achieve the overall goal of the sprint.

In the meeting, the scrum team shows what they accomplished during the sprint. Typically this takes the form of a demo of the new features. The sprint review meeting is intentionally kept very informal, typically with rules forbidding the use of PowerPoint slides and allowing no more than two hours of preparation time for the meeting. A sprint review meeting should not become a distraction or significant detour for the team; rather, it should be a natural result of the sprint.

Artifacts

Product Backlog

The product backlog[5] is a prioritized "master" list containing every desired requirement or change to the product. Product backlog items can be technical tasks ("Refactor the Login class to throw an exception") or more user-centric ("Allow undo on the setup screen"). When using Scrum, it is not necessary to start a project with a lengthy, upfront effort to document all requirements. Typically, a Scrum team and its product owner begin by writing down everything they can think of easily. This is almost always more than enough for a first sprint. The product backlog is then allowed to grow and change as more is learned about the product and its customers. Figure 2 shows an example product backlog kept as a spreadsheet. The spreadsheet shows that each product backlog item is assigned a priority (Very High, High, etc.) by the product owner. The developers have developed estimates, but it is understood that they are very imprecise and are useful only for rough assignments of tasks into the various sprints.

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Figure 2: Example product backlog.

Sprint Backlog

The sprint backlog is the list of tasks that the Scrum team commits to complete in the current sprint. The team moves items from the product backlog to the sprint backlog based on the priorities set by the product owner and the team's perception of the time it will take to complete the various features. In doing so, the team expands each product backlog item into one or more sprint backlog tasks so they can more effectively share work during the sprint. Conceptually, the team starts at the top of the prioritized product backlog list and draws a line after the lowest of the high priority items they feel they can complete.

[5] The term "backlog" can get confusing because it's used for two different things. The product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.

In practice it is not unusual to see a team select, for example, the top five items and then two items from lower on the list, but that are associated with the initial five. It is critical that the team selects the items and size of the sprint backlog. Because they are the ones committing to completing the tasks, they must be the ones to choose what they are committing to. As shown in Figure 3, the sprint backlog is also very commonly maintained as a spreadsheet.

Tasks	Mon	Tue	Wed	Thu	Fri
Code user interface	8	4	8		
Code middle tier	16	12	10	7	
Test middle tier	8	16	16	11	8
Write online help	12				

Figure 3: Example sprint backlog.

Burndown Charts

During a sprint the scrum master maintains the sprint backlog by updating it to reflect which tasks are completed and how long the team thinks it will take to complete those that are not yet done. The estimated work remaining in the sprint is calculated daily and graphed, resulting in a sprint burndown chart as shown in Figure 4.

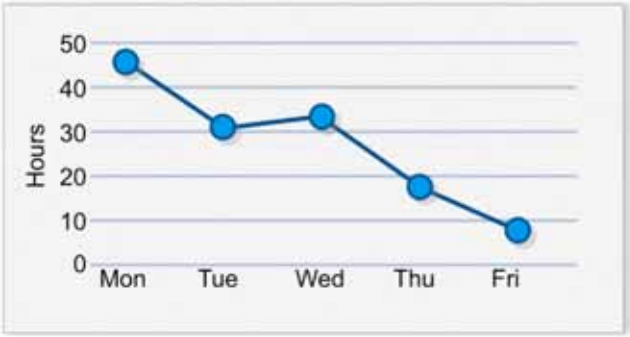

Figure 4: Example sprint burndown chart.

The team does its best to pull the right amount of work into the sprint, but sometimes too much or too little work is pulled in during the sprint planning meeting. In this case, the team needs to add or remove tasks. The sprint burndown chart in Figure 4 shows that the team had pulled in too much work initially so that the amount of work remaining actually increased on Wednesday due to the time required for the first task being increased.