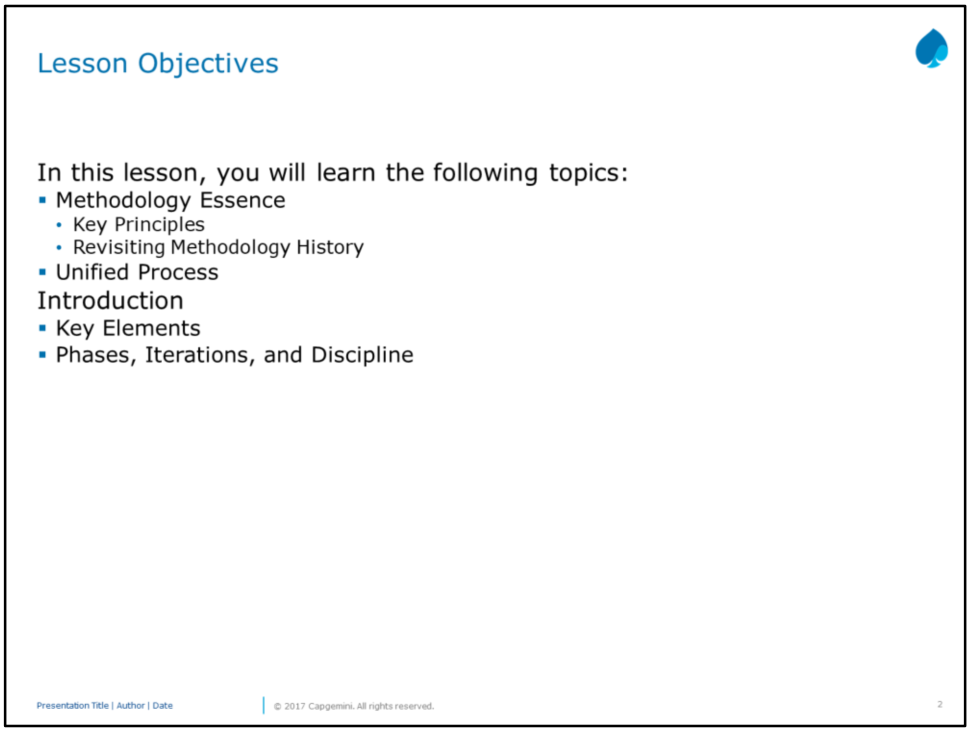
Introduction to Software Design & Architecture

Lesson 1: Methodology Overview

Capgemini



Lesson Objectives

In this lesson, you will learn the following topics:
- Methodology Essence
  - Key Principles
  - Revisiting Methodology History
- Unified Process

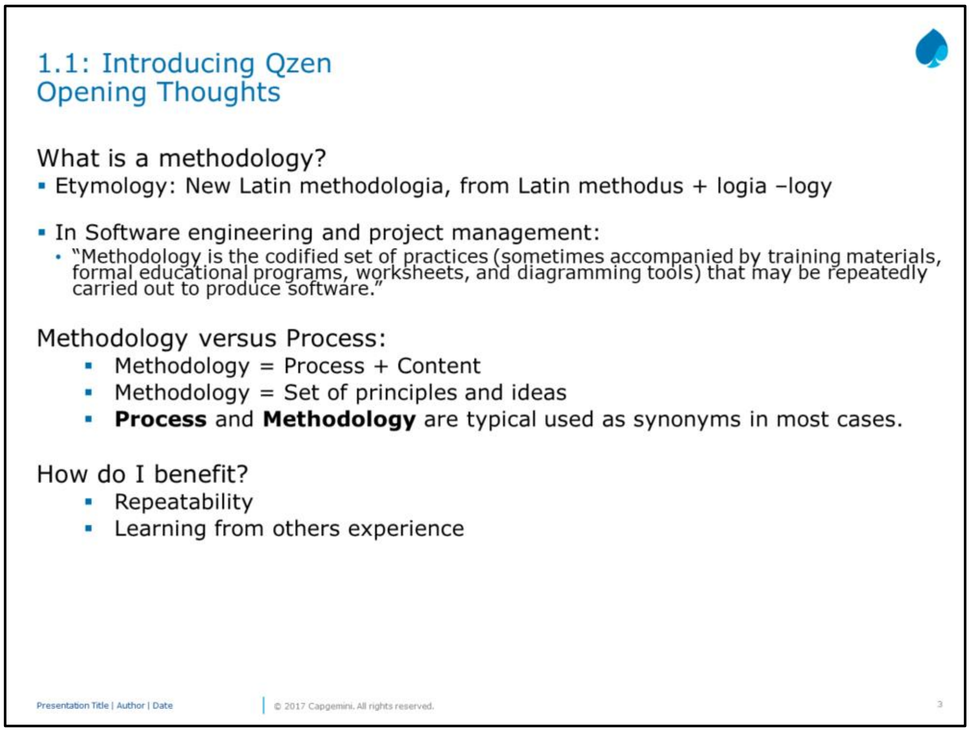
Introduction
- Key Elements
- Phases, Iterations, and Discipline

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

2

<u>Lesson Objectives</u>:
OOAD course uses Qzen Methodology. Hence we begin this course with a broad level understanding of the methodology and key concepts and principles that are associated with it.
Our focus will be on **Architecture and Design discipline**.



1.1: Introducing Qzen
Opening Thoughts

What is a methodology?

- Etymology: New Latin methodologia, from Latin methodus + logia –logy
- In Software engineering and project management:
  - "Methodology is the codified set of practices (sometimes accompanied by training materials, formal educational programs, worksheets, and diagramming tools) that may be repeatedly carried out to produce software."

Methodology versus Process:

- Methodology = Process + Content
- Methodology = Set of principles and ideas
- **Process** and **Methodology** are typical used as synonyms in most cases.

How do I benefit?

- Repeatability
- Learning from others experience

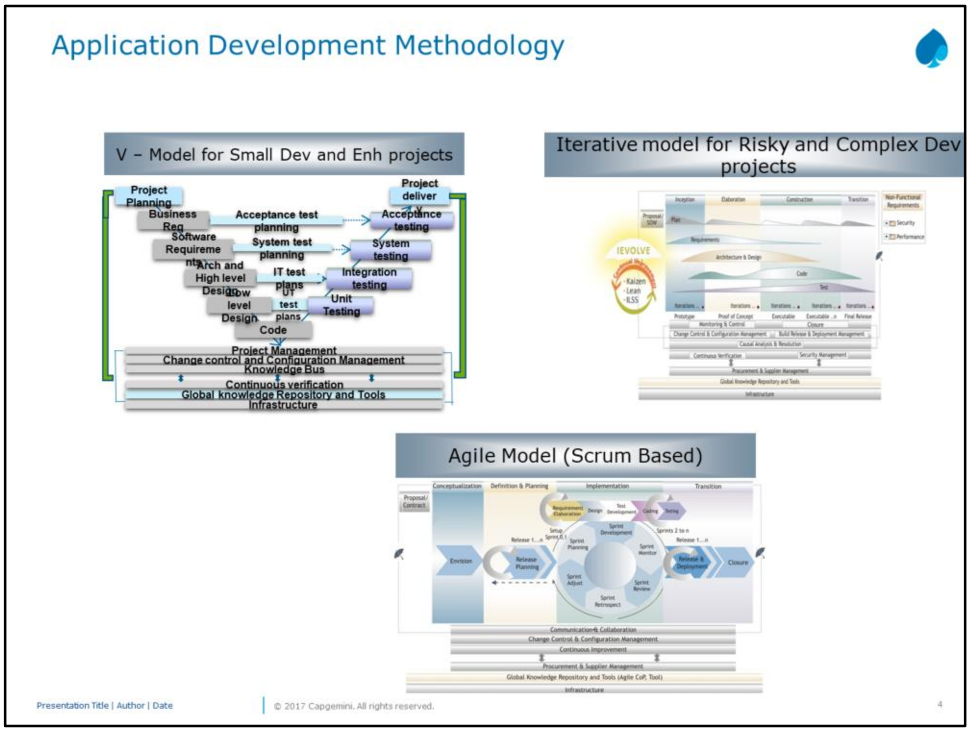
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 3

<u>Opening Thoughts</u>:

- Methodology is a body of methods, rules, and postulates employed by a discipline.
- It is a particular procedure or set of procedures.
- It is the analysis of the principles or procedures of inquiry in a particular field, or "a collection of theories, concepts or ideas".

- A process typically talks about what, when, and how aspects of a development approach. Along with these details, the methodology encompasses related set of principles and ideas for choosing a certain approach for development.
- The terms do tend to get used interchangeably.

- A methodology captures best practices for effective software development.
- A methodology is well defined. Hence there is the advantage of being able to repeatedly apply it with expectations of similar results across domains and projects. Especially in the context of Qzen the scenarios that work in "iGATE context" have been considered while structuring the methodology.



Application Development Methodology

V – Model for Small Dev and Enh projects

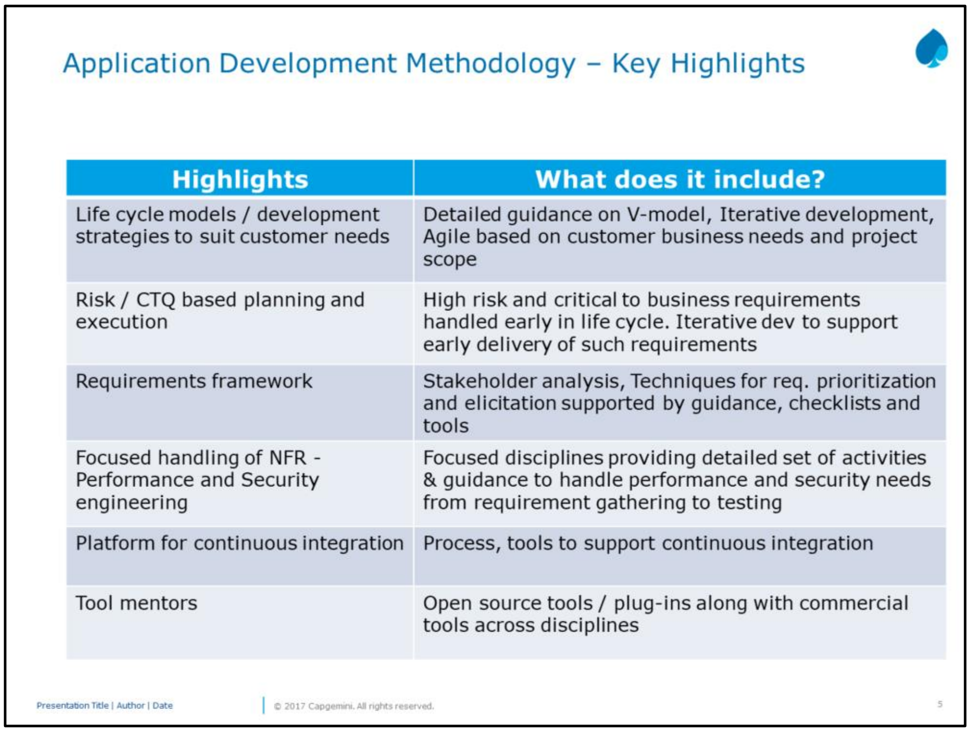
Iterative model for Risky and Complex Dev projects

Agile Model (Scrum Based)

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

4

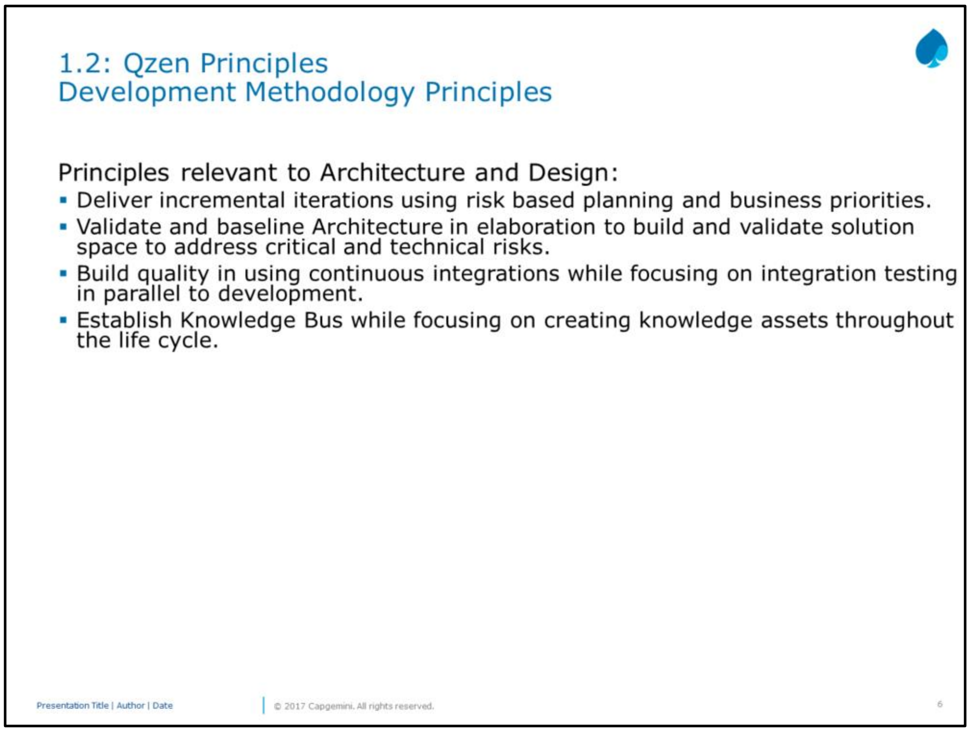


Application Development Methodology – Key Highlights

Highlights	What does it include?
Life cycle models / development strategies to suit customer needs	Detailed guidance on V-model, Iterative development, Agile based on customer business needs and project scope
Risk / CTQ based planning and execution	High risk and critical to business requirements handled early in life cycle. Iterative dev to support early delivery of such requirements
Requirements framework	Stakeholder analysis, Techniques for req. prioritization and elicitation supported by guidance, checklists and tools
Focused handling of NFR - Performance and Security engineering	Focused disciplines providing detailed set of activities & guidance to handle performance and security needs from requirement gathering to testing
Platform for continuous integration	Process, tools to support continuous integration
Tool mentors	Open source tools / plug-ins along with commercial tools across disciplines

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 5

1) Life cycle model selection is based on project classification and customer needs. Projects with life cycle scope to use iterative development to manage risks and complexities. For restricted project scope such as only LLD, Code and UT V model to be used. Agile model also available.
2) Requirements framework include detailed process, checklists, templates which aid requirement elicitation as well as analysis.
Elicitation workbook is comprehensive workbook for stakeholder analysis, requirement gathering /elicitation, prioritization etc.
Multiple prioritization techniques are discussed such as mind maps, QFD.
Exhaustive templates available for documenting BRD and various components of SRS (use cases, user stories, NFR template, UI specs, report specs etc.)
Guidance available of UI designing
3)Complete life cycle activities for performance and security engineering include steps to be taken at various project stages along with guidance and checklists. E.g. performance pattern and techniques, security checklist etc.
4) Tool mentors include tools available in iGate – open source as well licensed.



