Software Engineering

Agility and Process

Dr. Farrukh Zeshan

Agile Software Development

- Agile software development is an iterative approach to software development that emphasizes flexibility and collaboration. (small, working increments of software in short sprints, rather than one large release)

1. Identify the problem: Start by understanding the problem that needs to be solved and defining the goal of the software.
2. Form a cross-functional team: Assemble a team of developers, designers, testers, and other stakeholders who will be responsible for building the software. Giving them the freedom and authority to make decisions about how best to complete their work.

Agile Software Development

3. Create a product backlog: Write down all the features and requirements of the software in the form of user stories. This is called the product backlog.

4. Prioritize the backlog: Prioritize the user stories in the backlog based on their importance and dependencies.

5. Plan the sprint: Select the most important user stories from the backlog and plan the sprint. A sprint is a short iteration of work, usually two to four weeks, during which the team will work on delivering a usable portion of the software.

Agile Software Development

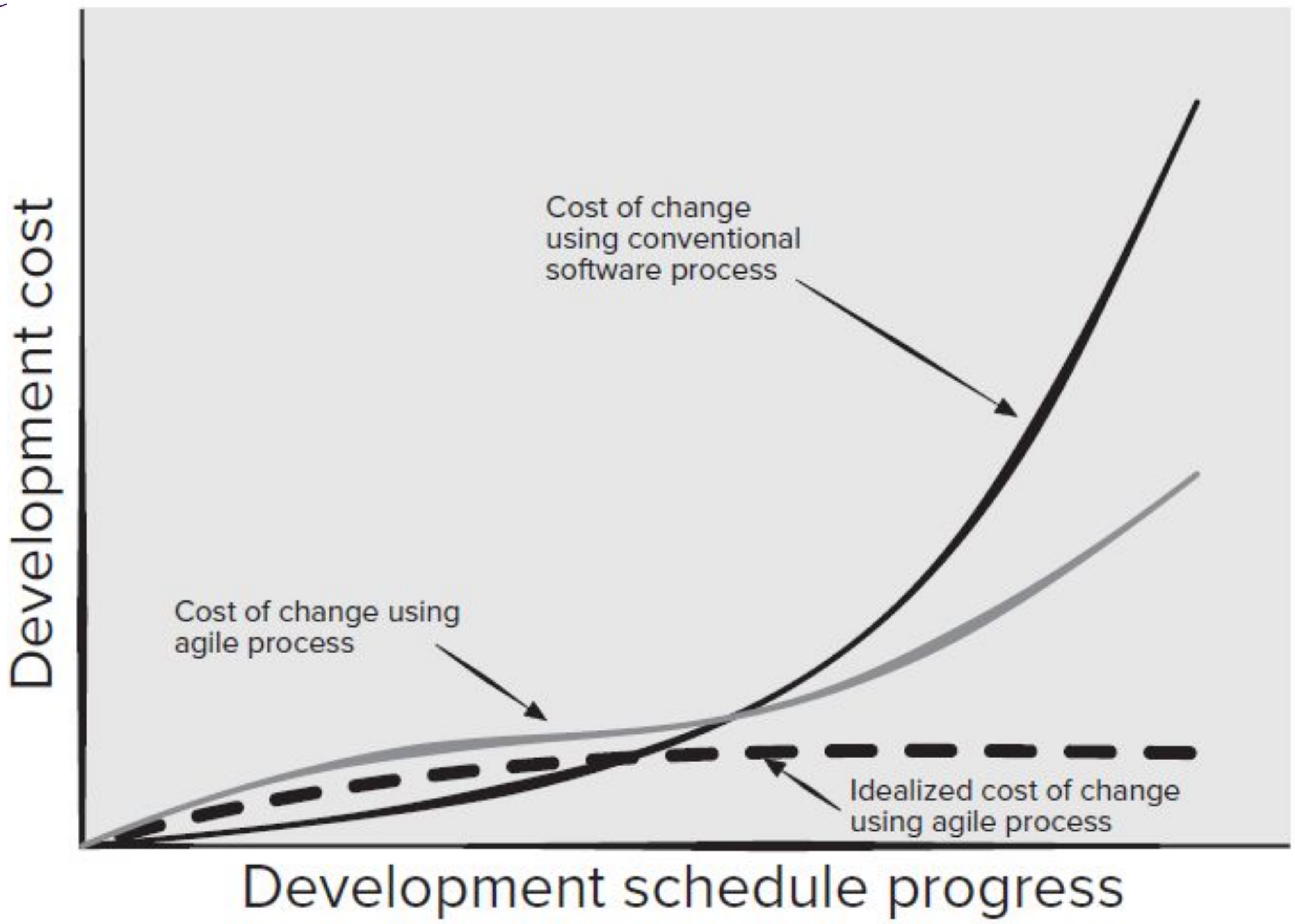
6. Start the sprint: Begin working on the user stories selected for the sprint. Use daily stand-up meetings to communicate progress and identify any obstacles that need to be addressed.

7. Continuously integrate and test: As the team works on the user stories, continuously integrate the changes into the codebase and test the software to ensure it works as expected.

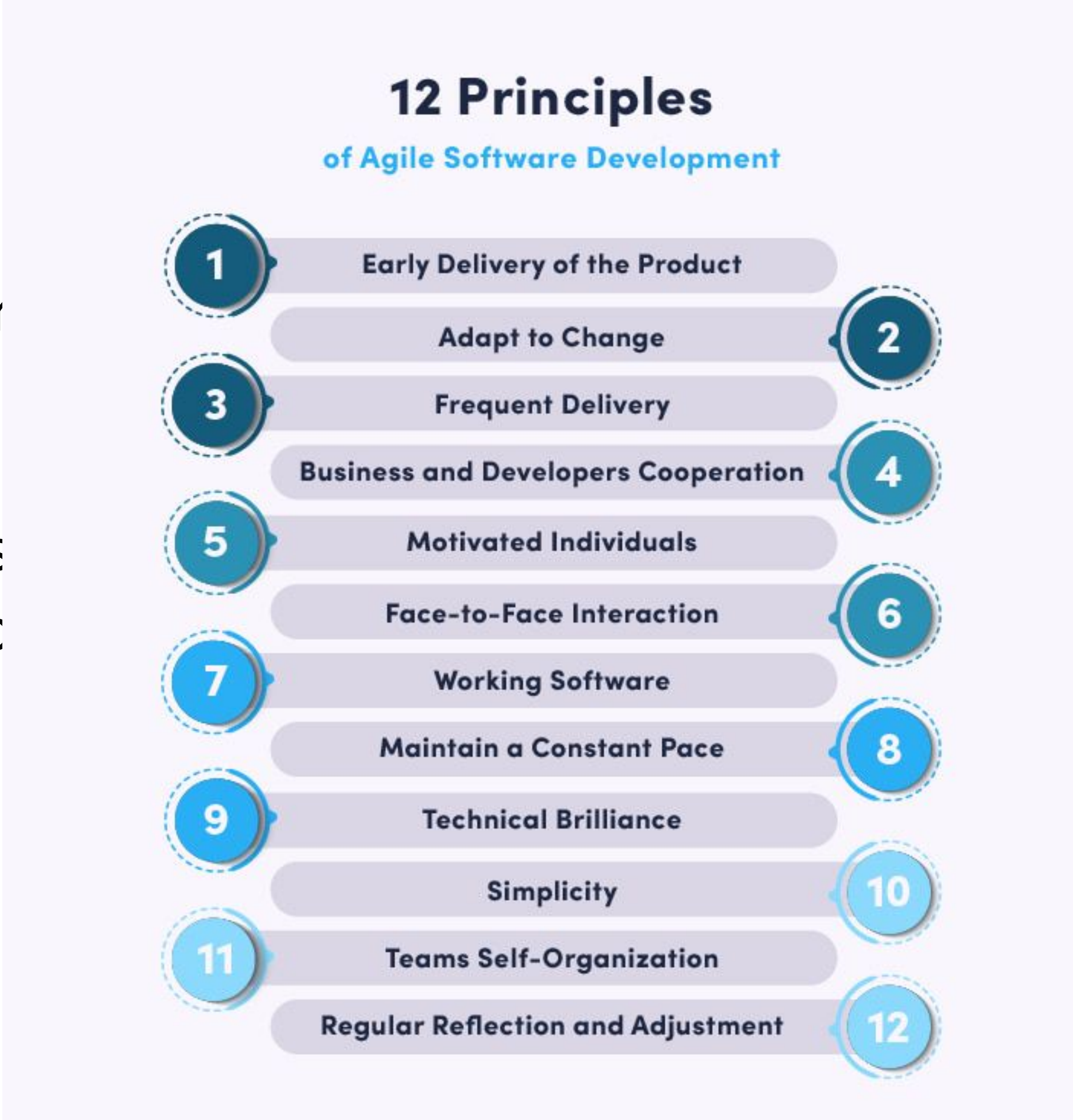
Agile Software Development

8. Review and retrospective: At the end of the sprint, conduct a review with stakeholders to demonstrate the work completed and gather feedback. Also, have a retrospective with the team to reflect on what went well and what can be improved.

9. Repeat: Repeat the sprint process, incorporating feedback from the review and retrospective, until the software is complete and meets the desired specifications.


Cost of change using conventional software process
Cost of change using agile process
Idealized cost of change using agile process
Development cost
Development schedule progress

