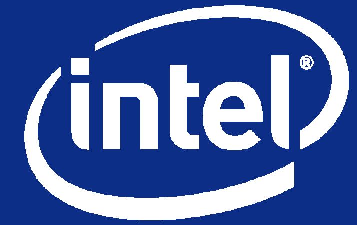
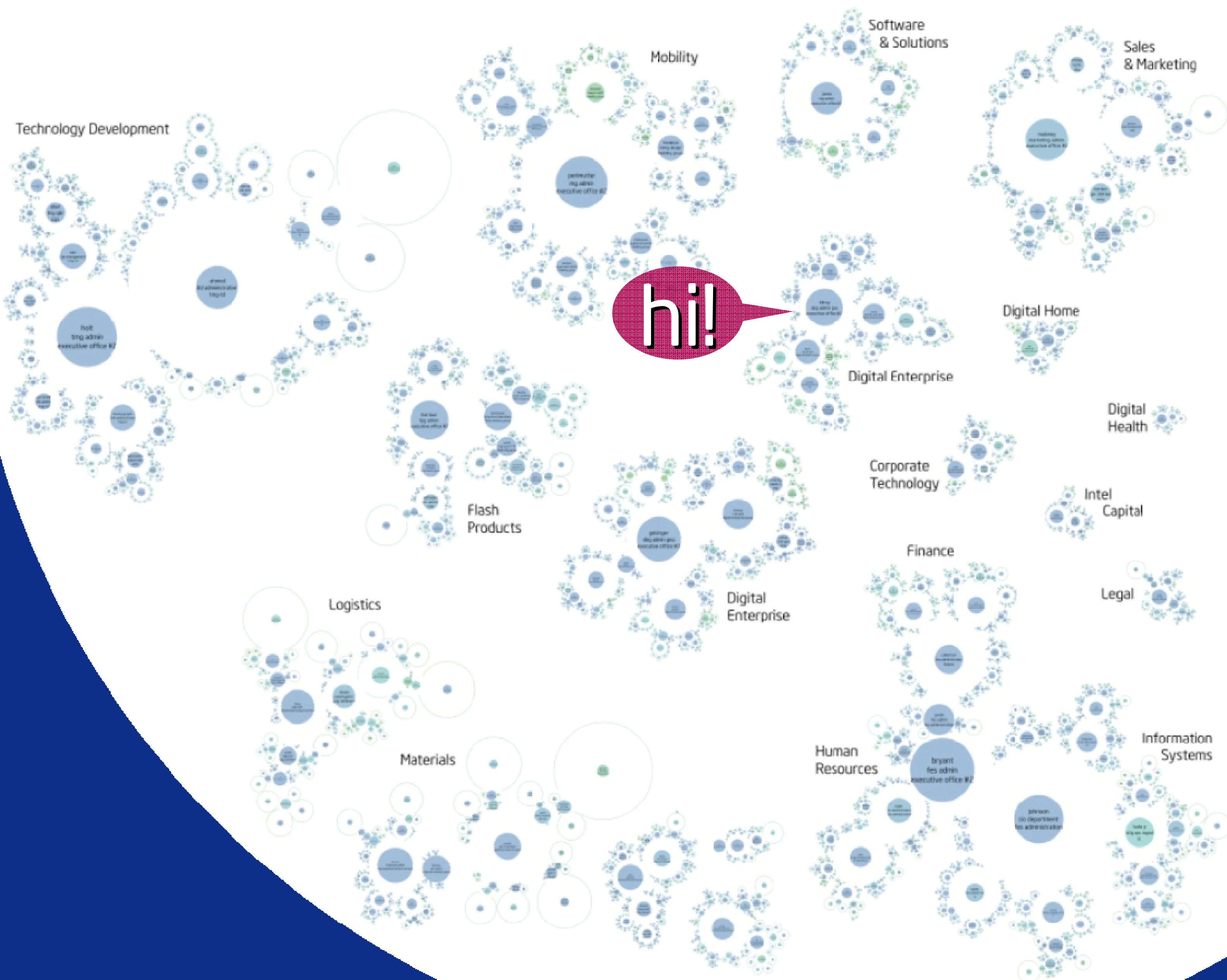


Are Life Cycles Still Relevant?

Erik Simmons, PNSQC 2009

With thanks to Brian Bramlett and Sarah Gregory



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Prologue: Moving Quality Forward

What's in a word?

Moving: Latin – "to change, exchange, go in/out, quit"; change position, reallocate, shift to a new place

Quality: Latin – "of what kind"; fitness for use, comparative worth, degree of excellence

Forward: Old English/Latin – "turn to the front (fore+ward)"; ahead, at the front, expected in the future, without customary restraint

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.
intel

Moving Quality Forward

A tale of a fish, a hedgehog, and an old dog…

…Who is next?

But now, the big questions:

Are life cycles still relevant today?

Can a life cycle help Move Quality Forward?

