Introduction to Agile Development

{How to eat an elephant}

INNTOPIA

INNOVATIVE TRAVEL TECHNOLOGY

One Bite At A Time

This Morning…

- What is Agile, why do it?
- A brief background of Agile disciplines
- Our approach from 10k feet to the ground
- Summary and next steps
- Questions

Trends in Software Business: Implications for Agility

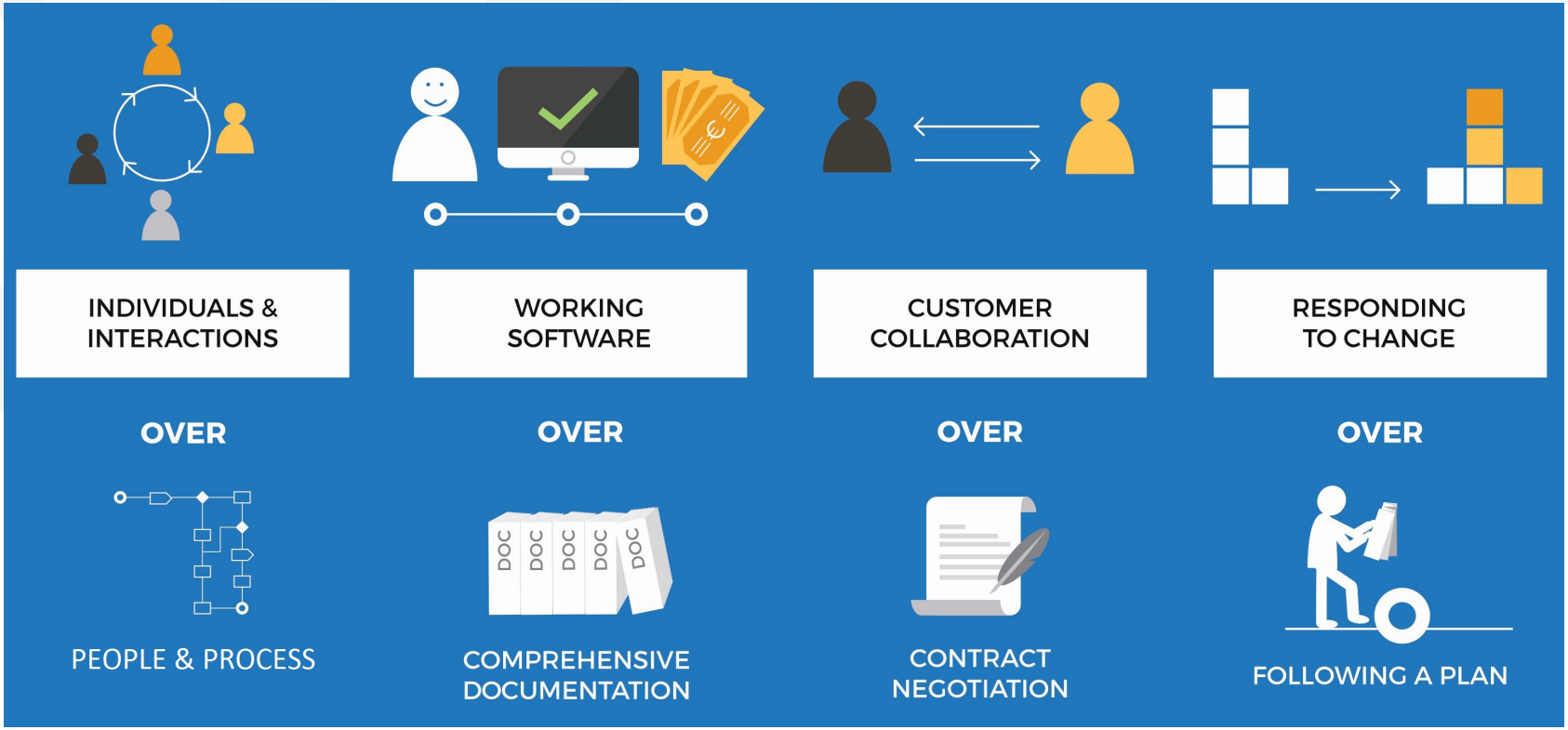
Trend	Implication
User expectations for functionality and quality are increasing	Higher demand for user-driven, flexible development
Cycle times are shorter	Reacting to industry trends, technical change and customer needs is paramount
Systems are becoming more complex and more integrated	Sound planning, architecting and integrity control is required
New business opportunities require exploration of new product concepts	Novel features need to be innovated and identified

What is Agile Development?

- Agility != “no process”

- Agility = Ability to react to changing requirements **quickly**, **appropriately**, and **effectively** to get product into the hands of **customers** more quickly.

Principles of Agile Methods:



While there is value in the items on the bottom, we value the items on the top more

Three Flavors of Agile

- Extreme Programming (XP)
- SCRUM
- KANBAN

Extreme Programming ('XP' or 'Lean')

- Perhaps the best-known and most widely used agile method.
- XP takes an 'extreme' approach to iterative development
  - New versions may be built several times per day ('continuous integration')
  - Increments are delivered to customers every 2 weeks
  - All tests must be run for every build an the build is only accepted if tests run successfully

Common components of XP

- **The planning game**: At the start of each iteration, customers, managers, and developers meet to flesh out, estimate, and prioritize requirements for the next release. The requirements are called "user stories" and are captured on "story cards"
- **Small releases**: Working versions are put into production anywhere from every few days to every few weeks
- **Tests**: Developers work test-first; they write unit tests for their code before they write the code itself. Customers write functional tests for each iteration and at the end of the iteration, all tests should run
- **Refactoring**: As developers work, the design should be evolved to keep it as simple as possible. Constant code improvement makes reacting to change easier.
- **Pair Programming**: Two developers sitting at the same machine write all code. Helps with cross pollination of technical and business knowledge. Encourages refactoring and serves as an informal review process
- **Continuous Integration**: Developers integrate new code into the system as often as possible. Automated tests are run after integration or the new code is discarded

Common components of XP (2)

- **Collective ownership**: The code is owned by all developers and they may make changes anywhere in the code at anytime they feel necessary
- **On-site customer**: A customer works with the development team at all times to answer questions, perform acceptance tests, and ensure that development is progressing as expected
- **40 hour work weeks**: Requirements should be selected for each iteration such that developers do not need to put in overtime
- **Open workspace**: Developers work in a common workspace set up with individual workstations around the periphery and common development machine in the center

SCRUM

- Iterative lifecycle focusing on communication, particularly feedback loops (Sprints and Scrums)
- Without major feature creep in an iteration, teams build useful, demonstrable product functionality
- Scrum's goals are to optimize development environment, reduce organizational overhead and closely synchronize market requirements with iterative releases
- All Scrum meetings and sprints are time boxed to maintain efficiency and promote production.

Common components of SCRUM

- **Pre-sprint planning**: Features and functionality are selected from the "Product backlog" and placed into the "sprint backlog". Since tasks in the backlog are generally at a higher level of abstraction, pre-sprint planning also identifies a Sprint Goal reminding developers why the tasks are being performed

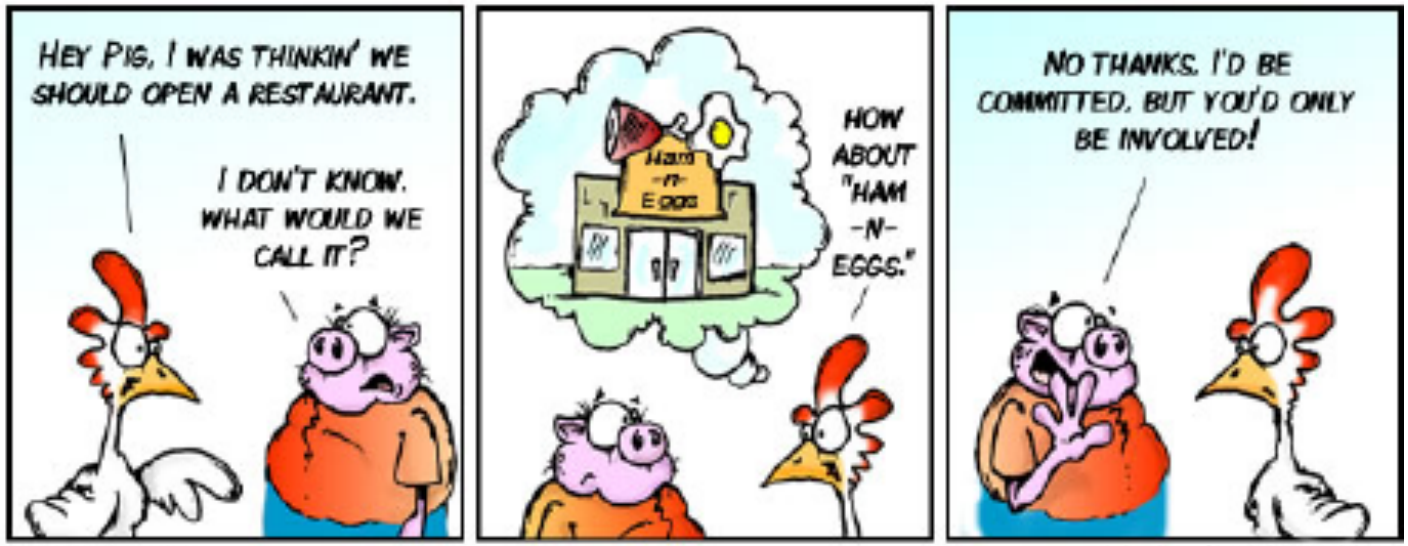
- **Sprint**: Teams are handed their sprint backlog and "told to sprint to achieve their objectives". Tasks in the sprint backlog are frozen and remain unchangeable for the duration of the sprint. Team members choose the tasks they want to work on and begin development. Scrum meetings are held every morning to enhance communication and inform customers, developers, and managers on the status of the project, identify any problems and keep the entire team focused on a common goal.

- **Post-sprint review**: After every sprint, a post-sprint review meeting is held to analyze project progress and demonstrate the current system.

