Software Engineering

Lecture 6
Agile Development

Mr. Noor Ul Arfeen

Common Fears for Developers

The project will produce the wrong product

The project will produce a product of inferior quality

The project will be late

We’ll have to work 80 hour weeks

We’ll have to break commitments

We won’t be having fun

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

Yielding …

Rapid, incremental delivery of software

An Agile Process

Is driven by customer descriptions of what is required (scenarios)

Recognizes that plans are short-lived

Develops software iteratively with a heavy emphasis on construction activities

Delivers multiple ‘software increments’

Adapts as changes occur

Principles of agile methods

Principle	Description
Customer involvement	The customer should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and design the system so that it can accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process used. Wherever possible, actively work to eliminate complexity from the system.

Agile Process Models

Extreme Programming (XP)

Scrum

Adaptive Software Development

Dynamic System Development Method (DSDM)

Crystal

Feature Driven Development

Agile Modeling (AM)

Extreme Programming (XP)

Perhaps the best-known and most widely used agile method

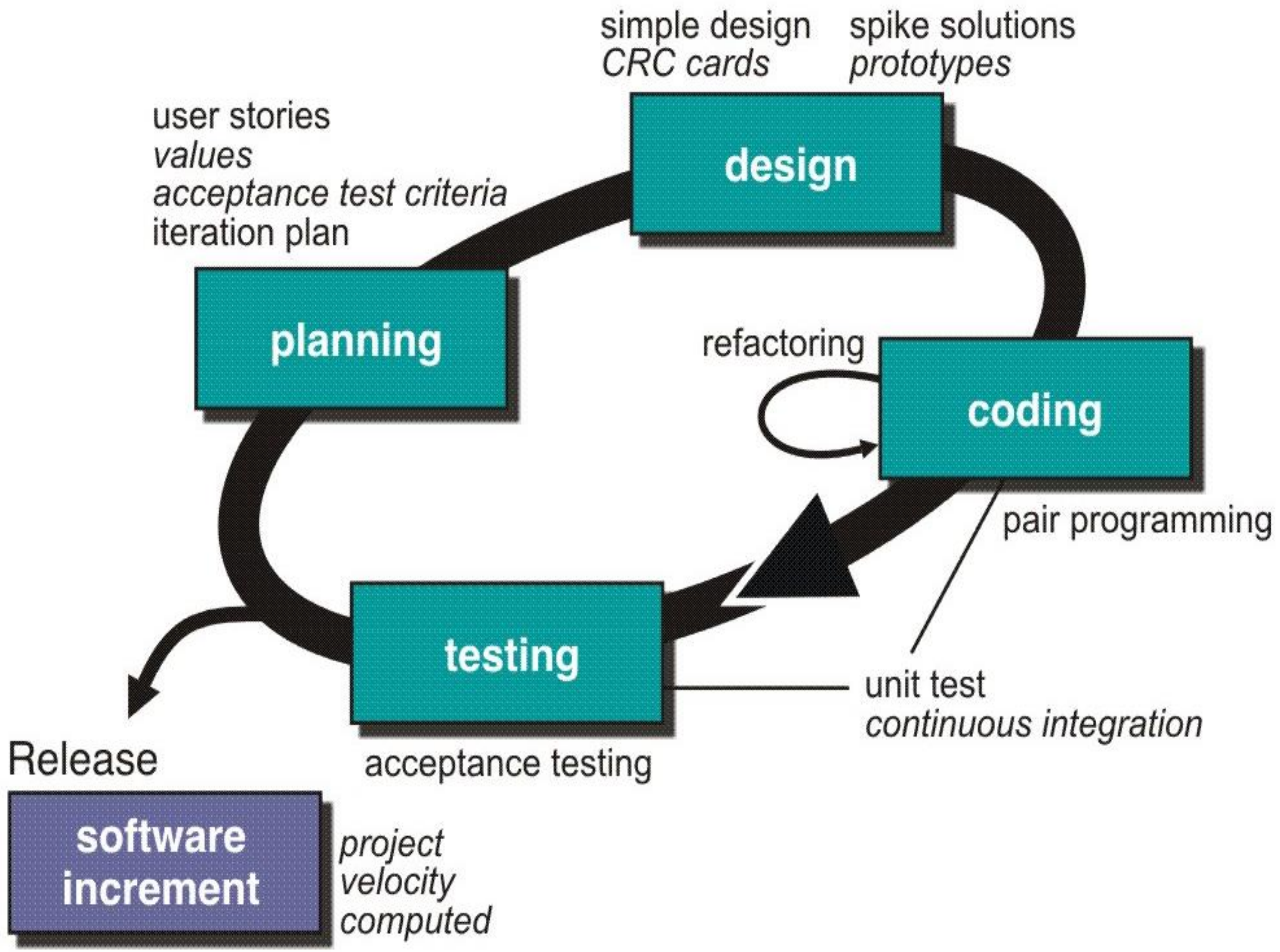
Extreme Programming (XP) takes an ‘extreme’ approach to iterative development

