Requirements Agility with the Rational Unified Process

Mark Lines

Co-founder, Unified Process Mentors

Mark@UPMentors.com

IBM Rational Software Development Conference 2008

WHERE TEAMS ARE R-HEROES

IBM

UPM UNIFIED PROCESS MENTORS

Rational. software

Disclaimers...

- Speaker will tend to make sweeping generalizations
- This is not an Agile methodology bashing session
- Not interested in methodology wars
  - There is no perfect methodology, ideal is a blend
  - Also see P06 “Process Wars” at 3:50pm today in the Sloane room
- We love Agile!
  - Much of our work these days is related to applying “practical” Agile techniques into organizations
- When I say “Unified Process”, you could substitute
  - Rational Unified Process, OpenUP, Agile UP, Others...
  - Practices are consistent

Agenda

- “Classic” Agile vs. Agile Unified Process
- Requirements styles:
  - Use Cases
  - User Stories
- Successfully applying User Stories on a Unified Process project
- Tips from the Trenches

Preliminaries

- We all understand that:
  - Iterative development is far superior to a linear process that assumes that all can be understood and planned upfront prior to coding (waterfall)
  - Agile approaches are beneficial to cutting costs and delivering working software earlier to stakeholders
  - Agile methodologies include:
    - Extreme Programming (XP)
    - SCRUM
    - DSDM
    - Crystal
    - FDD
    - Unified Process ??

“Classic” Agile vs. Agile Unified Process

- Agile principles:
  - Deliver functionality of high business value first
  - Frequent and incremental deployments of functionality
  - Continuously reassess priorities and finish project when “done”
  - Design as you go, don’t waste time on work not related to current iteration
  - “Just In Time” Everything
- Unified Process principles (agrees with the above, but)
  - A Vision “end state” is important to understand before investing (business case)
  - Architecture matters
  - Risk deferral is not acceptable (risk mitigation priorities may conflict with user priorities)
  - Stakeholders want commitments before the end state is achieved

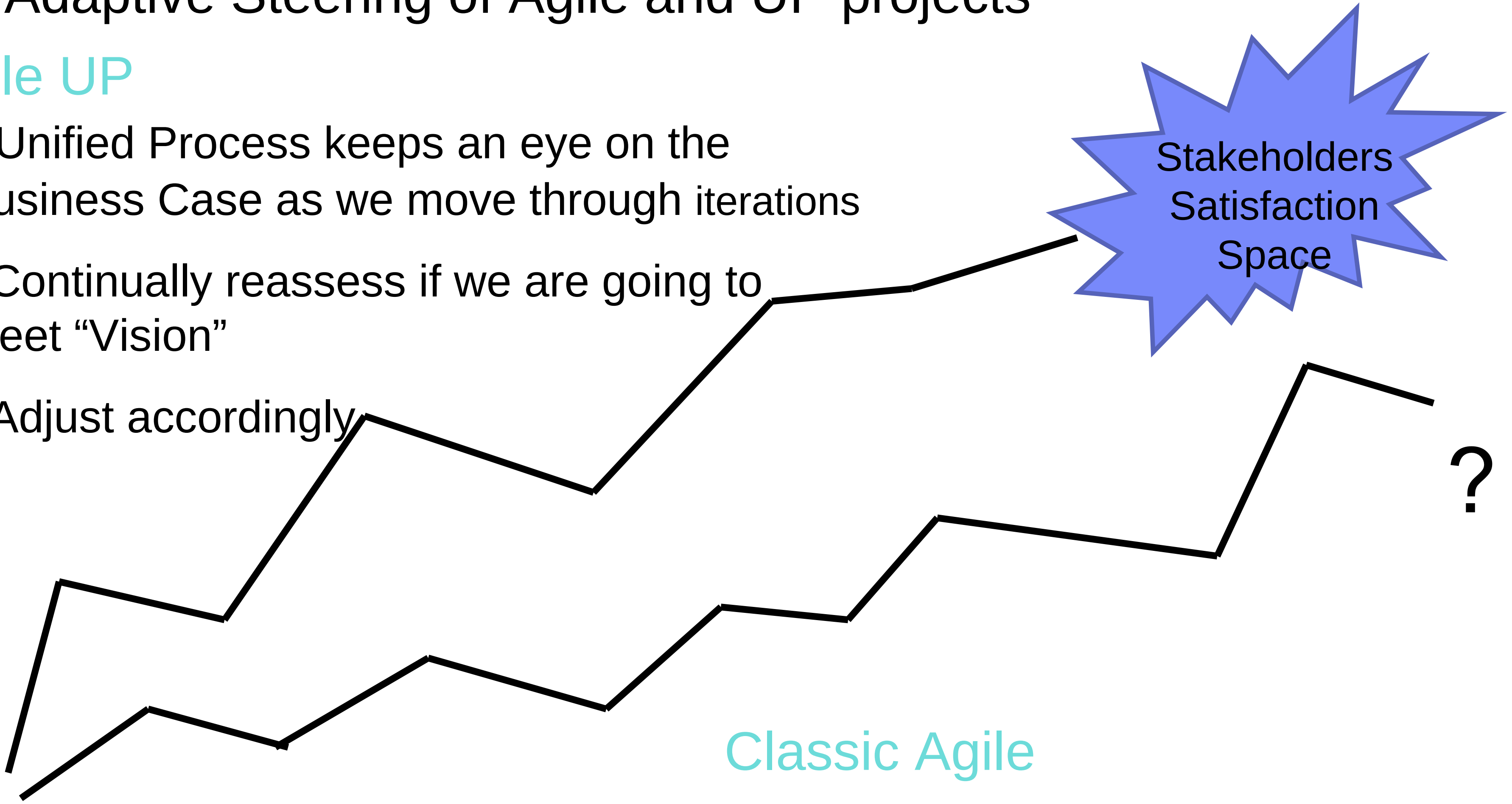
Adaptive Steering of Agile and UP projects