- **Team size:** Development personnel are split into teams of up to seven people. A complete team should at least include a developer, quality assurance engineer, and a product owner

Common components of SCRUM (2)

- **Iteration Length**: Durations are commonly held at 2 weeks
- **Support for distributed teams**: A project may consist of multiple teams that could easily be distributed
- **Done Work**: Every sprint results in completion of some done work that has business value

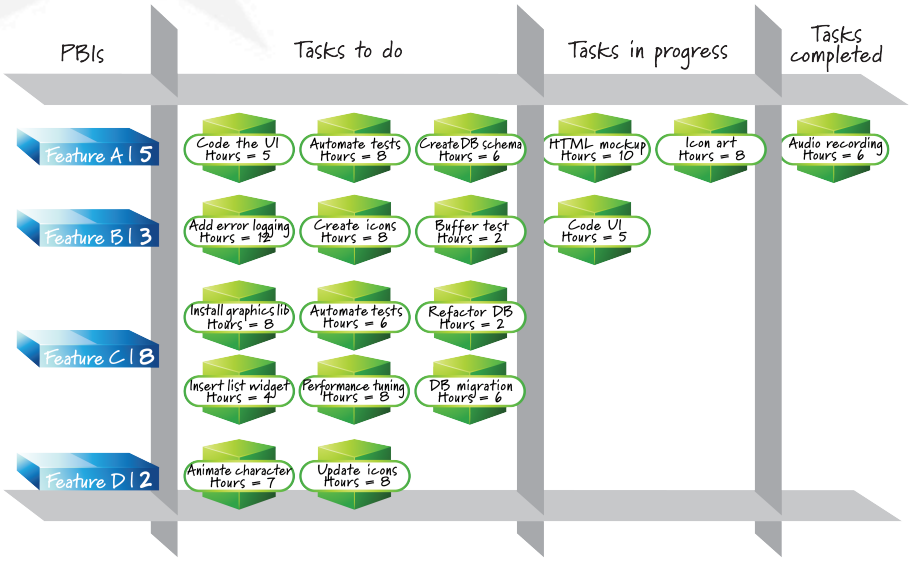


KANBAN

- Continuous WIP (Work In Process) flow
- Ever changing
- Flexibility in planning
- Minimizing cycle time
- Efficiency through focus

Common components of KANBAN

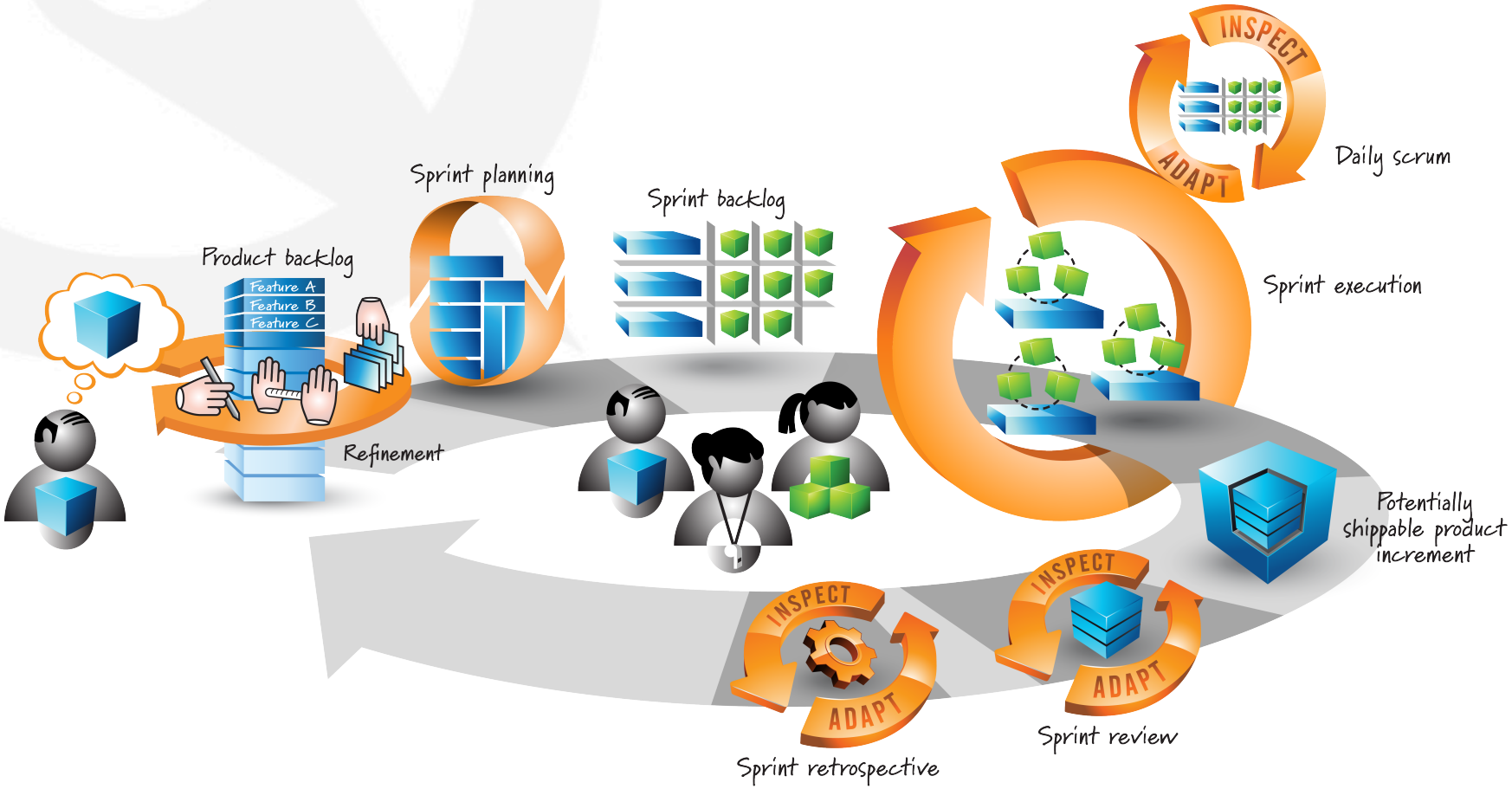
- **WIP Board**: Work is visualized in a simple flow from To Do ➔ Doing ➔ Done
- **No iterations**: Kanban does not contain the concept of iteration. Instead, there is a continuous flow of work in and out.
- **Pre-Development Focused**: Kanban is most effective for things that are constantly changing and it, therefore, a great fit for the analysis phase where requirements are fluid and negotiations are taking place.



Agile at Inntopia

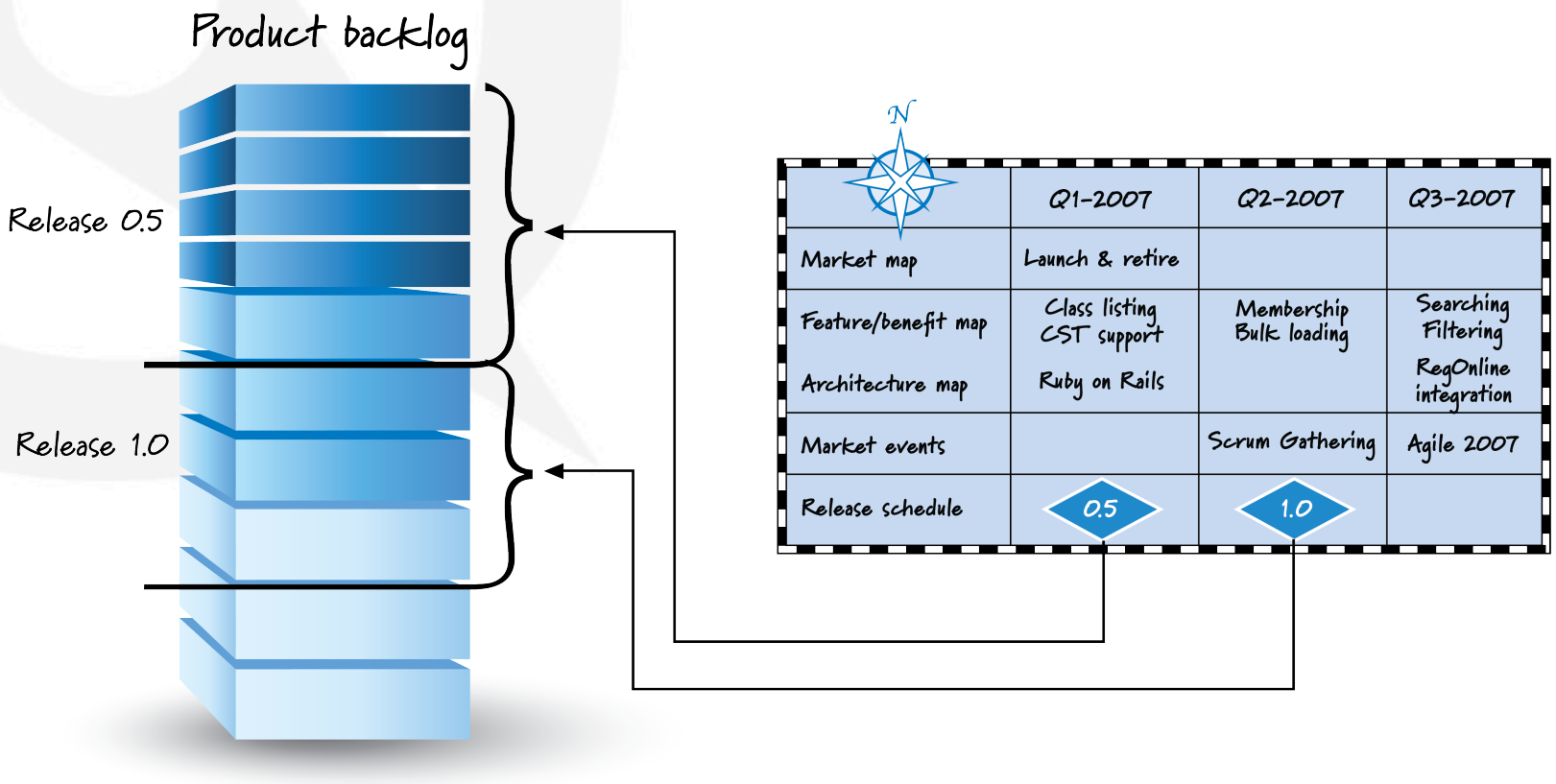
A Inntopia Approach…

Utilize Scrum but blend other topics of agile development where needed



Product Backlog

Product Backlog



How do we get there?