1.2: Qzen Principles
Development Methodology Principles

Principles relevant to Architecture and Design:

- Deliver incremental iterations using risk based planning and business priorities.
- Validate and baseline Architecture in elaboration to build and validate solution space to address critical and technical risks.
- Build quality in using continuous integrations while focusing on integration testing in parallel to development.
- Establish Knowledge Bus while focusing on creating knowledge assets throughout the life cycle.

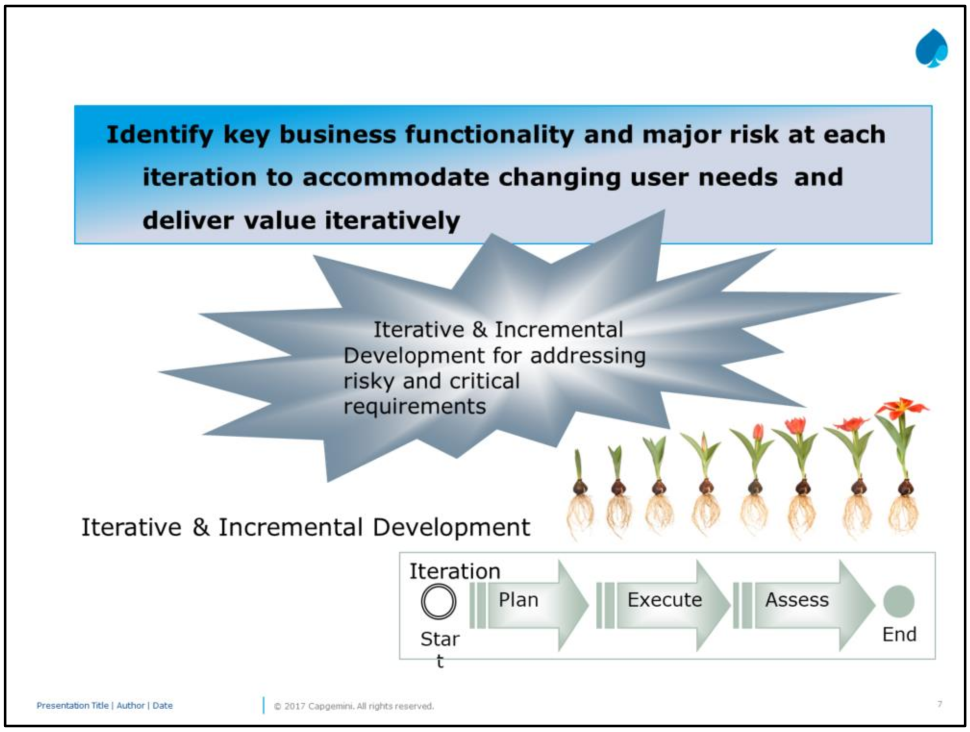
Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

6

Qzen Principles:

- The above slide lists the key driving principles for Qzen.
- Qzen is based on concepts of **Unified Process**.
- The first three principles are integral to the Unified Process. The remaining principles are derived from organization’s experiences of executing projects in a global development environment.

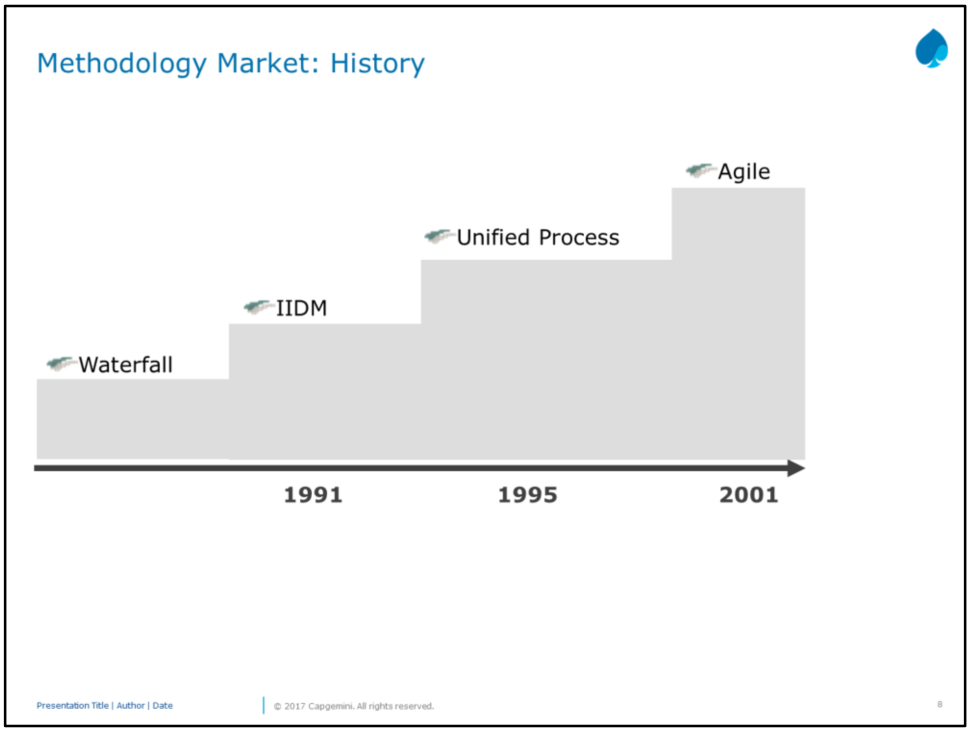


Identify key business functionality and major risk at each iteration to accommodate changing user needs and deliver value iteratively

Iterative & Incremental Development for addressing risky and critical requirements
Iterative & Incremental Development
Iteration
Start
Plan
Execute
Assess
End

<u>Delivering Incremental Iterations</u>:

- In an iterative and incremental life cycle, development proceeds as a series of iterations that evolve into the final system. While planning iterations, the following factors are taken into consideration:
  - Requirement priority
  - Critically to customer
  - Risks foreseen in the project
  - Total scope of the project in terms of effort and schedule
- Qzen methodology gives detailed guidelines on iteration planning.



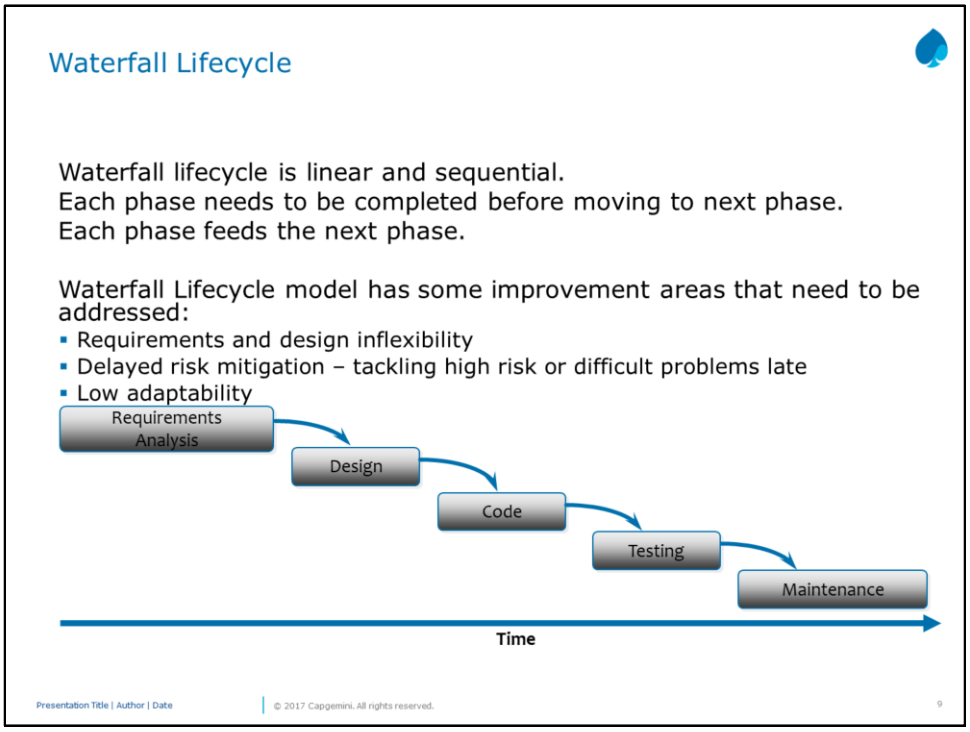
Methodology Market: History

Waterfall
IIDM
Unified Process
Agile
1991
1995
2001

© 2017 Capgemini. All rights reserved.

Methodology Market: History:

- Software Systems have become more complex, and development methodologies have also evolved over a period of time.
- There are several development models available in the arena of software engineering, some of the key ones are listed in the slide.



Waterfall Lifecycle

Waterfall lifecycle is linear and sequential.
Each phase needs to be completed before moving to next phase.
Each phase feeds the next phase.

Waterfall Lifecycle model has some improvement areas that need to be addressed:

- Requirements and design inflexibility
- Delayed risk mitigation – tackling high risk or difficult problems late
- Low adaptability

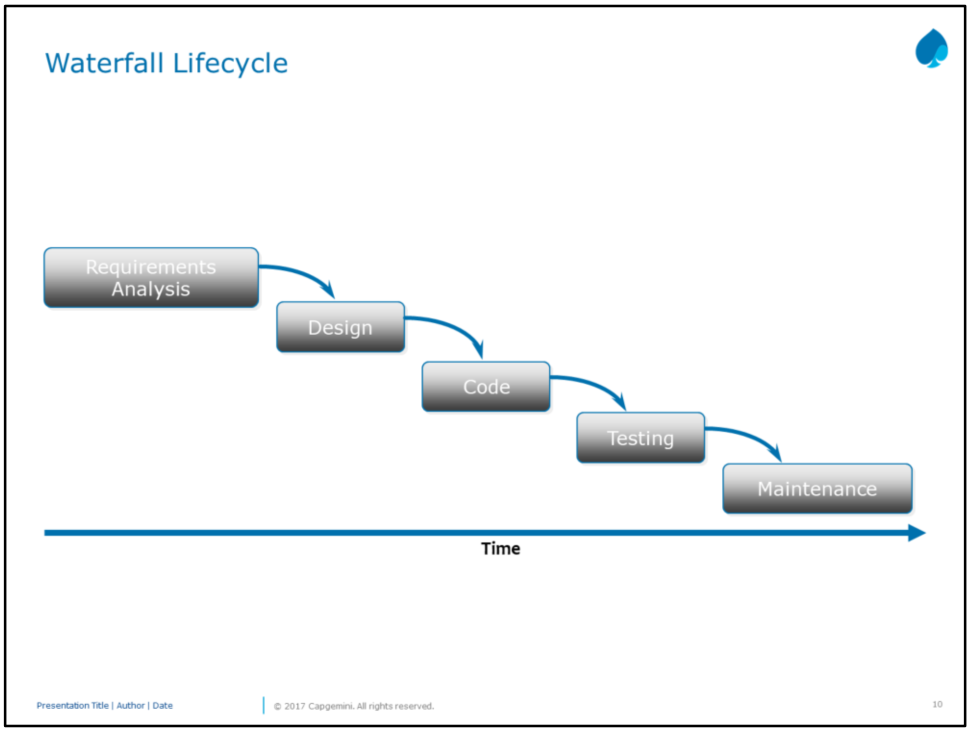
© 2017 Capgemini. All rights reserved.

<u>Waterfall Lifecycle</u>:

- The **Waterfall** or the **Classic Lifecycle model** is modeled after the conventional engineering cycle. It suggests a systematic, sequential approach to software development.
- Proposed by Winston Royce in 1970, it is the oldest and most widely used paradigm for software engineering.
- It is interesting to note that in the original Waterfall model, there were provisions for **feedback loops**. However, most organizations treat it as a strictly **linear** process. System development begins at system level, and proceeds through analysis, design, coding, testing, and maintenance.
- It is important to note that the next phase begins when the previous phase ends.

<u>Waterfall Lifecycle – Improvement Areas</u>:

- Rarely do projects follow a strict linear approach as defined in Waterfall Lifecycle. For example: The model assumes that all requirements are explicitly stated before the next phase begins. What happens if the requirements are not explicitly stated? If they are changed? In such cases, it is very difficult to accommodate such changes in this model.
- This leads to risks that need to be mitigated later in the lifecycle. Typically it is only when integration begins that the system starts falling apart!!
- Despite these limitations, this model has a definite and important place in software engineering because it provides a template in which the methods of analysis, design, coding, and testing can be placed.
- The advantages of the Waterfall model reside in its simplicity of approach. It is a good model to apply for development of systems that are well understood.



Waterfall Lifecycle

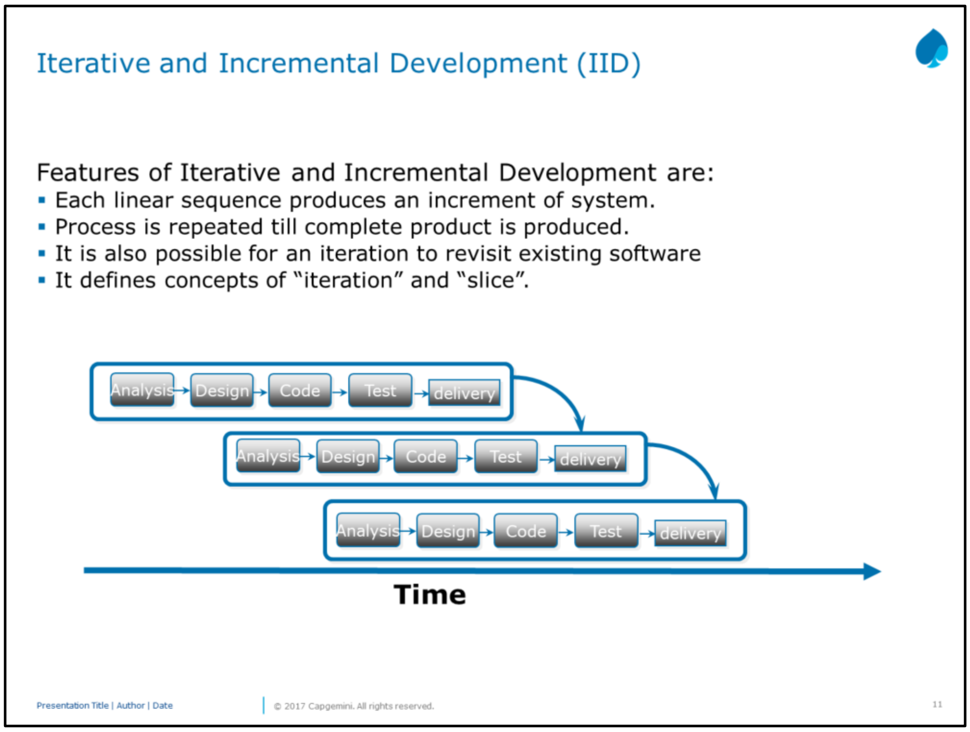
© 2017 Capgemini. All rights reserved.