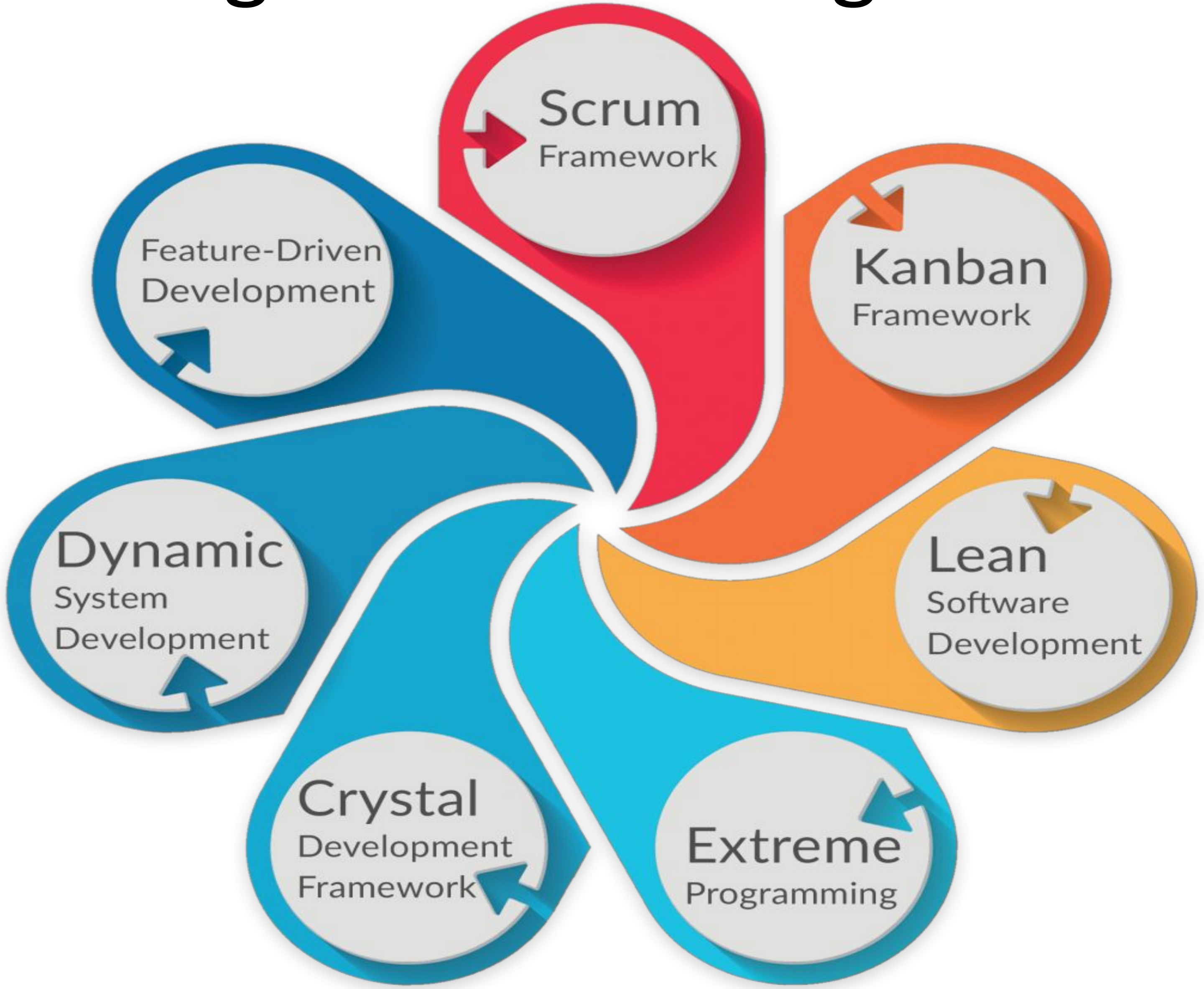
- Principles for those software organizations that want to achieve agility.


12 Principles

of Agile Software Development

1 Early Delivery of the Product
2 Adapt to Change
3 Frequent Delivery
4 Business and Developers Cooperation
5 Motivated Individuals
6 Face-to-Face Interaction
7 Working Software
8 Maintain a Constant Pace
9 Technical Brilliance
10 Simplicity
11 Teams Self-Organization
12 Regular Reflection and Adjustment

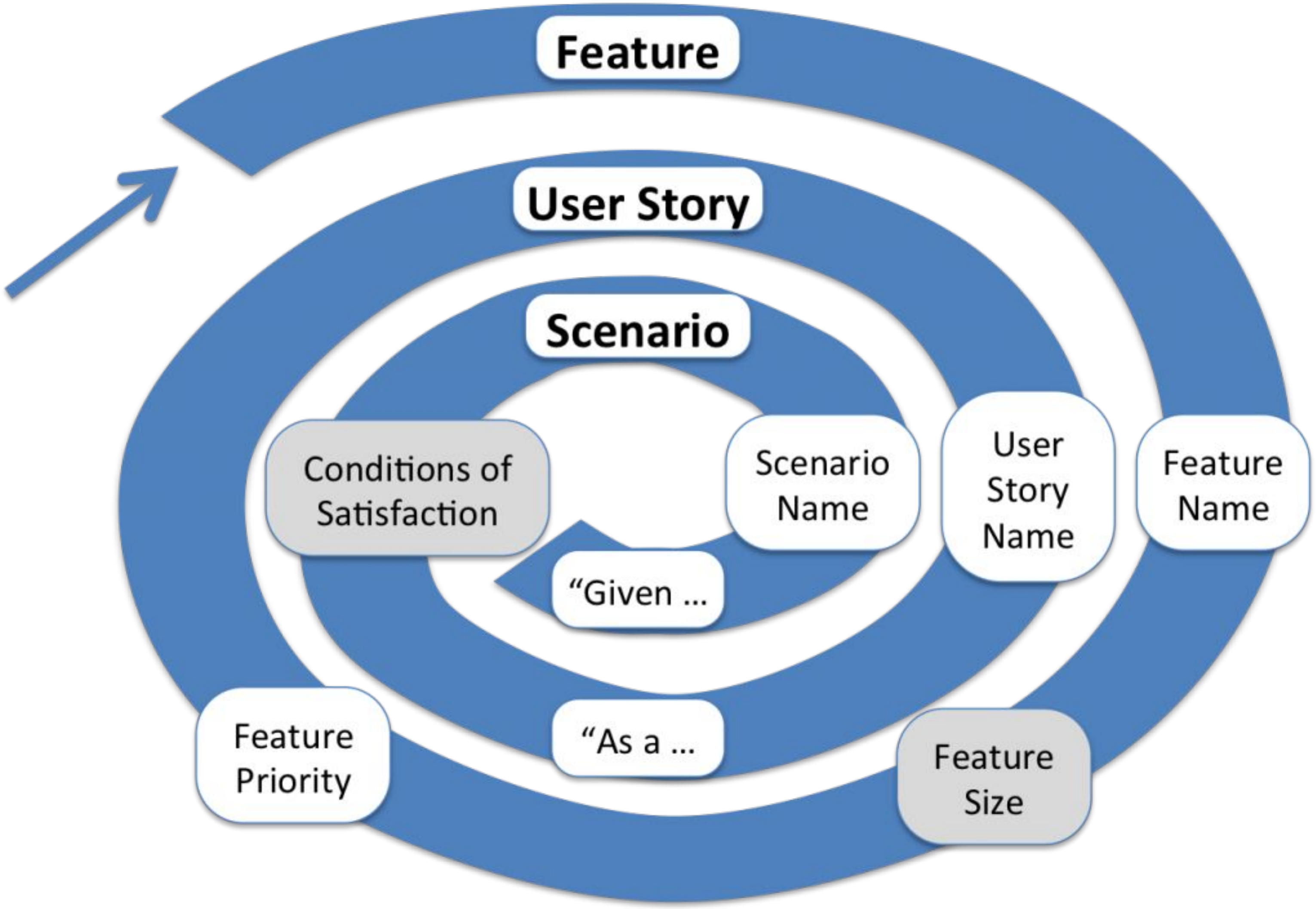
Agile Methodologies



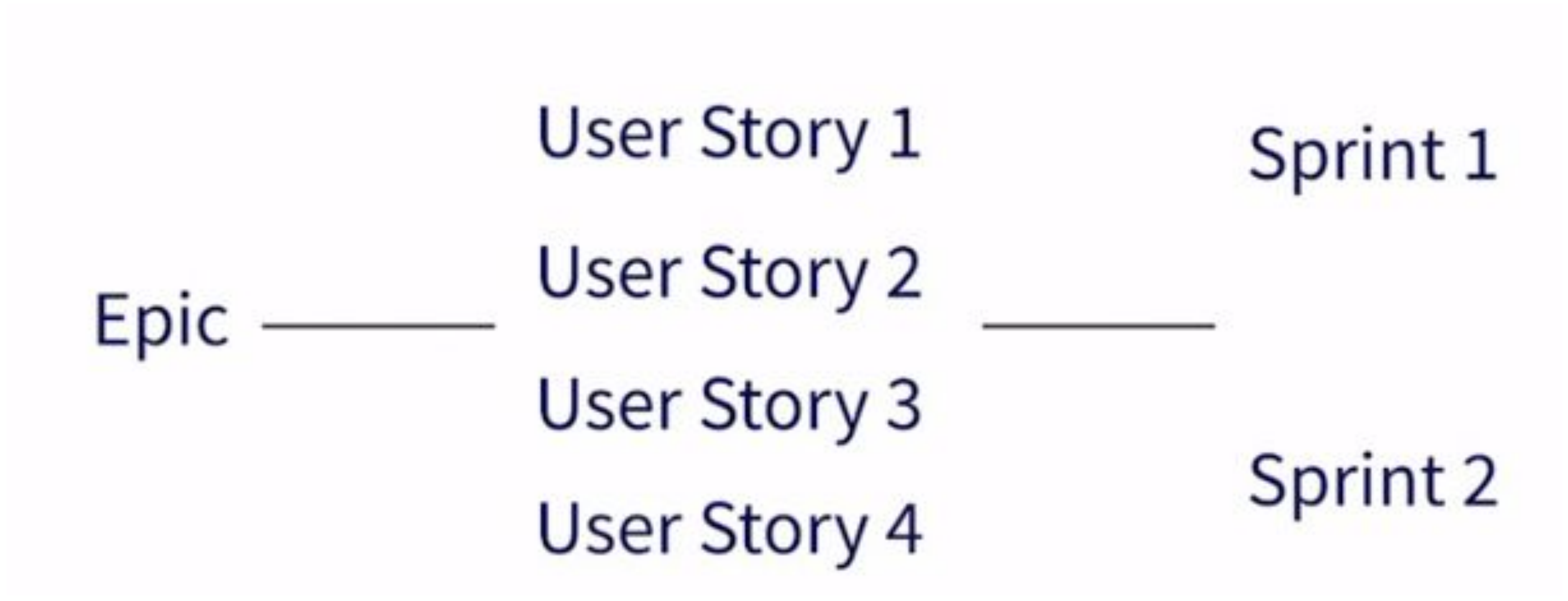
4 Values of Agile Manifesto?