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Well... What exactly is a life cycle?

a set of practices?

a process?

a style?

a process 'skeleton'?

an approach?

a philosophy?

a method? ...or methodology?

a model? ...of what?

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Which of these might be life cycles?

The waterfall

Stage-Gate®

Scrum

Agile methods

Interative and Incremental Development

CMMI^sm^

Test-first design

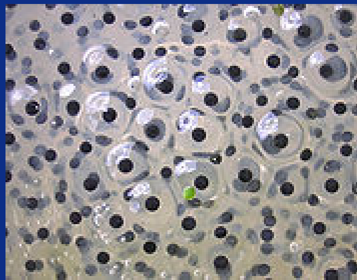
Plan-Do-Study-Act

Rapid prototyping

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Or is *this* a life cycle…



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Software life cycle definitions

"...A set of interrelated activities that result in the development or assessment of software products. Each activity consists of tasks. The life cycle processes may overlap one another."

Source: IEEE Stds Glossary

"The period of time that begins when a software product is conceived and ends when the software is no longer available for use. "

Source: NASA GSFC, Software Assurance Glossary

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.
intel

More definitions of software-related life cycles

The product life cycle "goes through many phases, involves many professional disciplines, and requires many skills, tools and processes…"

A Software release life cycle "is composed of different stages that describe the stability of a piece of software…"

Source: Wikipedia

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

More definitions of software-related life cycles

But, the system development life cycle is defined as "…the process of creating or altering systems…"

Finally, "the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system: the software development process."

Source: Wikipedia

So, a life cycle seems to relate methodologies to processes within phases or stages

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

So, what is a software development life cycle?

A software development life cycle is a framework that organizes methodologies, activities, practices, and processes in a specific and useful way

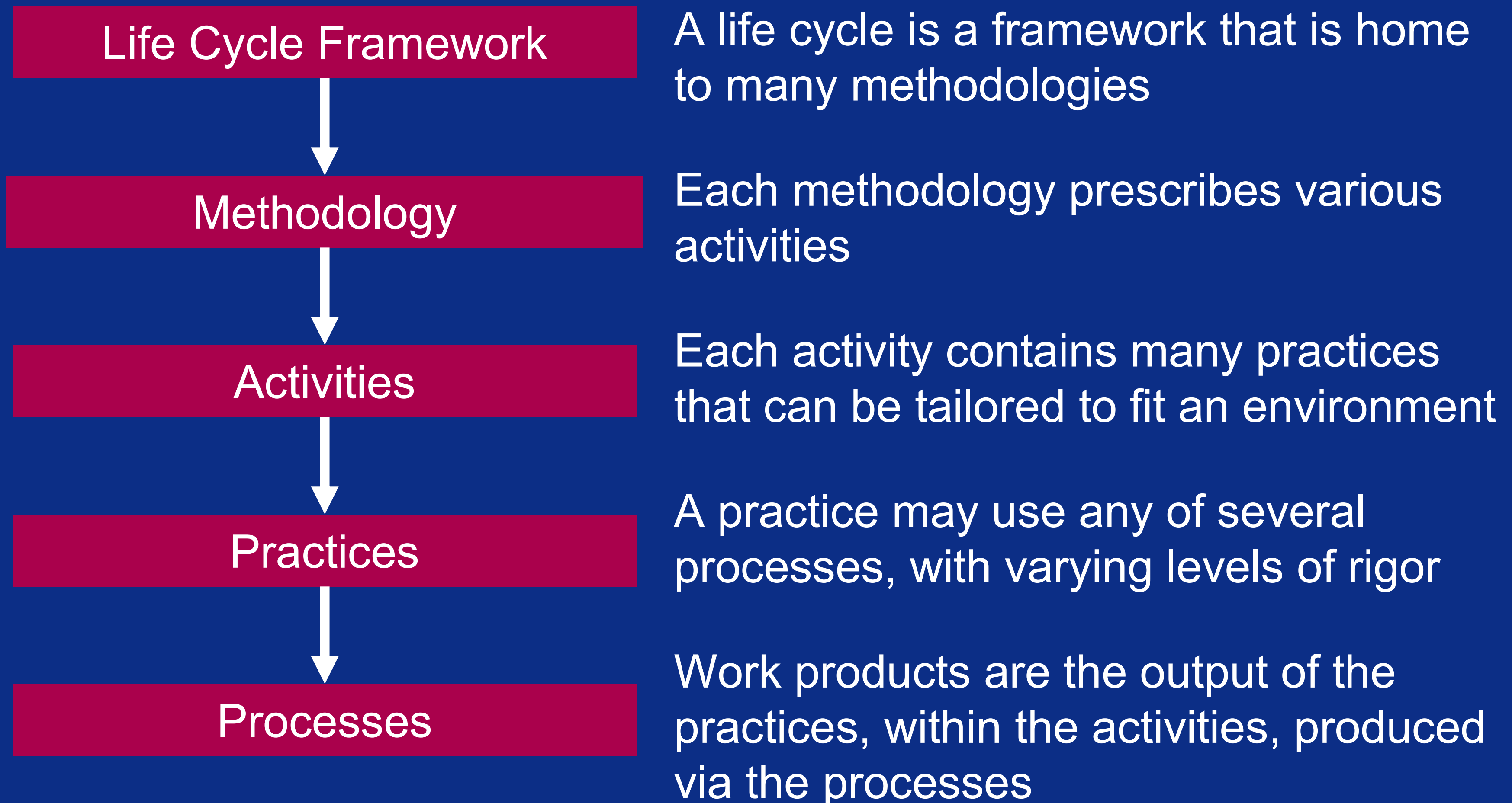
It is not "just" a picture, or a process, or a set of activities, though it may include all these things