User Stories

Why?
Because they address communication problems

How?
We talk about behaviors and INVEST

Story: Account Holder withdraws cash

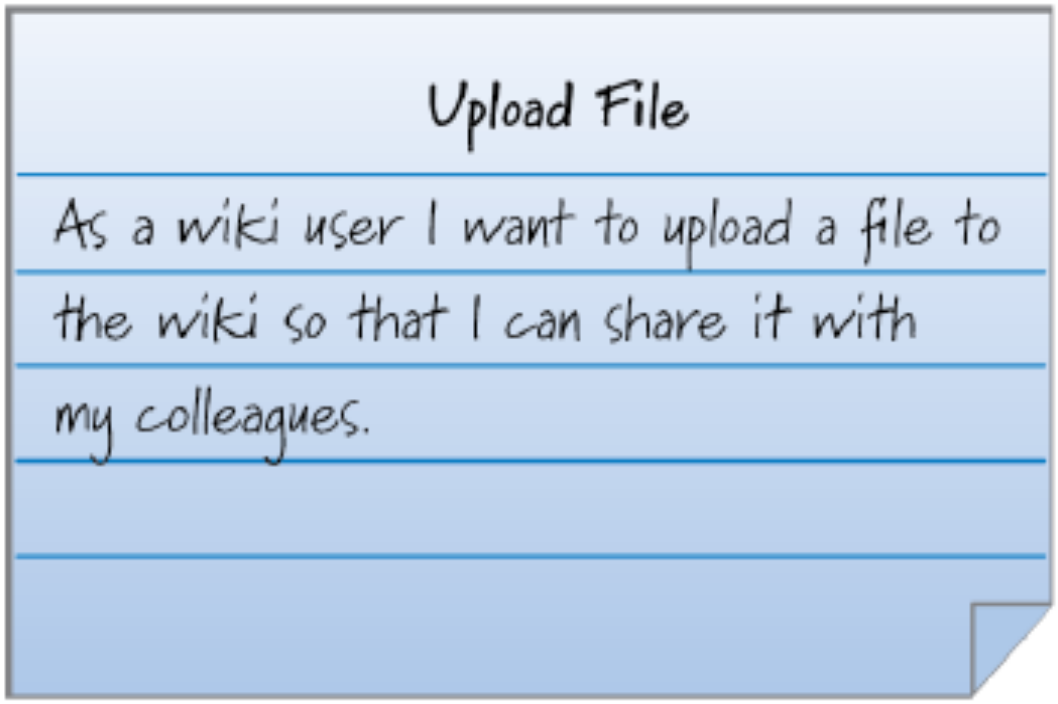
As an Account Holder
I want to withdraw cash from an ATM
So that I can get money when the bank is closed

Scenario 1: Account has sufficient funds
Given the account balance is $100
And the card is valid
And the machine contains enough money
When the Account Holder requests $20
Then the ATM should dispense $20
And the account balance should be $80
And the card should be returned

Scenario 2: Account has insufficient funds
Given the account balance is $10
And the card is valid
And the machine contains enough money
When the Account Holder requests $20
Then the ATM should not dispense any money
And the ATM should say there are insufficient funds
And the account balance should be $20
And the card should be returned

Scenario 3: Card has been disabled
Given the card is disabled
When the Account Holder requests $20
Then the ATM should retain the card
And the ATM should say the card has been retained

Scenario 4: The ATM has insufficient funds
…



Definitions of:
Ready (DoR) & Done (DoD)

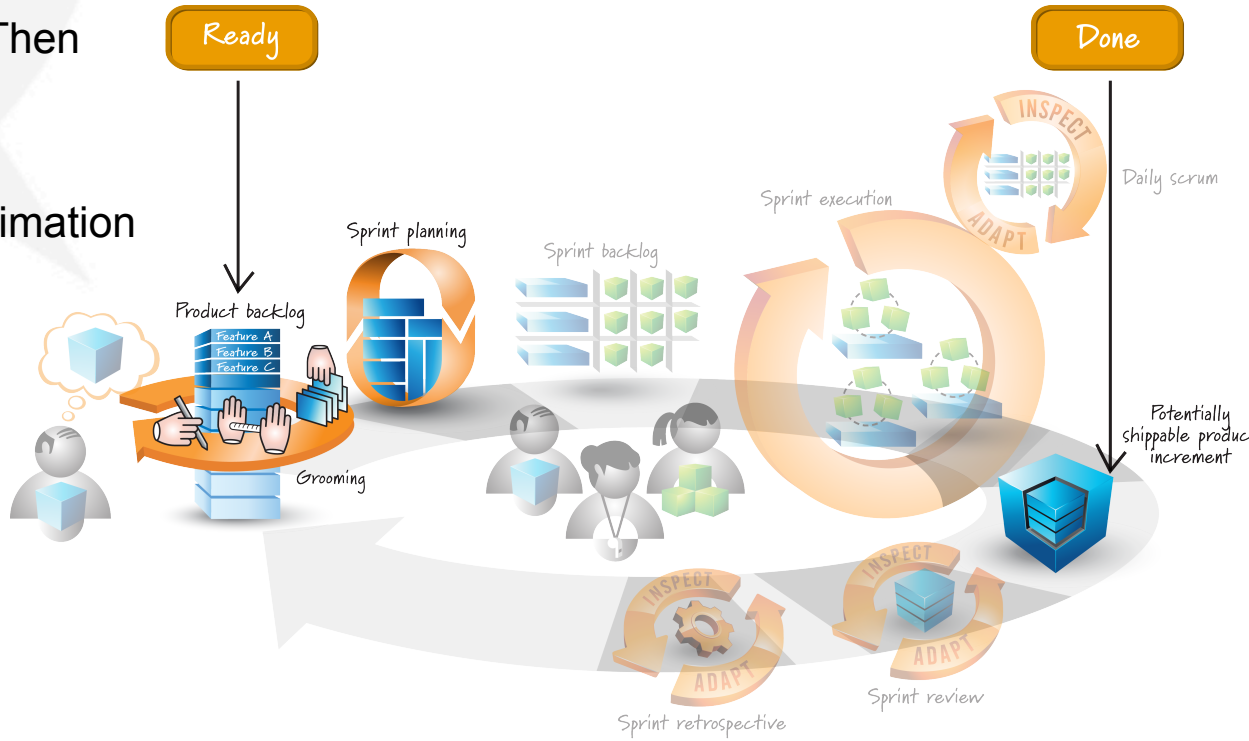
Ready?

Define what "**Ready**" means

- Common practices
  - Acceptance Criteria
  - Business Value defined
- Done When Statements
  - Given → When → Then
- Test Cases

Goals:

- Improved accuracy of estimation
- Reduce cost of re-work
- Increased Velocity
- Quality of work



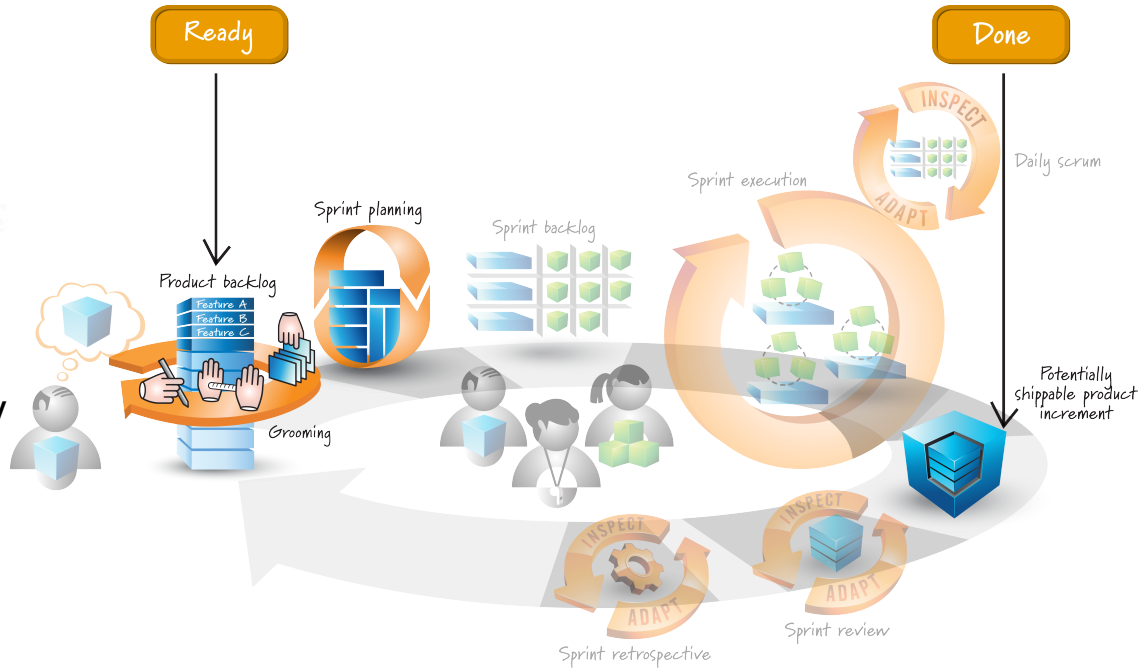
Done?

Define what "**Done**" means

- Common development practices
  - Refactor
  - Unit test
  - Continous Integration
- Done When Statements
  - Given → When → Then
- Test Cases

Goals:

- Improved code quality
- Reduce cost of re-work
- Increased Velocity
- Ability to add new functionality



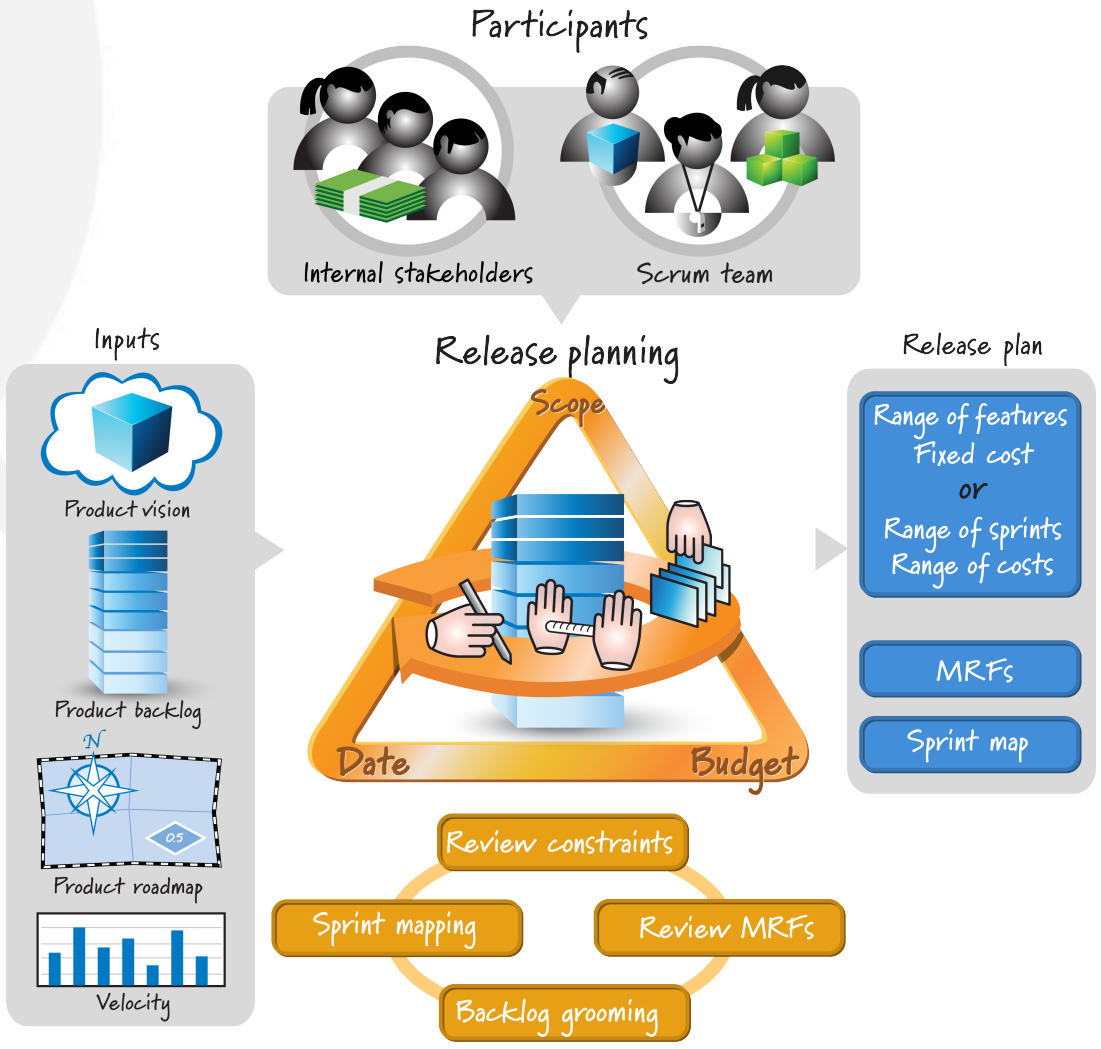
Planning & Estimating

Release Planning

Based on estimations of Features and Stories being broken out across future Sprints.