- Individuals and interactions over Processes and tools
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile Requirements



Agile Requirements

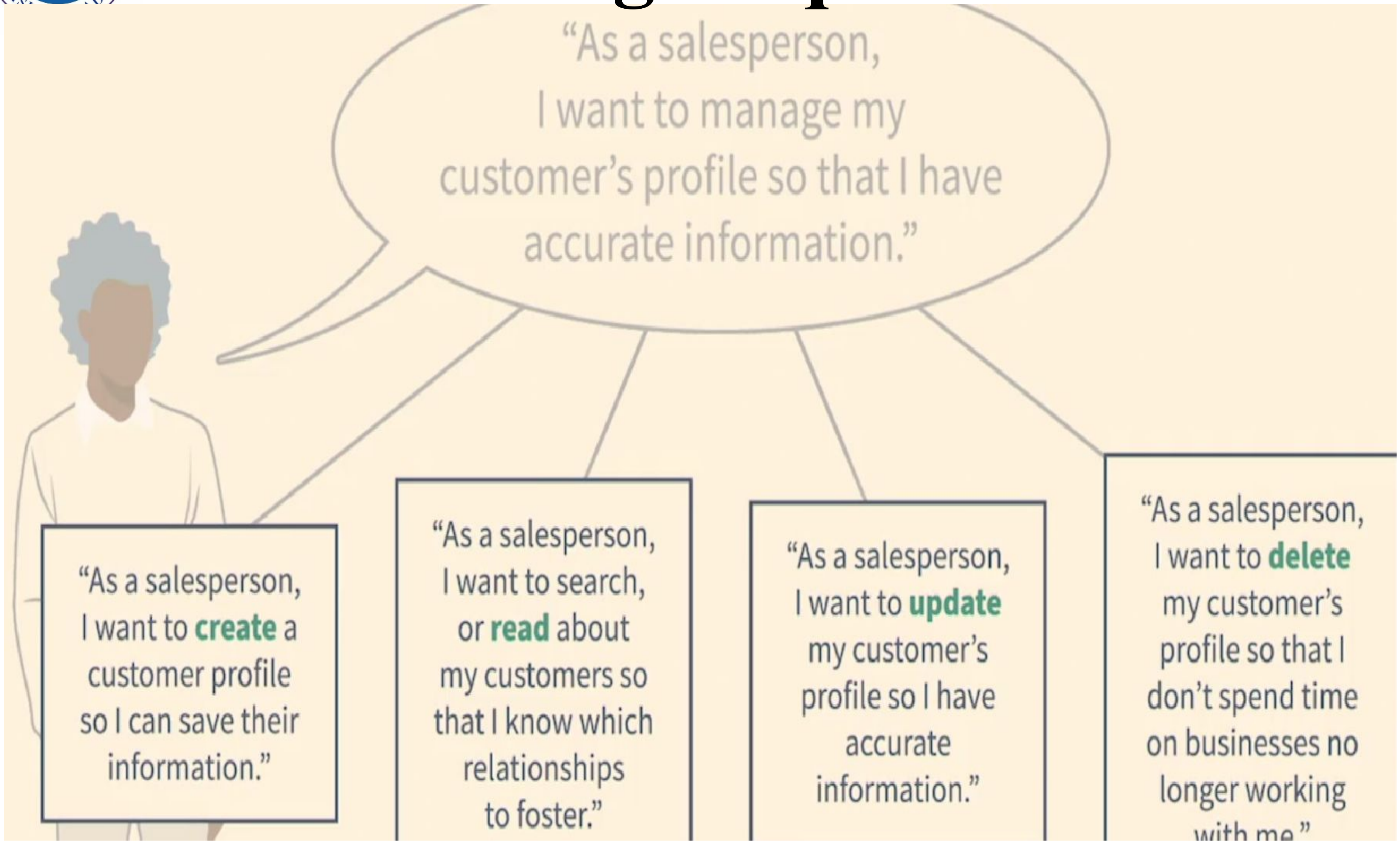


Refining Requirements

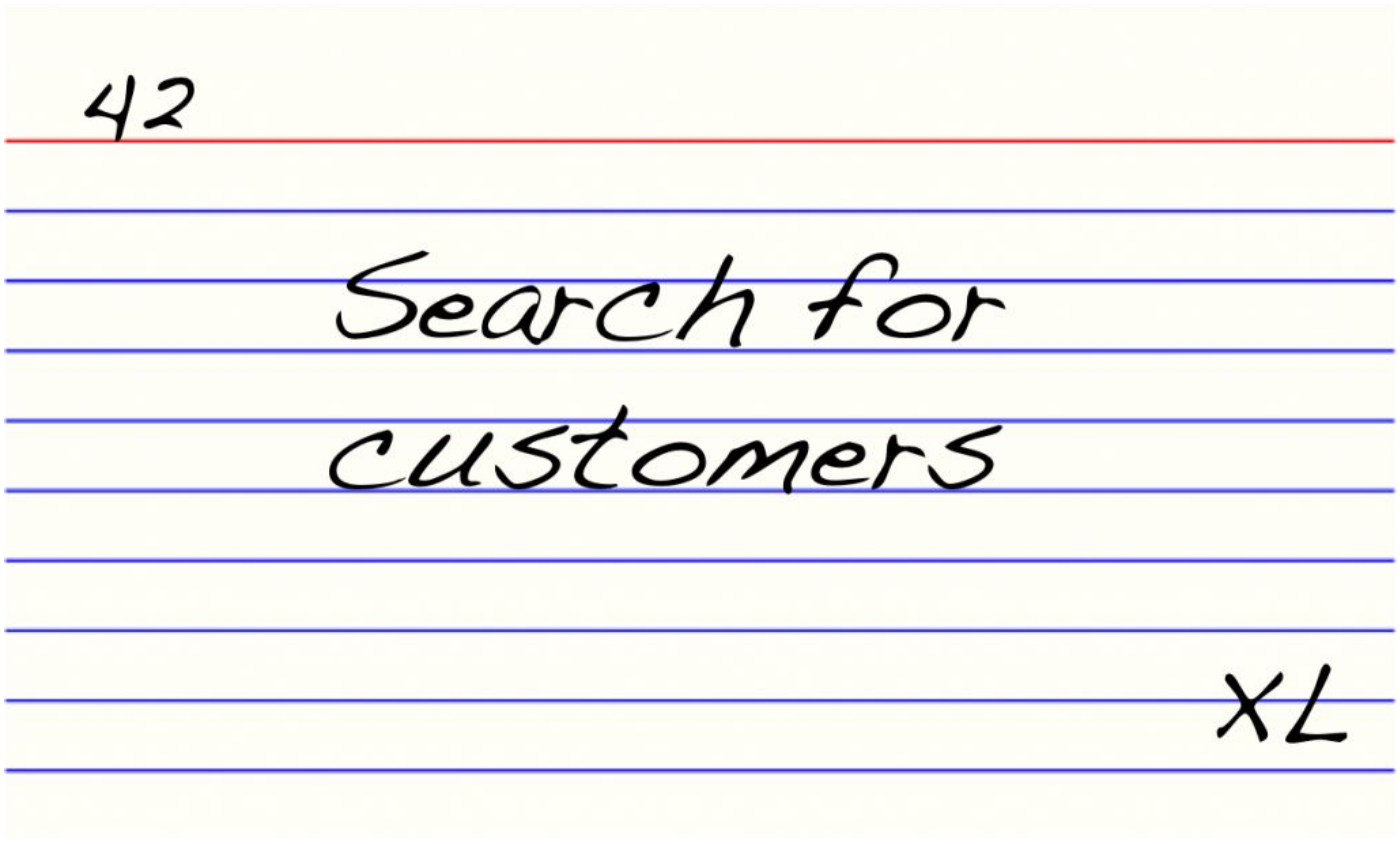
1. Feature
    1. Name
    2. Size [Optional]
    3. Priority

2. User Story
    1. Name
    2. “As <user role> I want to <action> so that <goal>”
    3. Conditions of Satisfaction [Optional]

Refining Requirements


"As a salesperson, I want to manage my customer's profile so that I have accurate information."
"As a salesperson, I want to **create** a customer profile so I can save their information."
"As a salesperson, I want to search, or **read** about my customers so that I know which relationships to foster."
"As a salesperson, I want to **update** my customer's profile so I have accurate information."
"As a salesperson, I want to **delete** my customer's profile so that I don't spend time on businesses no longer working with me."

Feature



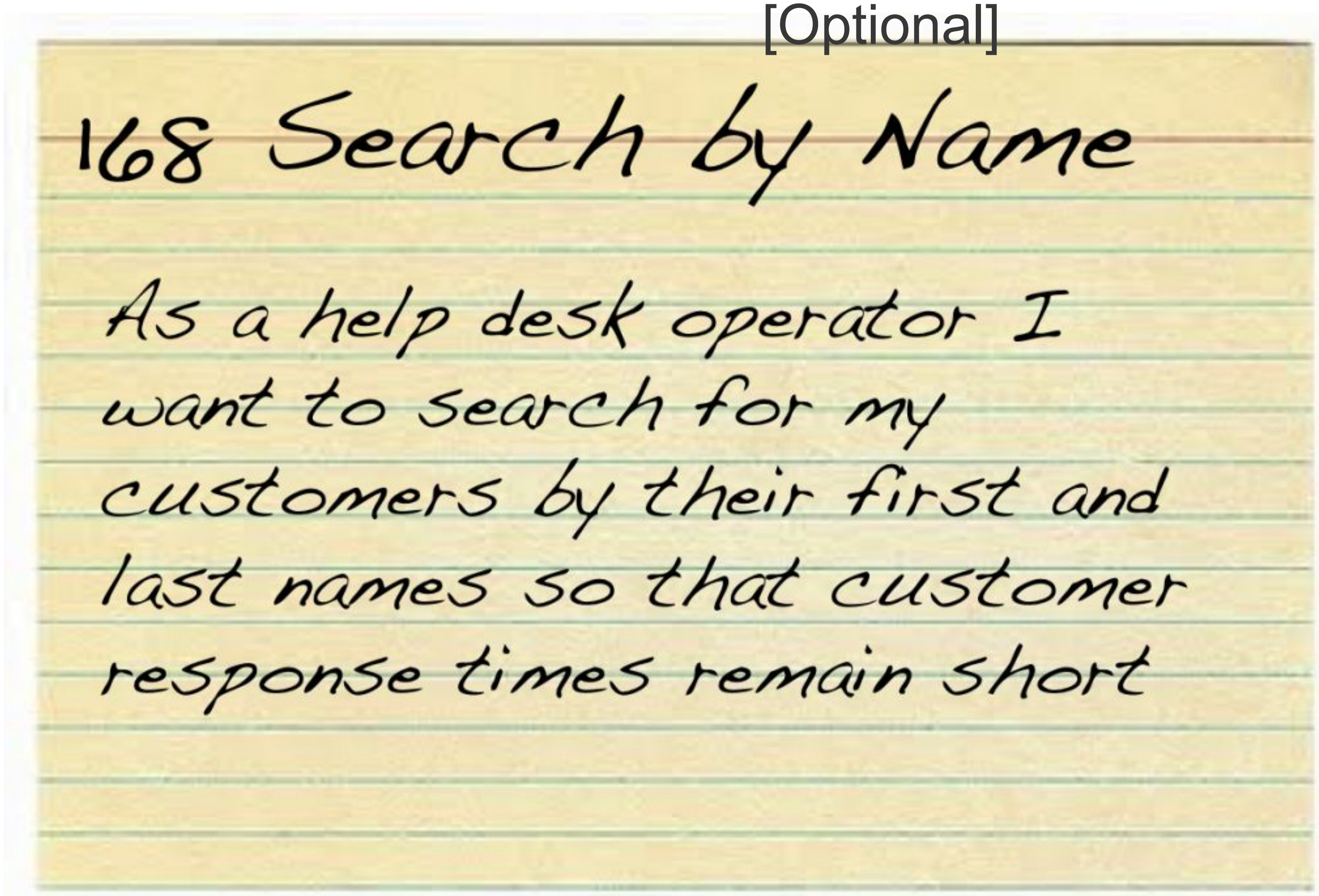
42

Search for customers

XL

User Story

1. User Story
   1. Name
   2. "As <user role> I want to <action> so that <goal>"
   3. Conditions of Satisfaction [Optional]

