COMP 354
Introduction to Software Engineering

Greg Butler

Computer Science and Software Engineering
Concordia University, Montreal, Canada

Office: EV 3.219 Email: gregb@cs.concordia.ca

Winter 2015

Course Web Site:
http://users.encs.concordia.ca/~gregb/home/comp354-w2015.html

Software Process — Recap

The objective of software development is to efficiently and predictably deliver a software product that meets the needs of the community of its stakeholders.

A process is a set of ordered steps intended to reach an objective.

A software development process (or model) is an approach to **building**, **deploying** and **maintaining** software. The motivation behind defining a process for software development is to manage the size and complexity of software systems.

The Unified Process

Created by Rational (IBM) in 1997

Some of the roots in "spiral model" of Barry Boehm
Core initial development around 1995-98
Phillippe Kruchten chief architect of UP/RUP

Iterative, not agile

Risk-driven development

early iterations focusing on creation of the core architecture and driving down the high risks

2-6 week iterations

The Unified Process — Key practices and guidelines

Short time-boxed iterations

Develop high-risk elements in early iterations

Deliver value to customer

Accommodate change early in project

Work as one team

The Unified Process

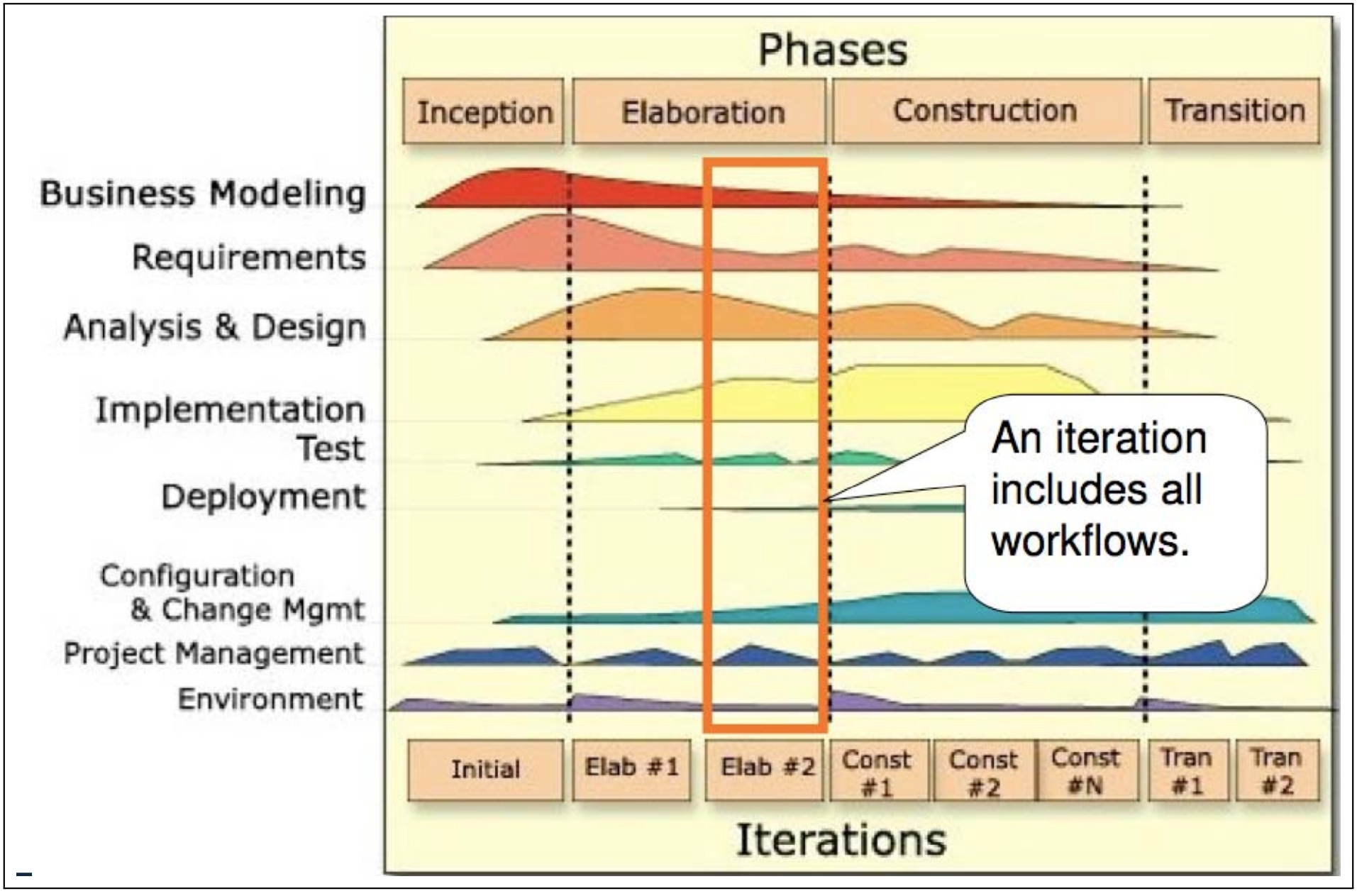
Technically not a process

... but a framework from which an iterative and incremental development process can be derived

Defines 4 sequential phases

Inception	Defines the scope of the project
Elaboration	Implements core architecture and critical (high risk, high value) use cases. So as to "drive-down" the risk of the entire project.
Construction	Implements secondary (non-critical) use cases. Alpha-testing
Transition	Deploys the system into "production". Beta-testing

The Unified Process



The Unified Process — Key ideas and practices

An iterative process framework

Project lifecycle phases

Identifies workers, activities, artifacts

Promotes certain practices

- Develop software iteratively
- Manage requirements
- Use component-based architectures Visually model software
- Verify software quality
- Control changes to software

The Unified Process — Characteristics

- Iterative process framework, typically customized to be a process description for the organization
- All work products ("artifacts") are optional and their order arbitrary.
- Products serve as common vocabulary for the team.