<u>Waterfall Lifecycle</u>:

- The **Waterfall** or the **Classic Lifecycle model** is modeled after the conventional engineering cycle. It suggests a systematic, sequential approach to software development.
- Proposed by Winston Royce in 1970, it is the oldest and most widely used paradigm for software engineering.
- It is interesting to note that in the original Waterfall model, there were provisions for **feedback loops**. However, most organizations treat it as a strictly **linear** process. System development begins at system level, and proceeds through analysis, design, coding, testing, and maintenance.
- It is important to note that the next phase begins when the previous phase ends.

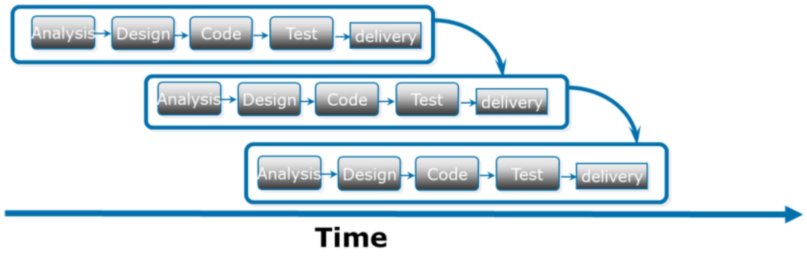
<u>Waterfall Lifecycle – Improvement Areas</u>:

- Rarely do projects follow a strict linear approach as defined in Waterfall Lifecycle. For example: The model assumes that all requirements are explicitly stated before the next phase begins. What happens if the requirements are not explicitly stated? If they are changed? In such cases, it is very difficult to accommodate such changes in this model.
- This leads to risks that need to be mitigated later in the lifecycle. Typically it is only when integration begins that the system starts falling apart!!
- Despite these limitations, this model has a definite and important place in software engineering because it provides a template in which the methods of analysis, design, coding, and testing can be placed.
- The advantages of the Waterfall model reside in its simplicity of approach. It is a good model to apply for development of systems that are well understood.



Iterative and Incremental Development (IID)

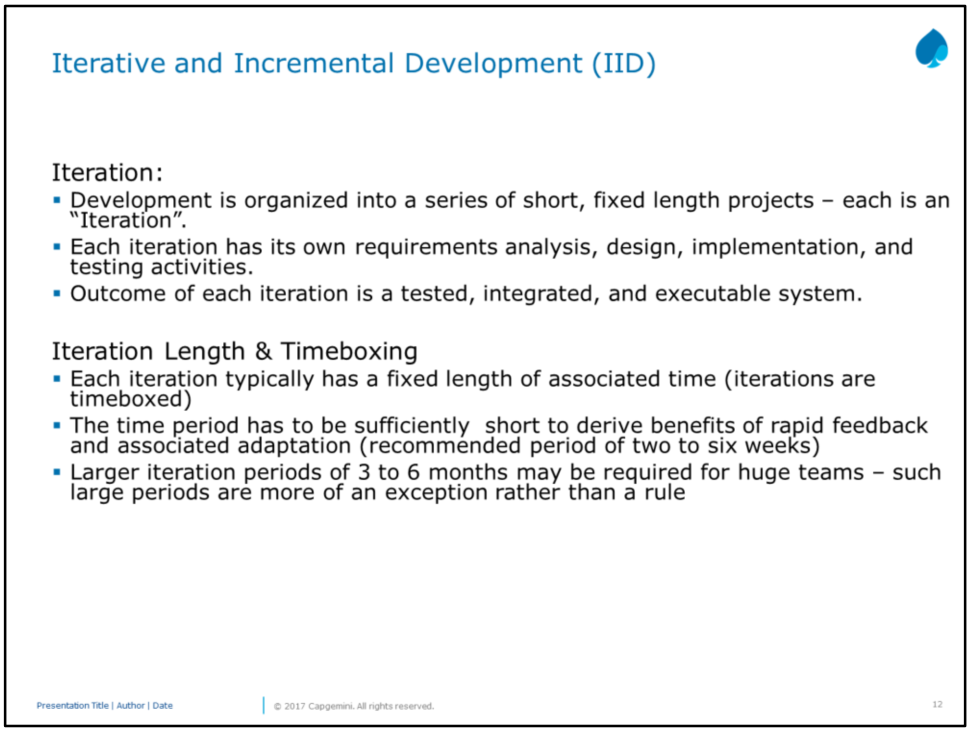
Features of Iterative and Incremental Development are:
- Each linear sequence produces an increment of system.
- Process is repeated till complete product is produced.
- It is also possible for an iteration to revisit existing software
- It defines concepts of "iteration" and "slice".



© 2017 Capgemini. All rights reserved.

<u>Iterative and Incremental Development</u>:

- Let us now discuss the **IID model**, which is an evolutionary model. This model combines the **Linear Sequential model**, with the iterative philosophy of the prototyping approach.
- The incremental model applies linear sequence in a staggered fashion as time progresses. Each sequence produces a deliverable increment of the software.
- The model is also iterative. Once the basic requirements are addressed in the first increment - thereby giving a core product - the core product can be evaluated by the customer. As a result, additional features and functionality can be given that can be taken up in the next increment. The process can be continued till the complete product is produced.
- **Example:** Let us try to understand this better with an example of a Word processing system. Using the incremental paradigm, the basic editing and file management features can be delivered in the first increment. Subsequent increment can address sophisticated editing features and other document processing capabilities. The third increment can add on spell checks and grammar checks. In this way, the process can be repeated until the product is complete as required.



Iterative and Incremental Development (IID)

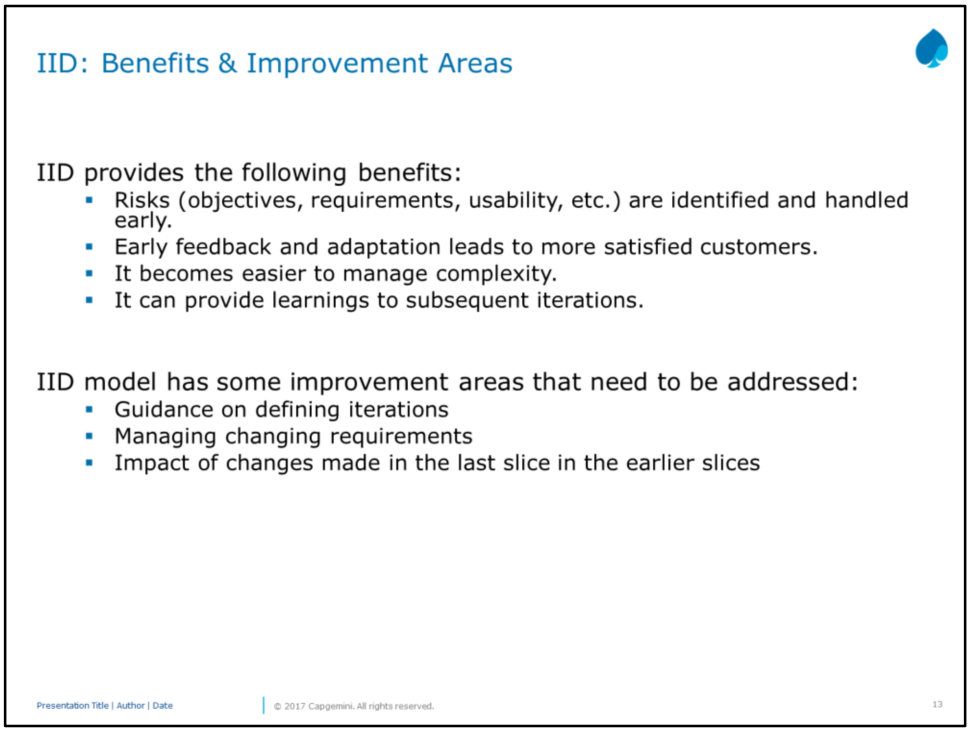
Iteration:
- Development is organized into a series of short, fixed length projects – each is an "Iteration".
- Each iteration has its own requirements analysis, design, implementation, and testing activities.
- Outcome of each iteration is a tested, integrated, and executable system.

Iteration Length & Timeboxing
- Each iteration typically has a fixed length of associated time (iterations are timeboxed)
- The time period has to be sufficiently short to derive benefits of rapid feedback and associated adaptation (recommended period of two to six weeks)
- Larger iteration periods of 3 to 6 months may be required for huge teams – such large periods are more of an exception rather than a rule

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 12

<u>Iterative and Incremental Development (IID)</u>:
- The concept of **iteration** is important! Note that an iteration is like a mini waterfall and has to end with an executable release.
- How long is an iteration? It is typically two to six weeks long as suggested in the guideline. However, its length could vary depending on the kind of project one is working on.

Here are the Motivations for Timeboxing
- Focus on Getting things done
- Prioritization and Decisiveness
- Team Satisfaction
- Stakeholder Confidence



IID: Benefits & Improvement Areas

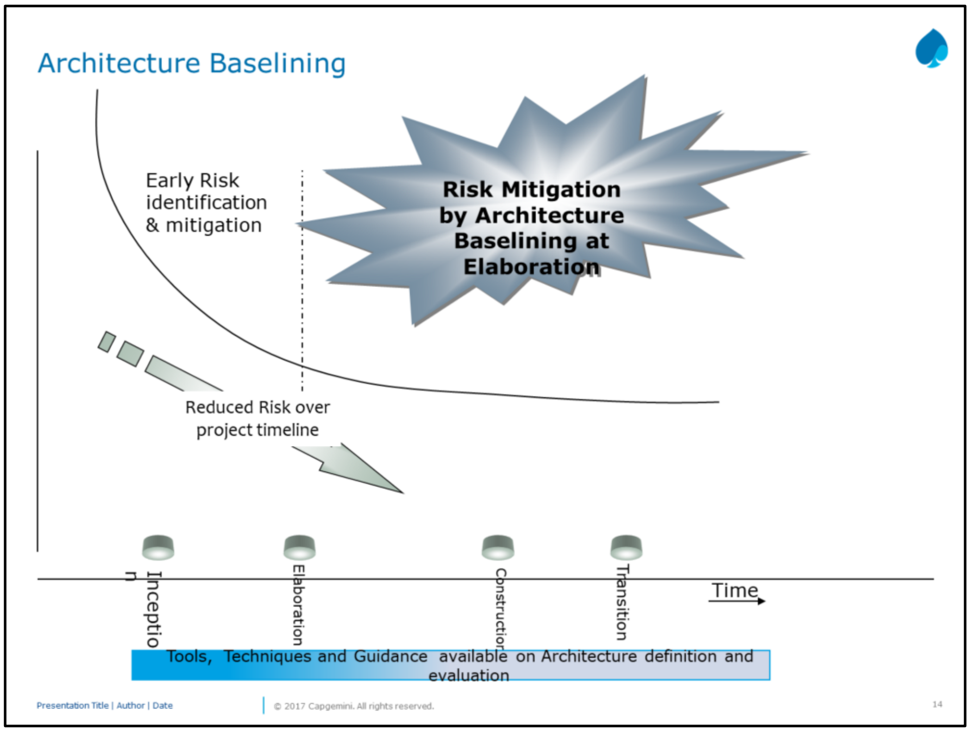
IID provides the following benefits:
- Risks (objectives, requirements, usability, etc.) are identified and handled early.
- Early feedback and adaptation leads to more satisfied customers.
- It becomes easier to manage complexity.
- It can provide learnings to subsequent iterations.

IID model has some improvement areas that need to be addressed:
- Guidance on defining iterations
- Managing changing requirements
- Impact of changes made in the last slice in the earlier slices

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 13

<u>Benefits of Iterative Development</u>:
- This model can be used to deliver product in phases - each phase moving towards completing required functionality. This way, it will be suitable to work on big or complex systems.
- Risks are now managed earlier in the lifecycle because high risk items are considered in the earlier iterations. Customer participation and feedback ensures that the software is better aligned to customer expectations.

<u>IID: Improvement Areas</u>:
- IID by itself does not include how to define iterations; and what is to be considered in each iteration. The methodology which bases itself on IID will have to provide this guidance.
- Making a paradigm shift to IID can be challenging for practitioners who are used to the traditional waterfall approach. The biggest challenge is ensuring that scope of requirements do not change across the iterations.
- The Project Manager will have a crucial role to play in ensuring the success of the IID approach.

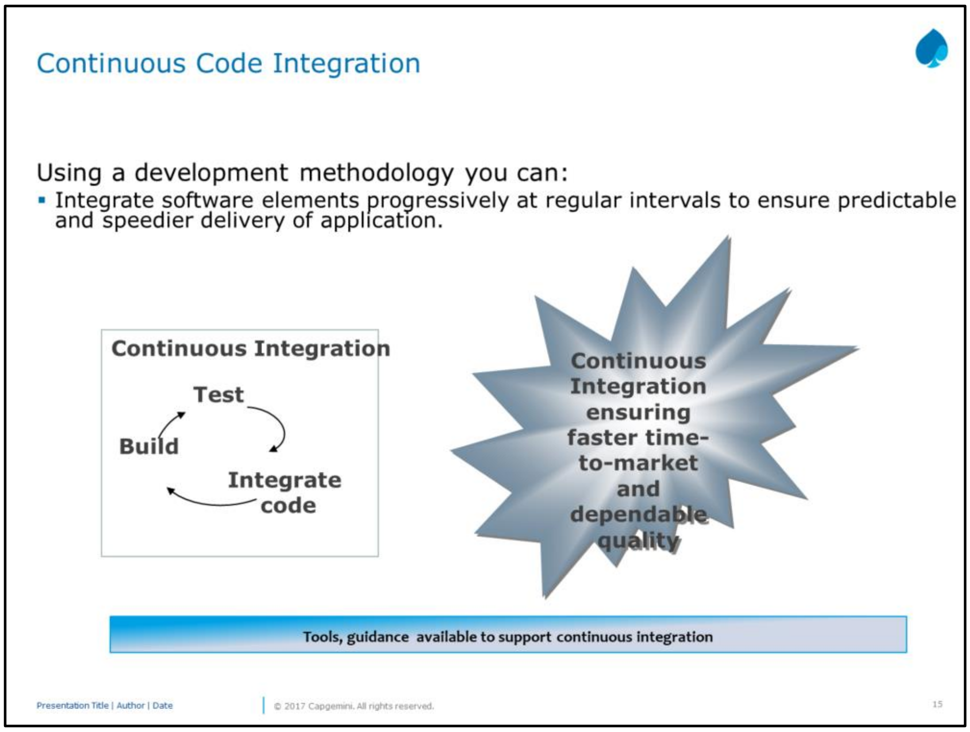


Architecture Baselining

Early Risk identification & mitigation
Risk Mitigation by Architecture Baselining at Elaboration
Reduced Risk over project timeline
Inception
Elaboration
Construction
Transition
Time
Tools, Techniques and Guidance available on Architecture definition and evaluation
Presentation Title | Author | Date
© 2017 Capgemini. All rights reserved.
14

Architecture Baselining:

- The methodology focuses on mitigating key technical risks early in the lifecycle and developing an architecture baseline. Architecture is considered as one of the biggest risks. Hence it is worked upon in the earlier iterations.
- Late-phase discovery of design defects results in costly over-runs and/or project cancellation.
- Key risks addressed:
  - Technical risks by implementing and testing the architecture
  - Business risks by implementing and testing key functionality
  - Team- and tool-oriented risks by having the team going through the full Software lifecycle implementing real code, by using the tools at hands
- Schedule and cost estimates can be radically improved because we:
  - Have mitigated key risks
  - Understand a vast majority of the requirements
  - Understand the building blocks that need to be implemented and tested
  - Understand the building blocks that can be acquired and at what cost
  - Understand the effectiveness of the team



Continuous Code Integration

Using a development methodology you can:

- Integrate software elements progressively at regular intervals to ensure predictable and speedier delivery of application.