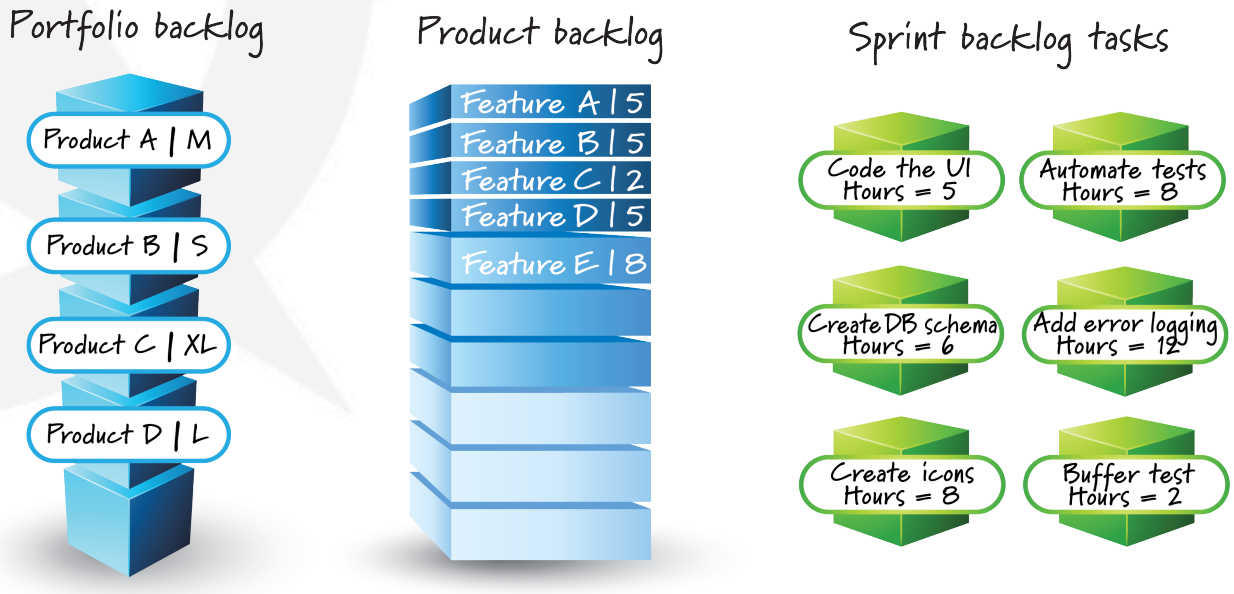
Dependent Features and Stories are either culled or cut down so as to meet INVEST

Not longer than 1 Quarter, shorter releases are preferable.



Agile Planning

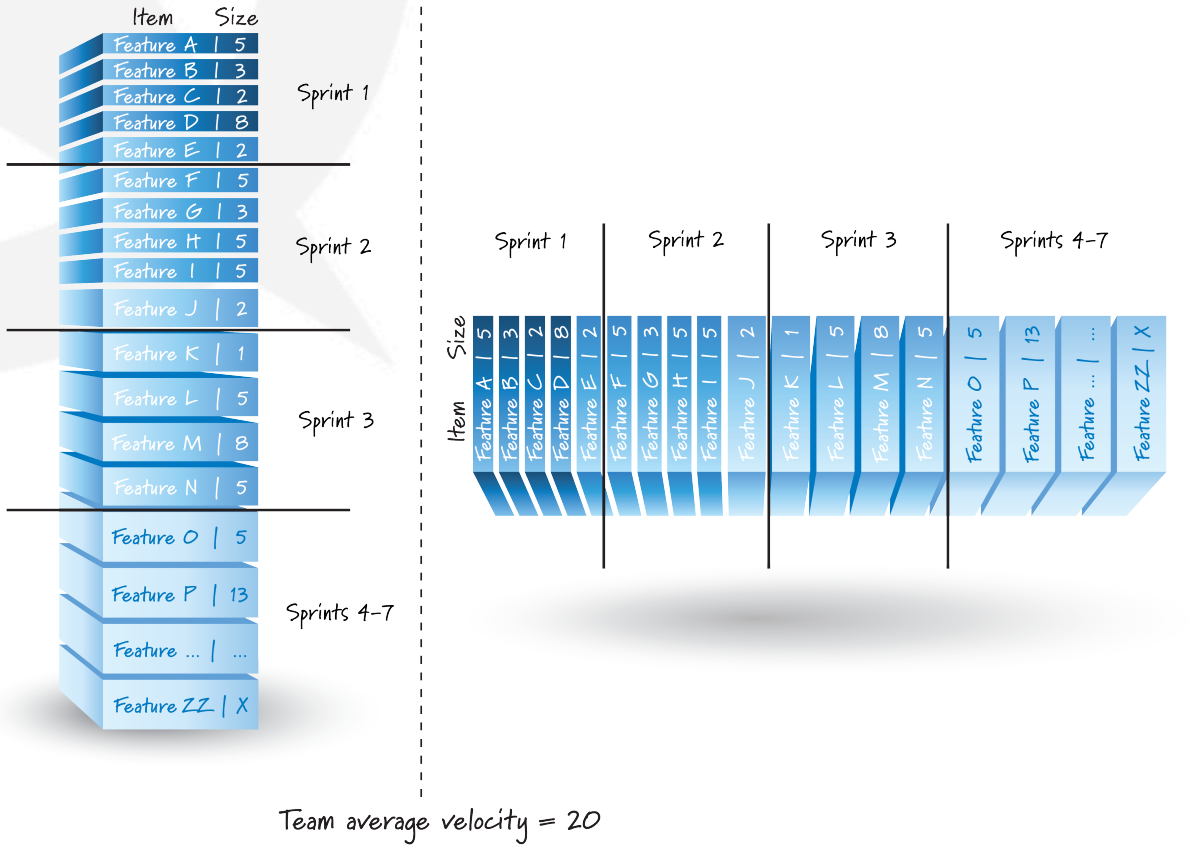
Predictive and Flexible



Item	Portfolio backlog	Product backlog	Sprint backlog tasks
Unit	T-shirt sizes	Story points / ideal days	Ideal hours / effort-hours
When	Portfolio planning	Product backlog grooming	Sprint planning

Product Increment Planning

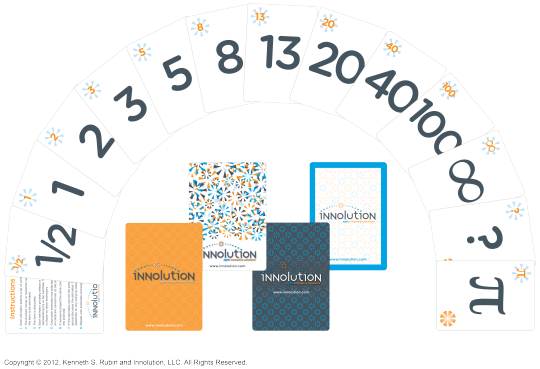
Planning feature sets in 8 week increments



Estimation

Planning Poker

- All team members place their "Bet" on the effort required to complete a backlog item
- Modified Fibonacci scale used to determine level of effort (1,2,3,5,8,13,21…)
- Wide discrepancies discussed
- Consensus

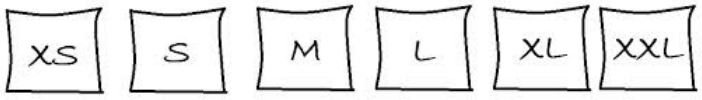


Relative Estimation

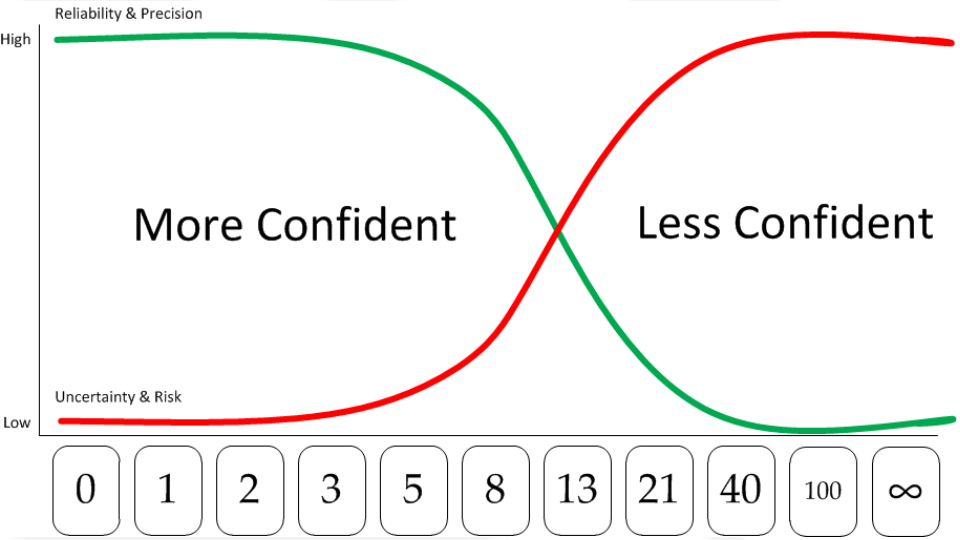
- Sized based on already sized well known reference stories

