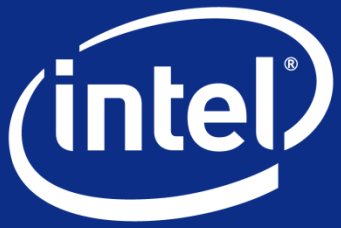


How Common is Common Enough?

Sarah Crary Gregory
Senior Partner
Intel Emergent Systems and Coaching

Finding Common Ground

...A story about a challenging situation* not dissimilar to others at a particular company.

"Common enough?"

How dissimilar is it to others?

Is it enough like what I experience in my own industry or research?

Are there ways in which I might appropriate what is common to my own experience?

Why even ask the question?

Hypothetical program –
“Codename: Blue”.

Blue includes silicon, firmware, and application-level software. Some parts of Blue are developed jointly with a team from a subsidiary company.

Blue is subject to many regulatory requirements, and also must comply with various standards and protocols. The team is globally-distributed, with program management, architecture, silicon development, software development, testing, sales, and support across many different sites.

Codename: Blue

Headcount: ~160

Life-of-program turnover: ~15%

Sites: 7

Countries: 5

100% overlap hours/week: 0

Strategic importance: High

“How common is common enough?”

Definitions: “Common” and “Enough”

Common *(ˈkämən)*

- (adj) occurring, found or done often, prevalent
- Synonyms: usual, ordinary, familiar, regular, frequent, recurrent, everyday

“The teams used a ***common*** dictionary to define terms used in the specification.”

Enough *(iˈnəf)*

- (det) as much or as many as **required**
- (adv) to the **required** degree or extent (after an adjective, adverb, or verb)

“***Enough*** participants joined the user acceptance testing to confirm that the interface met usability targets.”

Both commonality and sufficiency (“enoughness”)

“Common Enough” practice – between two extremes

The commonality of any practice among adopting teams exists along a spectrum, with two relatively well-defined endpoints.

Uniformity

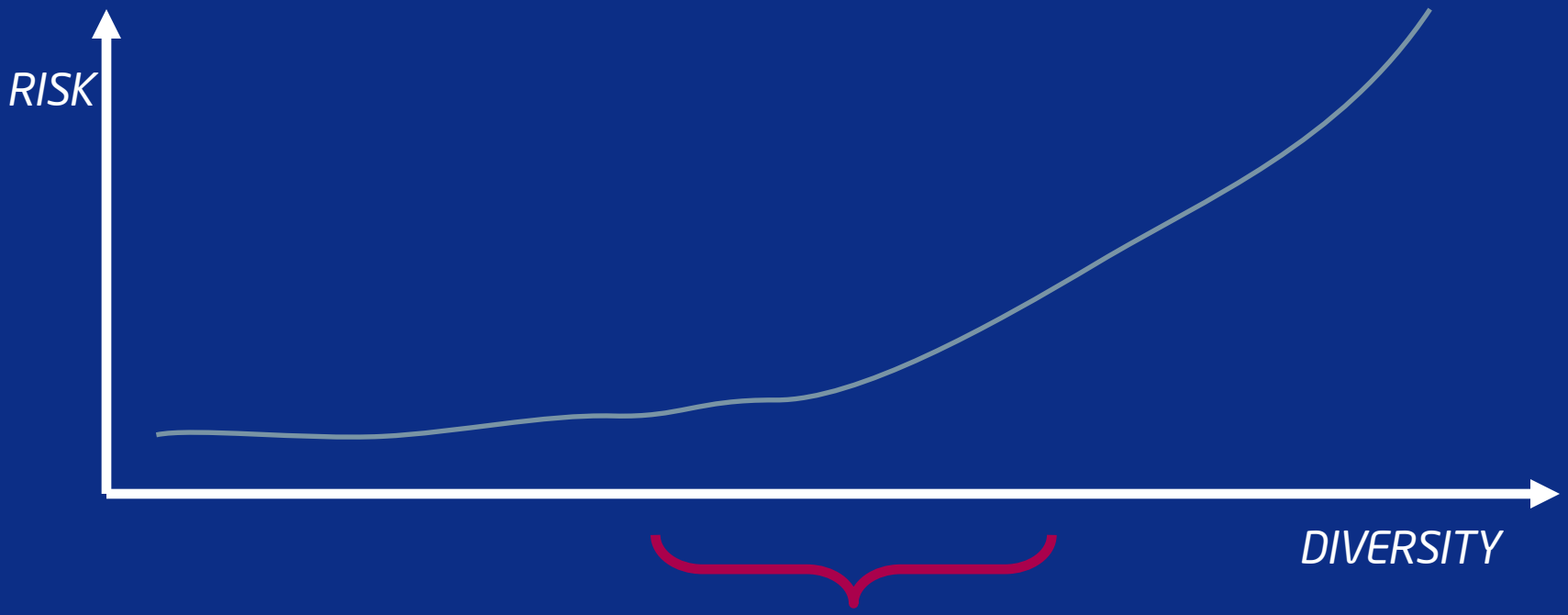
(Homogeneity)