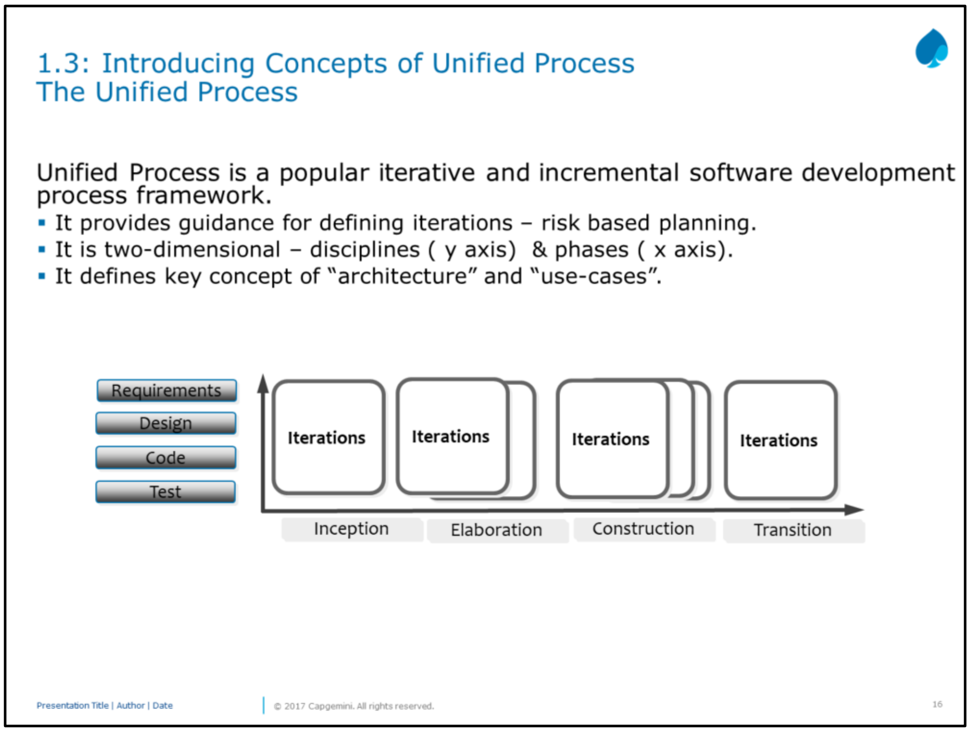
Continuous Integration
Test
Build
Integrate code
Continuous Integration ensuring faster time-to-market and dependable quality

Tools, guidance available to support continuous integration

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 15

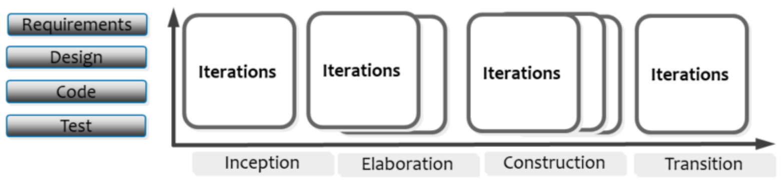
Building Quality In:

- Methodology provides for **incremental deliveries** or **builds** at defined frequencies. There is continuous integration of code with the rest of the system.
- Here are some practices through which **continuous integration** can be achieved:
  - Maintaining code in the source repository
  - Understanding and automating the build
  - Having a separate integration environment
  - Programmers sync up with the configuration management source code, integrate, and compile (in IDE) all the code in a system at least once a day
- Separate Development and Test teams can help in conducting integration testing in parallel with development.



1.3: Introducing Concepts of Unified Process
The Unified Process

Unified Process is a popular iterative and incremental software development process framework.
- It provides guidance for defining iterations – risk based planning.
- It is two-dimensional – disciplines (y axis) & phases (x axis).
- It defines key concept of "architecture" and "use-cases".

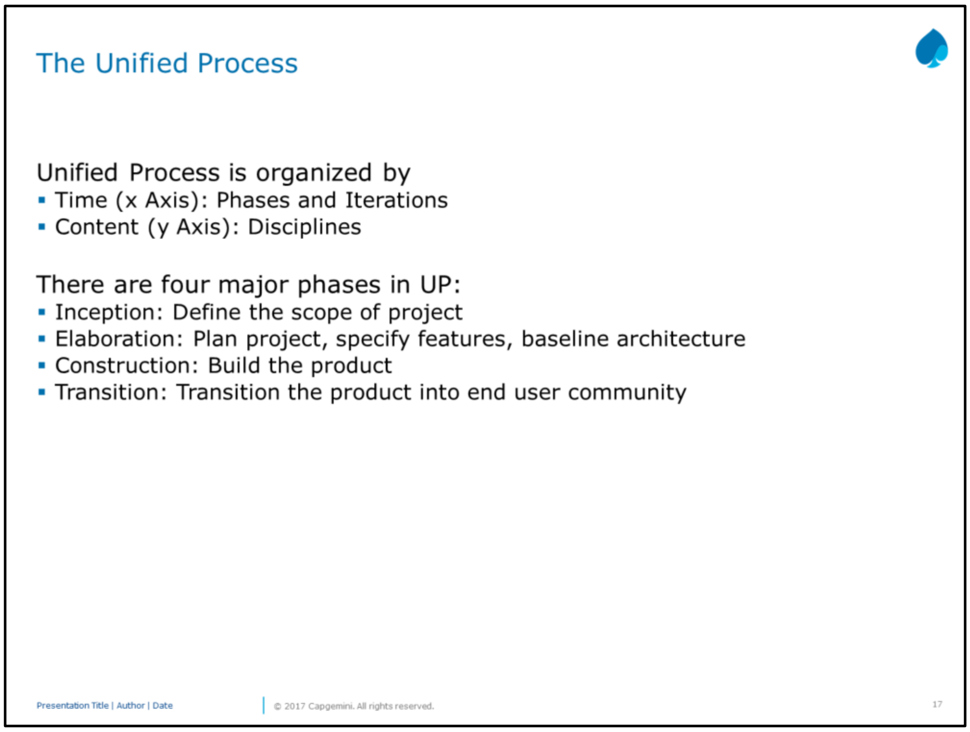


Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

16

<u>The Unified Process</u>:
- **Qzen** is a variant of the **Unified Process (UP)**, thus imbibing the concepts and best practices of the Unified Process.
- UP is a popular model for developing Object-Oriented systems. The key features and concepts of UP are enumerated in the subsequent slides.
- UP is a process framework. It means that one can customize the process to suit organization and project requirements.



The Unified Process

Unified Process is organized by
- Time (x Axis): Phases and Iterations
- Content (y Axis): Disciplines

There are four major phases in UP:
- Inception: Define the scope of project
- Elaboration: Plan project, specify features, baseline architecture
- Construction: Build the product
- Transition: Transition the product into end user community

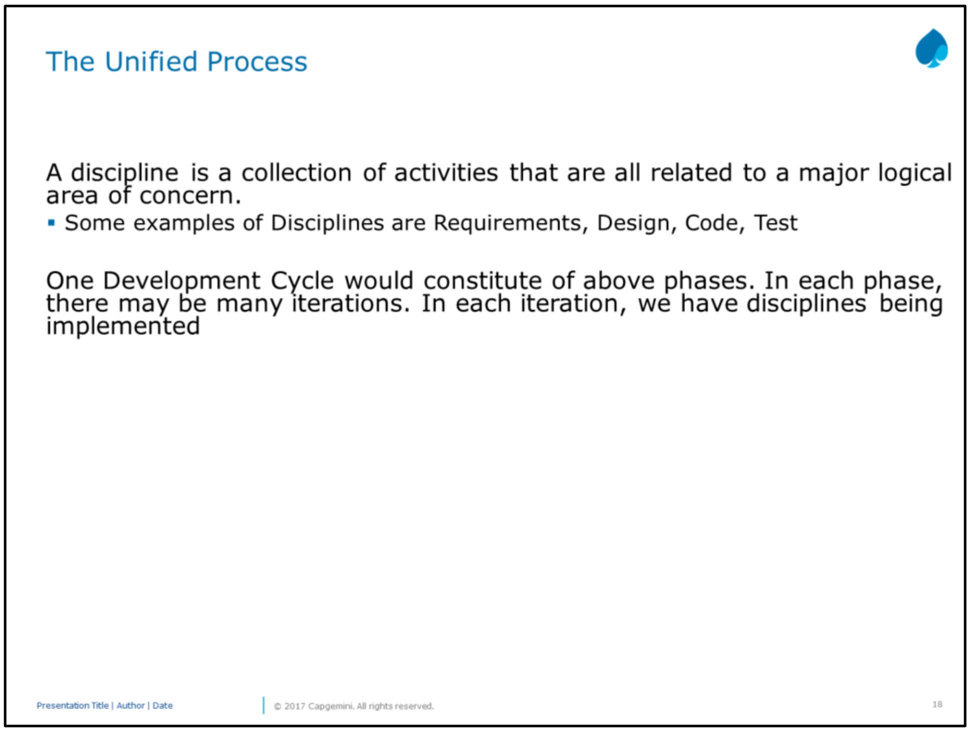
Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 17

The x axis of the Unified process is the dimension of time, and refers to the lifecycle aspect of the process i.e. how the process rolls out within the duration of a project. The y axis is the dimension of the contents and refers to the Disciplines, which group activities logically by nature.

Phases:
- Inception: A feasibility phase which helps in establishing the objectives and vision for the system & arrive at a Go/ No Go Decision. Objective is NOT to define ALL/ MOST of requirements or come up with detailed estimates for the project.
- Elaboration: Most requirements are discovered and stabilized & major risks are retired. Core Architecture is built & skeleton is ready.
- Construction: Putting the "Flesh on the Skeleton". Application is built, alpha tested & prepared for Beta Testing and Deployment. User Guides, Help & Manuals etc. need to be developed in this phase.
- Transition: System is put into Production (Beta Testing, Fine Tuning, Data Migration, Roll Out etc…).

Discipline: A set of activities and related artifacts in one area.

Phases constitute of iterations, in each iterations we would have disciplines; however focus & activities of disciplines would vary based on phase of development cycle.



The Unified Process

A discipline is a collection of activities that are all related to a major logical area of concern.

- Some examples of Disciplines are Requirements, Design, Code, Test

One Development Cycle would constitute of above phases. In each phase, there may be many iterations. In each iteration, we have disciplines being implemented

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 18

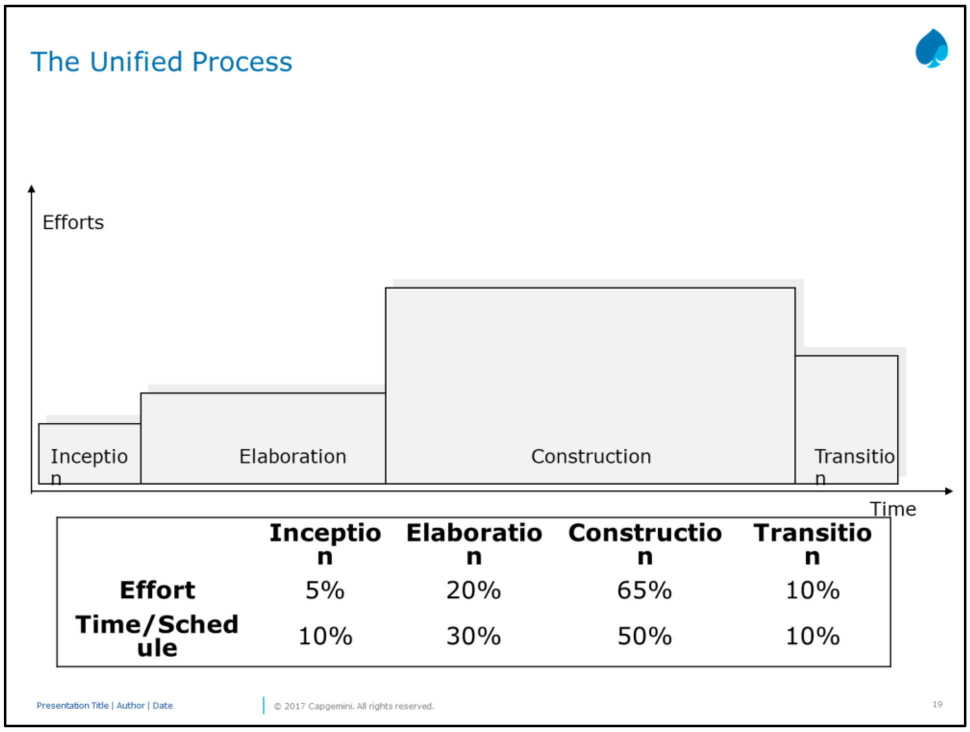
The x axis of the Unified process is the dimension of time, and refers to the lifecycle aspect of the process i.e. how the process rolls out within the duration of a project. The y axis is the dimension of the contents and refers to the Disciplines, which group activities logically by nature.

Phases:

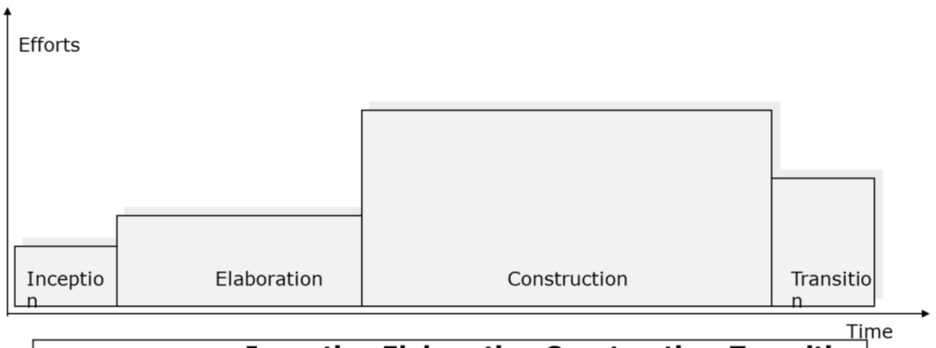
- Inception: A feasibility phase which helps in establishing the objectives and vision for the system & arrive at a Go/ No Go Decision. Objective is NOT to define ALL/ MOST of requirements or come up with detailed estimates for the project.
- Elaboration: Most requirements are discovered and stabilized & major risks are retired. Core Architecture is built & skeleton is ready.
- Construction: Putting the “Flesh on the Skeleton”. Application is built, alpha tested & prepared for Beta Testing and Deployment. User Guides, Help & Manuals etc. need to be developed in this phase.
- Transition: System is put into Production (Beta Testing, Fine Tuning, Data Migration, Roll Out etc…).