Agile UP

- Unified Process keeps an eye on the Business Case as we move through iterations

- Continually reassess if we are going to meet “Vision”

- Adjust accordingly

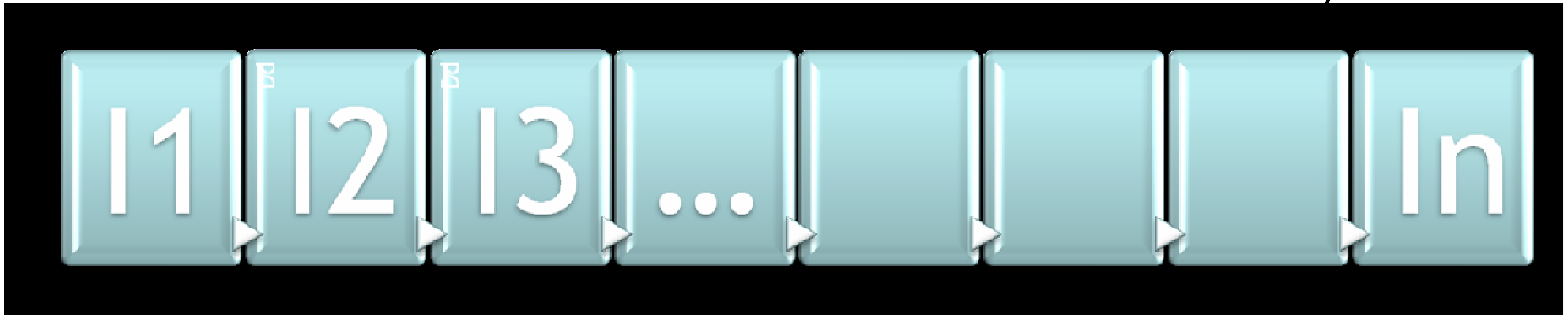


Classic Agile

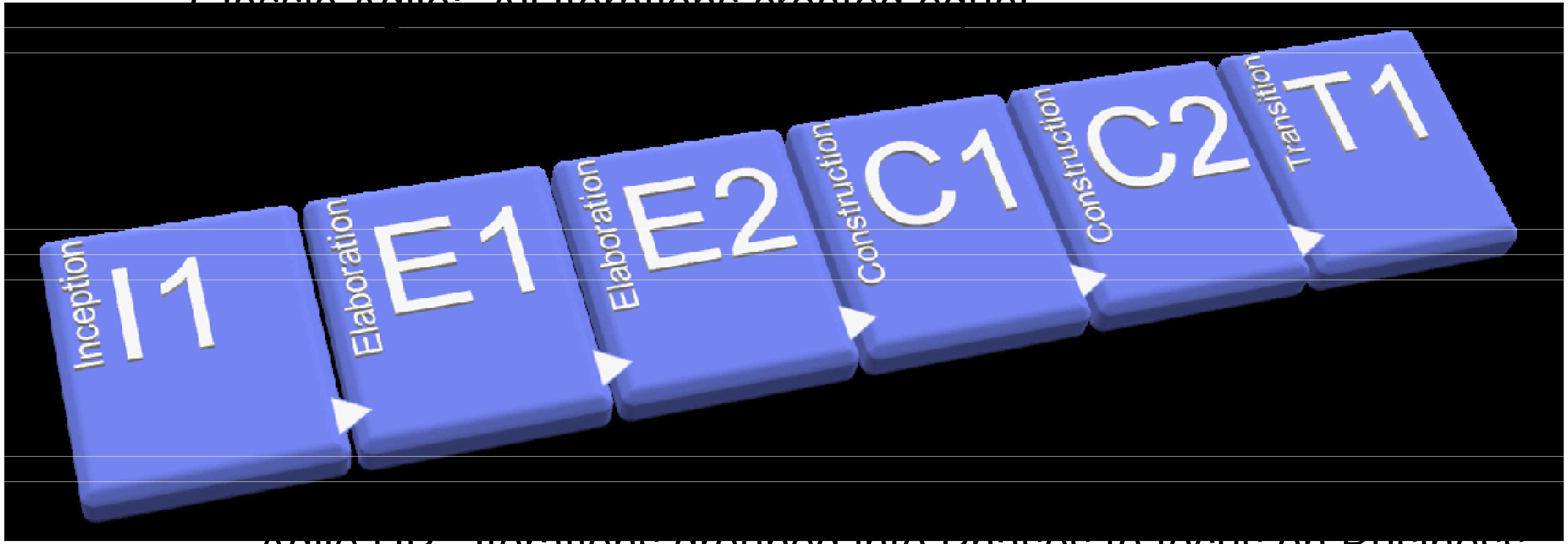
- Worry only about current iteration

- We will know when we are done

Unified Process Executes Iterations with End-state continuously in focus



Classic Agile: All Iterations created equal



Agile UP: Iterations grouped into Phases to focus on Business Milestones

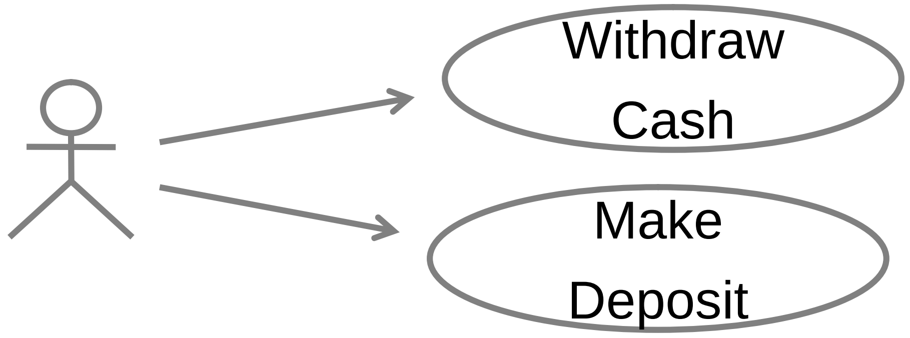
Let’s Assume the following UP principles…

- Value of a Business Case before substantial investment
- Value of making a commitment on deliverables BEFORE actual delivery (i.e. some predictability)
- Surprises late in the project are not good
  - Removing uncertainties early as possible is beneficial

Requirements Styles – Use Cases

Use Cases

- Describe a real “usage” scenario of interaction between a user (actor) and the system
- Puts requirements into a context of a workflow



Use Case example (outline level)

Withdraw Cash

- Basic Flow
  - Customer Logs on
  - Customer selects to withdraw cash
  - System validates withdrawal
  - System debits account with withdrawal amount
  - System dispenses cash
  - Use Case ends
- Alternative Flows
  - Print Receipt
  - Invalid Pin #
- Scenarios
  - Customer withdraws cash (Basic Flow)
  - Customer withdraws cash and prints receipt (Basic Flow + Print Receipt)

“detailing” of these flows