There is a quasi-hierarchical relationship among the lifecycle, activities, practices, and processes

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

From framework to processes



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

From framework to processes
Requirements engineering

Life Cycle Framework

↓

Methodologies

↓

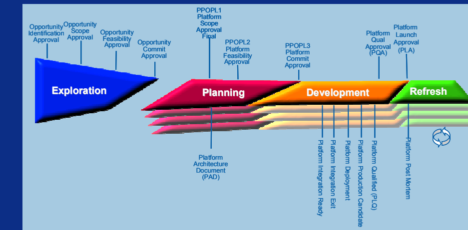
Activities

↓

Practices

↓

Processes

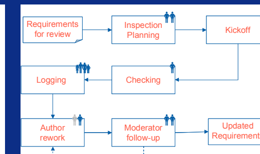


Requirements Engineering

Elicitation	Analysis & Validation	Specification	Verification	Management
Gathering Requirements from stakeholders	*Assessing, negotiating, and ensuring correctness of requirements*	*Creating the written requirements specification*	*Assessing requirements for quality*	*Maintaining the integrity and accuracy of the requirements*

Elicitation planning, Specification Quality Control, Planguage, Formal Specification, QFD, peer reviews, prioritization, ethnography, participant observation, Analysis of Competing Hypotheses, prototyping, etc.

1. **Gather raw requirements**, including sources, all stakeholders, and rationale
2. **Specify top-level requirements** (*ends*, not *means*)
3. **Determine design**:
   3.1 **Analyze requirements**, including stakeholder value, delivery order, and system scope
   3.2 **Find and specify design ideas** to meet the requirements
   3.3 **Evaluate the design ideas** against requirements using Impact Estimation
   3.4 **Repeat steps 1-3** until a reasonable balance between costs and requirements is achieved
4. **Select design ideas** and produce Evolutionary Delivery plan
5. **Manage Evolutionary Delivery project**

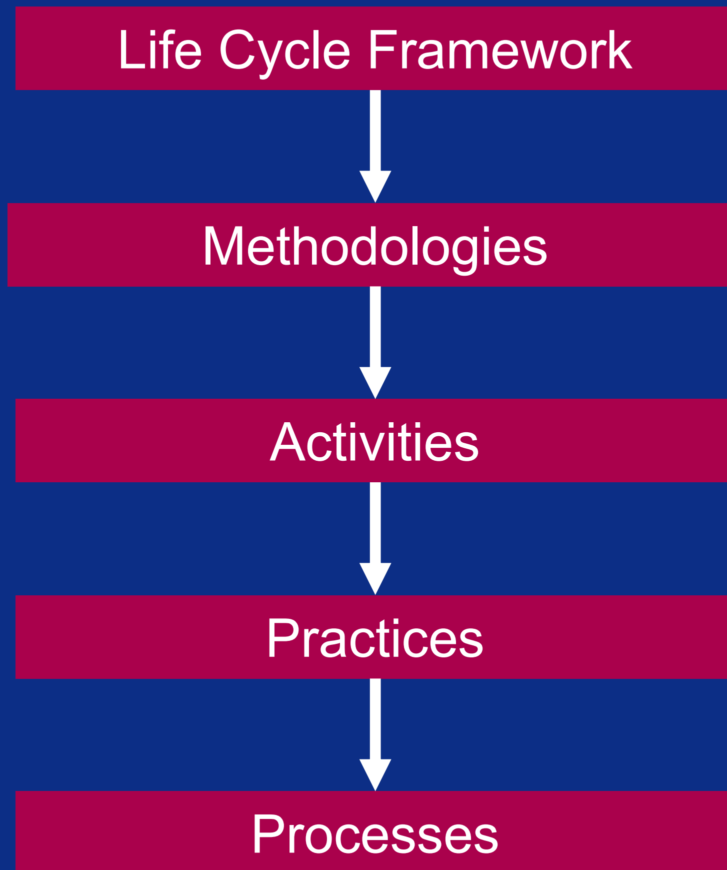


1. A full-time customer representative authors *stories* that illustrate the main requirements of the system and corresponding acceptance tests
2. Developers create estimates for each story; stories with estimates longer than about 2 weeks are decomposed into smaller blocks and re-estimated
3. Developers work to implement one story at a time, in priority order, using the customer representative to provide details and resolve questions
4. A story is complete when the application passes all acceptance tests written for it