Diversity

(Heterogeneity)

“Well-defined” does not automatically mean “often-occurring.” How many projects precisely follow the script of their methods or best practices? Are any projects subject to complete diversity, with no common practice, terminology, or process at all?

“Common Enough” – diversity and risk



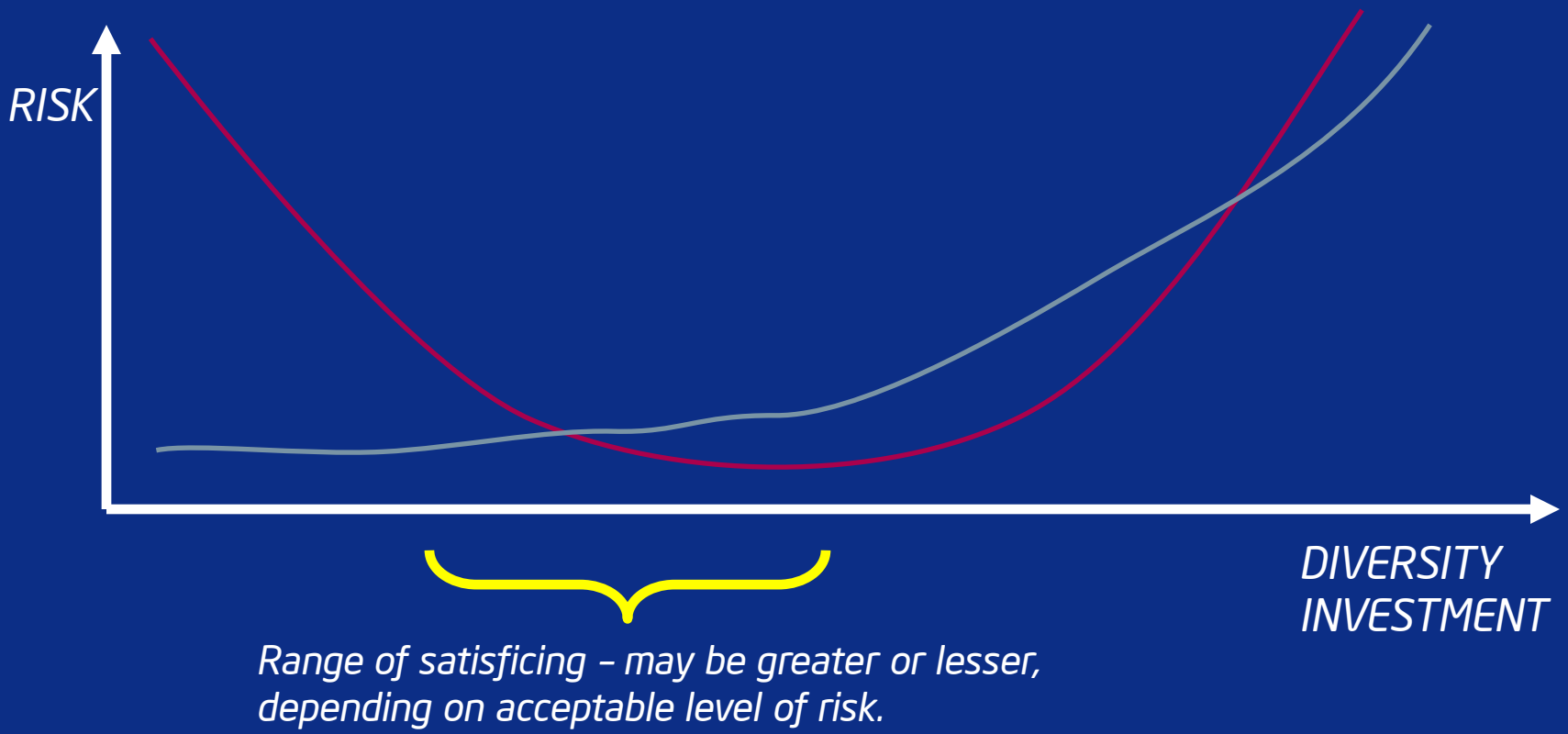
Taken on their own, a specific set of homogenous practices implemented uniformly will generally entail little risk.