Discipline: A set of activities and related artifacts in one area.

Phases constitute of iterations, in each iterations we would have disciplines; however focus & activities of disciplines would vary based on phase of development cycle.



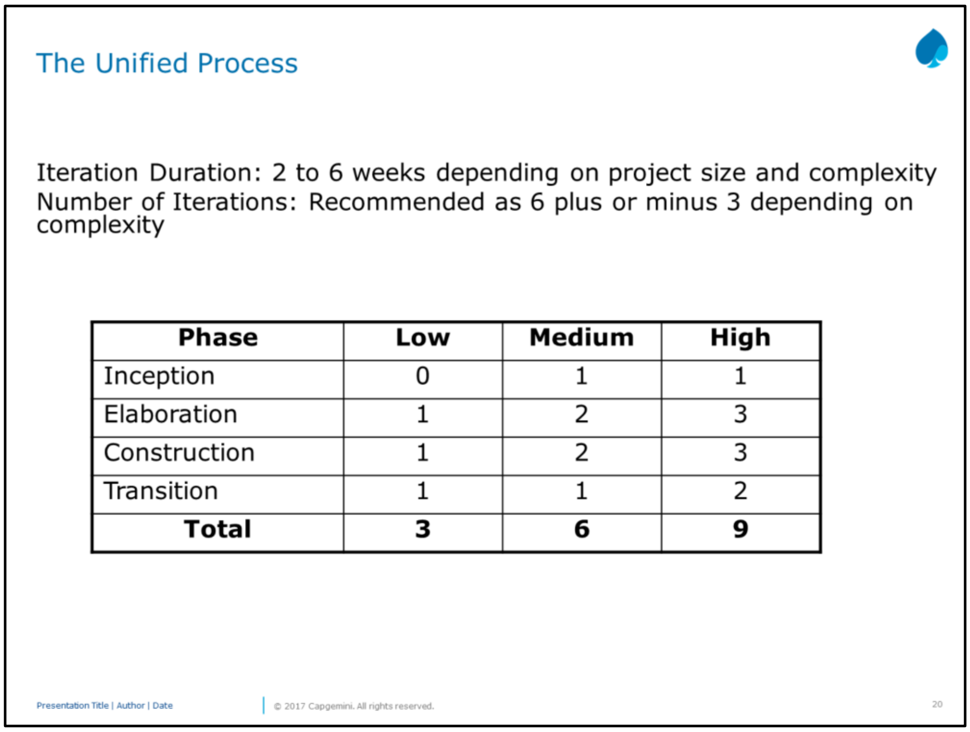
The Unified Process



	Inceptio n	Elaboratio n	Constructio n	Transitio n
Effort	5%	20%	65%	10%
Time/Sched ule	10%	30%	50%	10%

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 19

This slide talks about typical time and efforts spent on each of the phases of Unified Process.



The Unified Process

Iteration Duration: 2 to 6 weeks depending on project size and complexity
Number of Iterations: Recommended as 6 plus or minus 3 depending on complexity

Phase	Low	Medium	High
Inception	0	1	1
Elaboration	1	2	3
Construction	1	2	3
Transition	1	1	2
Total	**3**	**6**	**9**

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 20

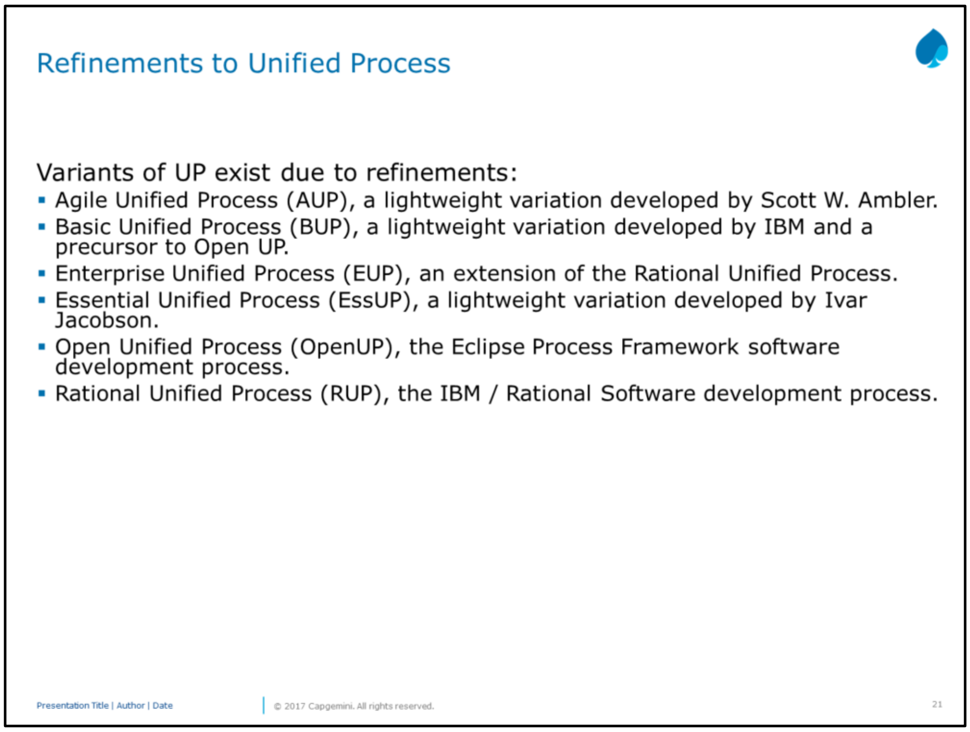
We have discussed about iteration duration earlier. In terms of number of iterations, the recommendation is 6 plus or minus 3, depending on project size and complexity.

Inception may have no real iterations if it consists only of planning and marketing. Iteration will be needed if prototyping comes into play.

Elaboration iterations may depend on the complexity of the architecture, and the expertise of the team.

Multiple iterations of constructions are required for coding and robust testing.

Transition would require at least one iteration; more iterations if project transition is complex.



Refinements to Unified Process

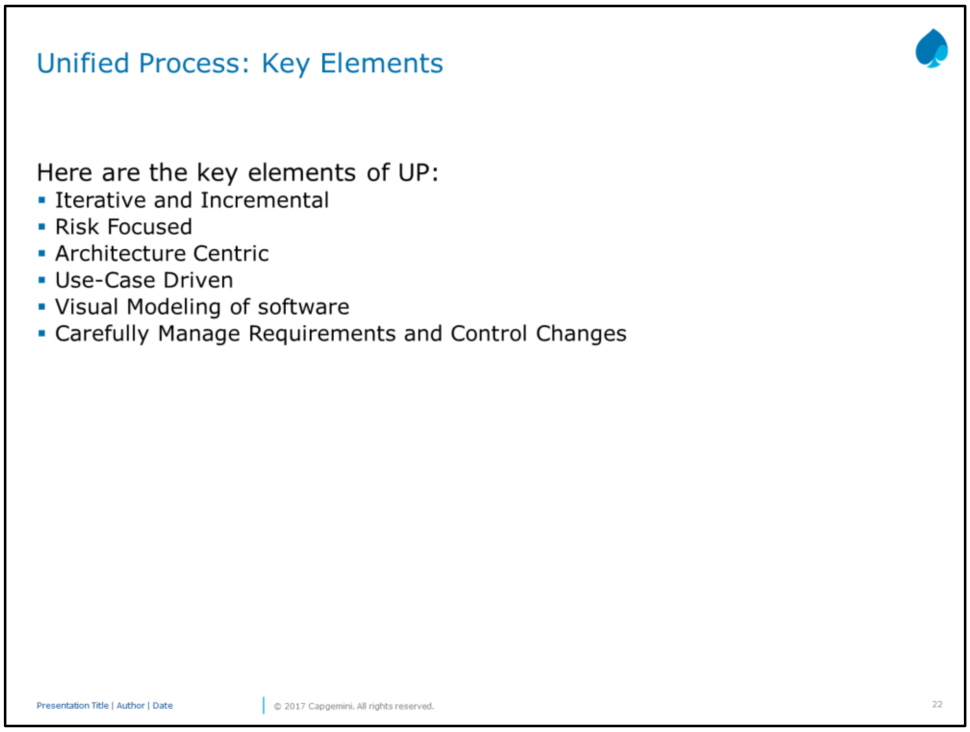
Variants of UP exist due to refinements:

- Agile Unified Process (AUP), a lightweight variation developed by Scott W. Ambler.
- Basic Unified Process (BUP), a lightweight variation developed by IBM and a precursor to Open UP.
- Enterprise Unified Process (EUP), an extension of the Rational Unified Process.
- Essential Unified Process (EssUP), a lightweight variation developed by Ivar Jacobson.
- Open Unified Process (OpenUP), the Eclipse Process Framework software development process.
- Rational Unified Process (RUP), the IBM / Rational Software development process.

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 21

Refinements to Unified Process:

- UP has many variants.
- The refinements are usually in terms of variations in the workflows, artifacts, and so on. Some of the commonly known refinements are RUP from IBM Rational, OpenUP which is the Eclipse Process Framework software development process, AUP developed by Scott Ambler as a lightweight variation, EUP as an extension of RUP, and so on.



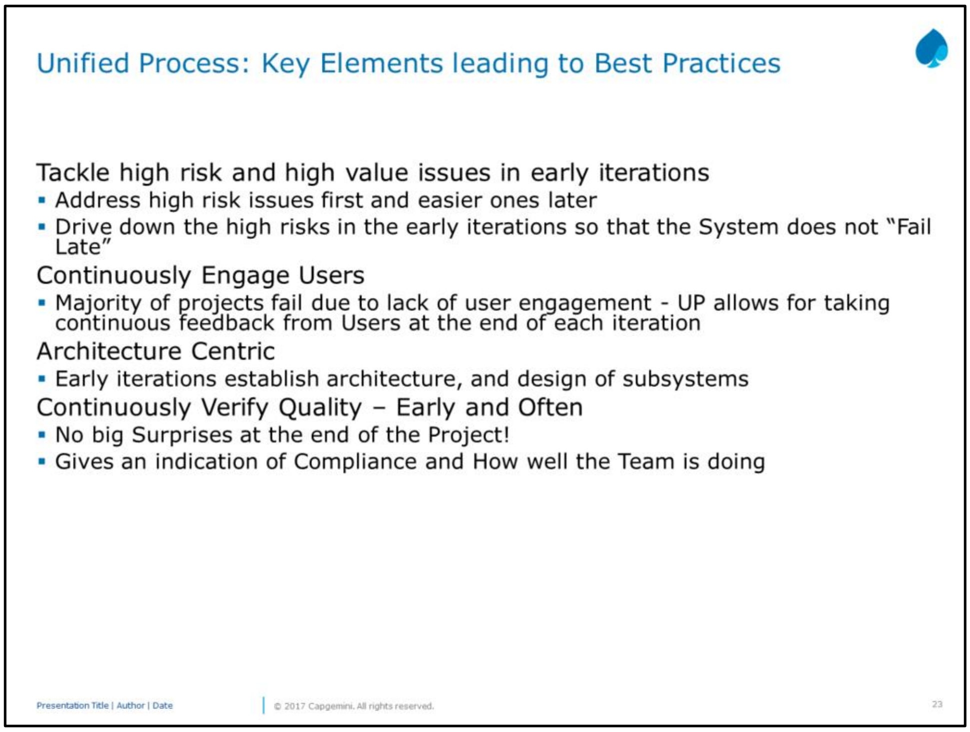
Unified Process: Key Elements

Here are the key elements of UP:
- Iterative and Incremental
- Risk Focused
- Architecture Centric
- Use-Case Driven
- Visual Modeling of software
- Carefully Manage Requirements and Control Changes

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 22

UP Key Features and Concepts:

- **Iterative and Incremental Development:** We have discussed this earlier.
- **Risk-Focused:** UP focuses on addressing the high risk items first so that chances of failure of the system later on is minimized. So initial iterations would work with high risk or high value features.
- **Architecture-Centric:** Architecture is considered as one of the highest risks, so it must be addressed early in the lifecycle.
- **Use-Case-Driven:** Functional requirements are captured as use-cases. In UP, use-cases drive the planning and implementation of system.
- Some other key elements include:
  - **Visual Modeling of Software:** UP recommends visual modeling of software. UML provides standards for visual modeling. It provides standard semantic notations, diagrams, and models for visual representation of the system. These visual representations allow developers to design a better system and allows the development team to communicate decisions unambiguously.
  - **Carefully Manage Requirements and control changes:** Most projects fail due to poor requirements management. UP emphasizes on carefully managing requirements with tools support. Similarly, UP recommends version control of all project artifacts.



Unified Process: Key Elements leading to Best Practices

Tackle high risk and high value issues in early iterations
- Address high risk issues first and easier ones later
- Drive down the high risks in the early iterations so that the System does not “Fail Late”

Continuously Engage Users
- Majority of projects fail due to lack of user engagement - UP allows for taking continuous feedback from Users at the end of each iteration

Architecture Centric
- Early iterations establish architecture, and design of subsystems

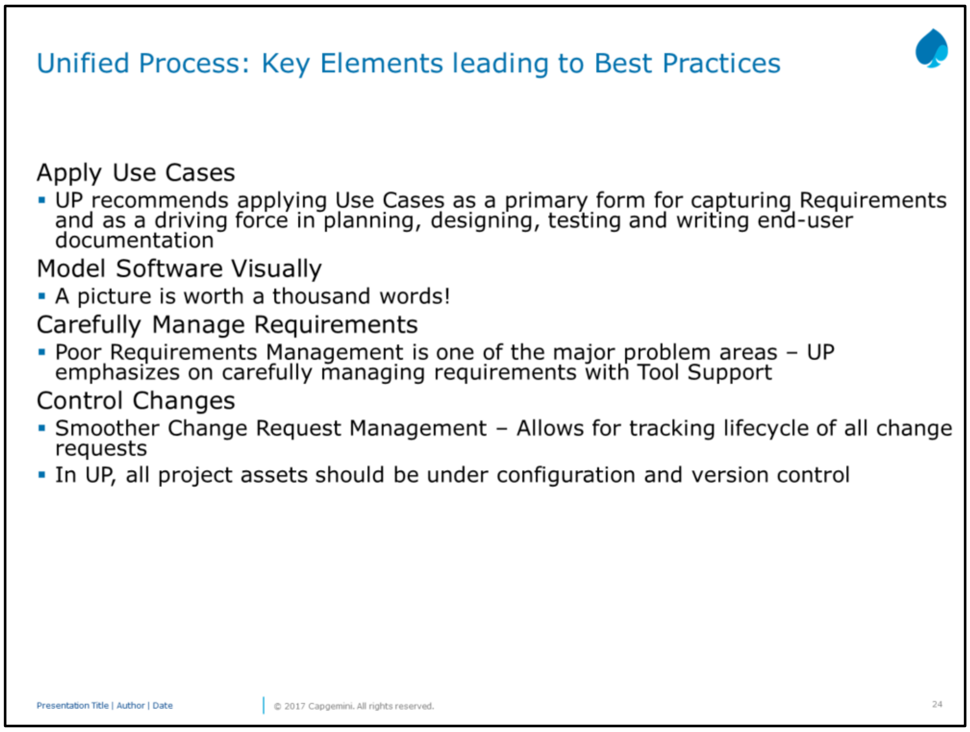
Continuously Verify Quality – Early and Often
- No big Surprises at the end of the Project!
- Gives an indication of Compliance and How well the Team is doing

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

23

The key elements lend themselves to several best practices of software development as outlined in this slide.