- ❑ New versions may be built several times per day;
- ❑ Increments are delivered to customers every 2 weeks;
- ❑ All tests must be run for every build and the build is only accepted if tests run successfully

XP Values

- ❑ Communication
- ❑ Simplicity
- ❑ Feedback
- ❑ Courage
- ❑ Respect

Extreme Programming (XP)



Extreme Programming (XP)

XP Planning

- ❑ Begins with the creation of user stories
- ❑ Agile team assesses each story and assigns a cost
- ❑ Stories are grouped to for a deliverable increment
- ❑ A commitment is made on delivery date
- ❑ After the first increment project velocity is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)

XP Design

- ❑ Follows the KIS (keep it simple) principle
- ❑ For difficult design problems, suggests the creation of spike solutions - a design prototype
- ❑ Encourages refactoring - an iterative refinement of the internal program design

XP Coding

- ❑ Recommends the construction of a unit test for a story before coding commences
- ❑ Encourages pair programming

XP Testing

- ❑ All unit tests are executed daily
- ❑ Acceptance tests are defined by the customer and executed to assess customer visible functionality

XP and agile principles

Incremental development is supported through small, frequent system releases

Customer involvement means full-time customer engagement with the team

People not process through pair programming, collective ownership and a process that avoids long working hours

Change supported through regular system releases

Maintaining simplicity through constant refactoring of code

Customer involvement

Customer involvement is a key part of XP where the customer is part of development team

The role of the customer is:

- ❑ To help develop stories that define the requirements
- ❑ To help prioritize the features to be implemented in each release
- ❑ To help develop acceptance tests which assess whether or not system meets its requirements

Requirements scenarios

In XP, user requirements are expressed as scenarios or user stories

These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates

The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates

Story card for document downloading

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it - this can either be through a subscription, through a company account or by credit card.

After this, you get a copyright form from the system to fill in and, when you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system if printing has been successful.

If the article is a print-only article, you canÕt keep the PDF version so it is automatically deleted from your computer.

Refactoring

Refactoring is the process of code improvement where code is reorganised and rewritten to make it more efficient, easier to understand, etc

Refactoring is required because frequent releases mean that code is developed incrementally and therefore tends to become messy

Refactoring should not change the functionality of the system

Automated testing simplifies refactoring as you can see if the changed code still runs the tests successfully