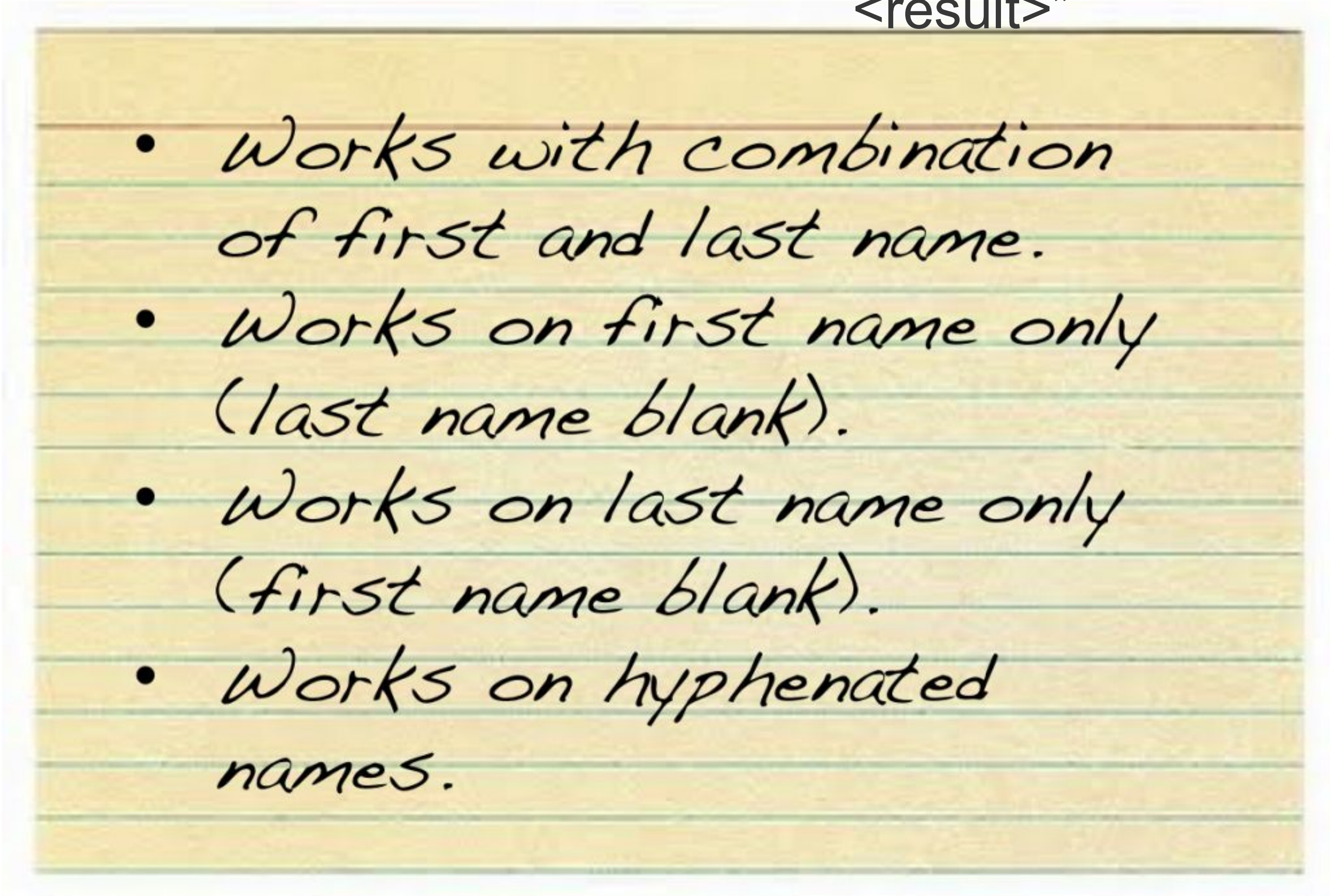


168 Search by Name

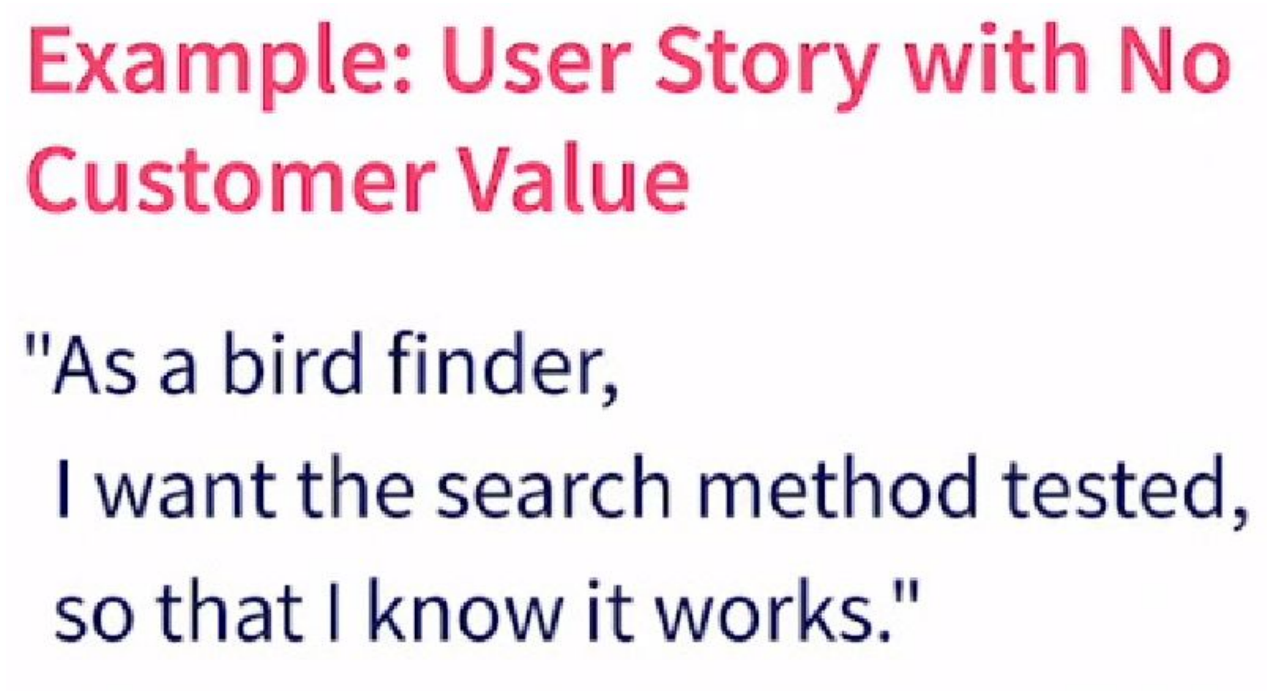
As a help desk operator I want to search for my customers by their first and last names so that customer response times remain short

Scenarios

Scenario
1. Name
2. “Given <precondition> When <trigger> Then <result>”



- Works with combination of first and last name.
- Works on first name only (last name blank).
- Works on last name only (first name blank).
- Works on hyphenated names.



Example: User Story with No Customer Value

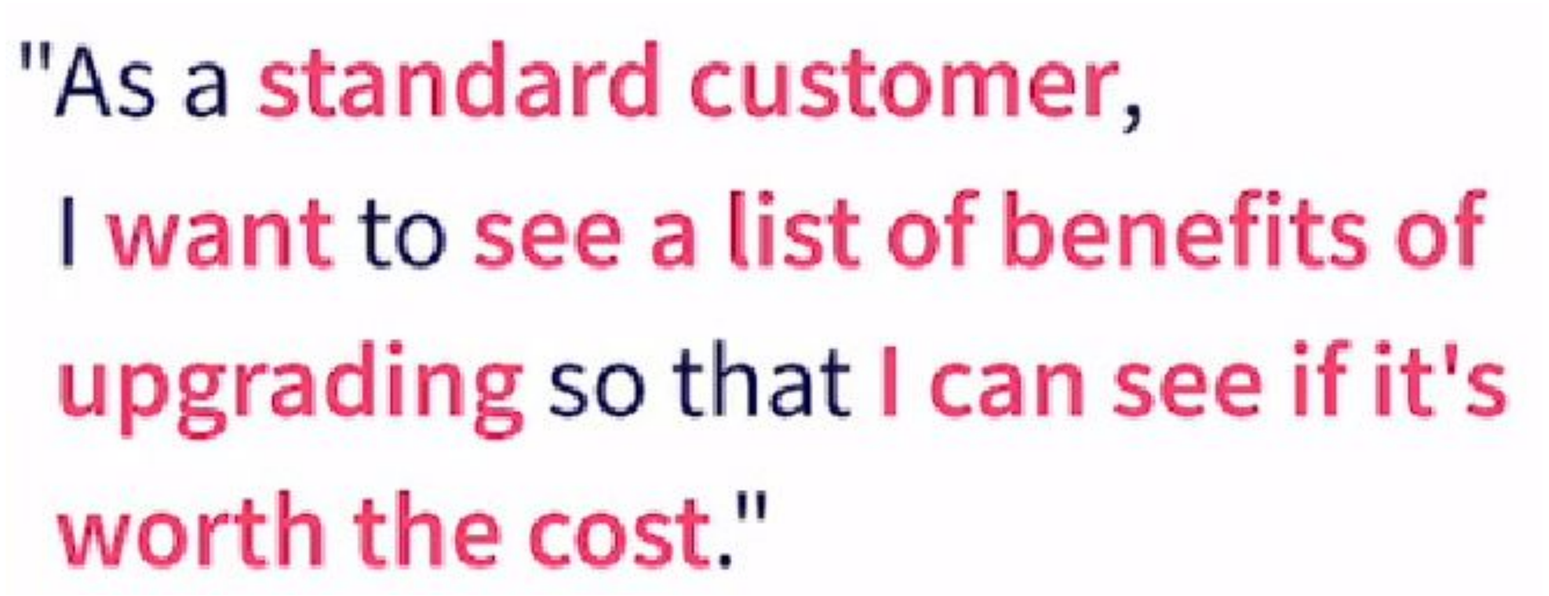
"As a bird finder,
I want the search method tested,
so that I know it works."