Deciding to Implement this Scenario in an iteration requires...

Use Case Scenario Example (detailed)

Scenario: Withdraw Cash and Print Receipt

Basic Flow

1. Customer Logs on

   Customer inserts debit card
   System prompts for PIN #
   Customer enters PIN, system validates (according to BR123)
   System presents ATM Menu

2. Customer selects to withdraw cash

   Customer selects withdraw cash option
   System requests amount
   Customer enters amount
   System validates amount (according to BR124)

....

7. System offers to print Receipt

   System offers to print Receipt
   Customer select not to print Receipt

8. System returns debit card to customer

9. Use Case ends.

Alternative Flows

A1 Print Receipt

At Step 7, “System offers to print Receipt” of the Basic Flow, Customer select to print receipt. System prints receipt. Return to Step 8 of the Basic Flow

Detailed Use Cases are detailed requirements

- A typical Detailed level Use Case specification is 4 – 20 pages long
- Should be kept up to date after the system goes into production
- Provides benefits downstream in the lifecycle:
  - User documentation
  - Testing scenarios
  - Input to Object-oriented Analysis & Design/ Services design
- Can be difficult in that workflow is designed by users and analysts (rather than developers!)
- Scenarios should be detailed “just in time” in iteration in which they will be implemented
  - Alternatively, analysts can build up a backlog ahead of developers, in parallel as developers implement UCs

Do we need to Detail all Use Cases?

