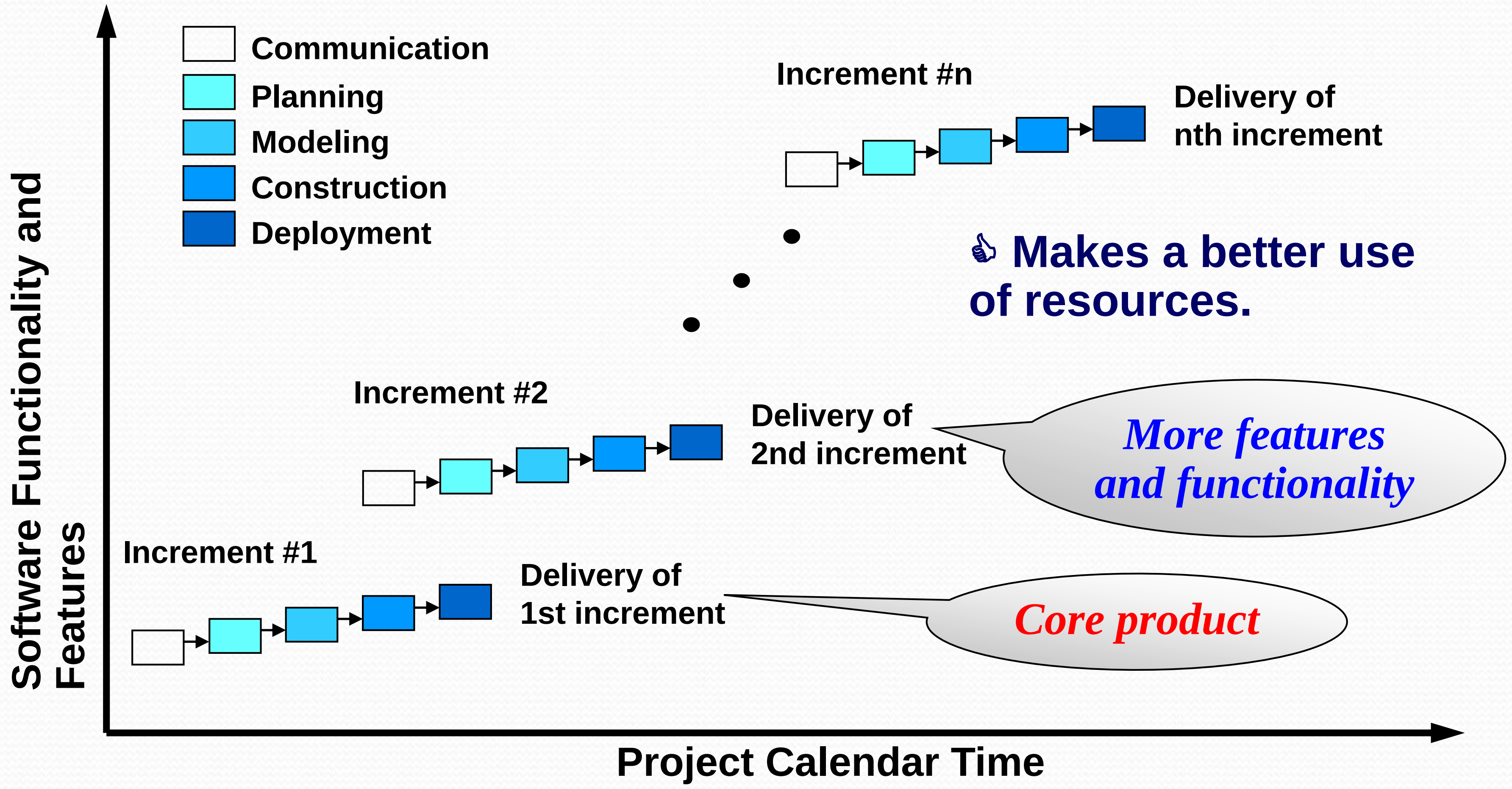
Agile Software Development

The Incremental Model - Remind


Communication
Planning
Modeling
Construction
Deployment
Increment #n
Delivery of nth increment
Increment #2
Delivery of 2nd increment
More features and functionality
Increment #1
Delivery of 1st increment
Core product
Software Functionality and Features
Project Calendar Time