Testing in XP

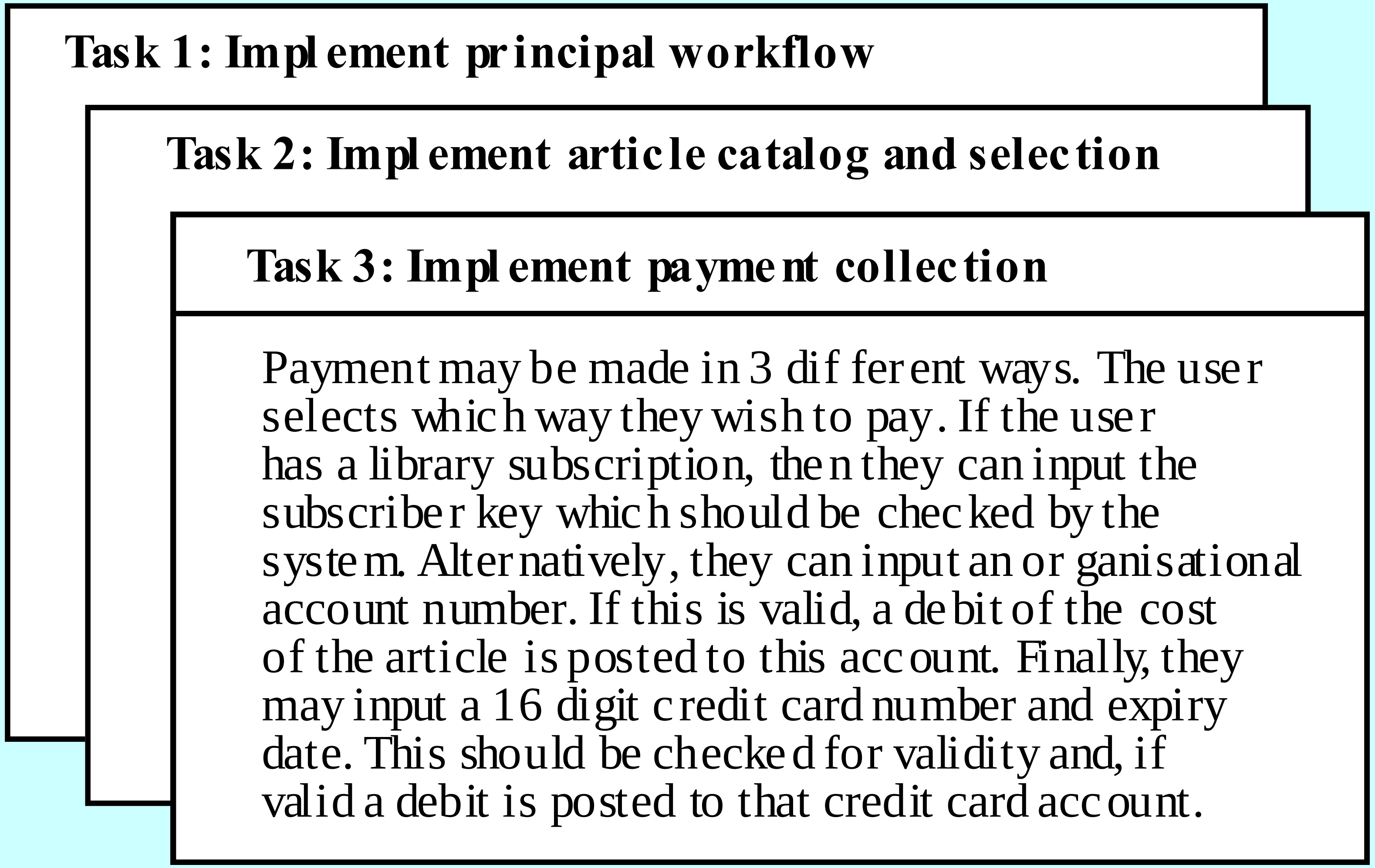
Test-first development

Incremental test development from scenarios

User involvement in test development and validation

Automated test harnesses are used to run all component tests each time that a new release is built