Unified Process: Key Elements leading to Best Practices

Apply Use Cases
- UP recommends applying Use Cases as a primary form for capturing Requirements and as a driving force in planning, designing, testing and writing end-user documentation

Model Software Visually
- A picture is worth a thousand words!

Carefully Manage Requirements
- Poor Requirements Management is one of the major problem areas – UP emphasizes on carefully managing requirements with Tool Support

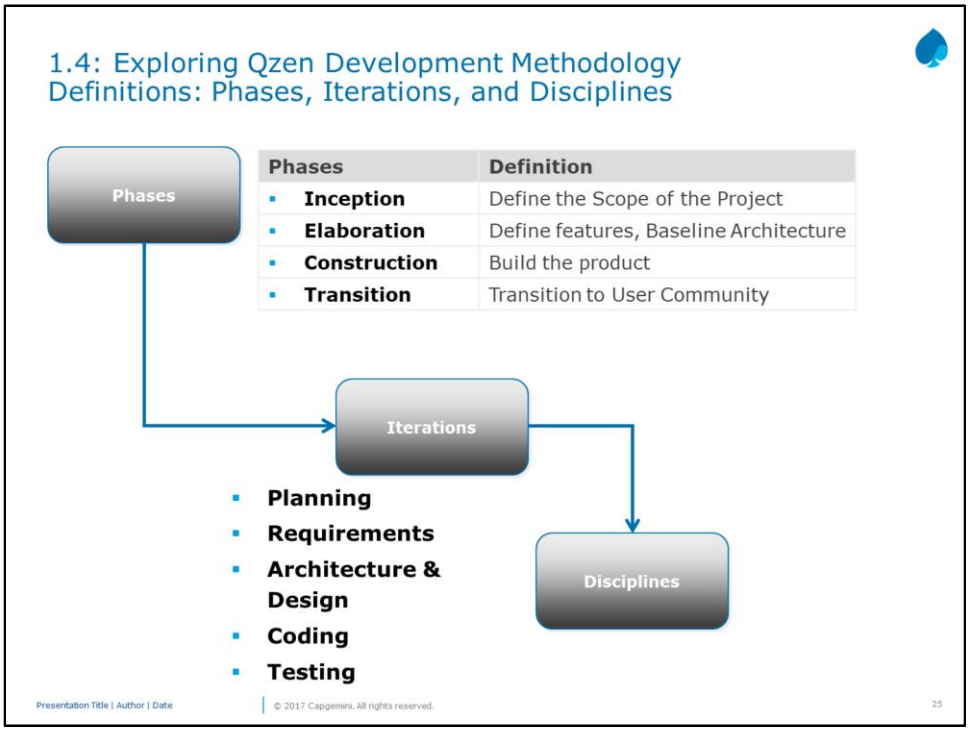
Control Changes
- Smoother Change Request Management – Allows for tracking lifecycle of all change requests
- In UP, all project assets should be under configuration and version control

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

24

The key elements lend themselves to several best practices of software development as outlined in this slide.



1.4: Exploring Qzen Development Methodology Definitions: Phases, Iterations, and Disciplines

Phases	Definition
▪ **Inception**	Define the Scope of the Project
▪ **Elaboration**	Define features, Baseline Architecture
▪ **Construction**	Build the product
▪ **Transition**	Transition to User Community

Phases
Iterations
Disciplines

- **Planning**
- **Requirements**
- **Architecture & Design**
- **Coding**
- **Testing**

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

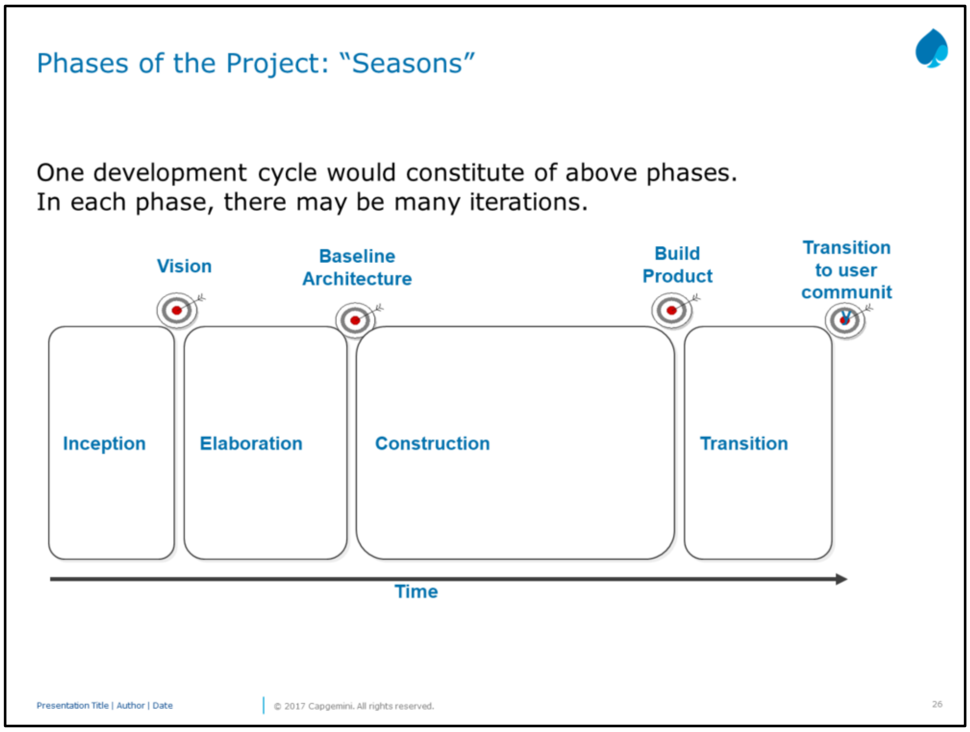
25

<u>Qzen – Phases and Disciplines</u>:

- The UP structure is organized into two dimensions, with **phases** and **disciplines**. These are explained in the context of QZen on this slide.

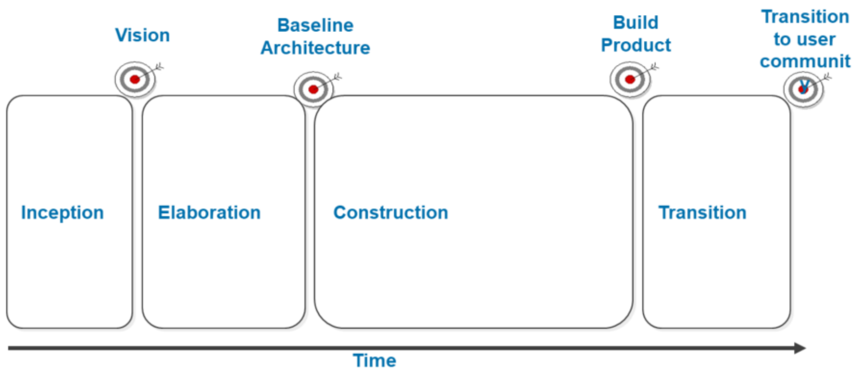
- There are four phases as listed in the above slide. The phase boundaries correspond to achievement of milestones in the lifecycle of the project. These are also crucial decision points in the life of the project. For example, at the end of **inception** stage, a decision needs to be taken on whether the project is viable and should funds be committed to the **elaboration** stage. Similar decisions need to be taken at the end of each stage.

- On a time axis, while executing the project, one progresses through the four phases. Each phase gets executed as one or more iterations. As seen earlier, each iteration is a mini project in itself going through the entire cycle from **planning** to **deployment**. Each stage is a UP discipline. A discipline is a workflow describing activities to be done, roles performing the activities, and artifacts that get created. The Qzen disciplines are listed in the above slide.



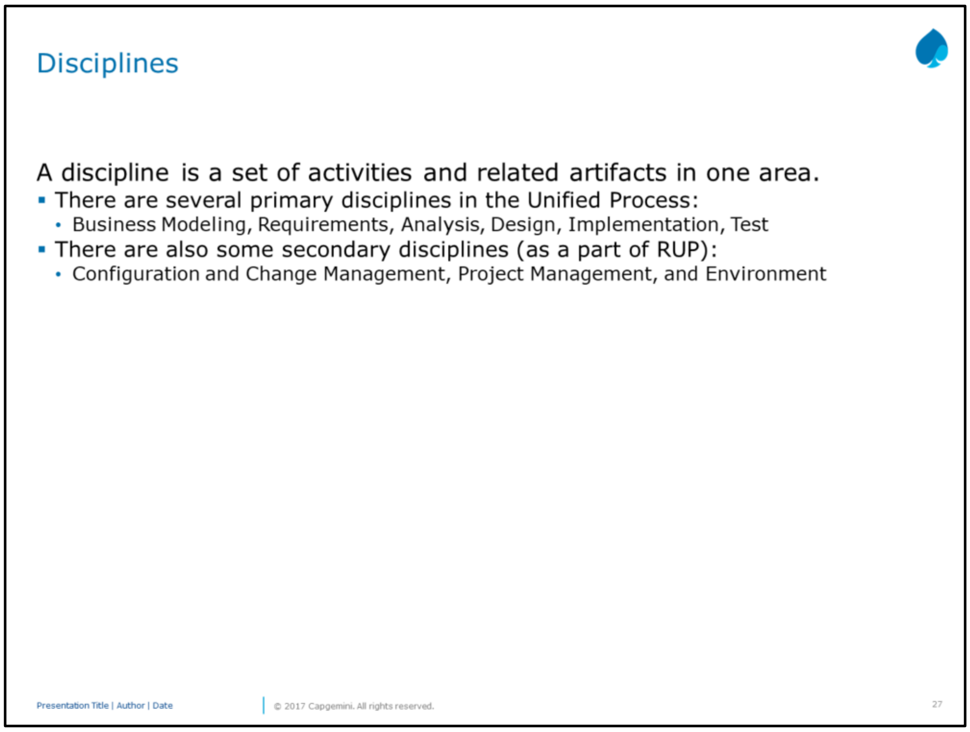
Phases of the Project: "Seasons"

One development cycle would constitute of above phases.
In each phase, there may be many iterations.



© 2017 Capgemini. All rights reserved.

Phases of the Project: "Seasons":
A project development cycle will include all the four phases. There would be one or more iterations in each phase. There are key milestones defined for each phase.



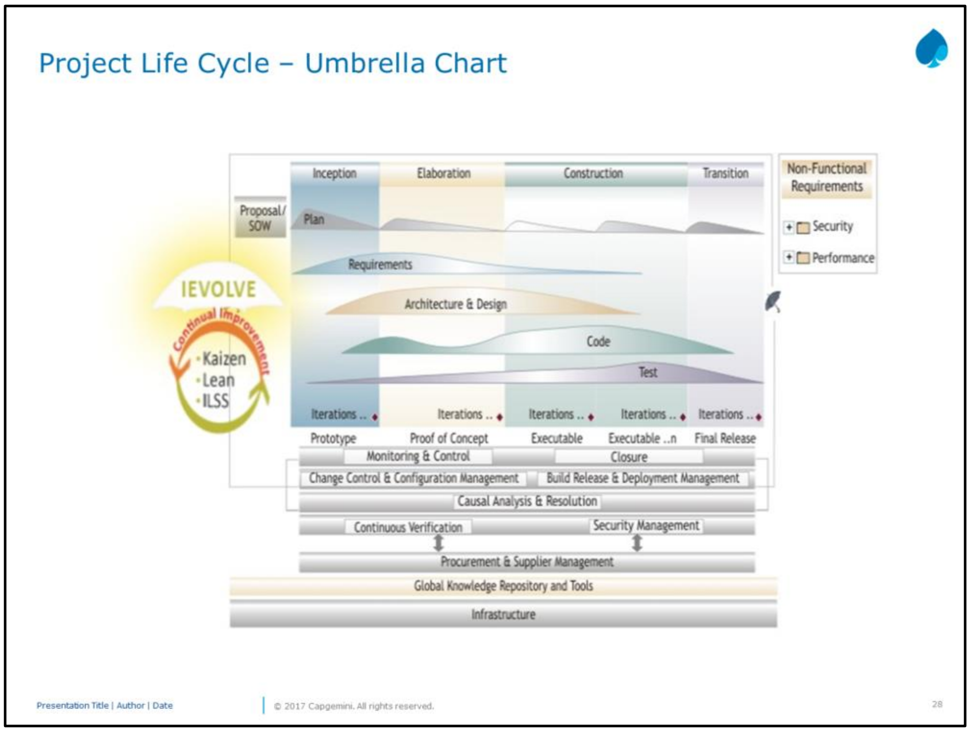
Disciplines

A discipline is a set of activities and related artifacts in one area.

- There are several primary disciplines in the Unified Process:
  - Business Modeling, Requirements, Analysis, Design, Implementation, Test
- There are also some secondary disciplines (as a part of RUP):
  - Configuration and Change Management, Project Management, and Environment

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 27

<u>Disciplines</u>:
A discipline will give the workflow of activities, along with guidance on how to perform the activities. Roles and responsibilities would be defined for each of the disciplines.