👍 **Makes a better use of resources.**

Incremental Development

- Incremental Development is a practice where the system functionalities are sliced into increments (small portions). In each increment, a slice of functionality is delivered by going through all the activities of the software development process, from the requirements analysis to the deployment.
- Incremental Development is often used together with the Evolutionary practice of Iterative Development in software development. This is referred to as Iterative and Incremental Development (IID)

Incremental Process Models

- Used in situations where the initial software requirements are reasonably well-defined, but the overall scope of development is not clear yet
- Used when there may be a need to provide a limited set of software functionality to users quickly, which will be refined in later releases
- Thus, a process model that produces the software in increments is much more suitable

Agile Software Development

- IID is the core of the agile methods
- Agile methods emphasis on building releasable software in short time periods.
- Agile methods emphasize working software as the primary measure of progress

What is ‘Agility’?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

=> Rapid, incremental delivery of software

Characteristics of Agile Software Development

- Light Weighted methodology
- Small to medium sized teams
- Vague and/or changing requirements
- Vague and/or changing techniques
- Simple design
- Minimal system into production

An Agile Process

- Is driven by customer descriptions of what is required (scenarios/user stories)
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

Agile and traditional approaches

- Requirements change midway or can be unclear
- Traditional:
  - Try to restrict change
  - Try to create predictive plan
- Agile:
  - Try to embrace change
  - Try to be adaptive

Agile and traditional approaches

- Scheduling:
  - Traditional processes : structured in phase, harder to measure progress
  - Agile processes: structured by feature, easier to measure progress
- Quality:
  - Traditional processes: testing at the end of the project
  - Agile: continuous testing, integration and review

(S. Govindara)

Agile methods

- Extreme Programming
- Scrum

Extreme programming (XP)

- Extreme (XP) programming is an approach of SD to develop software faster, incrementally and to produce software satisfied customer.
- Relies on constant code improvement, and user involvement in the development team.
- It presents a number of practices and principles, which makes XP successful and well known among all the agile software development methods

XP Values

- Simplicity - It is better to implement the simplest solution.
- Communication - Increases cooperation, group productiveness, and decreases mistakes.
- Feedback - Keeps the project on track. Feedback should be almost continuous.
- Courage - Required to be able to throw code away when it is appropriate, or to refactor late in the design to increase system performance.

XP Principles

- **Pair programming** - Ensures quality code. One programmer is thinking whether the approach will work, about testing, or ways to simplify the code while the other programmer writes the code. The roles of the two programmers may change often, and programmer pairs may change often.
- **Simple design** - Keep the design as simple as possible for the moment and don't add features that are not needed for current functionality. The reasoning behind this is that if a feature is not valuable now, it is not worth the investment until it is valuable.
- **Small releases** - There is a short time between versions.

XP Principles

- **Refactoring** - The code is restructured as necessary to improve structure and performance.
- **Testing and Integration** - Tests are written before code. Code is continuously integrated together.
- **Customer availability** - The customer is on site to answer questions or at least is readily accessible.
- **Standards of coding** - There are group rules for code so all members can understand the code readily.