- Not necessarily
- All process & documentation in an adaptive process such as the Unified Process is negotiable
- Early in the lifecycle, we work on the trickiest UCs, so detailing these Scenarios is advised
- Later in the lifecycle, detailing of all functionality may have diminishing returns. Eg) Scenarios to maintain reference tables. Really necessary?
- Some organizations ONLY outline all Use Cases! i.e. an Agile, minimalist approach to requirements with UCs
- Decide based on your circumstances

Comparison of Use Case to User Story

Use Case

Withdraw Cash

Basic Flow

- Customer Logs on
- Customer selects to withdraw cash
- System validates withdrawal
- System debits account with withdrawal amount
- System dispenses cash
- Use Case ends

Alternative Flows

- Print Receipt
- Maximum daily withdrawal exceeded

Scenarios

- Customer withdraws cash (Basic Flow)
- Customer withdraws cash and prints receipt (Basic Flow + Print Receipt)

User Story

US19 A user can withdraw cash from their bank account using the ATM

Note: The ATM connects to the bank using the Interac network

Test printing a receipt

Test with insufficient cash in the User's account

Test that daily withdrawal limit of £800 is not exceeded

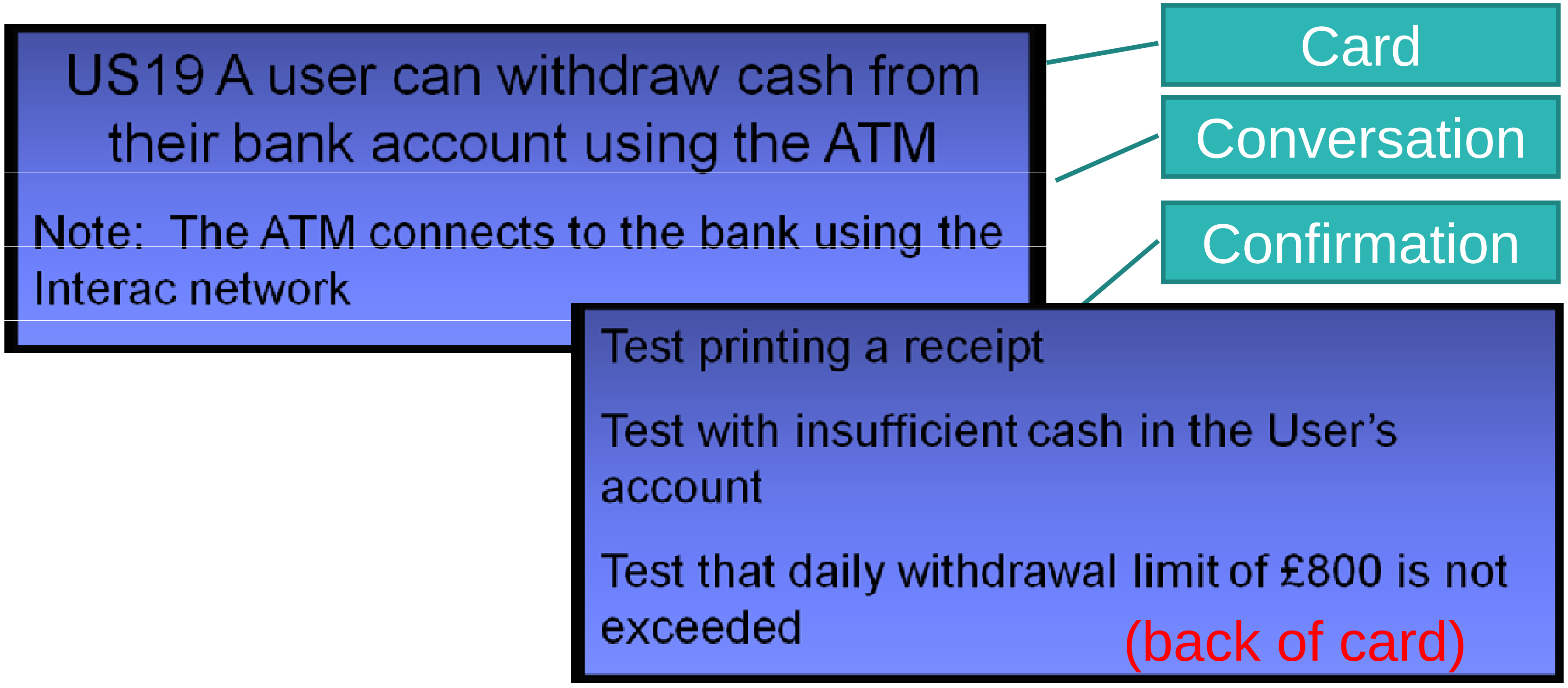
(back of card)

Evolving Requirements during the lifecycle – User Stories

- Classic Agile
  - User Stories at beginning of project
  - Not much investigation of functionality beyond general discussion (worry about it later when we build it)
  - Caution: May contain “scope bombs”

Requirements Styles – User Stories

- User Stories
  - One or two sentences to describe a “story” of functionality that is valuable to the User
  - “represent” requirements rather than “document” them