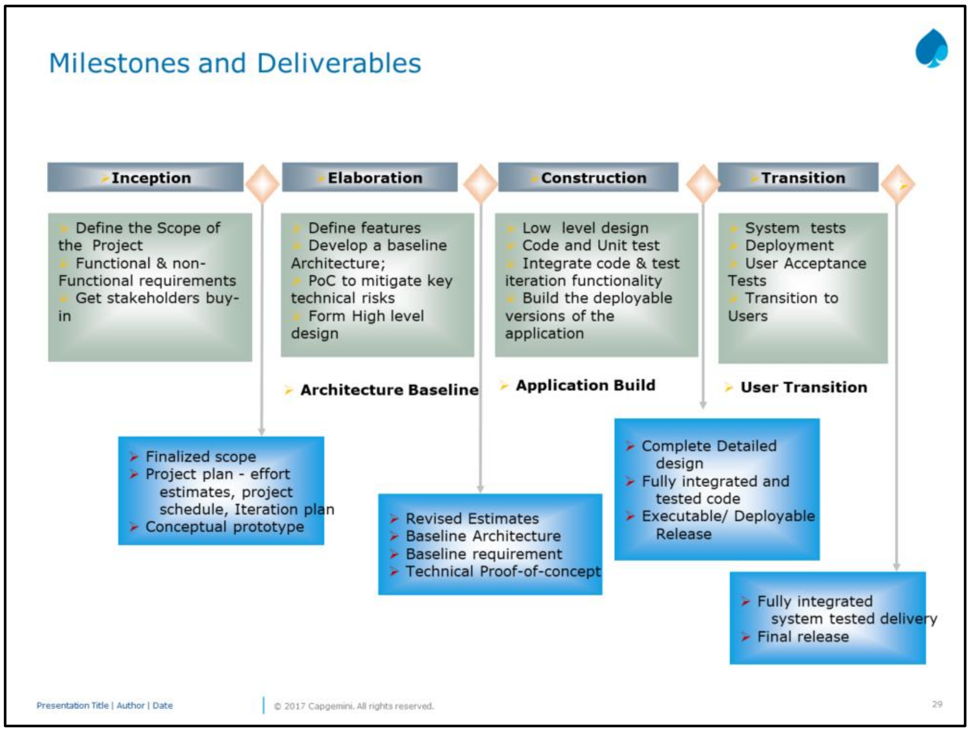


Project Life Cycle – Umbrella Chart

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 28

<u>Qzen Project Life Cycle: Umbrella Chart</u>:

- The above slide shows the Qzen Project Lifecycle for development projects. The horizontal time axis indicates the four phases as the project unfolds and progresses. The vertical axis is for disciplines, which are logical grouping of activities.
- The primary disciplines of Qzen are Plan, Requirements, Architecture and Design, Code, and Test. Key inputs, outputs, and roles for these disciplines are available in Qzen. There are several supporting disciplines, such as Change Control and Configuration Management, Build Release and Deployment Management etc.
- The humps indicate how the emphasis for different disciplines vary across the four phases.
- **Example:** You would have **Design** being done across the various phases. However, the point that varies is the amount of focus on Design. That is, there would be more of Design in the **Elaboration** phase and less of it in **Construction**, but Design as an activity would still be done. This is in contrast with the Waterfall approach, where once Design is completed, it is not revisited as the project progresses into coding and testing.
- Note that detailed guidance is also provided for two key quality attributes:
  - ➢ Performance
  - ➢ Security
- IEVOLVE provides Lean and Value innovation techniques that can be applied across phases and disciplines for continuous improvements.

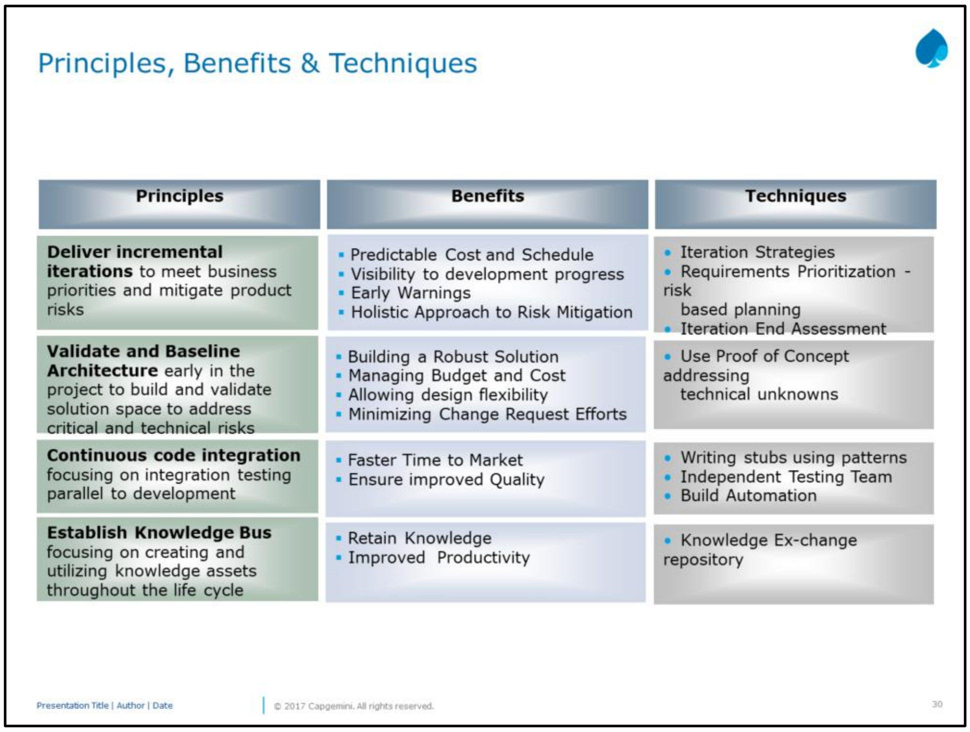


Milestones and Deliverables

Inception
Define the Scope of the Project
Functional & non-Functional requirements
Get stakeholders buy-in
Elaboration
Define features
Develop a baseline Architecture;
PoC to mitigate key technical risks
Form High level design
Construction
Low level design
Code and Unit test
Integrate code & test iteration functionality
Build the deployable versions of the application
Transition
System tests
Deployment
User Acceptance Tests
Transition to Users
Architecture Baseline
Application Build
User Transition
Finalized scope
Project plan - effort estimates, project schedule, Iteration plan
Conceptual prototype
Revised Estimates
Baseline Architecture
Baseline requirement
Technical Proof-of-concept
Complete Detailed design
Fully integrated and tested code
Executable/ Deployable Release
Fully integrated system tested delivery
Final release
Presentation Title | Author | Date
© 2017 Capgemini. All rights reserved.
29

<u>Milestones and Deliverables</u>:

- Phases map to types of risks for an iteration.
  - **Inception:** Risks associated with Scope
  - **Elaboration:** Risks associated with Architecture
  - **Construction:** Risks associated with Production
  - **Transition:** Risks associated with the Roll out
- You can use this knowledge to identify the right milestones. The scope of these phases along with the "deliverables" at the end of each phase are listed on the above slide.
- Artifacts are organized into sets, one set per discipline.
- In an iterative development process, the various artifacts are not produced in one phase, completed or even frozen before moving on to the next phase. Instead, the artifact sets evolve throughout the development cycle.

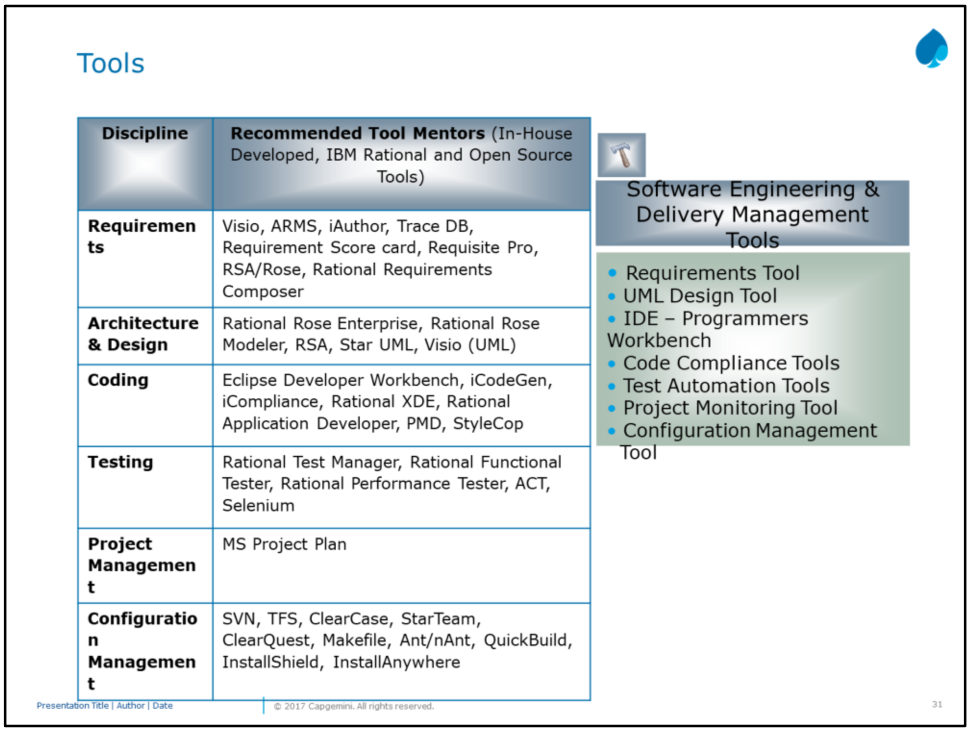


Principles, Benefits & Techniques

Principles	Benefits	Techniques
Deliver incremental iterations to meet business priorities and mitigate product risks	▪ Predictable Cost and Schedule ▪ Visibility to development progress ▪ Early Warnings ▪ Holistic Approach to Risk Mitigation	• Iteration Strategies • Requirements Prioritization - risk based planning • Iteration End Assessment
Validate and Baseline Architecture early in the project to build and validate solution space to address critical and technical risks	▪ Building a Robust Solution ▪ Managing Budget and Cost ▪ Allowing design flexibility ▪ Minimizing Change Request Efforts	• Use Proof of Concept addressing technical unknowns
Continuous code integration focusing on integration testing parallel to development	▪ Faster Time to Market ▪ Ensure improved Quality	• Writing stubs using patterns • Independent Testing Team • Build Automation
Establish Knowledge Bus focusing on creating and utilizing knowledge assets throughout the life cycle	▪ Retain Knowledge ▪ Improved Productivity	• Knowledge Ex-change repository

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 30

Summarizing the Principles, Benefits and Techniques



Tools

Discipline	Recommended Tool Mentors (In-House Developed, IBM Rational and Open Source Tools)
Requirements	Visio, ARMS, iAuthor, Trace DB, Requirement Score card, Requisite Pro, RSA/Rose, Rational Requirements Composer
Architecture & Design	Rational Rose Enterprise, Rational Rose Modeler, RSA, Star UML, Visio (UML)
Coding	Eclipse Developer Workbench, iCodeGen, iCompliance, Rational XDE, Rational Application Developer, PMD, StyleCop
Testing	Rational Test Manager, Rational Functional Tester, Rational Performance Tester, ACT, Selenium
Project Management	MS Project Plan
Configuration Management	SVN, TFS, ClearCase, StarTeam, ClearQuest, Makefile, Ant/nAnt, QuickBuild, InstallShield, InstallAnywhere

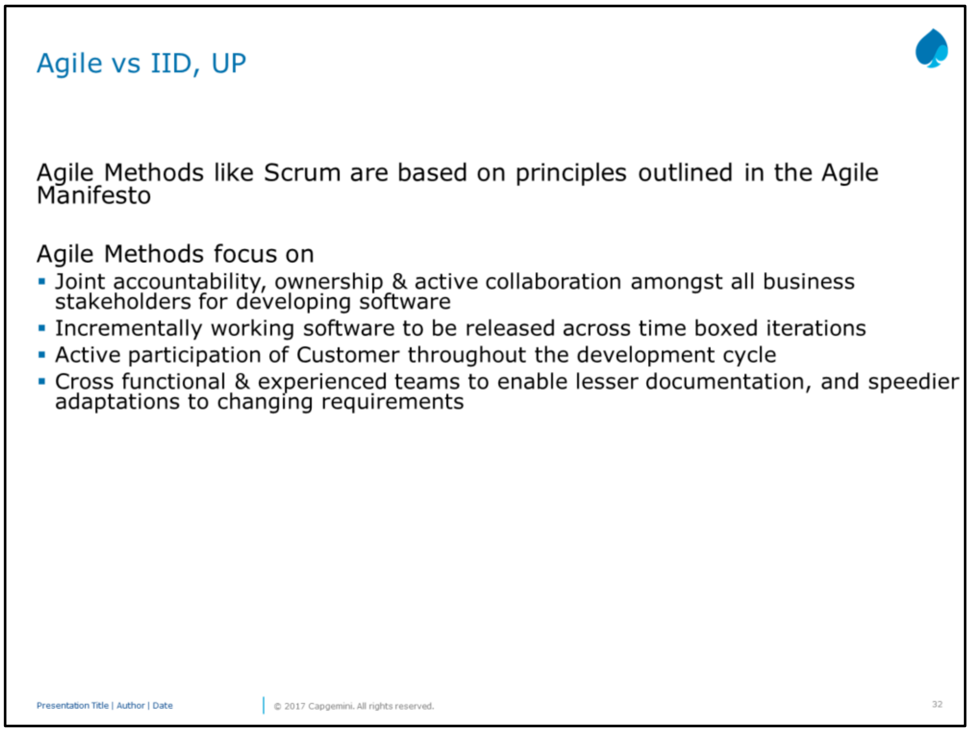
Software Engineering & Delivery Management Tools

- Requirements Tool
- UML Design Tool
- IDE – Programmers Workbench
- Code Compliance Tools
- Test Automation Tools
- Project Monitoring Tool
- Configuration Management Tool

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 31

Listing of various tools recommended by Qzen.

Guidance on using some of these tools are available in Qzen.



Agile vs IID, UP

Agile Methods like Scrum are based on principles outlined in the Agile Manifesto

Agile Methods focus on

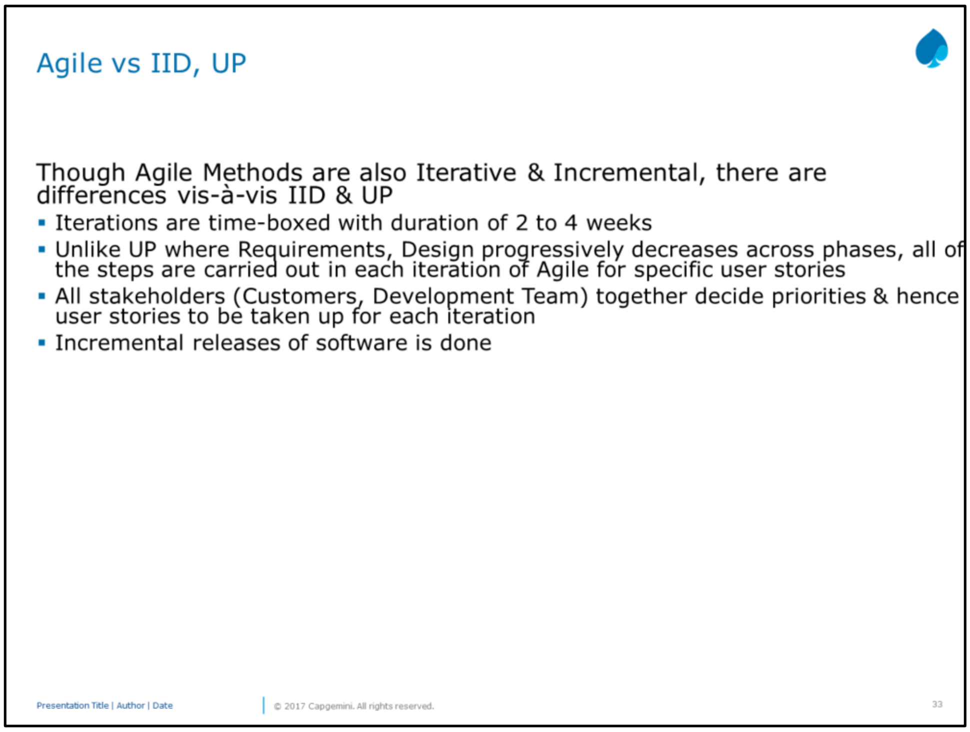
- Joint accountability, ownership & active collaboration amongst all business stakeholders for developing software
- Incrementally working software to be released across time boxed iterations
- Active participation of Customer throughout the development cycle
- Cross functional & experienced teams to enable lesser documentation, and speedier adaptations to changing requirements

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

32

We are briefly touching upon Agile here to compare it with IID & UP. For more guidance on Agile Project execution, pl. refer to Qzen.