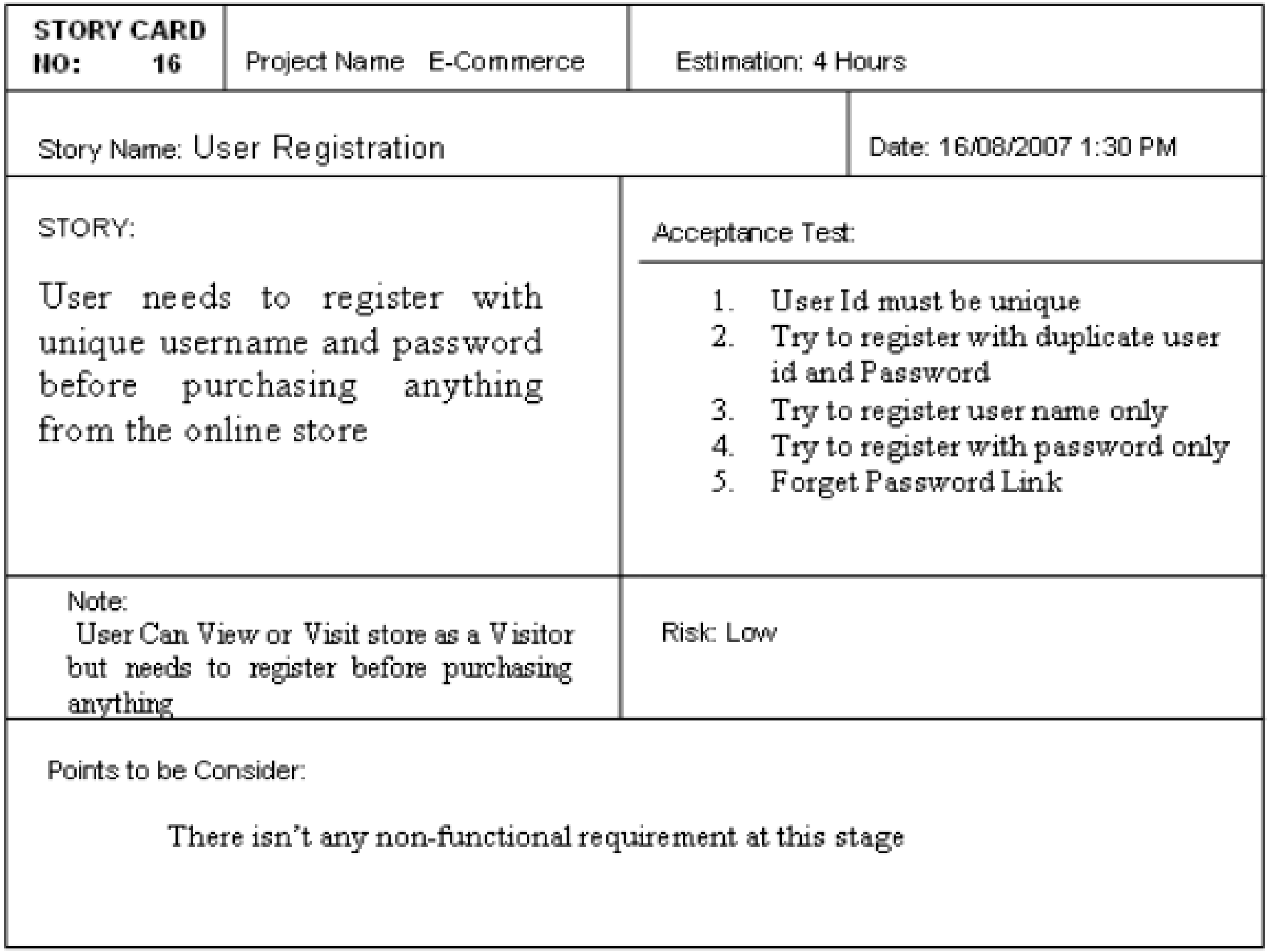
Development phases in XP

- XP planning
- XP design
- XP coding
- XP testing

XP planning

- Begins with the creation of **user stories** (by users)
- Team assesses each story and assigns a **cost** (rough estimates of the time to implement each story)
  - Cost from 1 to 3 weeks
  - Split story or combine stories
- Stories are grouped to for a **deliverable increment**
- A **commitment** is made on delivery date
  - These selected stories are then translated into individual programming tasks to be implemented during the iteration to complete the stories
- After the first increment **project velocity** is used to help define subsequent delivery dates for other increments

Example- Story card



STORY CARD NO: 16	Project Name E-Commerce	Estimation: 4 Hours
Story Name: User Registration		Date: 16/08/2007 1:30 PM

STORY:

User needs to register with unique username and password before purchasing anything from the online store

Acceptance Test:

1. User Id must be unique
2. Try to register with duplicate user id and Password
3. Try to register user name only
4. Try to register with password only
5. Forget Password Link

Note:
User Can View or Visit store as a Visitor but needs to register before purchasing anything

Risk: Low

Points to be Consider:

There isn't any non-functional requirement at this stage

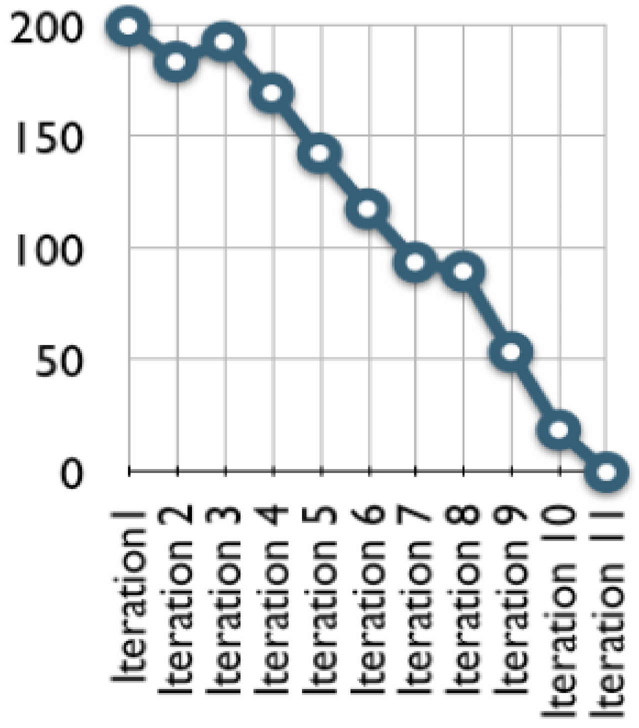
Project Velocity

- Velocity basically is how fast the software is being developed
- Velocity can mean the number of user stories that can be tacked in one iteration
- If you know the velocity you can make the ‘burndown curve’, that is, show how fast you are burning through stories
- Knowing this at an iteration level, and then at a release level help short and long-term project planning

Burndown Chart - Example

Burndown Chart

- The Vertical axis shows the number of user stories that have yet to be completed



XP Design

- Encourage the use of CRC (Class Responsibility Collaborator) cards
  - CRC cards identify and organize the classes that are relevant to the current software increment (aim to identify data and methods/activities involved in an increment)