US19 A user can withdraw cash from their bank account using the ATM

Note: The ATM connects to the bank using the Interac network

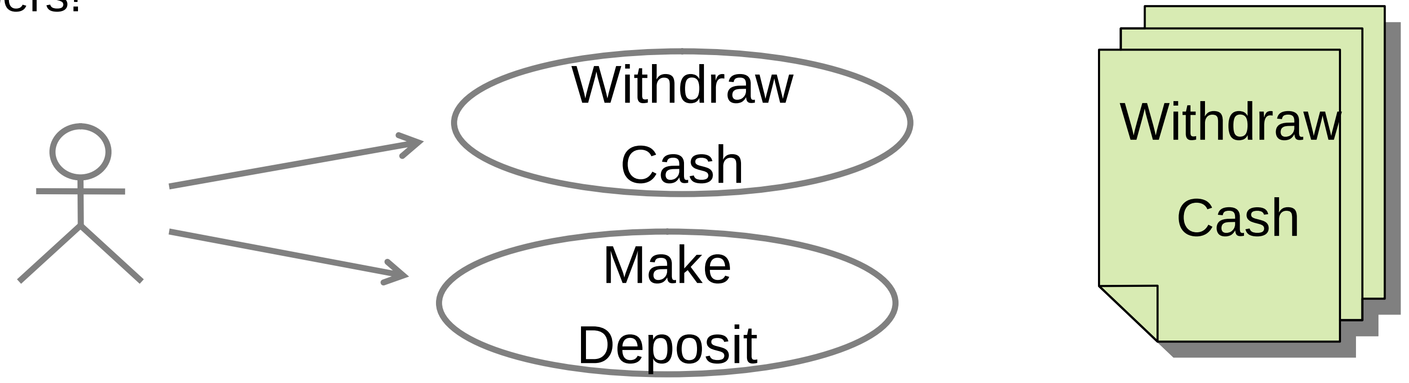
Test printing a receipt

Test with insufficient cash in the User’s account

Test that daily withdrawal limit of £800 is not exceeded

Evolving Requirements during the lifecycle – Use Cases

- Unified Process
  - Identify (via a Use Case diagram) and “Outline” all Use Cases in the first iteration (called Inception)
  - Outlining validates the *context* of the requirements in a workflow
  - Useful for fostering general understanding and aiding of risk identification
  - 2 pages per Use Case, double spaced, 1-2 hours to complete
  - Reminder: Sometimes hard, but better that users figure it out instead of developers!

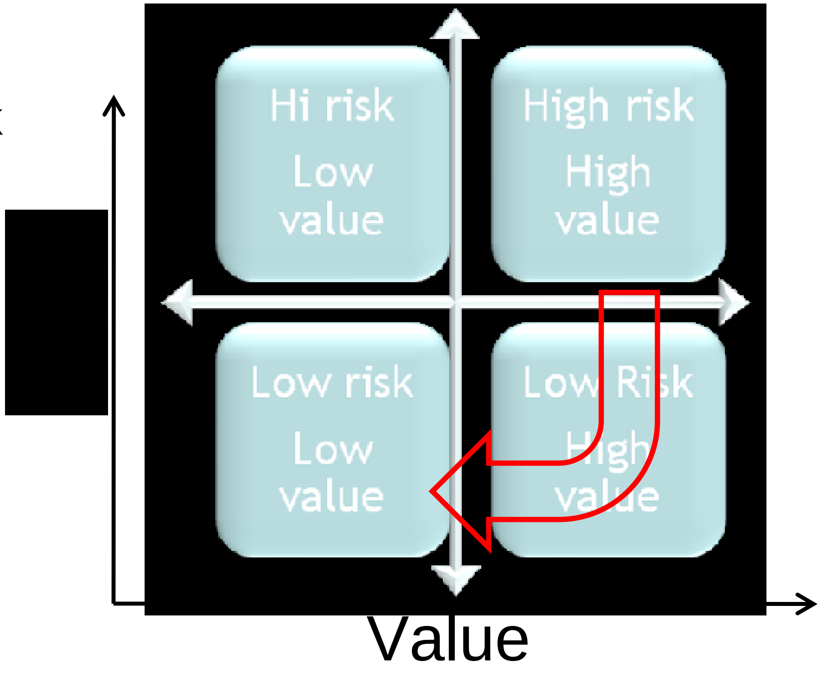


Comparison of status of Requirements EARLY in lifecycle

	Agile	Unified Process
Point in Lifecycle	Completed 1-2 days in, stories added as needed	1-2 weeks in (end of Inception), Scenarios added as needed, but NOT entire Use Cases
Size of requirements	Approx. 50-200 lines of text	10-30 pages
Comprised of:	10-30 index cards	Vision, UC diagram, Outlined UC Specs, System-wide Requirements (formerly Supplementary Specifications in RUP), Glossary
Lifetime of requirements	Cards ripped up at end of iteration in which built	Survive post-release of application