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

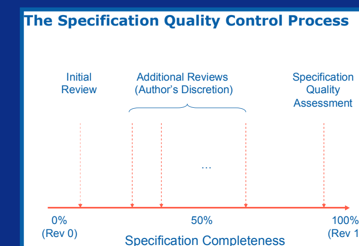
Example: Specification Quality Control as part of requirements verification



Requirements engineering

Verification
Assessing requirements for quality

Specification Quality Control

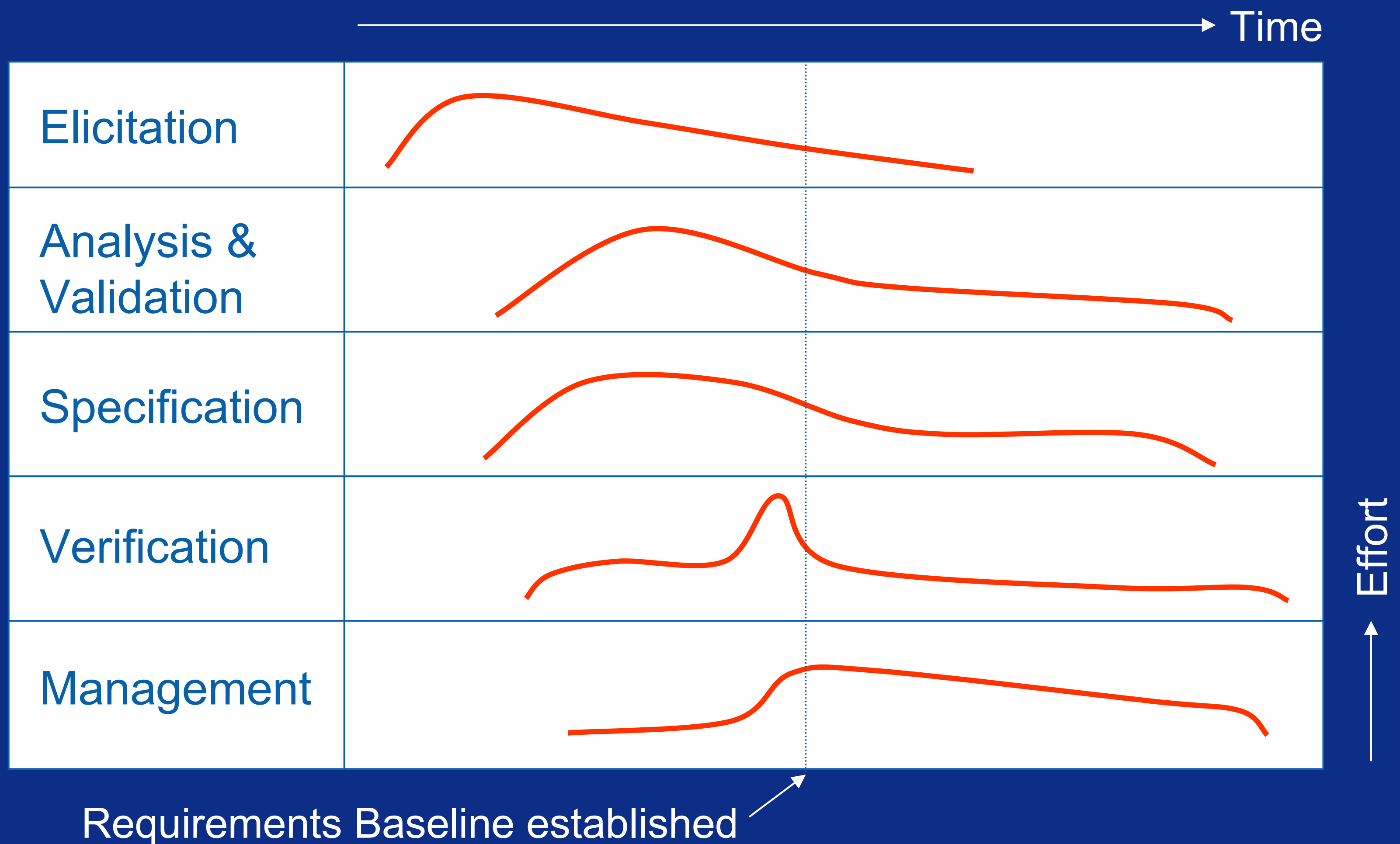


Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Activities are usually performed in parallel

Requirements engineering activities over time



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Additional views on the life cycle

But there are many other prevalent and useful ways to view a life cycle, such as:

A series of commitments

A series of decisions

A series of models, each with increasing fidelity, that culminates in the final, tested system

A sequence of phases, each with a particular purpose, separated by gates to screen for product and project health

A description of the flow of information from group to group, among specialties, across data systems

A set of states through which the project or product must pass

continues

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Additional views on the life cycle *(continued)*

A set of value-added interactions among producers and consumers (marketers, designers, coders, testers, etc.)