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype that is implemented and evaluated, aiming to lower risk in later implementations
- Encourages "refactoring"—an iterative refinement of the internal program design

CRC Cards

Class: Order	
Responsibilities	**Collaborations**
Check items are in stock	Order line
Determine the price	Order line
Check for valid payment	customer
Dispatch to delivery address	

XP Coding

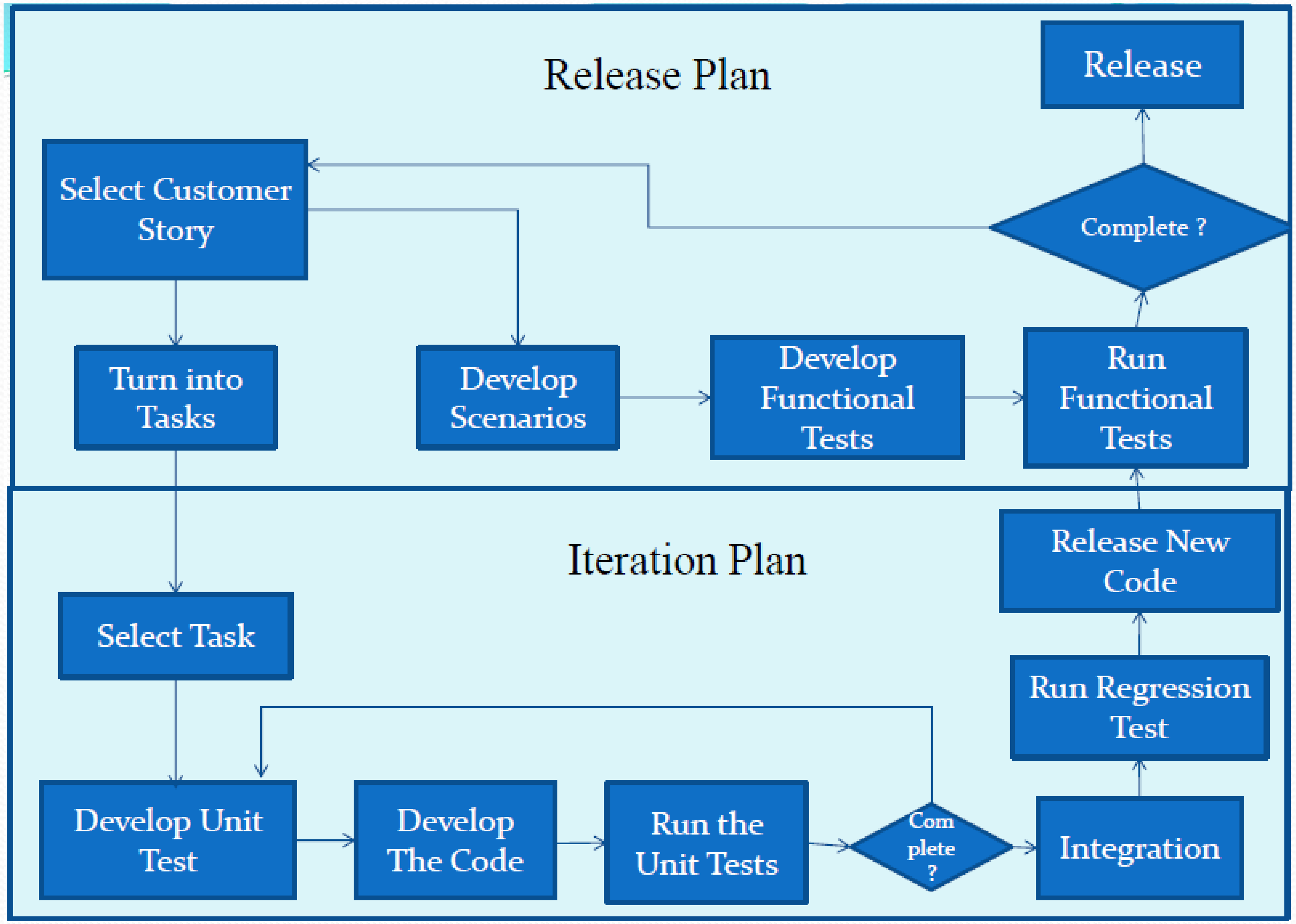
- Recommends the construction of a unit test for a story be done *before* coding commences
  - This helps the coder to focus on the task
  - Once the code is complete it can be tested immediately
- Encourages "pair programming"
  - Two people work together at the same workstation
  - Each takes on a slightly different role (e.g. coding details and coding standards/review)
- As pair programmers complete their work, their code is integrated with the code of others
  - Integration team or
  - Extension of pair programmers tasks