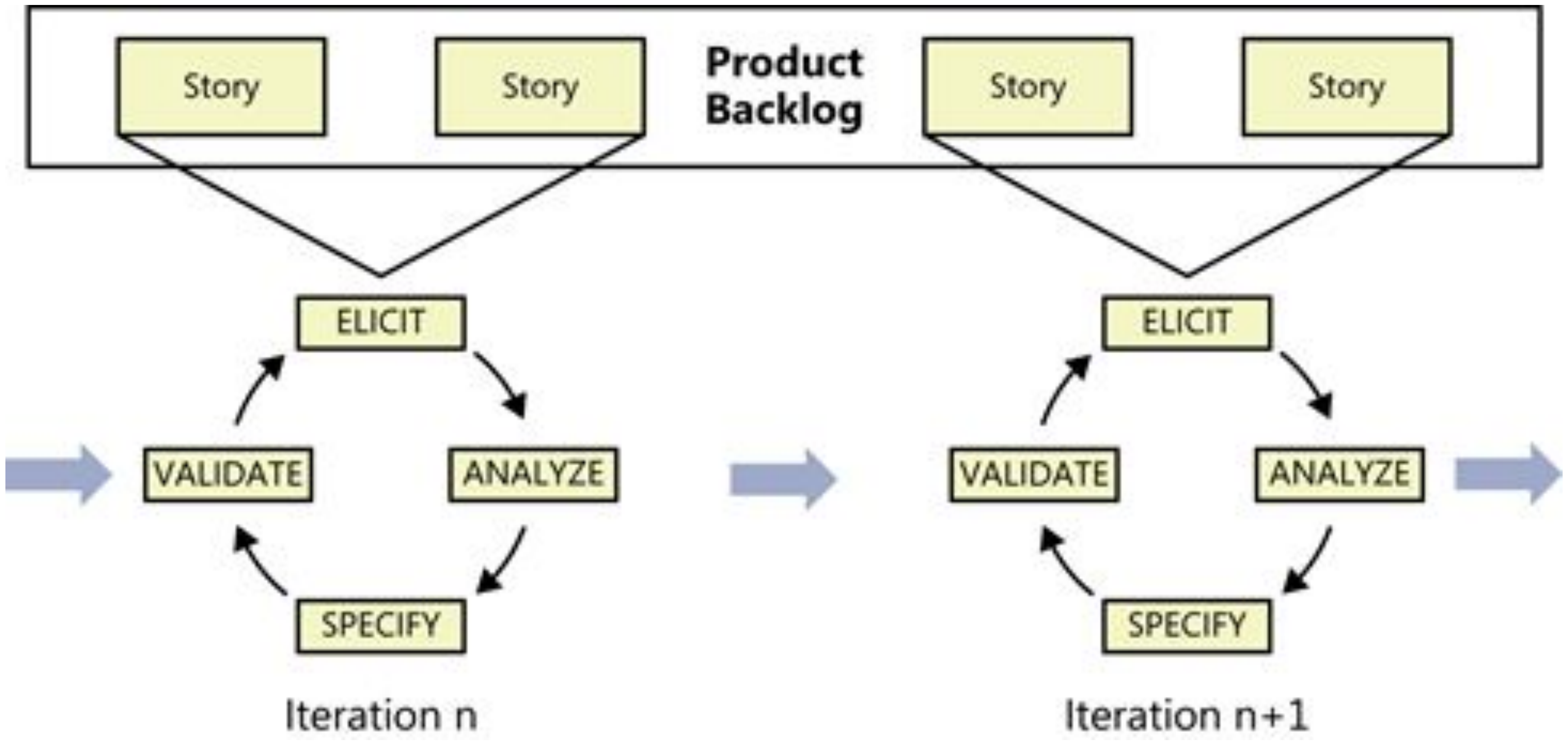
Example: Untestable User Story

"As a bird finder,
I want to see the list of birds quickly,
so I can identify my bird"

Velocity

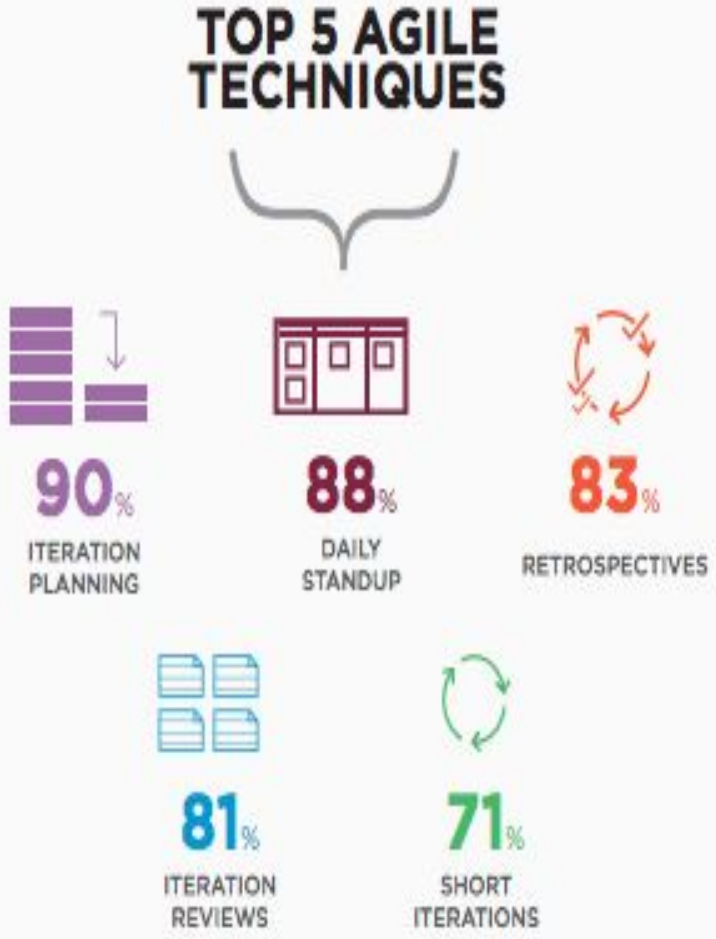
Rate at which teams can complete work within a time frame

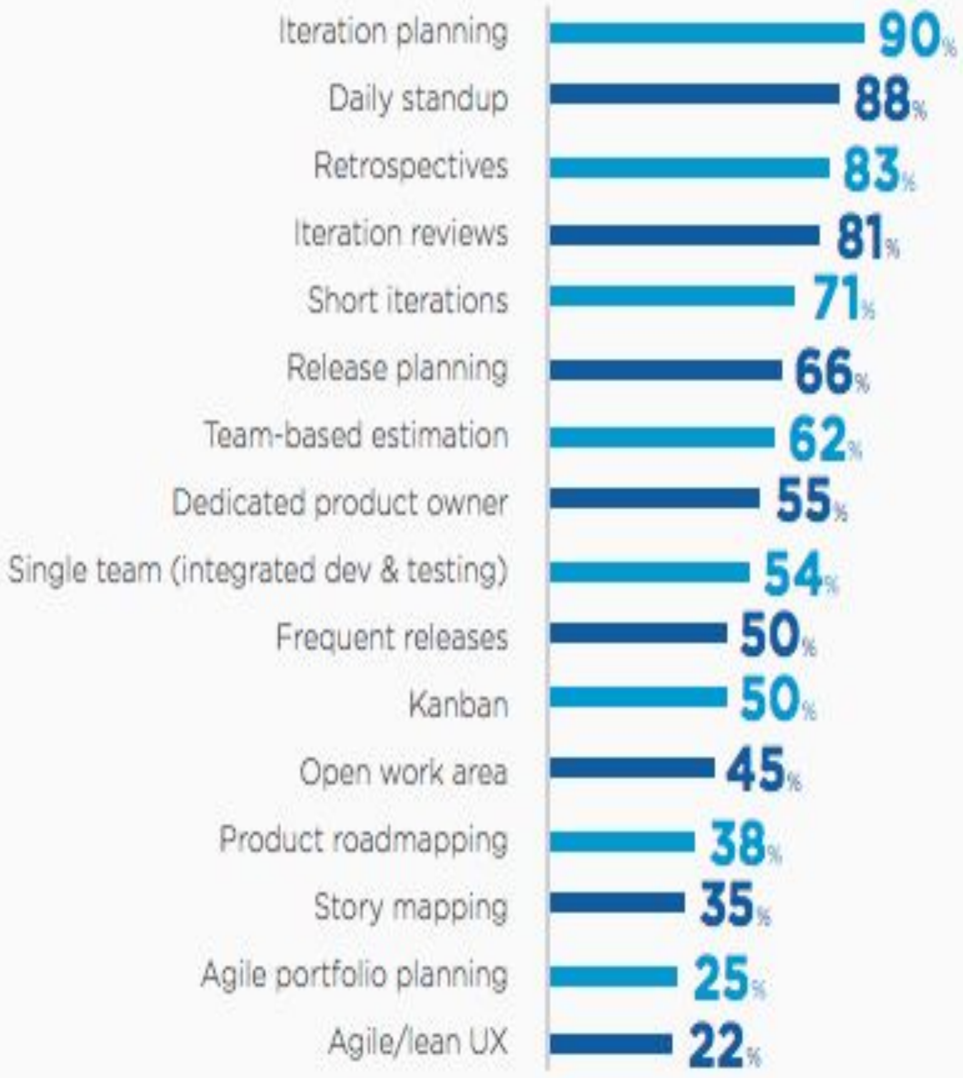

"As a standard customer,
I want to see a list of benefits of upgrading so that I can see if it's worth the cost."


Story
Story
Product Backlog
Story
Story
ELICIT
ANALYZE
SPECIFY
VALIDATE
Iteration n
ELICIT
ANALYZE
SPECIFY
VALIDATE
Iteration n+1