Task cards for document downloading



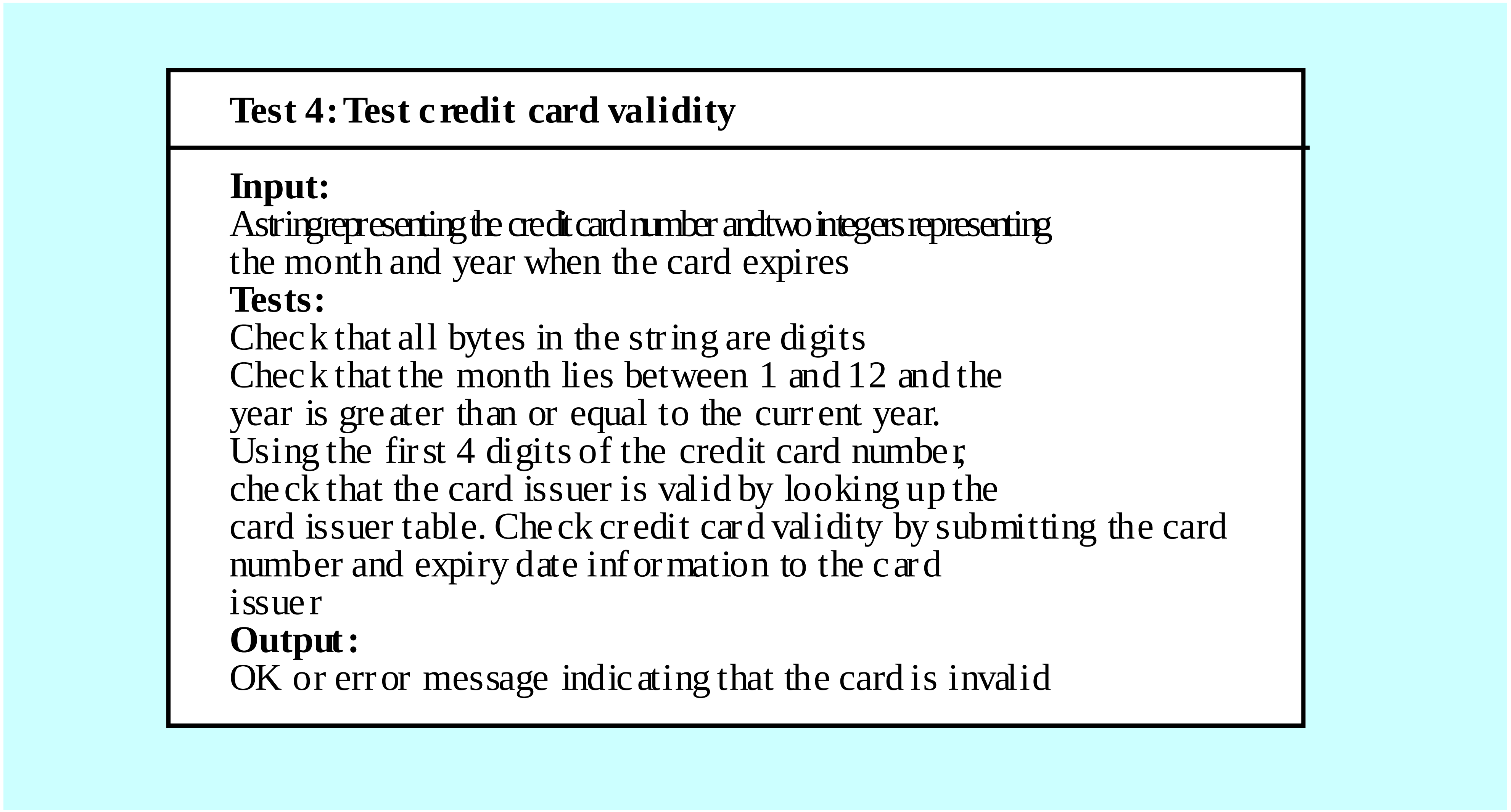
Task 1: Implement principal workflow

Task 2: Implement article catalog and selection

Task 3: Implement payment collection

Payment may be made in 3 different ways. The user selects which way they wish to pay. If the user has a library subscription, then they can input the subscriber key which should be checked by the system. Alternatively, they can input an organisational account number. If this is valid, a debit of the cost of the article is posted to this account. Finally, they may input a 16 digit credit card number and expiry date. This should be checked for validity and, if valid a debit is posted to that credit card account.

Text case description



Test 4: Test credit card validity

Input:
A string representing the credit card number and two integers representing
the month and year when the card expires
Tests:
Check that all bytes in the string are digits
Check that the month lies between 1 and 12 and the
year is greater than or equal to the current year.
Using the first 4 digits of the credit card number,
check that the card issuer is valid by looking up the
card issuer table. Check credit card validity by submitting the card
number and expiry date information to the card
issuer
Output:
OK or error message indicating that the card is invalid

Pair Programming

In XP, programmers work in pairs, sitting together to develop code

This helps develop common ownership of code and spreads knowledge across the team

It serves as informal review process as each line of code is looked at by more than 1 person

It encourages refactoring as the whole team can benefit from this

Measurements suggest that development productivity with pair programming is similar to that of two people working independently

Problems with XP

Customer involvement

- This is perhaps the most difficult problem. It may be difficult or impossible to find a customer who can represent all stakeholders and who can be taken off their normal work to become part of the XP team. For generic products, there is no ‘customer’ - the marketing team may not be typical of real customers

Architectural design

- The incremental style of development can mean that inappropriate architectural decisions are made at an early stage of the process
- Problems with these may not become clear until many features have been implemented and refactoring the architecture is very expensive

Test complacency

- It is easy for a team to believe that because it has many tests, the system is properly tested
- Because of the automated testing approach, there is a tendency to develop tests that are easy to automate rather than tests that are ‘good’ tests

Scrum

Scrum is an agile method that focuses on managing iterative development rather than specific agile practices