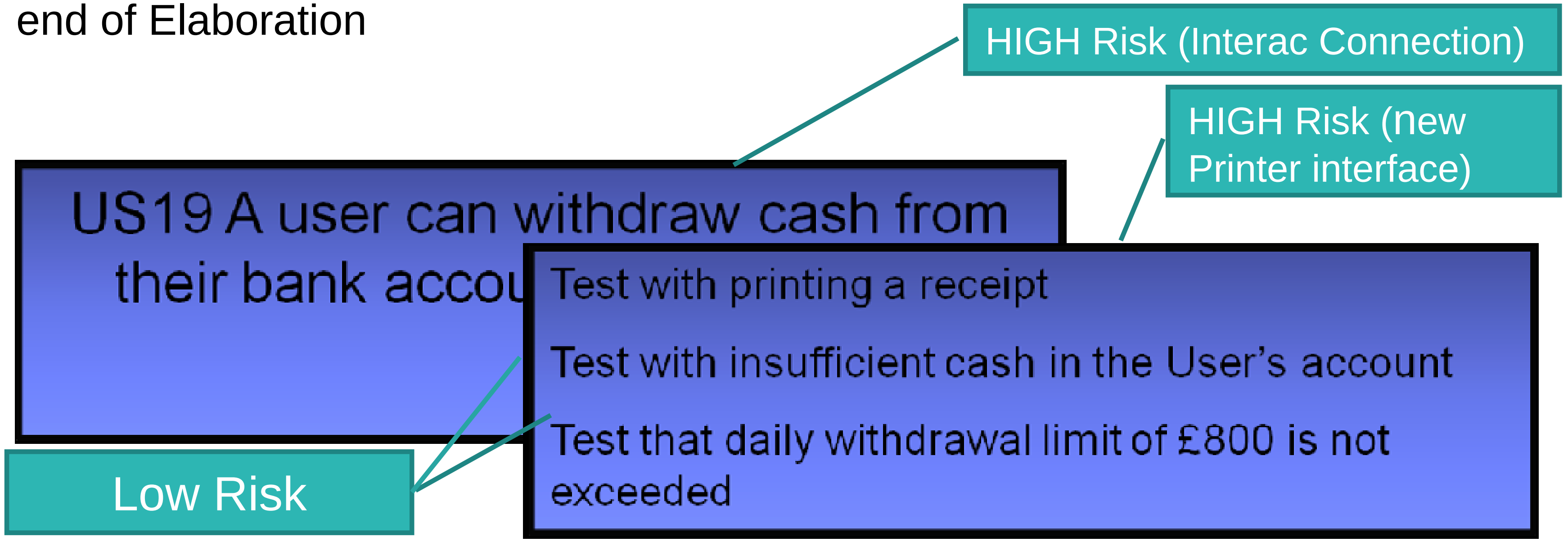
Implementing Your Requirements in Iterations

- Classic Agile
  - Pick the stories for current iteration off the stack with highest business value
    - “juicy bits”
- Unified Process
  - Pick the UC Scenarios that mitigate Risk
    - Architectural coverage
    - “tricky bits”
  - Priority setting changes in Construction
  - Try to combine both priorities



Where User Stories planning gets "sticky"

- User Story is a collection of Scenarios
- UP implements set of Scenarios in each iteration
- User Story contains a mix of high & low risk scenarios
- Implementing an entire User Story wastes time getting to an Elaboration milestone of mitigating highest risks
  - Key differentiator of UP is that it provides a CREDIBLE plan to stakeholders at end of Elaboration



HIGH Risk (Interac Connection)

HIGH Risk (new Printer interface)

US19 A user can withdraw cash from their bank accou

Test with printing a receipt

Test with insufficient cash in the User's account

Test that daily withdrawal limit of £800 is not exceeded

Low Risk

Solution – Split User Story into 2 Stories

US19 A user can withdraw cash from their bank account using the ATM

Test with printing a receipt

HIGH Risk

Implement in early iterations (Elaboration)

US20 The ATM system needs to validate certain conditions when the User withdraws cash

Note: Needs US19 to be implemented first

Test with insufficient cash in the User's account

Test that daily withdrawal limit of £800 is not exceeded

Low Risk

Implement in late iterations (Construction)

Tips from the Trenches – “Endless Projects”

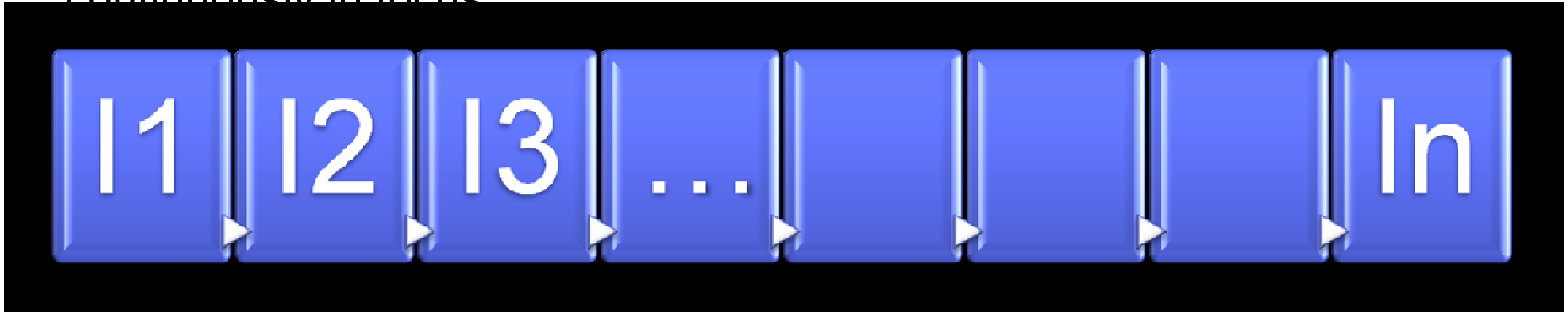
Beware the endless cycles of iterations!