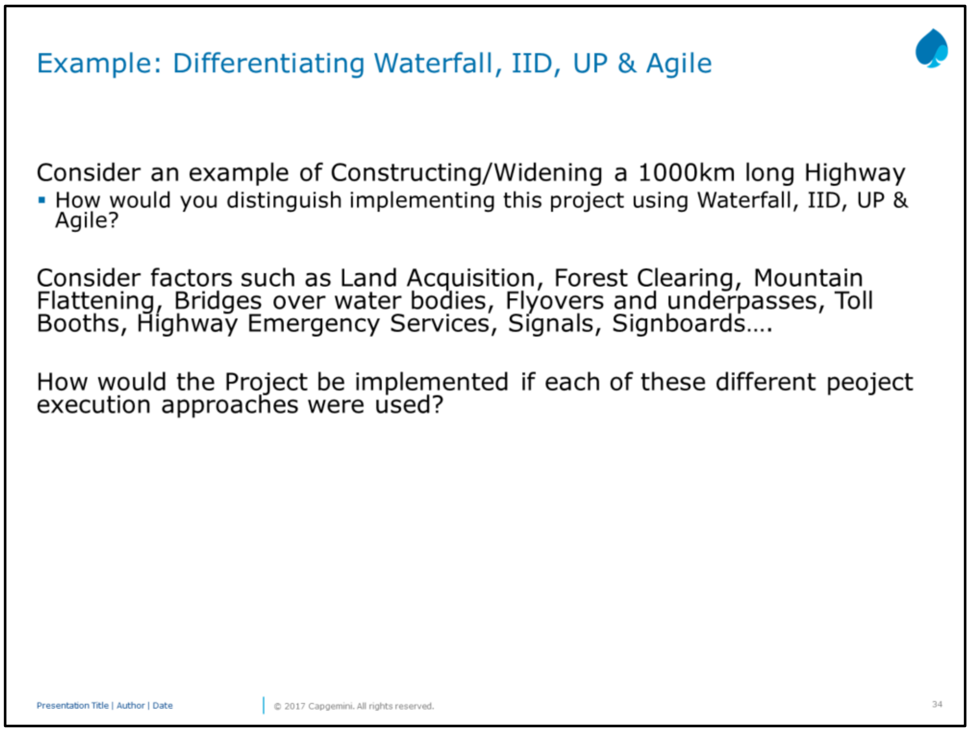
Agile vs IID, UP

Though Agile Methods are also Iterative & Incremental, there are differences vis-à-vis IID & UP

- Iterations are time-boxed with duration of 2 to 4 weeks
- Unlike UP where Requirements, Design progressively decreases across phases, all of the steps are carried out in each iteration of Agile for specific user stories
- All stakeholders (Customers, Development Team) together decide priorities & hence user stories to be taken up for each iteration
- Incremental releases of software is done

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 33

We are briefly touching upon Agile here to compare it with IID & UP. For more guidance on Agile Project execution, pl. refer to Qzen.



Example: Differentiating Waterfall, IID, UP & Agile

Consider an example of Constructing/Widening a 1000km long Highway

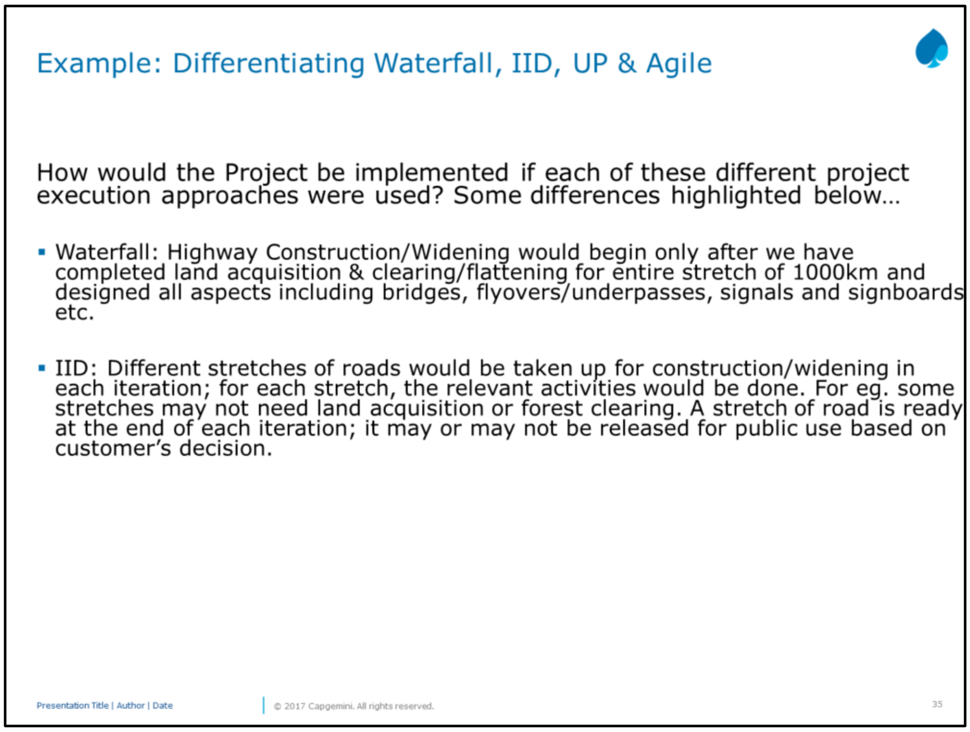
- How would you distinguish implementing this project using Waterfall, IID, UP & Agile?

Consider factors such as Land Acquisition, Forest Clearing, Mountain Flattening, Bridges over water bodies, Flyovers and underpasses, Toll Booths, Highway Emergency Services, Signals, Signboards….

How would the Project be implemented if each of these different peoject execution approaches were used?

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 34

Let us see this non software example to better understand the distinction of using these different methodologies.



Example: Differentiating Waterfall, IID, UP & Agile

How would the Project be implemented if each of these different project execution approaches were used? Some differences highlighted below…

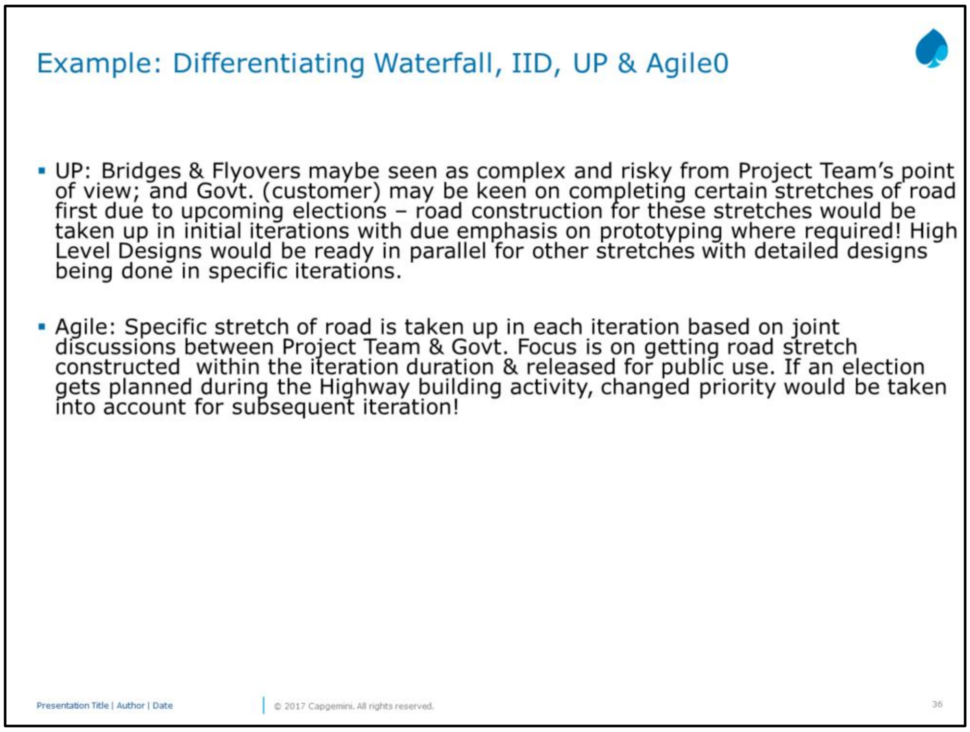
- Waterfall: Highway Construction/Widening would begin only after we have completed land acquisition & clearing/flattening for entire stretch of 1000km and designed all aspects including bridges, flyovers/underpasses, signals and signboards etc.
- IID: Different stretches of roads would be taken up for construction/widening in each iteration; for each stretch, the relevant activities would be done. For eg. some stretches may not need land acquisition or forest clearing. A stretch of road is ready at the end of each iteration; it may or may not be released for public use based on customer's decision.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

35

You could come up with more such differences based on the key principles and characteristics of each of these models.



Example: Differentiating Waterfall, IID, UP & Agile0

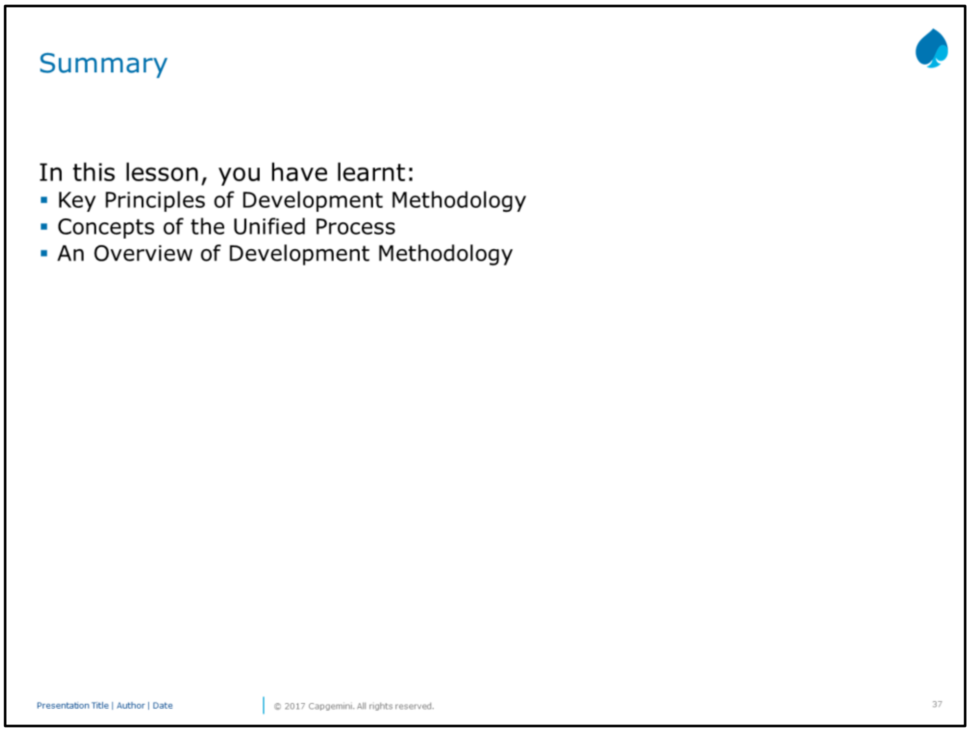
- UP: Bridges & Flyovers maybe seen as complex and risky from Project Team's point of view; and Govt. (customer) may be keen on completing certain stretches of road first due to upcoming elections – road construction for these stretches would be taken up in initial iterations with due emphasis on prototyping where required! High Level Designs would be ready in parallel for other stretches with detailed designs being done in specific iterations.
- Agile: Specific stretch of road is taken up in each iteration based on joint discussions between Project Team & Govt. Focus is on getting road stretch constructed within the iteration duration & released for public use. If an election gets planned during the Highway building activity, changed priority would be taken into account for subsequent iteration!

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

36

You could come up with more such differences based on the key principles and characteristics of each of these models.



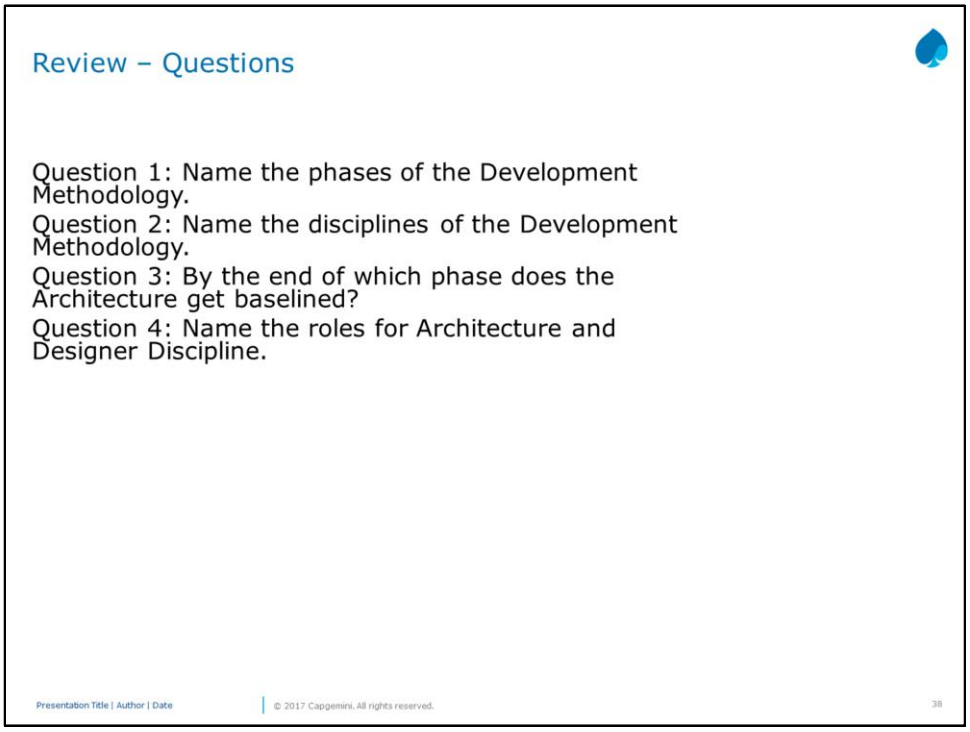
Summary

In this lesson, you have learnt:

- Key Principles of Development Methodology
- Concepts of the Unified Process
- An Overview of Development Methodology

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 37

Summary:
This concludes a brief overview of the Development Methodology.



Review – Questions

Question 1: Name the phases of the Development Methodology.

Question 2: Name the disciplines of the Development Methodology.

Question 3: By the end of which phase does the Architecture get baselined?

Question 4: Name the roles for Architecture and Designer Discipline.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

38