A "map" of the project's journey from start to finish, including milestones and points of interest along the way

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

What about Agile?

Agile development increased emphasis on practices and away from lifecycles

But the underlying lifecycle model is *at least* as important as the particular practices used by a team

- How much requirements and planning work is done first vs. postponed?
- How much iteration, and across which activities?
- How many increments?

"The lifecycle model is probably the strongest indication of whether a development approach is 'agile'." – Steve McConnell

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Work flow versus work state

Work flow: What tasks must be performed, by whom, and in what order?

Work State: What must be produced, for whom, by when, at what cost, and at what level of quality?

Both are necessary and valuable things; agile software development favors *state* over *flow* (see the Manifesto)

More generally, agility is greater when we define and focus on *ends* rather than *means*

To what degree should a life cycle prescribe work flow? What about work state?

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

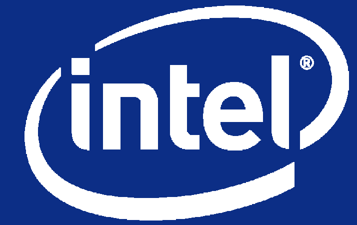
Two forms of control

Cybernetic control (a form of Closed Loop Control) is established in real time, using the difference between the desired output and the actual output to change the system's behavior

Open Loop Control uses sparse instructions to guide and control, assuming intelligence and skill on the part of the recipient. The controlling entity may not check back to see the effects of the control, and those effects may be delayed. Key to this control is a shared mental model

Where should a life cycle rely on open loop control, and where is cybernetic control appropriate?

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.



Software Development Challenges

(at least a few of the bigger ones)

Software development challenges

Problem complexity is increasing – we've solved most of the simple problems

Solution complexity is increasing – we're not finding many simple solutions to complex problems

Design tool complexity is increasing – multi-core, multi-threaded, distributed, cross-platform…yikes

Organizational complexity is increasing – larger software teams, distributed development, more cross-domain interactions, more dependencies

Software development process complexity is increasing – this is a natural response to solution complexity increases

Market forces are unrelenting – the expectations placed on teams have not relaxed even in the face of the other factors

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Complexity

A complex system is one that requires consideration of many parts, factors, or forces at the same time

- Simple systems have low coupling and high cohesion
- Complex systems have high coupling and low cohesion

Example: *Cross-cutting concerns*

- Security is a cross-cutting concern; it can't be understood completely by looking at product architecture, company organizational structure, etc.

Managing complexity is going to be central to future software development

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Coping with complexity

Traditional engineering techniques like reduction into parts help manage complexity, but are not enough by themselves

Agility, abstraction, and hierarchy are essential

Object-orientation is an example of a method that helps deal with complexity via abstraction and hierarchy (encapsulation, information hiding, inheritance, etc.)

How can the life cycle help in these areas?

We need to treat the life cycle more systematically and holistically – it's not just a picture...

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Boundary issues

Software development has a rapidly increasing number of boundaries

Boundaries between: Org groups, individuals, disciplines, phases, methodologies, data sets, activities, and a host of others

- Examples: Architecture vs. design; coding vs. testing

The life cycle, as a framework, defines the location and existence of many of these boundaries – but not the details

We need to find ways to work across all these boundaries

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Protocols – working across boundaries

One very good way to work across boundaries is through use of protocols

- "code of correct conduct", or "how unrelated objects communicate with each other"
- Protocols can be relatively simple, yet work in very complex settings (e.g., TCP/IP)

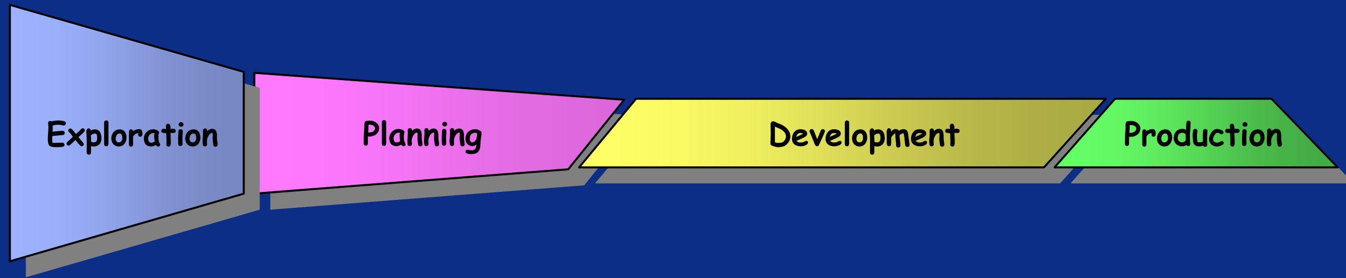
Example: Think about how commitments are made and viewed on an agile vs. traditional team