New stories may be added, defects, reprioritized

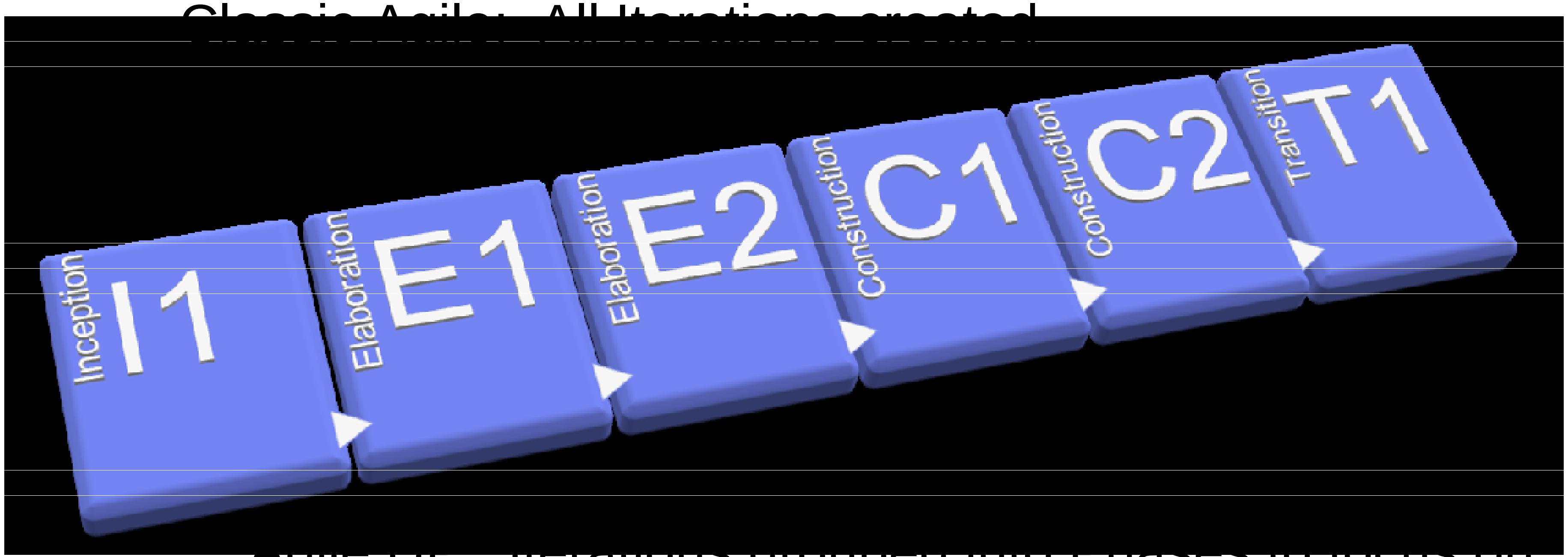
Agile takes cards off the top of the stack (new stories, defect fixes)

- Can become never ending!
- When are we done??
- This is where the discipline of the Unified Process can help us...

REVIEW - Unified Process Executes Iterations with End-state continuously in focus



Classic Agile: All Iterations created



Agile UP: Iterations grouped into Phases to focus on Business Milestones

Tips from the Trenches (cont.)