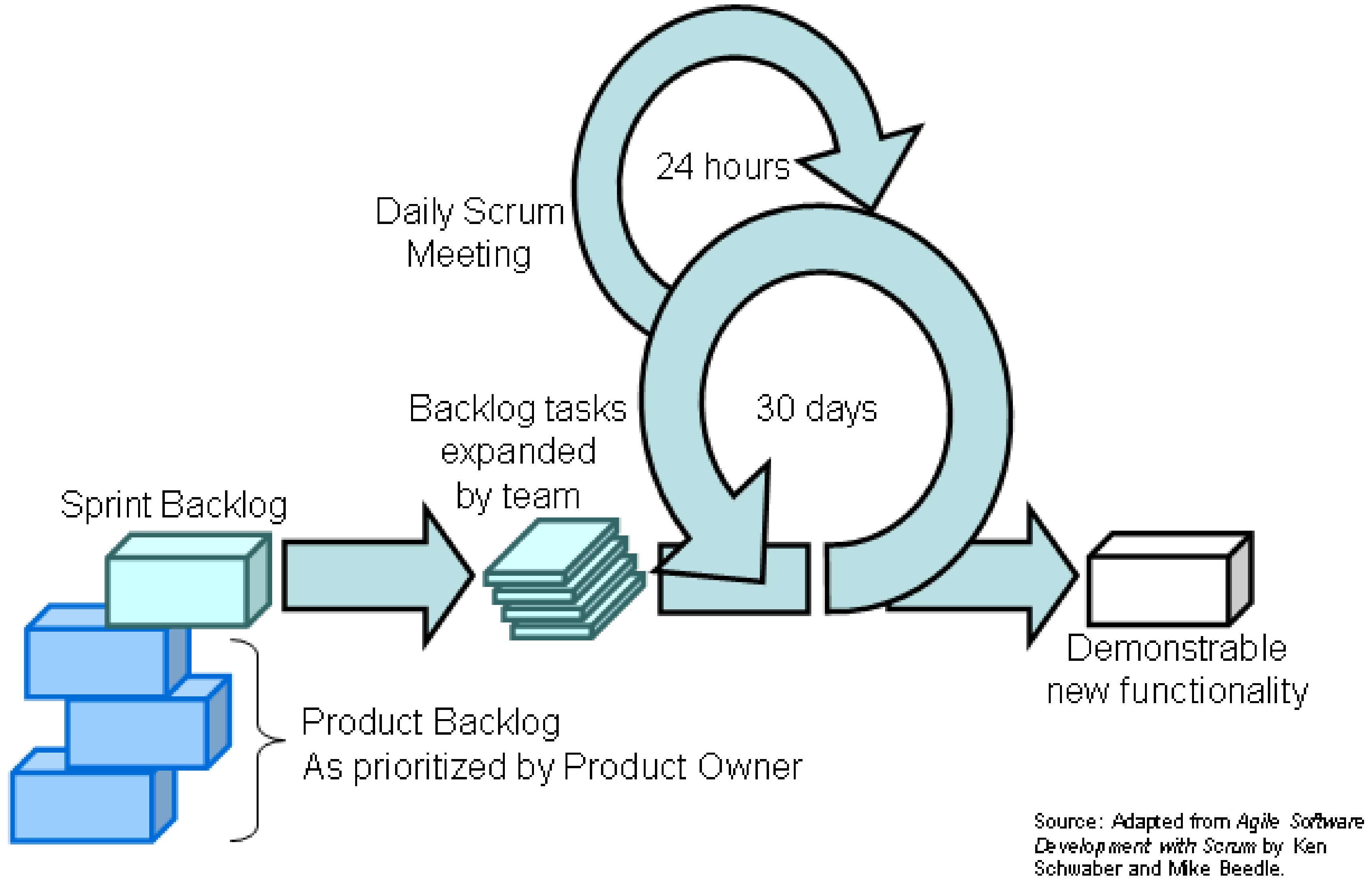
A scrum (short for scrummage) is a method of restarting play in rugby football that involves players packing closely together with their heads down and attempting to gain possession of the ball

Scrum

Emphasizes use of a set of software patterns

- ❑ Backlog
- ❑ Sprints
- ❑ Scrum meetings

Scrum


24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Demonstrable new functionality
Product Backlog
As prioritized by Product Owner
Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Scrum’s Roles

The Product Owner

The Scrum Master

The Team

Everyone else is not part of Scrum

Scrum’s Practices

The Sprint Planning Meeting

The Sprint

The Sprint Review Meeting

The Daily Scrum

Everything else is not part of Scrum

The Sprint Planning Meeting

Product Owner describes highest priority features to the Team

Team decides what the can commit to delivering in the Sprint

The Sprint Review Meeting

Time boxed to one hour of prep and four hours of meeting

Team demonstrates product increment to product owner’s satisfaction

Informality is encouraged. PowerPoint is discouraged

The Daily Scrum

Time boxed to fifteen minutes!

The Team and the Scrum Master only

What have you accomplished since yesterday?

Are your Sprint Backlog estimates accurate?

What are you working on today?

The Sprint Retrospective

Time boxed to three hours

Team, Scrum Master, and (optionally) Product Owner review the last Sprint

What went well?

What can be improved?