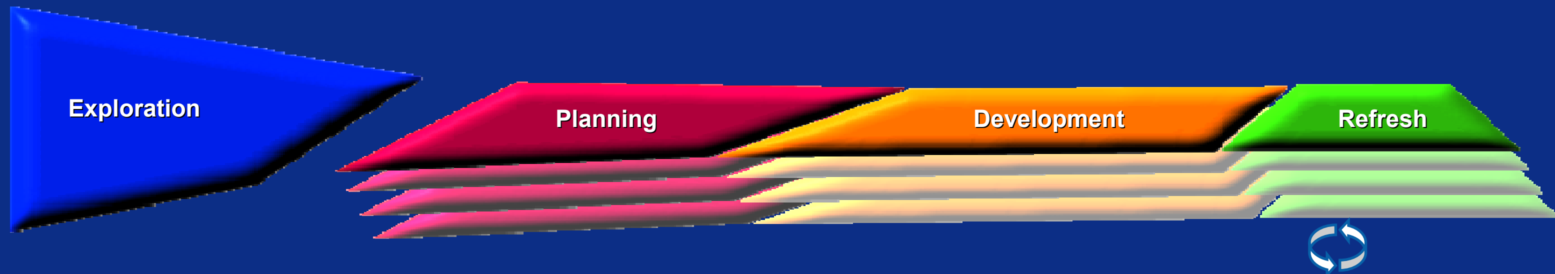
How do things change if we view commitment as a negotiated, cross-boundary communication device rather than a strict contract for enforcement?

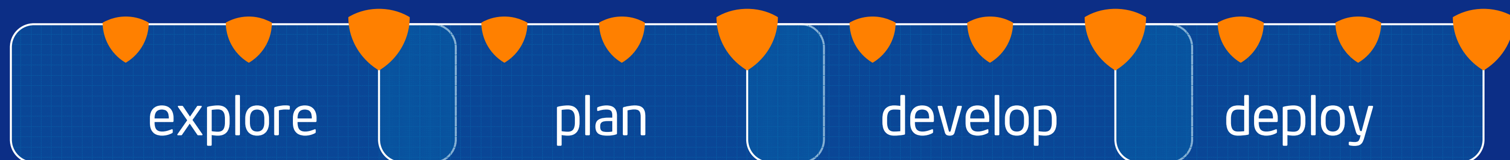
Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Boundary examples