Tee Shirt Sizing

- Shirt Sizes are ranges of Story Points used for high level estimating only



Scale

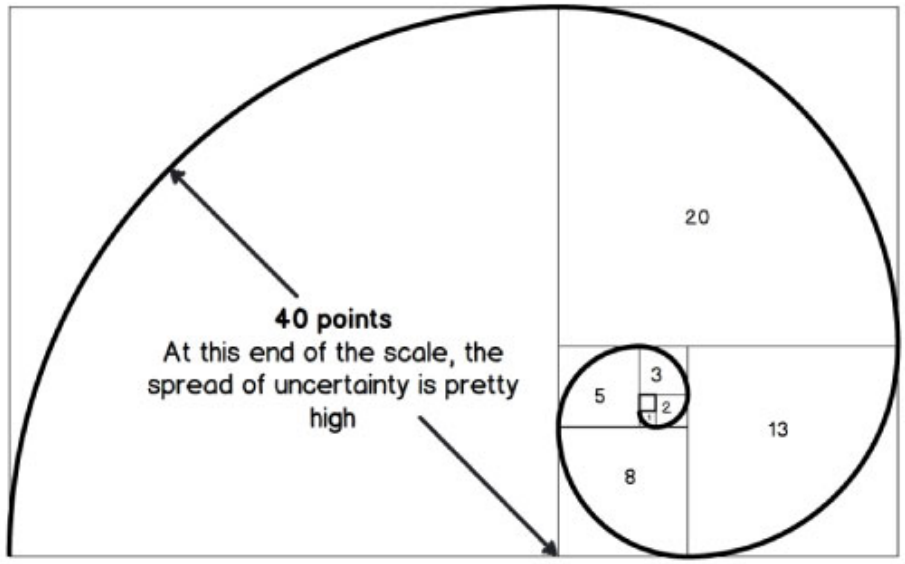


Care must be taken to estimate based on a scale that lends itself to success

Beyond 8 points risk exceeds confidence.