Actionable items are presented to the Product Owner for prioritization as non-functional requirements

Scrum’s Artefacts

The Product Backlog

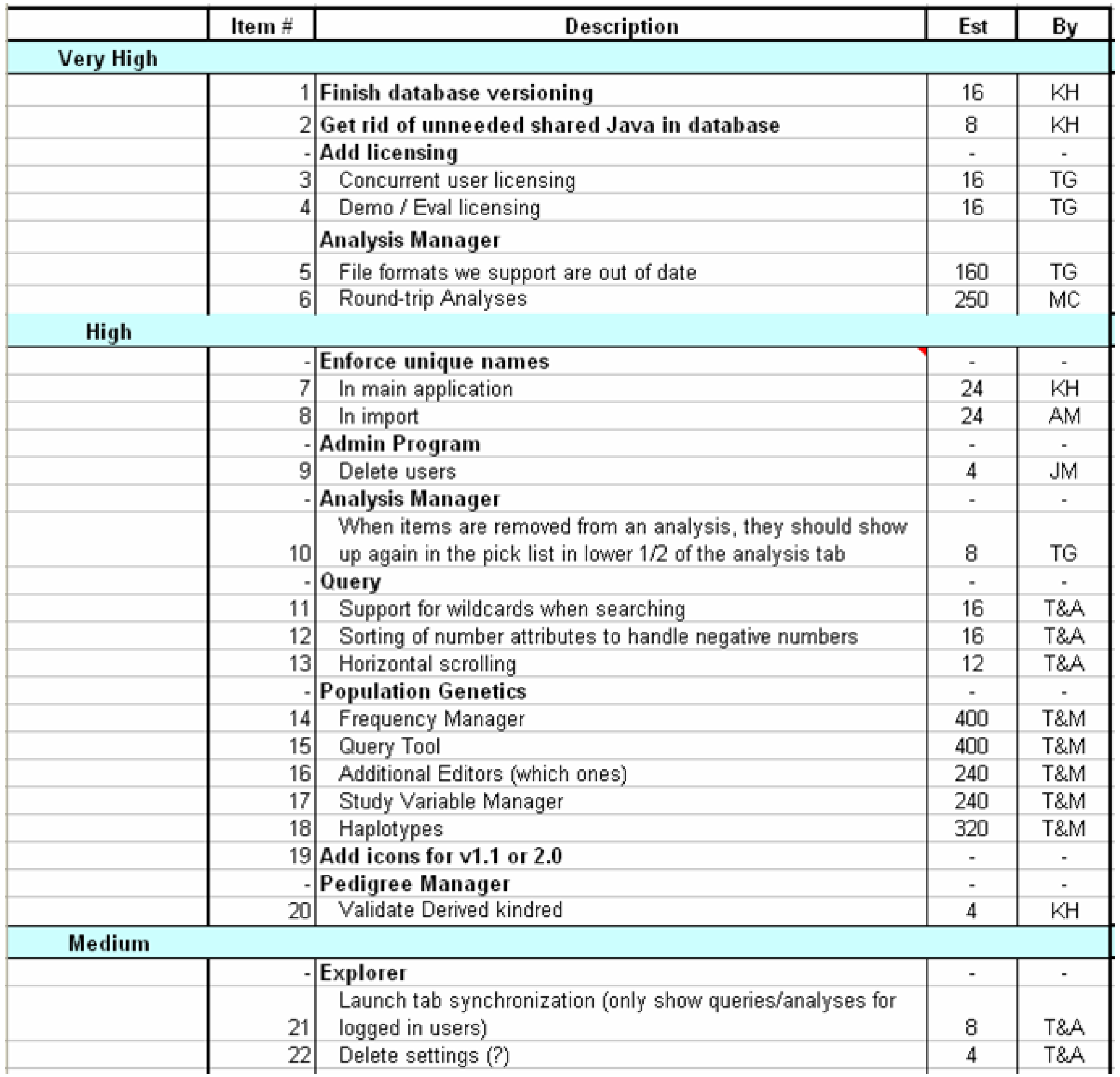
The Sprint Backlog

The Sprint Burndown Chart

The Product Increment

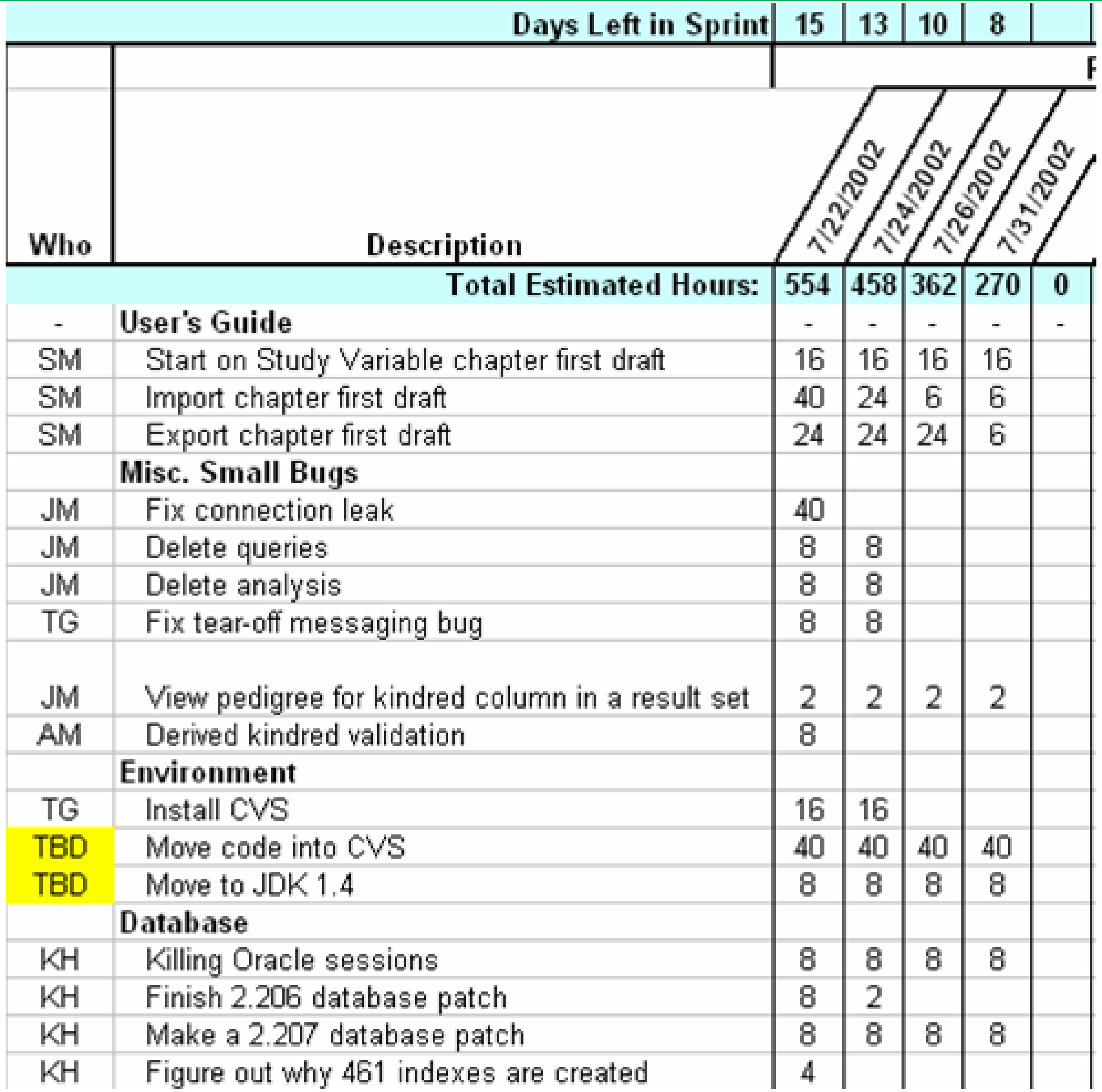
Everything else is not part of Scrum

The Product Backlog



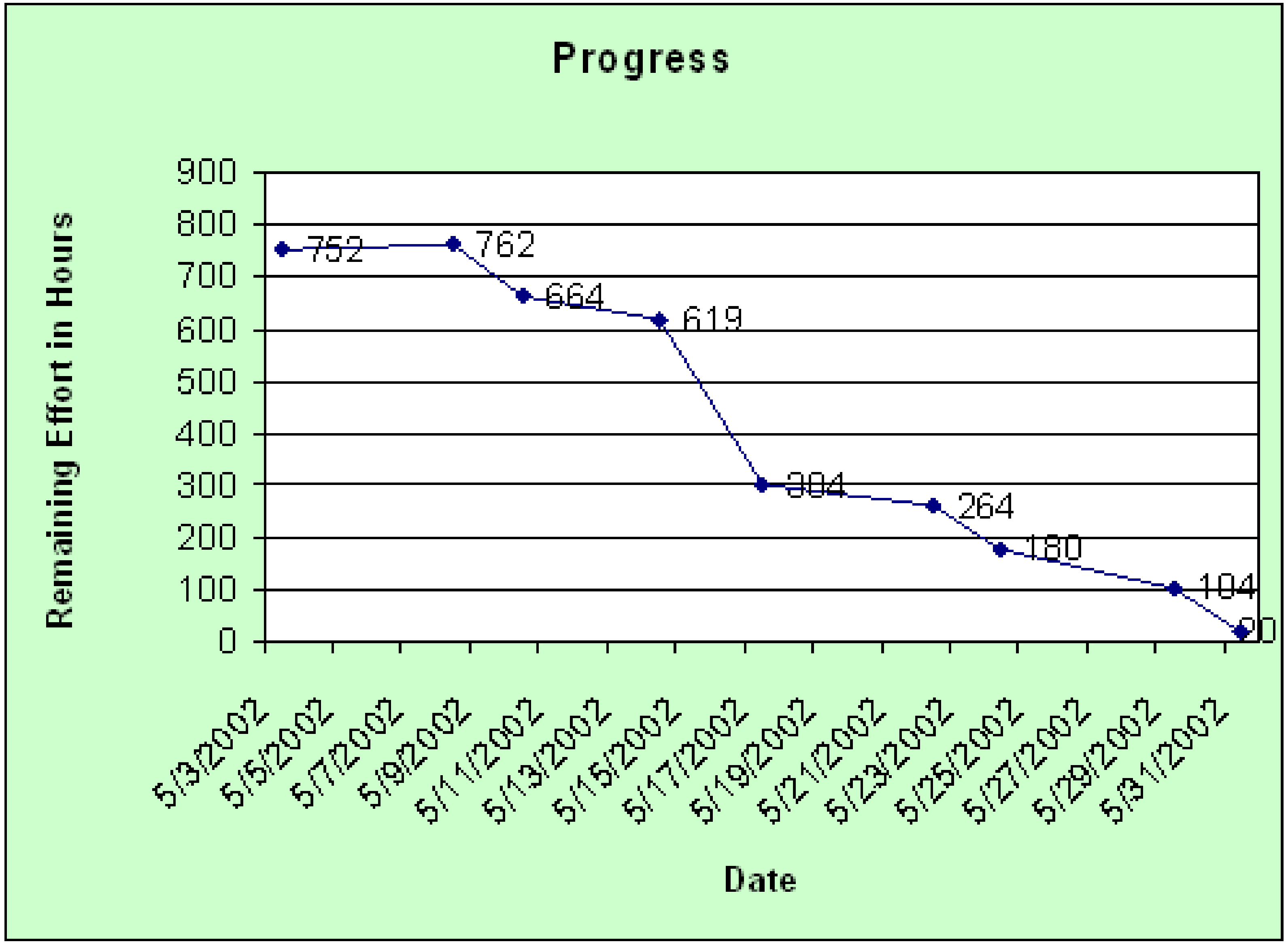
	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	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

The Sprint Backlog



Who	Description	7/22/2002	7/24/2002	7/26/2002	7/31/2002	
	Days Left in Sprint	15	13	10	8	