XP Testing

- Testing is done regularly

- “Acceptance tests” are defined by the customer from user stories and executed to assess customer-visible functionality

XP overall activities



XP sequence of activity

- A Release Plan is created from the User Stories. This plan sets out the overall project. Releases happen every 2-3 months.
- Iteration plans are then created for each individual iteration, using development time estimates for each user story.
- The customer specifies scenarios to show that a user story has been correctly implemented. A set of functional (or acceptance) tests is developed based on these. The customers are responsible for verifying the correctness of the acceptance tests, and reviewing test scores to decide which failed tests are of highest priority.
- Acceptance tests are also used as regression tests prior to the release of a new version of the software.

XP sequence of activity

- Each Iteration Plan is developed in detail just before the iteration begins and not in advance. Iterations are between 1 and 3 weeks in duration.

- User Stories are converted into implementation tasks, recorded on task cards. A programmers takes a task card, writes the unit test cases for the task, implements the code, and tests it.

- When the tests pass, the programmer then integrates the new code, runs regression tests, and releases the code for full functional testing.

XP sequence of activity

- After this, there is a tested and working software feature ready to demonstrate to the customer.

- Eventually, after all the iterations have been completed the product will be finished.

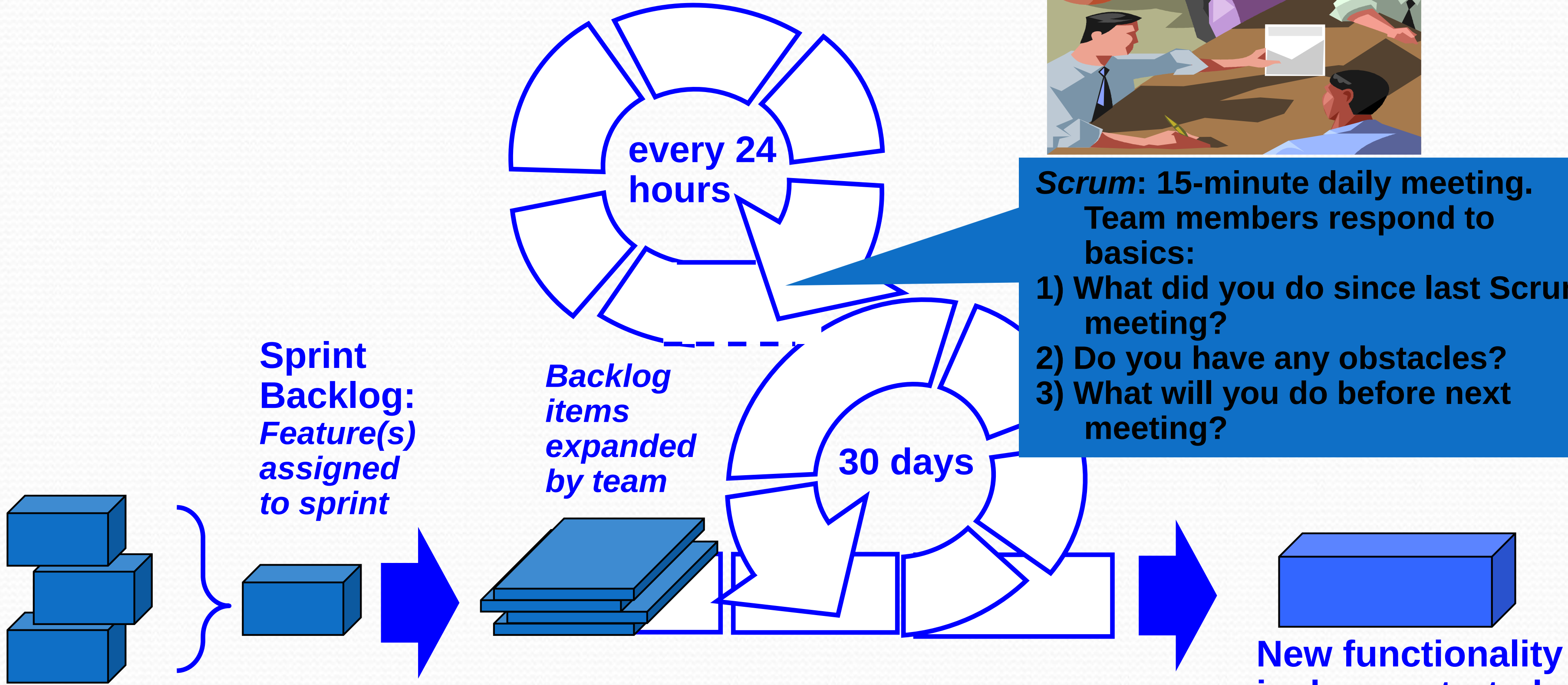
XP advantages

- Works best with small to medium size projects
- Works best for high risk projects where change is likely
- Works best for experienced, cohesive teams

Scrum method

- Scrum—distinguishing features
  - Development work is partitioned into a “product backlog”
  - Work occurs in “sprints” and is derived from a “backlog” of existing requirements (a collection of stories)
  - Meetings are very short and sometimes conducted without chairs
  - “demos” are delivered to the customer within the time-box allocated