Beyond 13 points chances for success are low

Beyond 20 points failure is assured



INNTOPIA

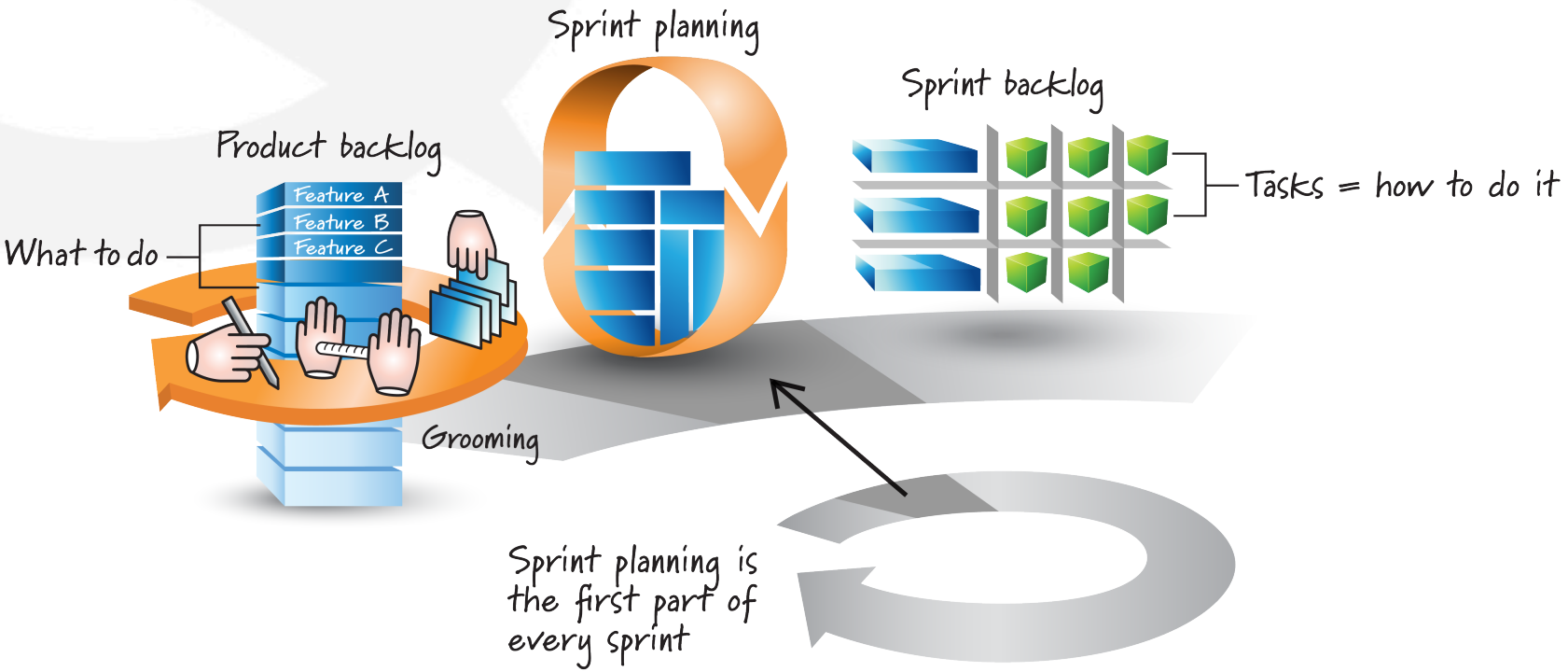
INNOVATIVE TRAVEL TECHNOLOGY

SCRUM Ceremonies

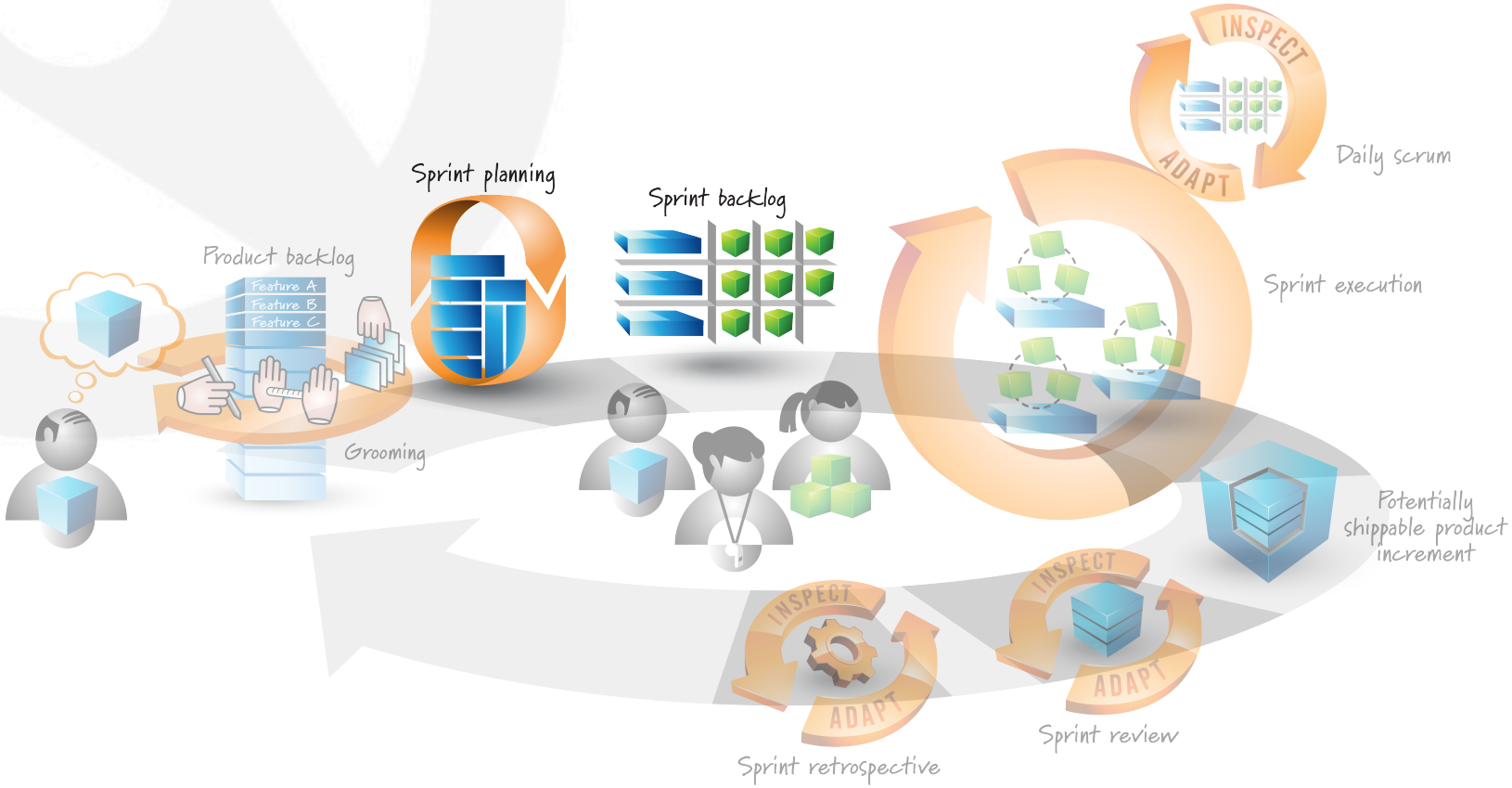
Backlog Grooming

Sprint Planning

Breaking Stories down to actionable, testable tasks

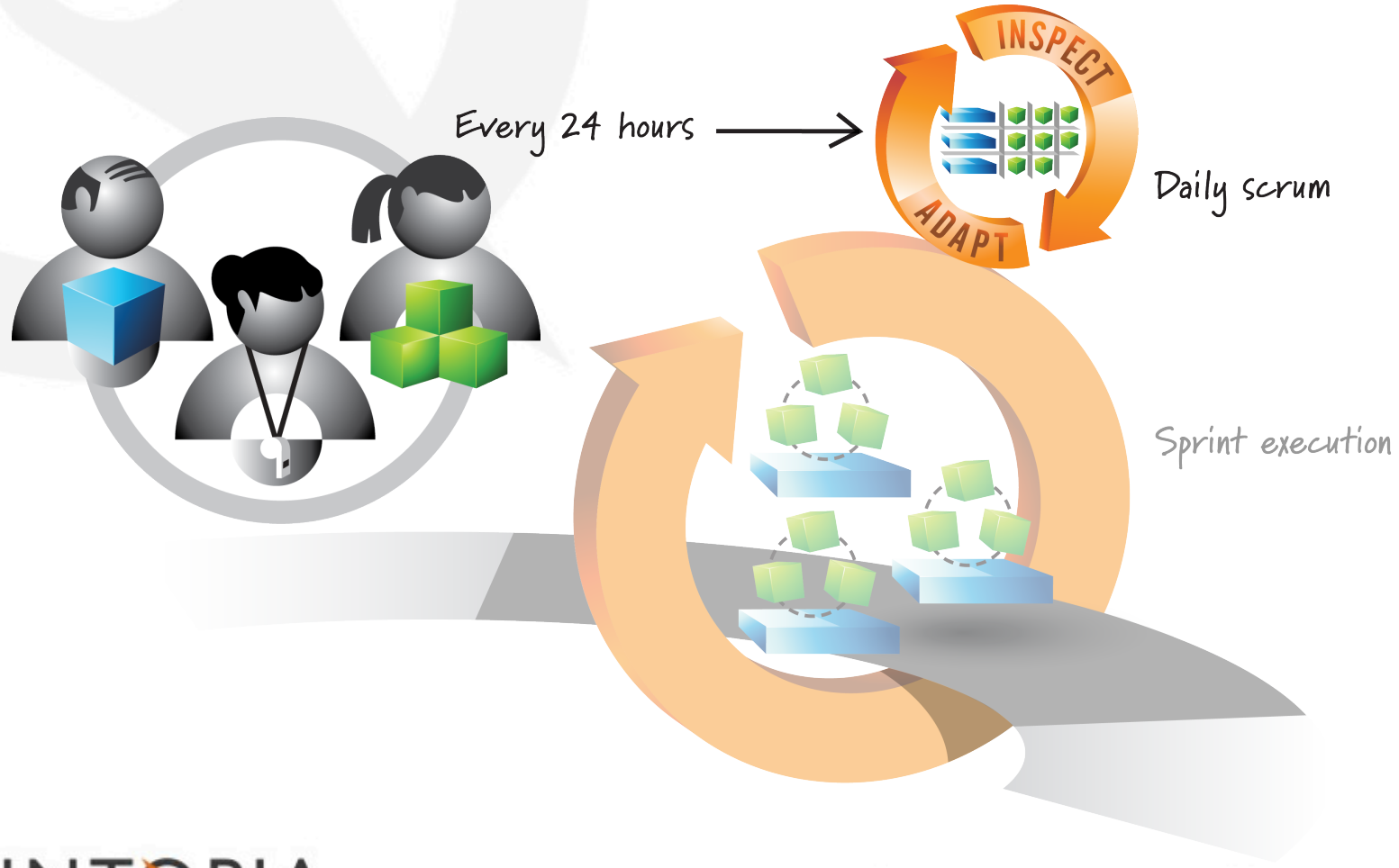


When Sprint Planning Happens

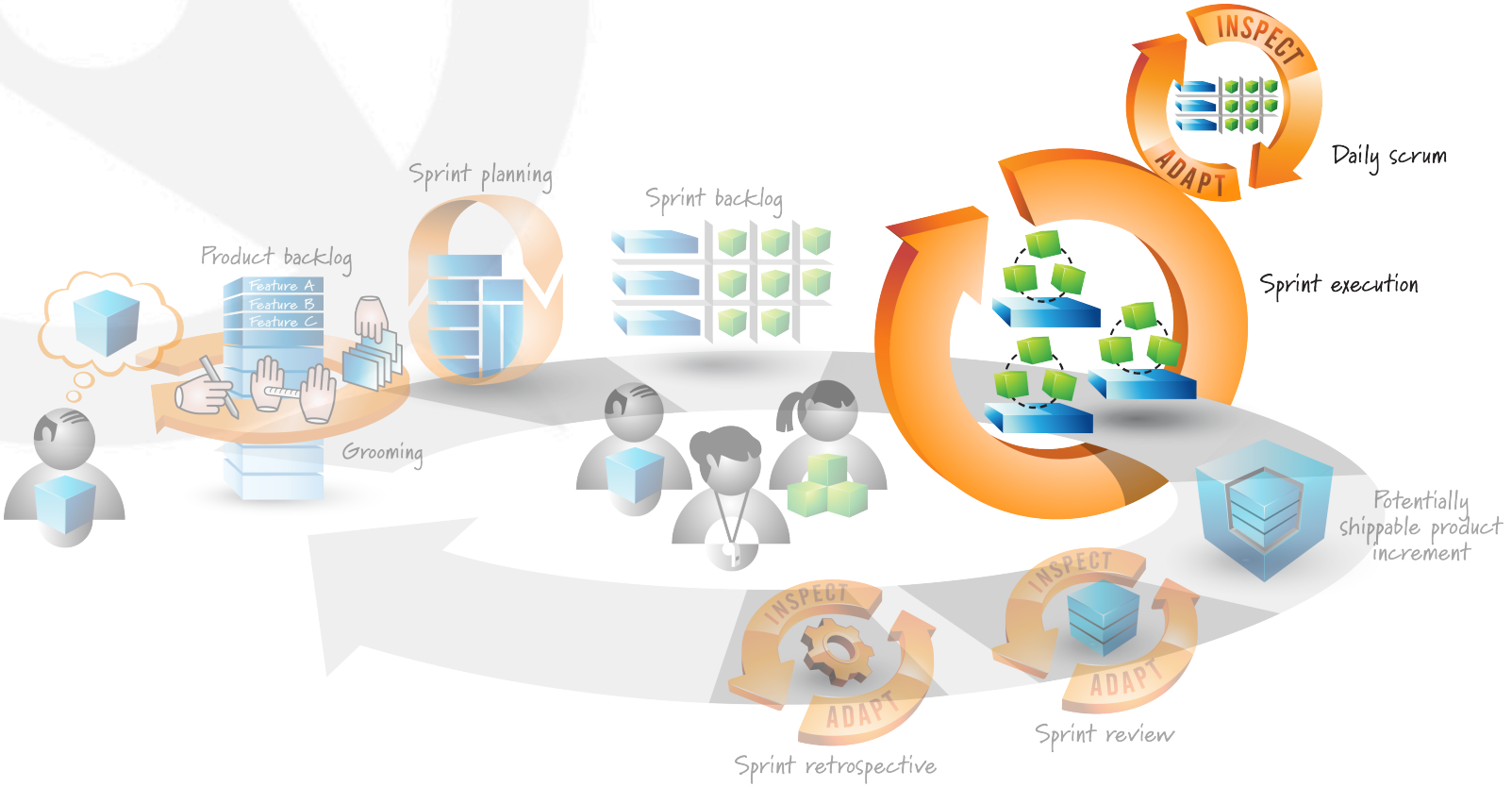


Daily Standup

What did I do – What will I do – What's blocking me.