Allow requirements change early in Lifecycle, but tighten up in Construction

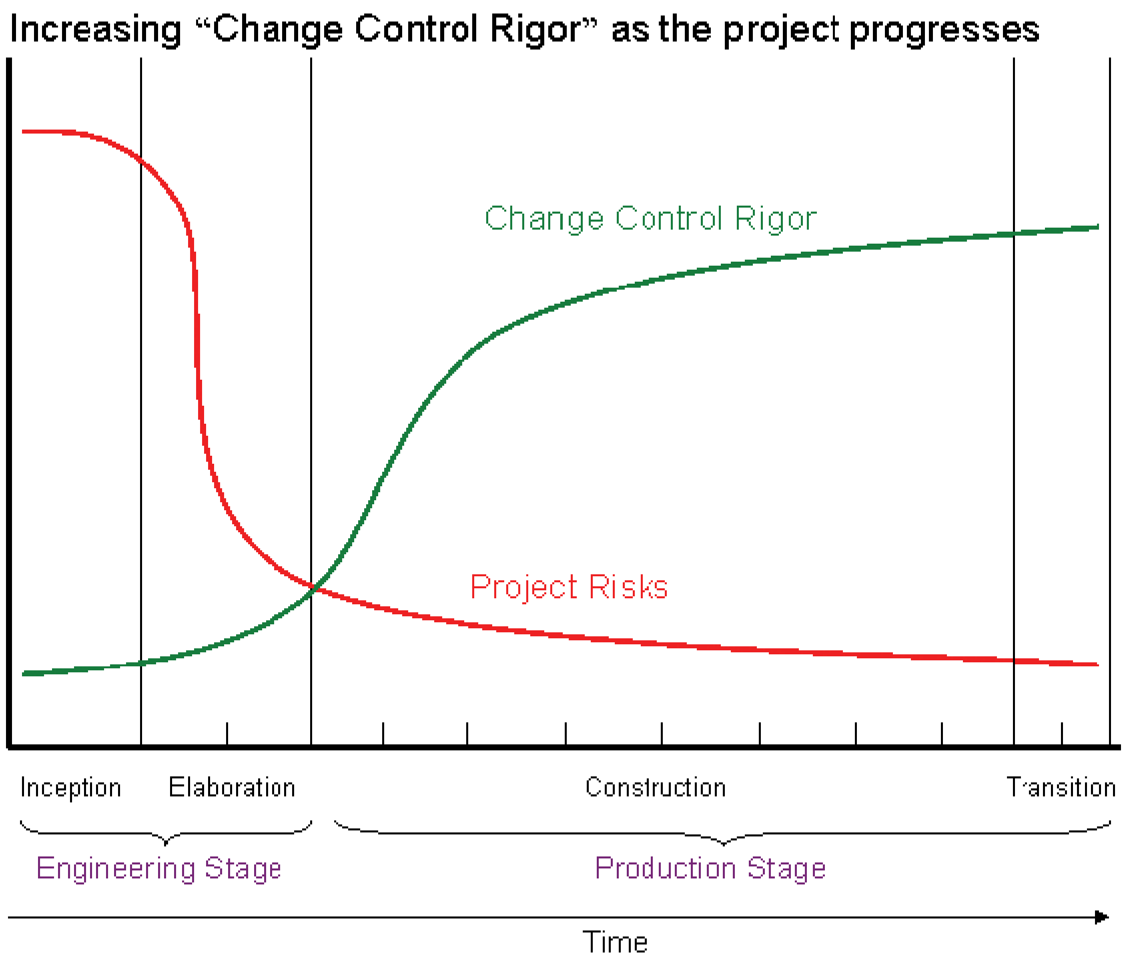
Phase	Iteration Length	Start	End
Inception			
I1	2 weeks	Jan 3	Jan 16
Elaboration			
E1	4 weeks	Jan 17	Feb 13
E2	4 weeks	Feb 14	Mar 13
Construction			
C1	3 weeks	Mar 14	Apr 3
C2	3 weeks	Apr 4	Apr 24
C3	3 weeks	Apr 25	May 15
Transition			
T1	2 weeks	May 16	May 29

Tips from the Trenches (cont.)

- The Unified Process assumes an overall plan with
  - Overall Vision, with a list of end-state Features
  - Overall Plan with a set schedule (Software Development Plan/Project Plan)
  - Usually unacceptable to tell management
    - "don't ask me to make commitments, we will know when we are done when we get there!"
  - UP controls requirements churn post-Elaboration

Phase	Iteration Length	Start	End
Inception			
I1	2 weeks	Jan 3	Jan 16
Elaboration			
E1	4 weeks	Jan 17	Feb 13
E2	4 weeks	Feb 14	Mar 13
Construction			
C1	3 weeks	Mar 14	Apr 3
C2	3 weeks	Apr 4	Apr 24
C3	3 weeks	Apr 25	May 15
Transition			
T1	2 weeks	May 16	May 29

Unified Process adjusts Project Management style during the Project Lifecycle vs. Classic Agile



Tips from the Trenches – “Team Buy-in”

- Agree amongst your team and stakeholders on acceptable level of requirement **“formality”**
  - Eg) Testers may expect more detailed requirements to test from that Analysts plan to produce
  - Eg) Production support/maintenance may expect detailed requirements to be able to support application
  - Sarbanes-Oxley, CMMI, etc may dictate a level of necessary formality
- Team should buy-in to the Iteration goal of building production-worthy, demonstrable, tested software
  - NOT just performing their “assigned activities”
    - (silo behaviour)

Summary

- Strive for minimum process without creating unreasonable project delivery risk
- Beware of Agile Hype
  - Seek to do what is practical for your organization
- Applications can be delivered much quicker and with higher quality with an adaptable and agile process
  - Key is understanding which shortcuts to take, in what circumstances
- A UP “agile coach” can help you through a few iterations

References

- Rational Edge article
- *Agile RUP: Experiences for the Trenches*
- Scott Ambler with Mark Lines, Joshua Barnes, Julian Holmes
  - http://www.UPMentors.com/publications
- Manifesto for Agile Software Development
  - http://agilemanifesto.org/
- IBM Rational Method Composer & RUP
  - http://www-306.ibm.com/software/awdtools/rmc/index.html

Questions?

Mark Lines

Mark@UPMentors.com

www.UPMentors.com

UPM > UNIFIED PROCESS MENTORS