Scrum Diagram


every 24 hours
30 days
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint

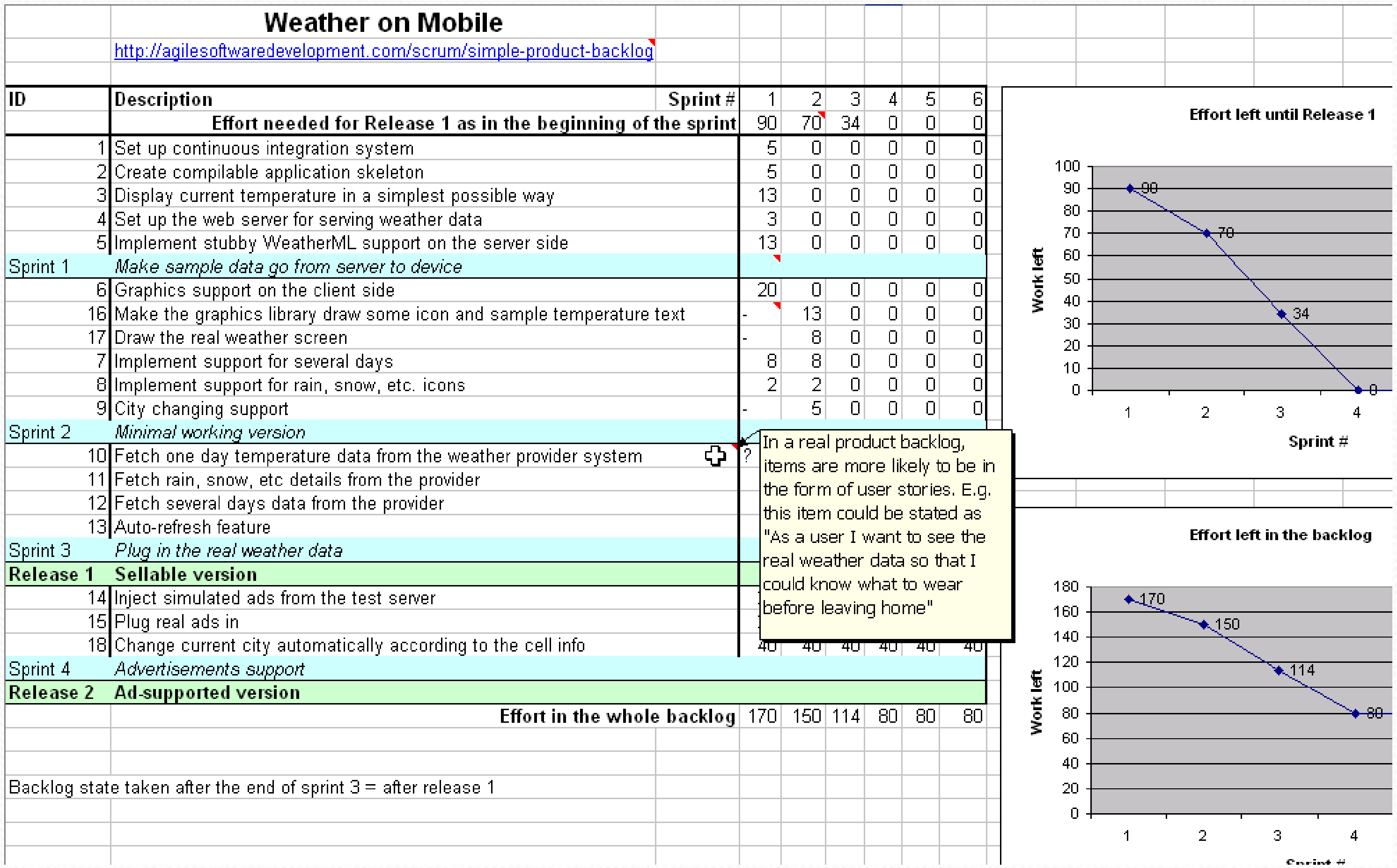
Scrum: 15-minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

Product Backlog:
Prioritized product features desired by the customer

SCRUM

- SCRUM starts with the **Product Backlog** which is a prioritized list of all product requirements (user stories)
- SCRUM teams take on as much of the product backlog as they think they can turn into an increment of product functionality within a 30-day iteration. This is called a **Sprint**.
- The team maintains a list of tasks to perform during each Sprint that is called the **Sprint Backlog**.

Example – Product Backlog



Weather on Mobile

http://agilesoftwaredevelopment.com/scrum/simple-product-backlog

ID	Description	1	2	3	4	5	6
	Sprint #	1	2	3	4	5	6
	Effort needed for Release 1 as in the beginning of the sprint	90	70	34	0	0	0
1	Set up continuous integration system	5	0	0	0	0	0
2	Create compilable application skeleton	5	0	0	0	0	0
3	Display current temperature in a simplest possible way	13	0	0	0	0	0
4	Set up the web server for serving weather data	3	0	0	0	0	0
5	Implement stubby WeatherML support on the server side	13	0	0	0	0	0
Sprint 1	*Make sample data go from server to device*						
6	Graphics support on the client side	20	0	0	0	0	0
16	Make the graphics library draw some icon and sample temperature text	-	13	0	0	0	0
17	Draw the real weather screen	-	8	0	0	0	0
7	Implement support for several days	8	8	0	0	0	0
8	Implement support for rain, snow, etc. icons	2	2	0	0	0	0
9	City changing support	-	5	0	0	0	0
Sprint 2	*Minimal working version*						
10	Fetch one day temperature data from the weather provider system	?					
11	Fetch rain, snow, etc details from the provider						
12	Fetch several days data from the provider						
13	Auto-refresh feature						
Sprint 3	*Plug in the real weather data*						
Release 1	**Sellable version**						
14	Inject simulated ads from the test server						
15	Plug real ads in						
18	Change current city automatically according to the cell info	40	40	40	40	40	40
Sprint 4	*Advertisements support*						
Release 2	**Ad-supported version**						
	Effort in the whole backlog	170	150	114	80	80	80