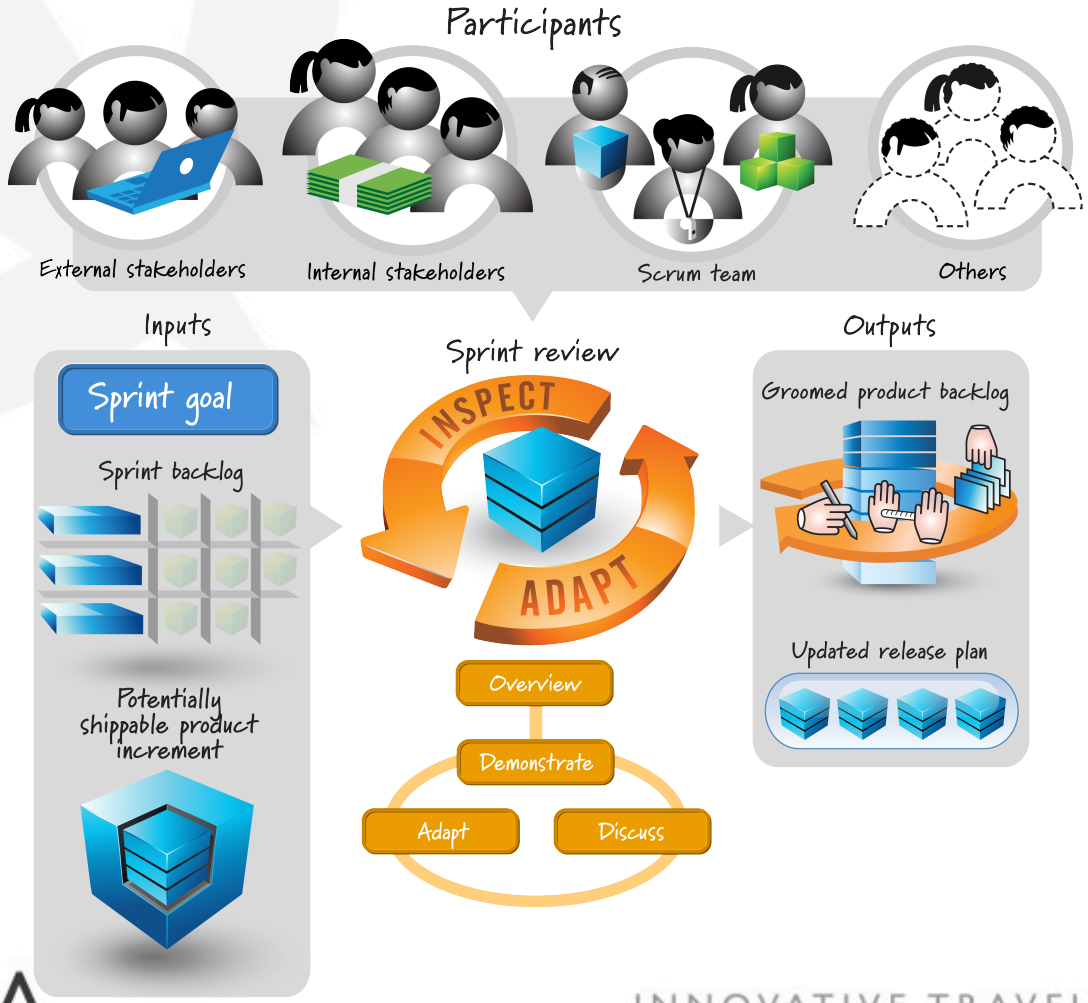


When Sprint Execution Happens



Sprint Review & Demo

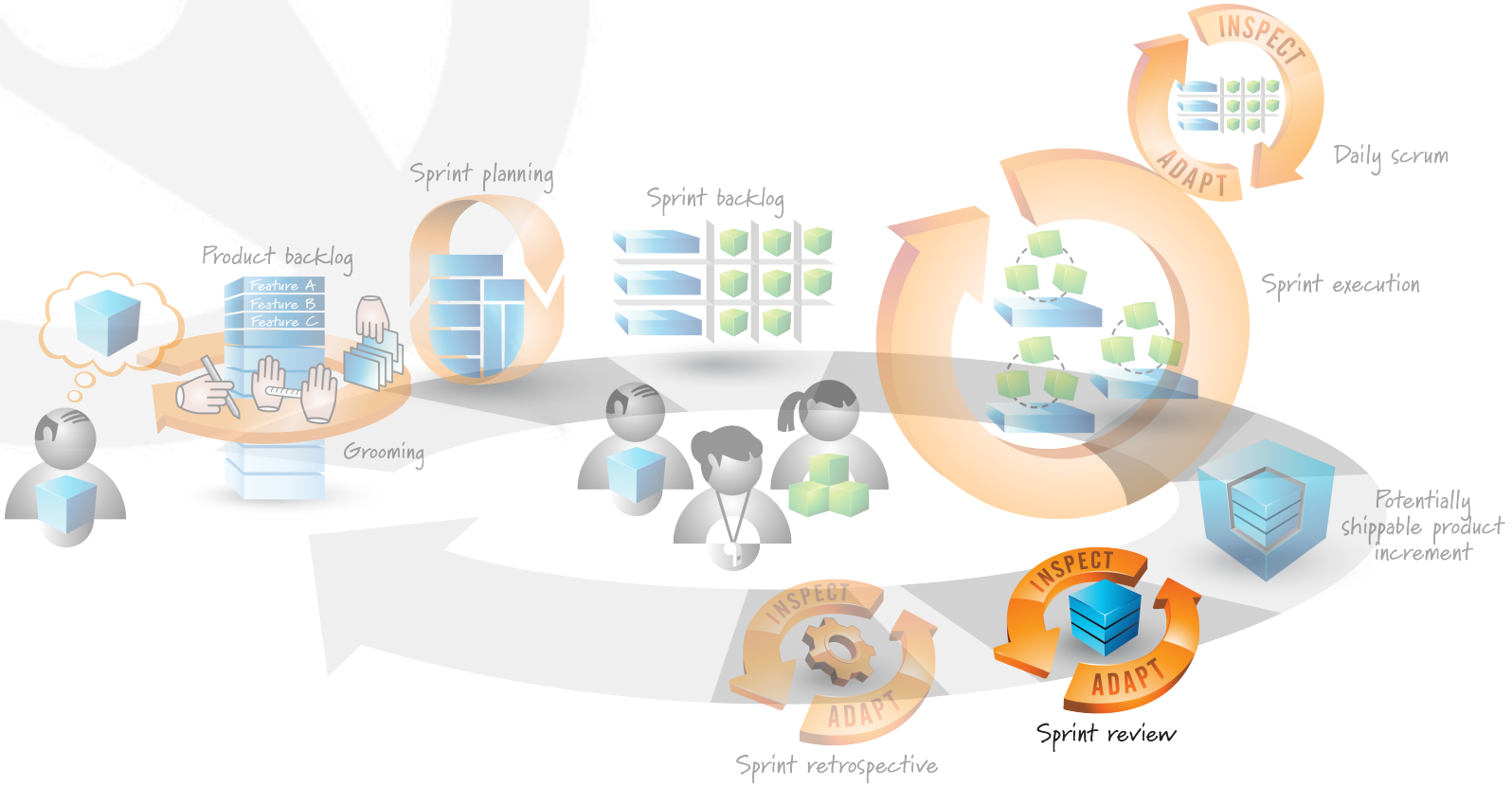
Live demo of potentially shippable product, all invited



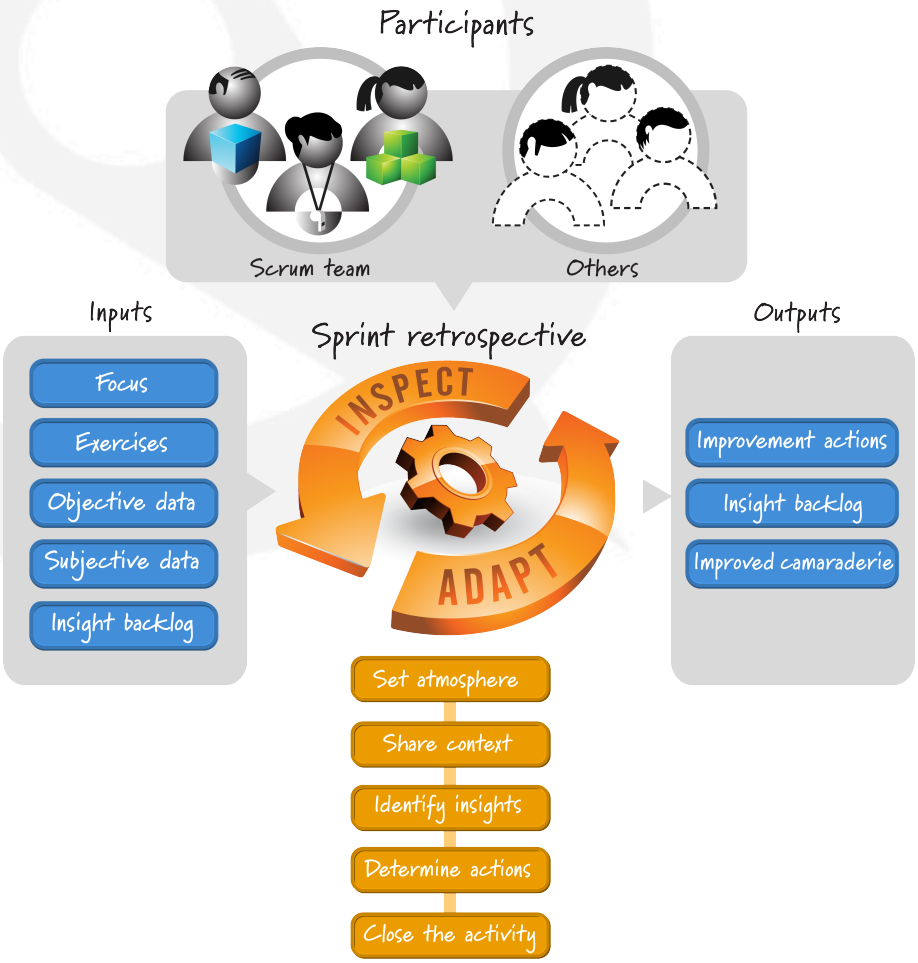
INNTOPIA

INNOVATIVE TRAVEL TECHNOLOGY

When Sprint Review Happens



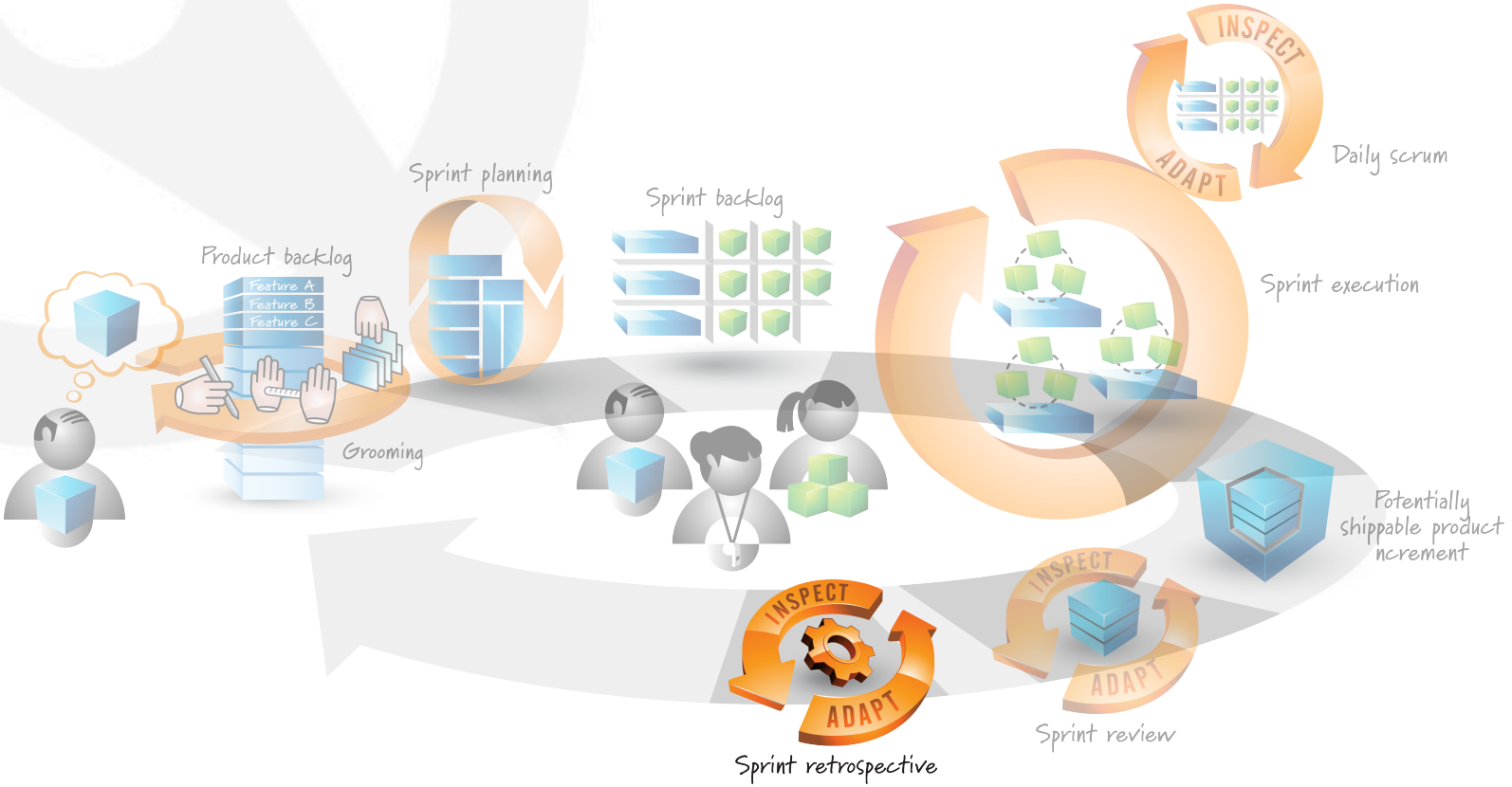
Retrospective



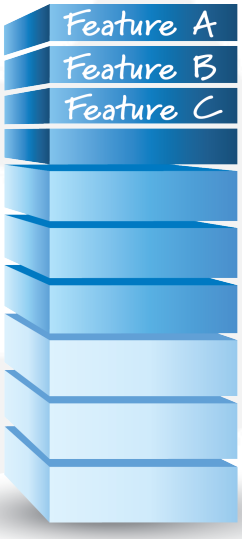
Sprint Retrospective

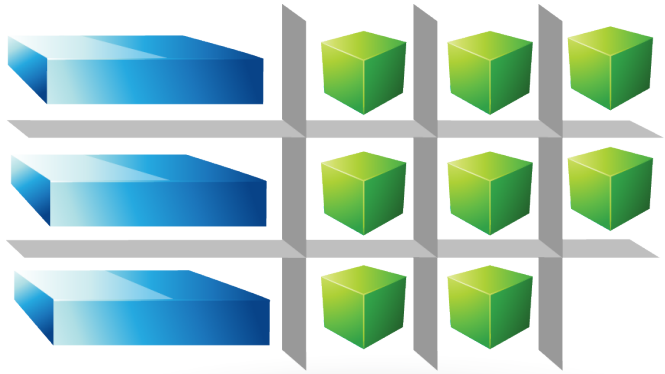
- Held at the end of every Sprint
- What worked well
- What needs improvement
- Progress towards release
- Only for team