- RUP is a process framework and licensed product (tool plus web pages)
- Artifacts are information abstractions, e.g. Vision or Risk List, organized in disciplines, e.g. Requirements Discipline

The Unified Process — Disciplines within iterations

Example disciplines

Requirements, Design, Project Management, Implementation

Development Case of UP

UP tailored for each project, choose sets of practices and work products to create ("less is better")

Disciplines addressed in each iteration

but to varying degree

The Unified Process — Life cycle in four phases

Inception

Business case, vision, identify high risks & 10% of key reqs in detail, estimate elaboration effort

Elaboration

Core & architecturally significant parts coded/tested, key risks identified/mitigated, 80% of major reqs evolved/defined

Construction

Builds remaining system in short iterations, efficient and predictable due to solid elaboration

Transition

Exposes release candidate for review/feedback, then deployment

The Unified Process — Some prominent work products

Vision

summary of objectives, features, business case

Software Architecture Document

Short learning aid to understand the system

Test Plan

summary of goals and methods of testing

Iteration Plan

detailed plan for the next iteration

Change Request

uniform way to track all requests for work, e.g. defects

The Unified Process — Some guidelines

- Attack risks early and continuously before they will attack you
- Stay focused on developing executable software in early iterations
- Prefer component-oriented architectures and reuse of existing components
- Baseline an executable architecture early

The Unified Process — Six Best Practices

Time-boxed iterations

Avoid attempting large, up-front requirements

Strive for cohesive architecture and reuse existing components

On large projects: reqs & core architecture developed by small co-located team; then early team members divide into sub-project leaders

Continuously verify quality

Test early, often, and realistically by integrating all software each iteration

The Unified Process — Six Best Practices

Visual modeling

Prior to programming, do at least some visual modeling to explore creative design ideas

Manage requirements

Find, organize, and track requirements iteratively through skillful means. Use tools.

Manage change

Disciplined configuration management and version control, change request protocol, base-lined releases at the end of each iteration

The Unified Process — Is it Agile?