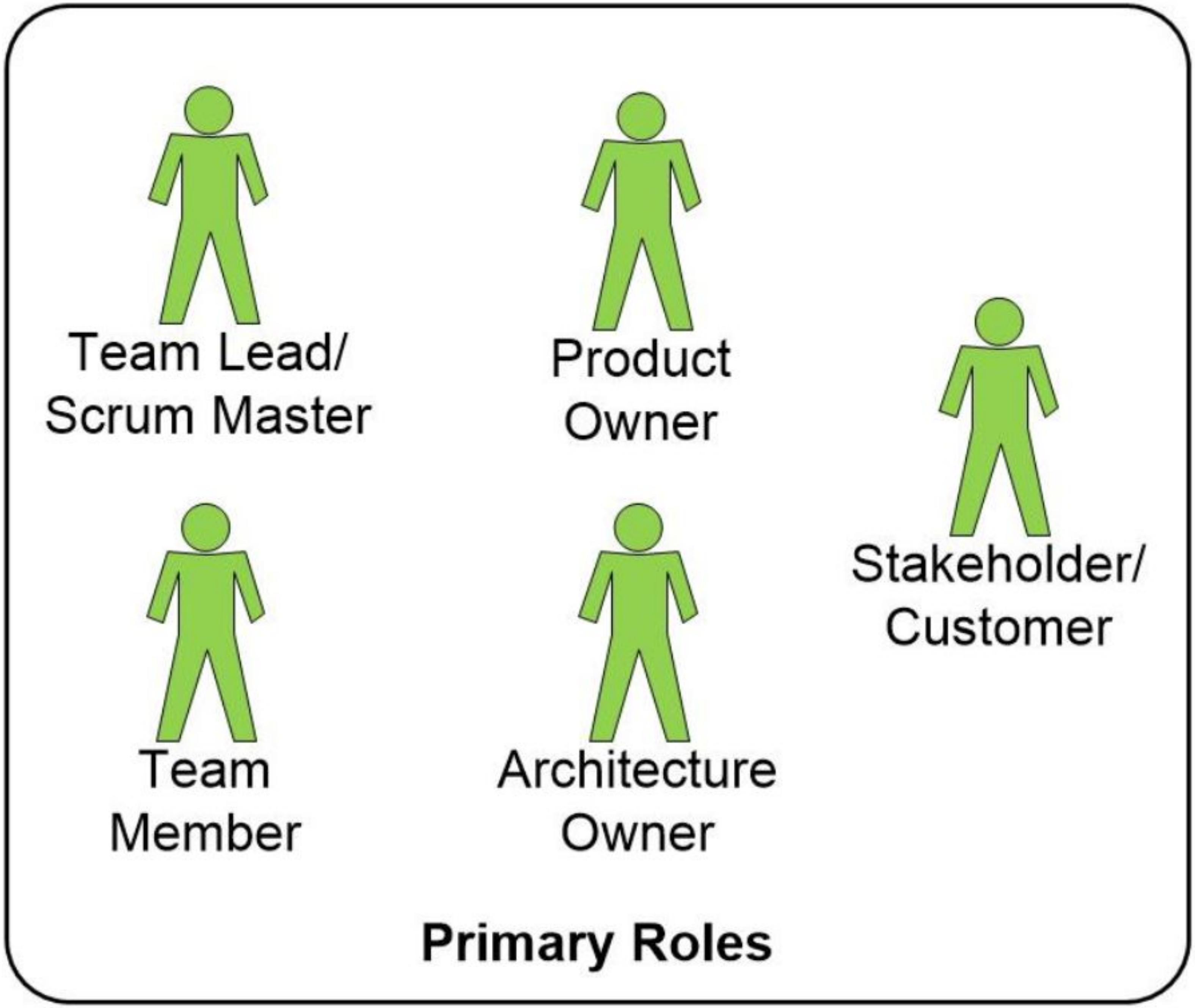
Agile Techniques Employed

From 2015 to 2016, the use of Kanban grew from 39% to 50%; iteration reviews increased from 54% to 81% and iteration planning went from 69% to 90%.





*Respondents were able to make multiple selections.


Team Lead/
Scrum Master
Product
Owner
Stakeholder/
Customer
Team
Member
Architecture
Owner
Primary Roles

Scrum Master

Dedicated role that protects the team and helps them improve

Product Owner (PO) Responsibilities

- Acts as the full-time business representative
- Reviews the team's work
- Ensures highest value is delivered
- Interacts with stakeholders

Scrum Master Responsibilities

- Protects the team and its processes
- Keeps the team working at a sustainable pace
- Acts as a spokesperson for the team
- Helps remove any roadblocks

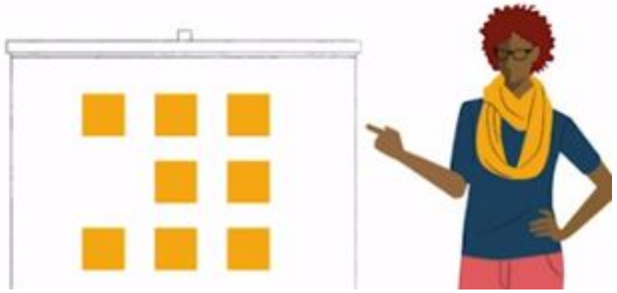
Team Composition

Ideal size: seven, plus or minus two members

User Story

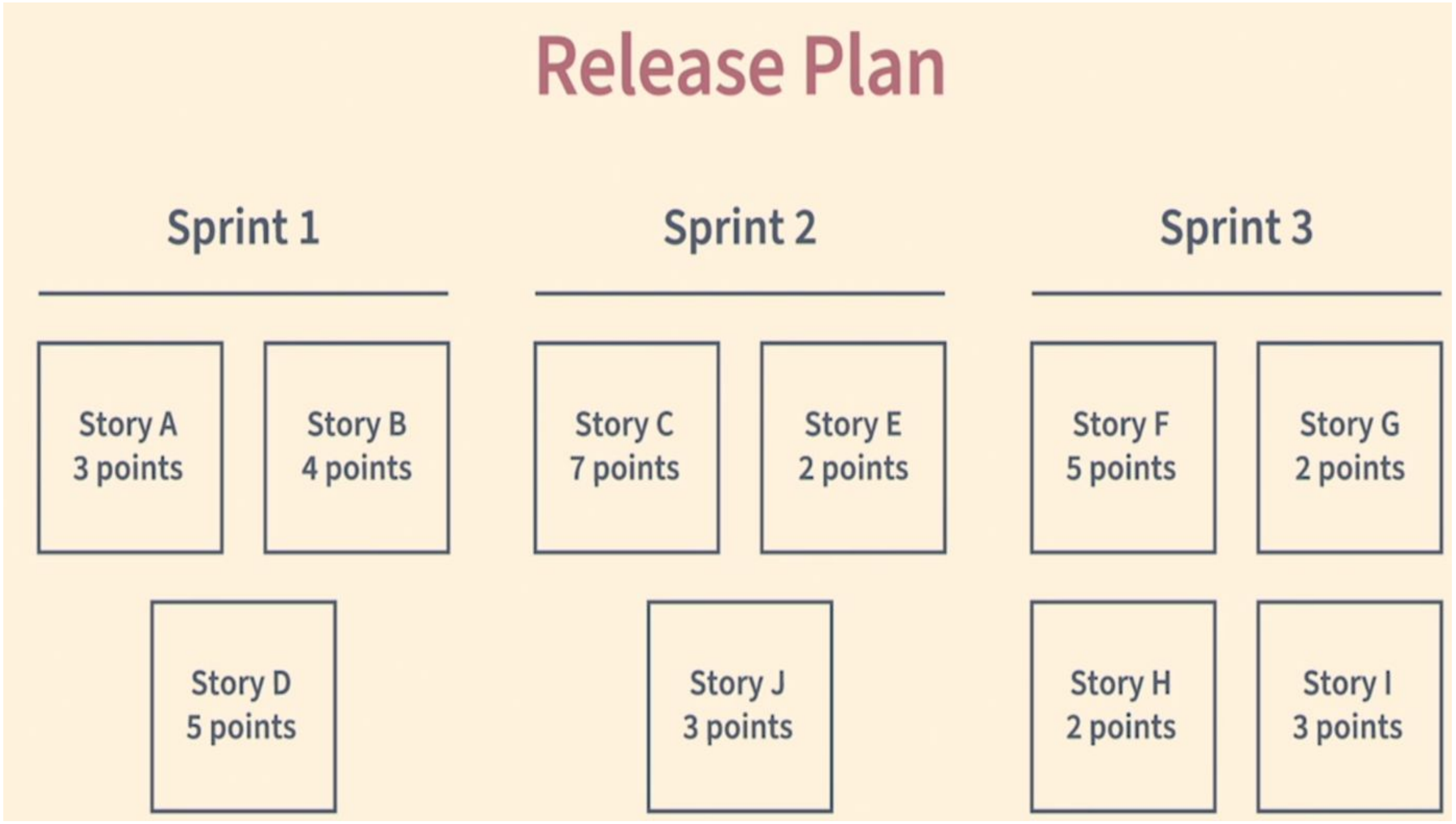
Detailed, valuable chunk of work a team can quickly deliver

1. Schedule-driven release plan
2. Functionality-based release plan

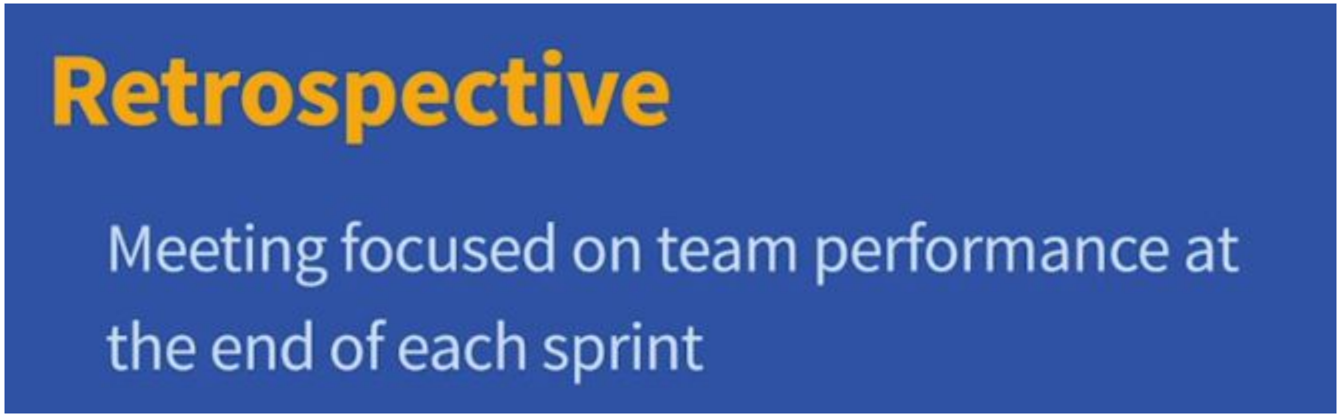


Release

Sprint 1		Sprint 2		Sprint 3	
Story A	3 Points	Story C	8 Points	Story G	5 Points
Story B	2 Points	Story E	2 Points	Story H	5 Points

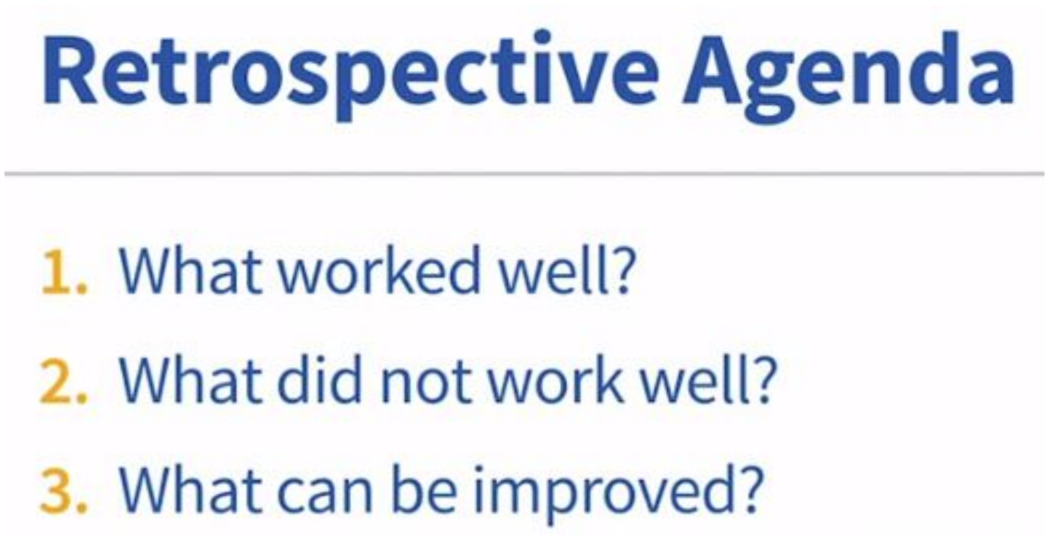


Release Plan
Sprint 1
Story A 3 points
Story B 4 points
Story D 5 points
Sprint 2
Story C 7 points
Story E 2 points
Story J 3 points
Sprint 3
Story F 5 points
Story G 2 points
Story H 2 points
Story I 3 points