When Sprint Retro Happens

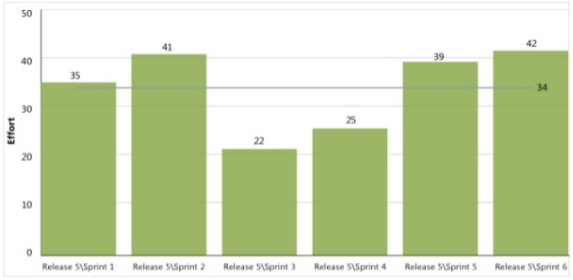


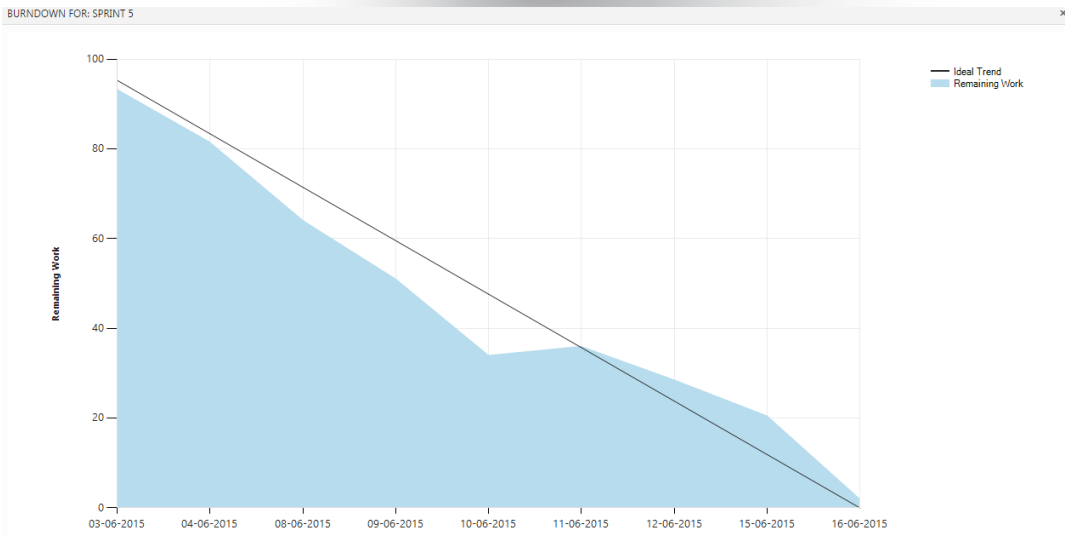
Scrum Artifacts





Velocity Chart





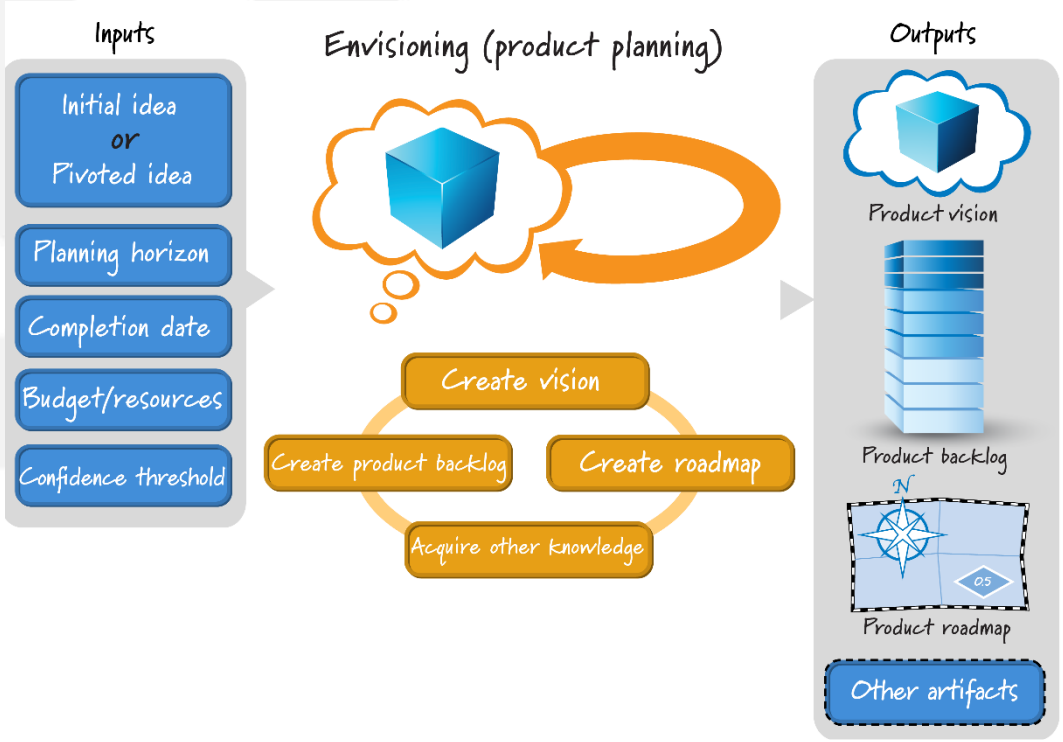
Request Workflow

Request is made to Inntopia

Every request for features or functions in Inntopia products is reviewed by the Product Management Team (PMT). The PMT evaluates requests based on a number of criteria and determines whether to devote resources to the request and then prioritizes it within the backlog of requests. Each request is subject to the following questions and, depending on the outcome, may or may not be passed to a Product Owner for discovery.

- Can we do it?
- Will we do it?
- Time?
- Cost?

Discovery Begins



Discovery is the process of working together to understand the desired feature(s), break them down into Product Backlog Items (PBIs) and ultimately into User Stories.

Discovery Cont.

Product Owner – Leads Discovery

The Product Owner's (PO) primary job is to maintain the Product Backlog of his/her team. The PO works closely with the stakeholders who have requested the functionality and the Inntopia stakeholders and team to ensure the requirements have been captured in enough detail that work can begin. To do this, the PO breaks a single request down into Features and User Stories as outlined below.

It is important to note that as soon as one Feature and its User Stories are complete, development can begin; there is no need for all Features and User Stories of a single request to be completed first. The process is iterative and Inntopia development will build the first feature while subsequent features are still in Discovery.

Features & Stories

Features

Features are stand-alone pieces of functionality that satisfy a business or system need. Features are also containers for multiple User Stories needed to define a feature. There is no specific format for a feature and the requirements are detailed in the stories. Features do, however, need to specify what is being asked for, what business case it solves, and how it will generate a return on its investment.

User Stories

User Stories are the preferred method for documenting requirements. They define system behaviors with concrete scenarios rather than ideas or large use cases that can cause confusion.

Communication

How & When

Continuous During Discovery

The PO is in regular communication with the partner upon commencement of discovery and continues until all the features and user stories are complete. In keeping with agile tenets, Inntopia expects high user involvement and embraces evolving and changing requirements.

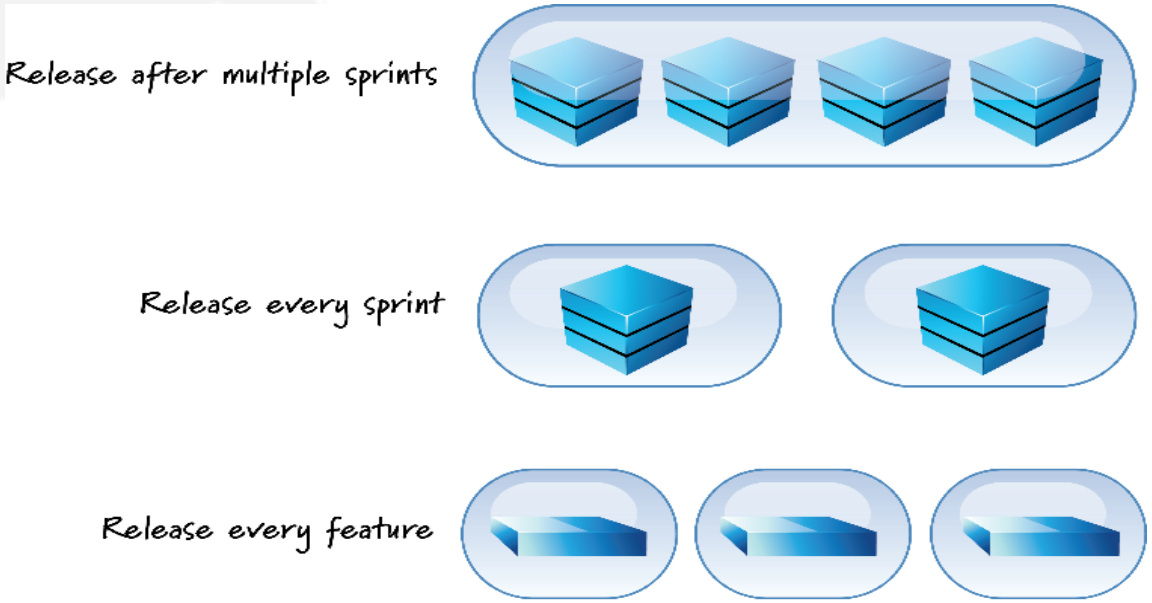
Biweekly Review of PSP

PSP is a Scrum term that means Potentially Shippable Product. The goal of each iteration (Sprint) is to create PSPs, meaning that User Stories are complete and ready to be deployed. It is important to note, however, that not all completed user stories have enough value to deploy or could stand alone and so *it is not automatic that Inntopia will deploy after every Sprint*.

At the end of every Sprint that involves work for a partner, Inntopia reviews the output with that partner. This provides real-time feedback for both the partner and the team as to progress and efficacy of the work being created and allows the partner to react to incremental development and create incremental change in the requirements. This communication is usually provided by the Senior Account Manager but it may also be delivered by the PO, or a technical resource.

Release

Inntopia releases code to production based on value, potentially every two weeks. Value is subjective and is decided through collaboration between Inntopia and its partners. We have the ability to craft releases based on features, Sprints, and combinations of both. In general, the approach is to release code early and often, as soon as there is value to functionality. In this way, we maximize the return on investment, and minimize the time to market on new features.



Complete

Inntopia does not consider a feature complete until the following conditions are met:

- All Stories of all Features accepted
- Customer UAT completed
- Deployment to Production completed

Questions?