UP can be used in a very traditional waterfall style or in an agile manner

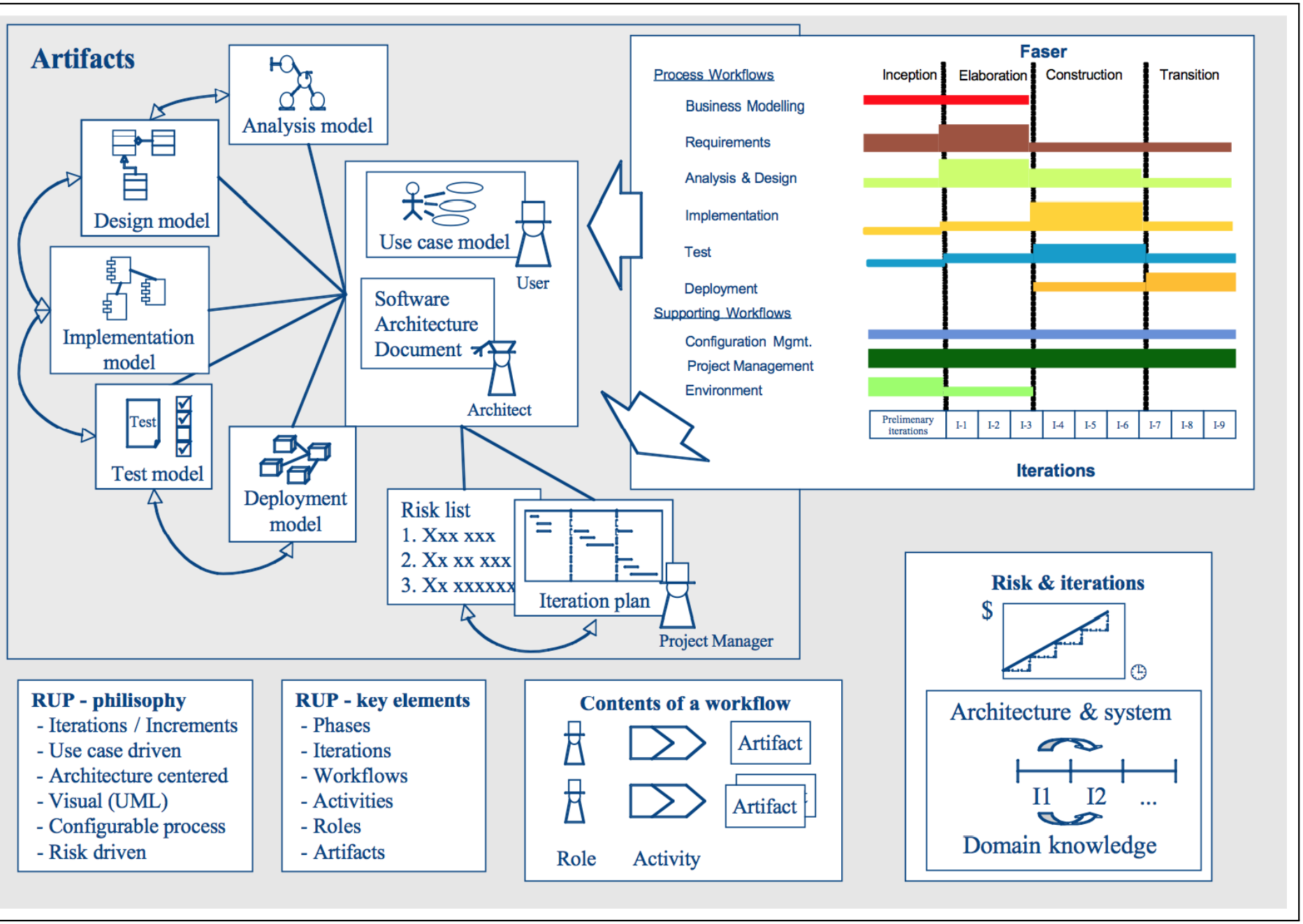
Martin Fowler

"You can use RUP as a agile process, or as a heavyweight process — it all depends on how you tailor it in your environment."