Retrospective

Meeting focused on team performance at the end of each sprint



Retrospective Agenda

1. What worked well?
2. What did not work well?
3. What can be improved?

Extreme Programming

- Stories (features client wants)
- Estimate duration and cost of each story
- Select stories for next build
- Each build is divided into tasks
- Test cases for task are drawn up first
- Pair programming
- Continuous integration of tasks

Extreme Programming

- For small-to-medium-sized teams developing software with vague or rapidly changing requirements
- Coding is the key activity throughout a software project
- Communication among teammates is done with code
- Life cycle and behavior of complex objects defined in test cases – again in code

Extreme Programming

1. Planning game – determine scope of the next release by combining business priorities and technical estimates
2. Small releases – put a simple system into production, then release new versions in very short cycle
3. Metaphor – all development is guided by a simple shared story of how the whole system works
4. Simple design – system is designed as simply as possible (extra complexity removed as soon as found)
5. Testing – programmers continuously write unit tests; customers write tests for features
6. Refactoring – programmers continuously restructure the system without changing its behavior to remove duplication and simplify

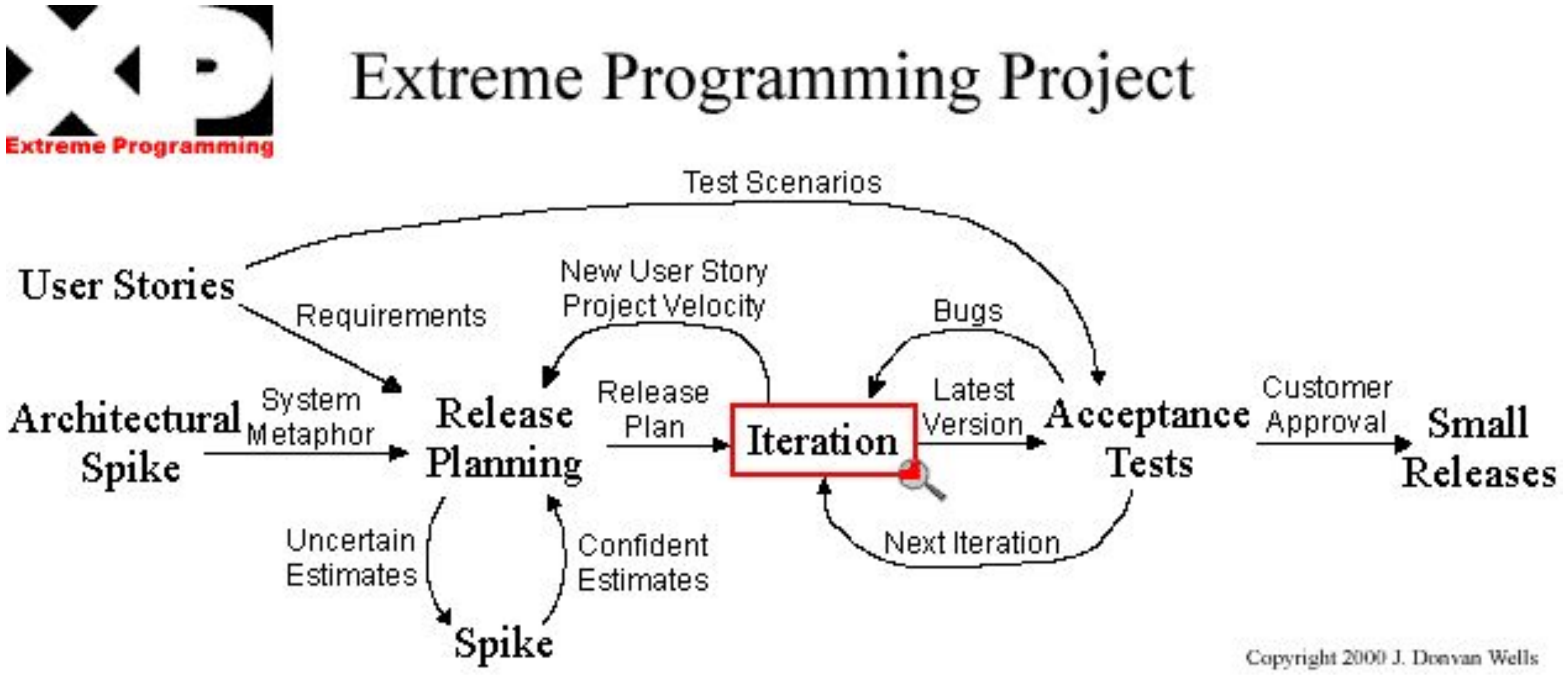
Extreme Programming

7. Pair-programming -- all production code is written with two programmers at one machine
8. Collective ownership – anyone can change any code anywhere in the system at any time.
9. Continuous integration – integrate and build the system many times a day – every time a task is completed.
10. 40-hour week – work no more than 40 hours a week as a rule
11. On-site customer – a user is on the team and available full-time to answer questions
12. Coding standards – programmers write all code in accordance with rules emphasizing communication through the code

Extreme Programming

- Review code all the time (pair programming)
- Everybody will test the code
- Everybody will design daily (refactoring)
- Everybody will work at defining and refining the architecture (metaphor)
- Build, integrate and test continuous

Extreme Programming



Unusual Features of XP

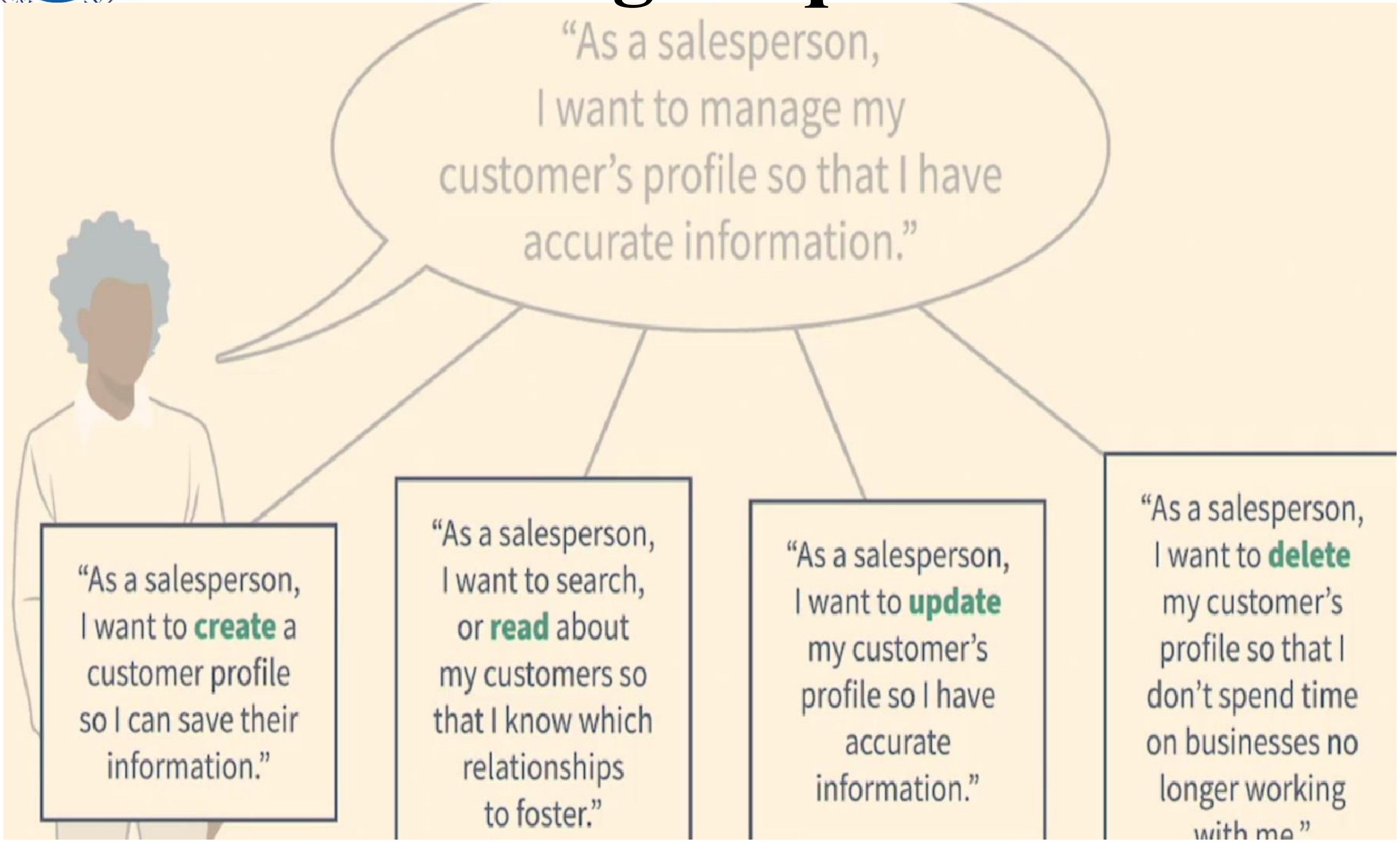
- Computers are put in center of large room lined with cubicles
- Client representative is always present
- Cannot work overtime for 2 successive weeks
- No specialization