	Total Estimated Hours:	554	458	362	270	0
-	**User's Guide**	-	-	-	-	-
SM	Start on Study Variable chapter first draft	16	16	16	16	
SM	Import chapter first draft	40	24	6	6	
SM	Export chapter first draft	24	24	24	6	
	Misc. Small Bugs					
JM	Fix connection leak	40				
JM	Delete queries	8	8			
JM	Delete analysis	8	8			
TG	Fix tear-off messaging bug	8	8			
JM	View pedigree for kindred column in a result set	2	2	2	2	
AM	Derived kindred validation	8				
	Environment					
TG	Install CVS	16	16			
TBD	Move code into CVS	40	40	40	40	
TBD	Move to JDK 1.4	8	8	8	8	
	Database					
KH	Killing Oracle sessions	8	8	8	8	
KH	Finish 2.206 database patch	8	2			
KH	Make a 2.207 database patch	8	8	8	8	
KH	Figure out why 461 indexes are created	4				

The Sprint Burndown Chart



Some reasons to avoid Scrum

Your current software development produces acceptable results

Your project cannot be decomposed into good, increment-able requirements (“big ball of mud”)

Your engineering practices embrace heavy, up-front design, the construction of baroque frameworks, and throw-it-over-the-wall attitudes towards QA

Nobody can agree on ‘done-ness’

Your management practices embrace ‘do it now and forget what I told you to do yesterday’