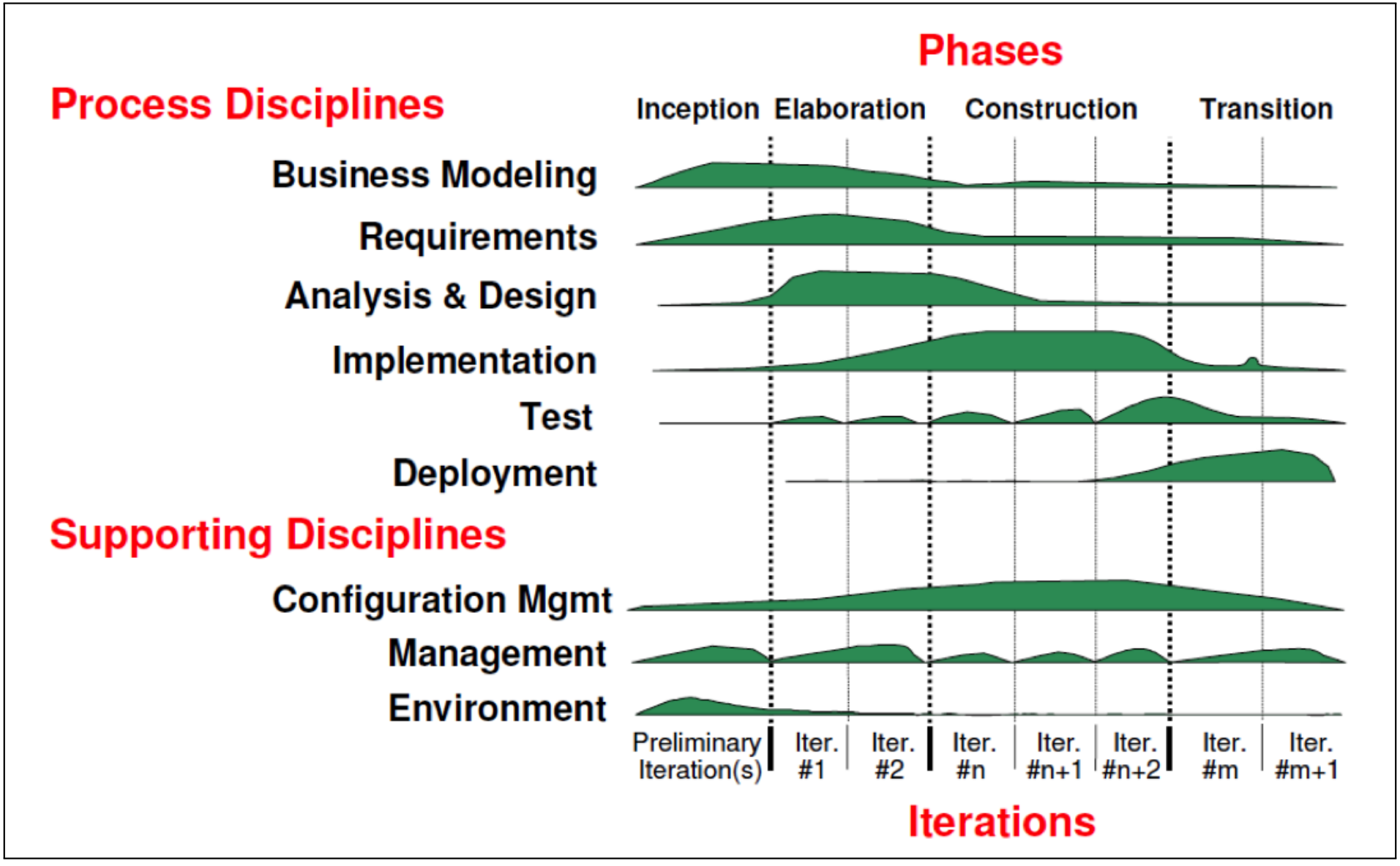
Craig Larman

is a strong proponent of using the RUP in an agile manner

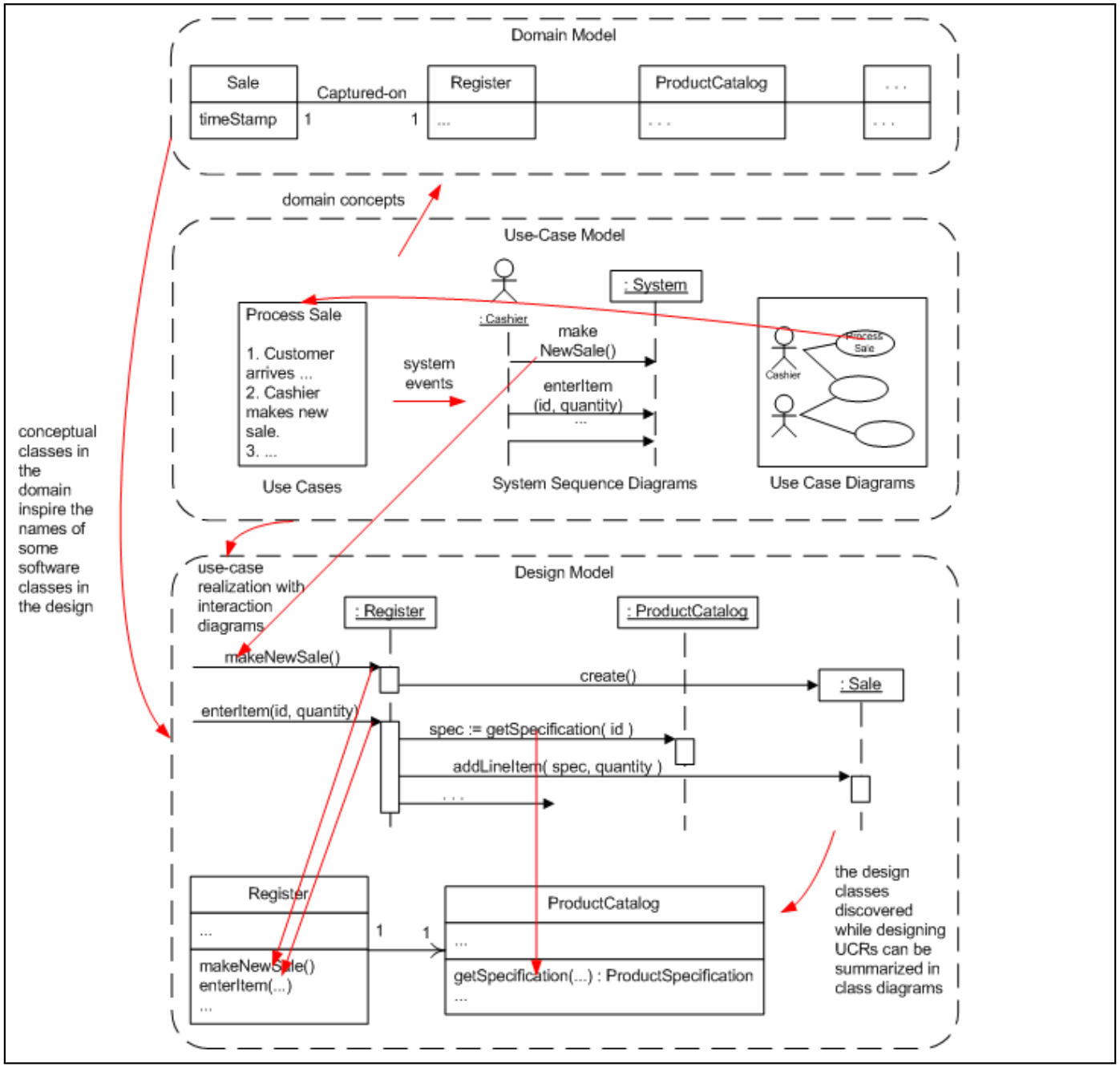
The Unified Process — Overview



The Unified Process



The Unified Process —- Larman Perspective



The Unified Process — Key Principles

Risk driven

Major risks "mitigated" during inception.
All risks mitigated by end of elaboration.

Use case driven

Use cases capture the functional requirements
use cases define the contents of the iterations
Each iteration realizes one or many use cases

Architecture-centric

Elaboration constructs a working architecture.

The Unified Process — Key Points

Framework offering guidance

with many optional artifacts

Roles

Iterations

with all disciplines used differently in different stage of the project

Architecture Centric

Use case driven

How does UP address the major problems in SE?

Requirements

Architecture

Change

Complexity