Evaluating XP

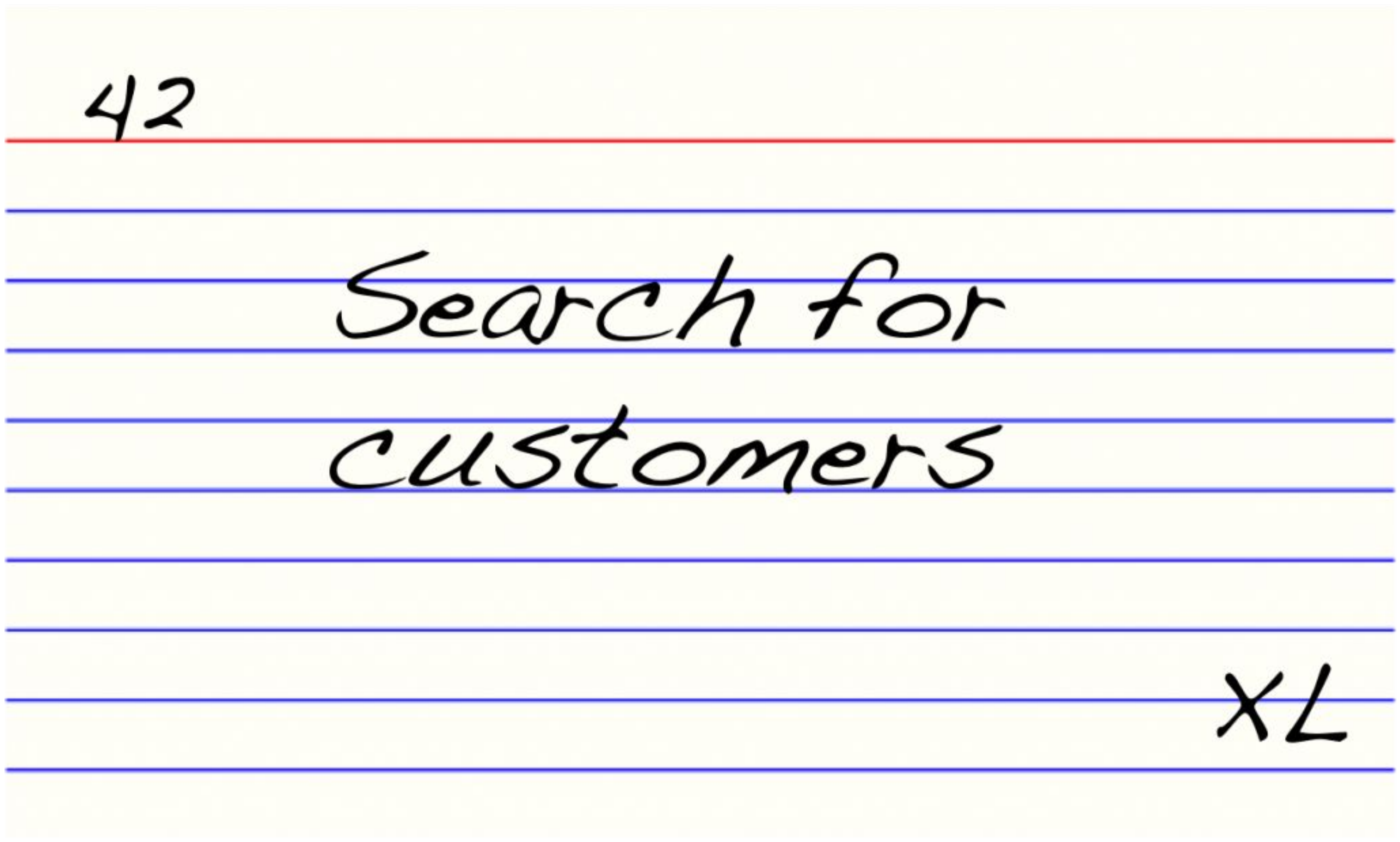
- XP has had some successes
- Good when requirements are vague or changing
- Too soon to evaluate XP

ABC department of a university want a students management system to keep record of teachers, courses, students, attendance, registered courses and students marks. System should also facilitate the university department to assign courses to the teachers. The student and teachers have unique user name and password. System should allow registered students to search and register courses. Teacher can upload, modify and view students marks and attendance while student can only view the semester results, registered courses and attendance reports.

Refining Requirements


"As a salesperson, I want to manage my customer's profile so that I have accurate information."
"As a salesperson, I want to **create** a customer profile so I can save their information."
"As a salesperson, I want to search, or **read** about my customers so that I know which relationships to foster."
"As a salesperson, I want to **update** my customer's profile so I have accurate information."
"As a salesperson, I want to **delete** my customer's profile so that I don't spend time on businesses no longer working with me."

Feature



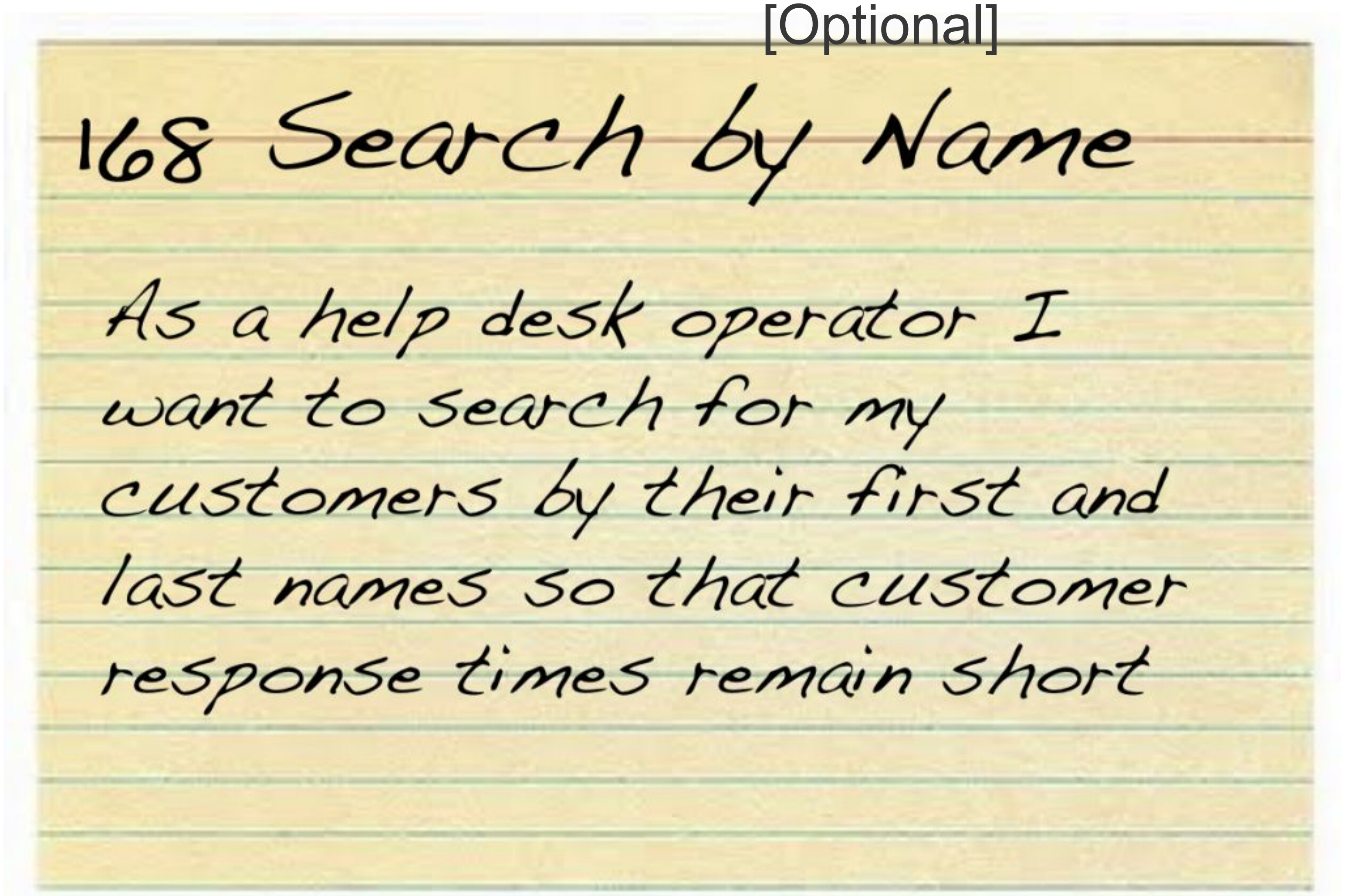
42

Search for customers

XL

User Story

1. User Story
   1. Name
   2. “As <user role> I want to <action> so that <goal>”
   3. Conditions of Satisfaction [Optional]



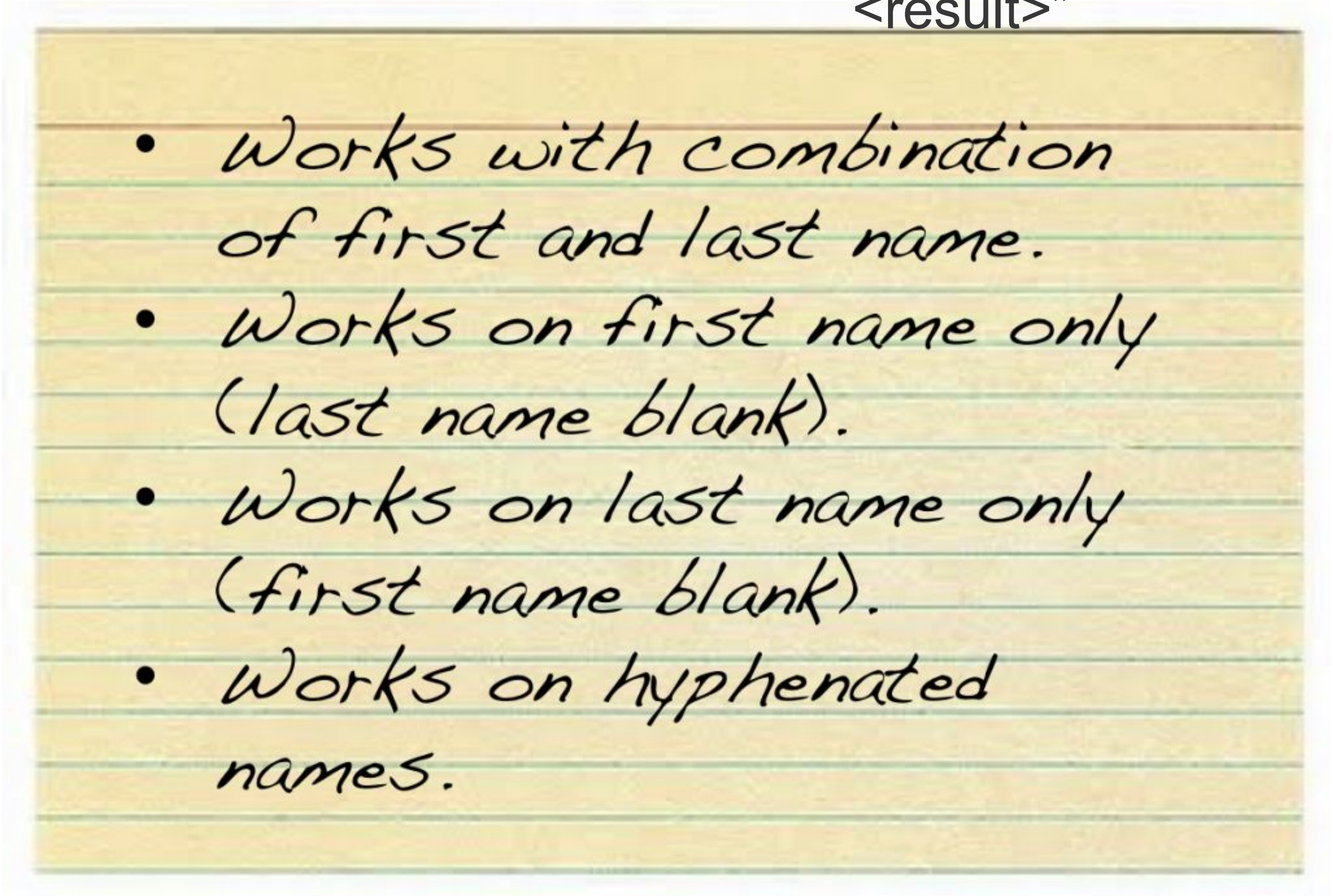
168 Search by Name

As a help desk operator I want to search for my customers by their first and last names so that customer response times remain short

Scenarios

Scenario

1. Name
2. "Given <precondition> When <trigger> Then <result>"



- Works with combination of first and last name.
- Works on first name only (last name blank).
- Works on last name only (first name blank).
- Works on hyphenated names.

Thank You