Backlog state taken after the end of sprint 3 = after release 1

In a real product backlog, items are more likely to be in the form of user stories. E.g. this item could be stated as "As a user I want to see the real weather data so that I could know what to wear before leaving home"

http://agilesoftwaredevelopment.com/scrum/simple-product-backlog

SCRUM

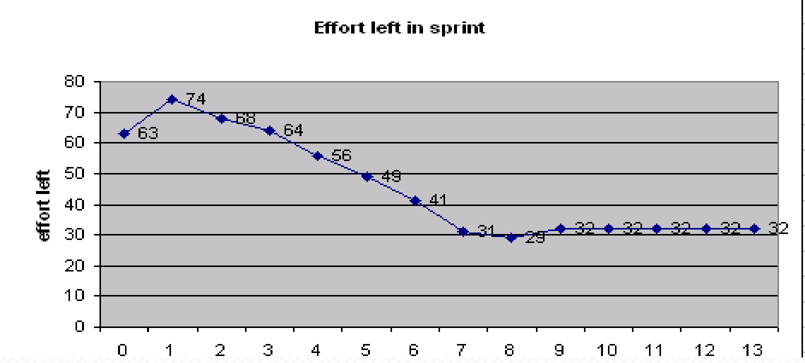
- Multiple teams can take on product increments in parallel, all working from the same Product Backlog. A project will have multiple sprints.
- Before a sprint starts a meeting is conducted to decide what is going to be developed and delivered in that particular sprint.

Example – Sprint Backlog

Sprint 3. Plug in the Real Weather

Story ID	Story/task	days in sprint / effort left													
		0	1	2	3	4	5	6	7	8	9	10	11	12	13
		63	74	68	64	56	49	41	31	29	32	32	32	32	32
10	**Fetch one day temperature data from the weather provider system**														
	Make our server connect and authenticate to the provider system	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	3	3	3
	Read provider's data directory	8	7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0
	Parse the current temperature out of the data	6	6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	1	1	1
	Push the temperature data to the client	16	16	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0
11	**Fetch rain, snow, etc details from the provider**														
	Parse snow/rain data from the provider's data	4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0
	Push the snow/rain data to the client	4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0
	Redesign client screen a bit			[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	3	3	3
	Refactor the server code										4	4	4	4	4
12	**Fetch several days data from the provider**														
	Parse the weather data in day packs	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	Push several days data to the client	3	3	3	3	3	3	3	3	3	3	3	3	3	3
13	**Auto-refresh feature**														
	Make the client ping server once per 4 hours	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	Make the server update the client	2	2	2	2	2	2	2	2	2	2	2	2	2	2

The team underestimated the difficulties of connecting to the provider's CORBA server. Even though some work has been already done, the remaining amount is much bigger, than the original estimation for the whole task



http://agilesoftwaredevelopment.com/scrum/simple-sprint-backlog

SCRUM

- After the completion of the sprint, a meeting is held to collect feedback from the team. This feedback helps in planning and working on the next sprint.
- As the development team starts to work on the next sprint, the testing team carries out functional testing of the features developed in the last sprint.
- This approach gives good results as the testing team works with the developers from the start of the project.

Scrum and XP

- Scrum teams work in iterations that are called sprints. These can last a little longer than XP iterations.
- Scrum teams do not allow changes to be introduced during the sprints. XP teams are more flexible with changes within an iteration as long as work has not started on that particular feature already.
- XP implements features in a priority order decided essentially by the customer, while in SCRUM there is more flexibility for additional stakeholders to influence the ordering.
- In XP unit testing and simple design practices are built in, while in SCRUM it is up to the team to organize themselves.