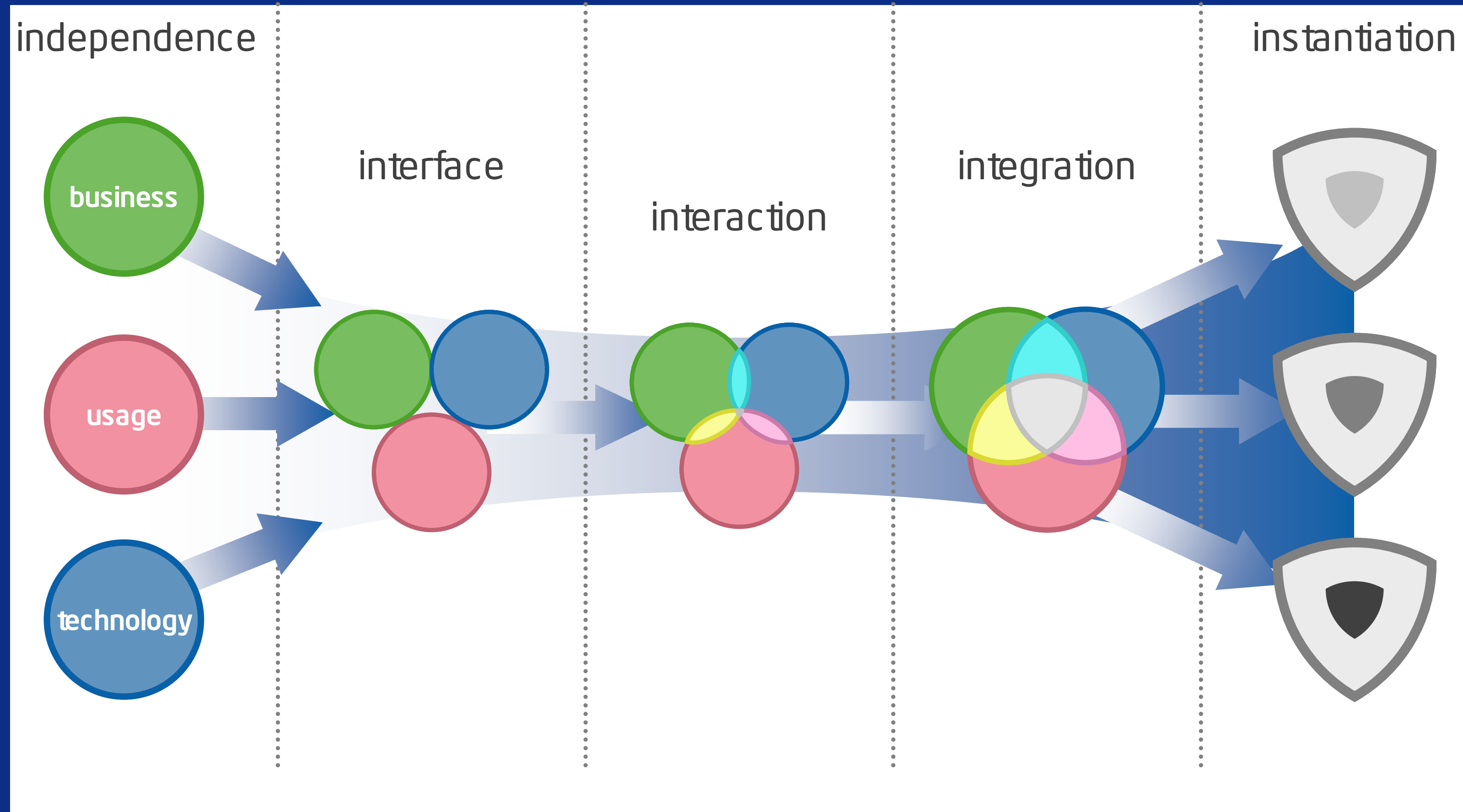




Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

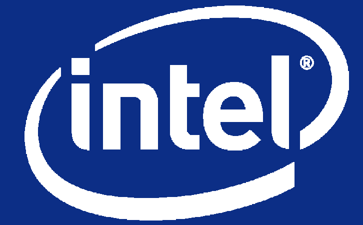
Boundary examples



Systems emerge as business, usage, and technology perspectives converge

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel



Improving the Life Cycle

A more systematic approach

The Viable Systems Model

Created by Stafford Beers based on the human nervous system

Used initially for organizational design and improvements, but can be applied much more generally

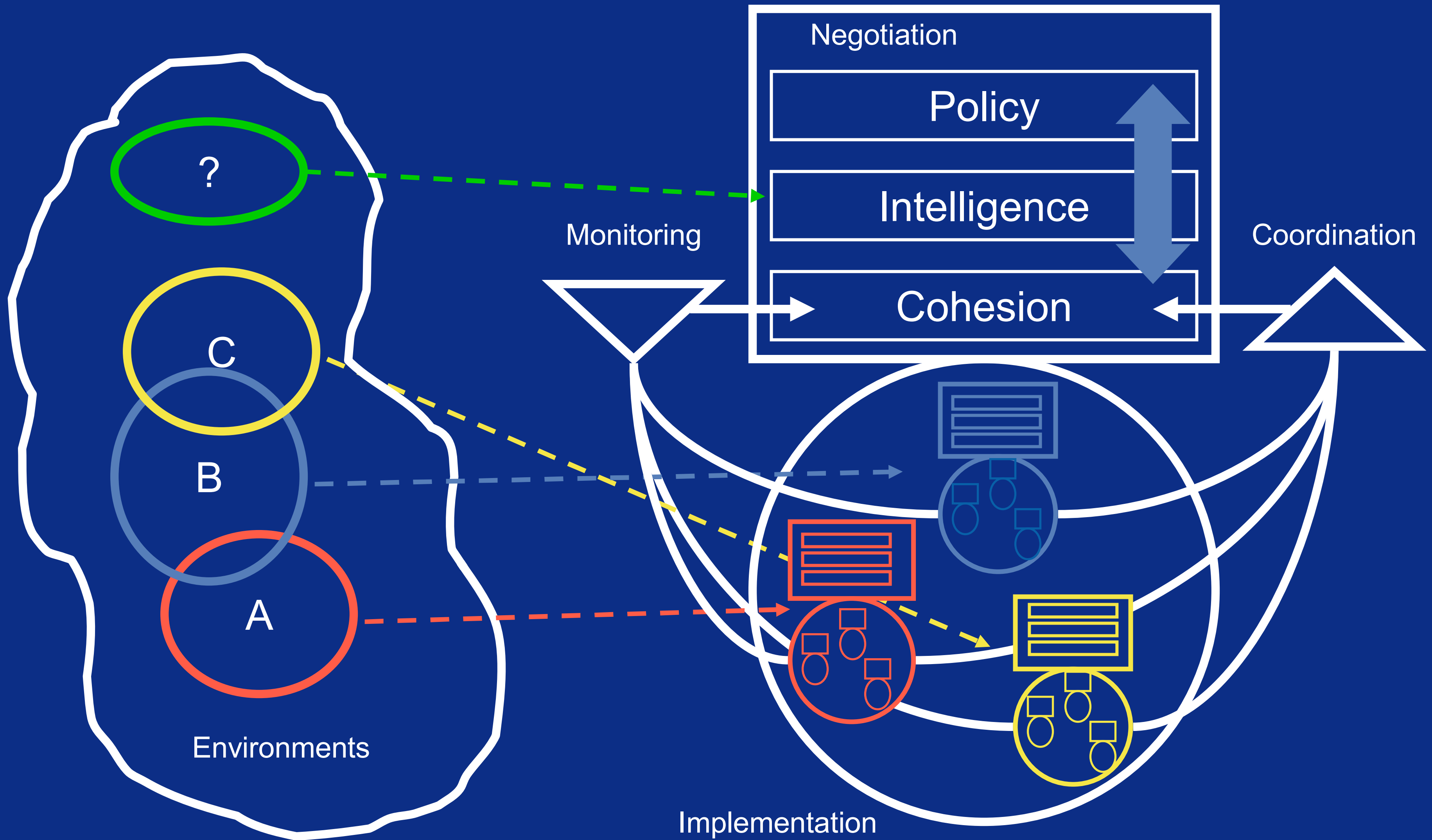
To maintain viability, a system must perform these five simultaneous functions:

1. Implementation *(engaging in different activities)*
2. Coordination *(preventing them from interfering)*
3. Monitoring *(managing them as a set)*
4. Intelligence *(thinking about the future)*
5. Policy *(input to values, purpose, direction, and identity)*

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

The Viable Systems Model



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

The VSM applied to life cycles

The VSM can be applied in two contexts relative to software development:

1. The software life cycle as a system
2. The broader system that produces software (including the lifecycle)

Each system needs to be viable in its own right

In which of the five VSM activities should the life cycle, as a system, engage?

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

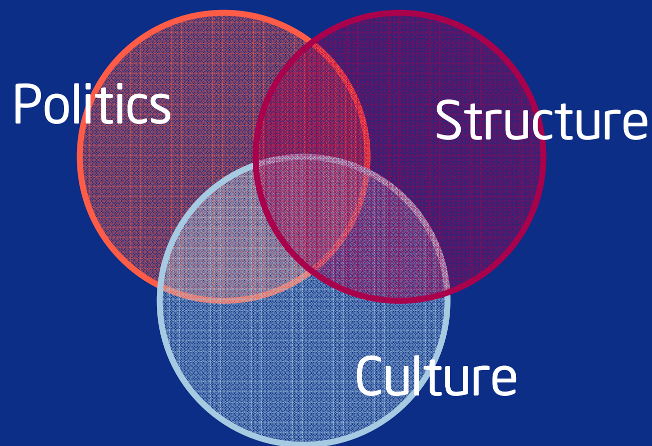
intel

Culture, structure, and politics

It's clear the life cycle has to specify structure using some combination of work state and work flow

But what about cultural? How much should a life cycle specify roles, interactions, cultural artifacts, and values?

What about politics (the science of group decision making) – should the life cycle also address the political dimension?



Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

The need for new techniques

A natural tendency in the face of complexity to “turn up” traditional management

Within a complex system, that can be harmful

Traditional management is largely linear, emphasizing decomposition, reduction, and managing one dimension at a time

- Agile development methods represent linear management techniques taken to a limit; they subdivide problems very finely
- How would story-based software development fare in a complex environment?

By definition, *you can’t simplify a complex system*

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Agile development, agility, and the life cycle

Agile software development started out as a revolution against overhead, too much documentation, and rigid adherence to process

- A decade later, it has changed significantly - b*ut it is being spoken of in the past tense by more and more people*

We must not lose the underlying concept of agility

- Agility implies a *certain immunity to inertia…*
- Agility creates the ability to respond to changing circumstances – new knowledge, new opportunities, unforeseen risks, etc.
- Agility is about sensing and adaptation

Agility is a powerful tool for coping with complexity

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Towards a new development paradigm

Direct control via traditional management needs to be augmented by a systems-based approach rooted in *design*

We need to invent new "non-linear" management techniques

Example: Managed collaboration

- Currently, we can *enable* collaboration, but we don't yet know how to effectively *manage* it
- We need to understand behavior in complex organizations, social net-working, and how to ensure that the right relationships and the right work emerges from the environment

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Life cycles – where next?

We have reached the limits of reductionism as a sufficient approach to problem solving

- Many, but not all, of the problems we will solve in the future require something more

What's next is a new approach with the ability to manage dynamic structures and relationships among many interconnected, interdependent parts

Software development in an increasingly complex world

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

The Life cycle of tomorrow

Because complexity is an increasing, intrinsic part of the environment, life cycles must adapt to remain relevant

More about	Less about
Coordination	Control
Self-organization	Fixed structure
Association	Partitioning of work
Focus on design	Focus on Implementation
Agility	Inertia
Expertise and influence	Authority and direction
Protocols, heuristics	Rules

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

Back to the questions

Is the life cycle still relevant?

Yes! And it can be even more relevant in the future – if we treat it as a system

Can the life cycle still help move quality forward?

You bet; and with increasing complexity being a big influence on future software, *we need the help*

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.

intel

Copyright © 2009 Intel Corporation. No part of this presentation may be reproduced in any way without written permission of Intel Corporation.