(Assuming, of course, that they are the *right* practices, and for some subset of programs.)

“Common Enough” – balancing investment and risk

Too much investment in driving commonality may increase, rather than decrease risk. Heterogeneous practice across different groups may be most effective, if local optimization can be bridged where necessary.

“Common Enough” – balancing diversity and investment



“Just enough” investment and common practice provides a satisfactory range of “enoughness” that allows work to move forward at an acceptable level of risk.

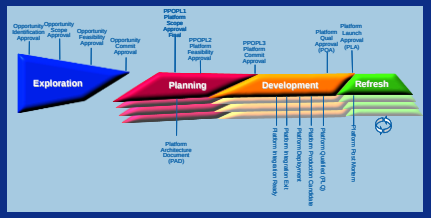
Factors influencing commonality

Precedented vs. Unprecedented Project	• Demonstrated common understanding? • Iteration of a previously built product?
Experience, size, distribution of team	• Experience level of team and individuals? • Small enough for everyone to know others? • Geographically distributed large team?
Cultural and interpersonal factors	• Different levels of language fluency? • Different communication patterns and expectations?
Domain, Organizational, and Technical Complexity	• How many senior managers are involved? • Where is the work emergent, and therefore less able to be defined?

From framework to data

Requirements engineering

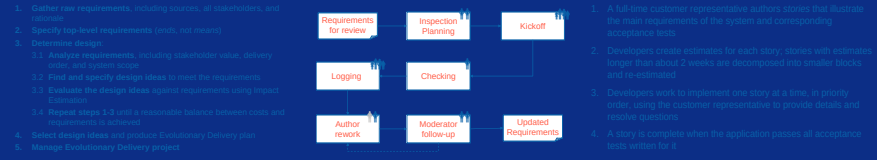
- Framework
- Methodologies
- Activities
- Practices
- Processes
- Tools
- Data



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, prototyping, etc.

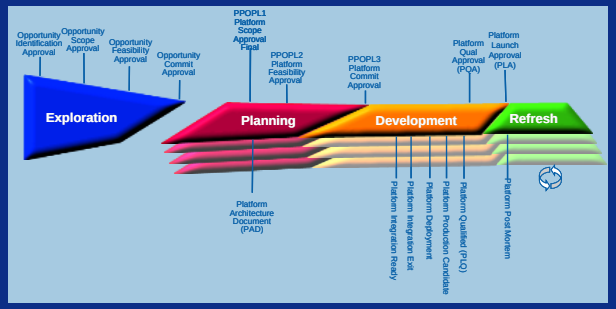


Many, many tools…

While streaming video, when a call comes in, the system shall pause the video and prompt the user to answer the call.

Framework

Framework



The **framework** organizes **methodologies** (and subsequent work) in a useful way.

Information at this level establishes the *taxonomy* and *terminology* used to communicate amongst team members as well as across group boundaries.

Too much commonality: Driving absolute consistency at all levels, in all ways, may lose important domain or disciplinary nuance, stifle innovation, and also be unnecessary effort.

Not enough commonality: Silos, with each group adopting (or inventing) a locally-optimized solution with little attempt to communicate across boundaries.

Methodologies

Requirements Engineering

Methodologies

Methodologies prescribe **activities** that enable the agent or actor to fulfil the objectives of the methodology

Too much commonality: Methods are identified as standards for all work, regardless of the nature of the work, the team, or other characteristics of the project at hand.

Not enough commonality: Every project begins anew, with no retrospective look at what worked before, and whether it might (or might not) work for the particular project.

Activities

Activities enable the agent or actor to fulfil the objectives of the methodology. As a discipline, RE has a set of activities that define a set of practices within them, .

Too Much Commonality: We have to elicit “all the requirements” before we can begin any design or development.

Not Enough Commonality: No assessment of the activities to determine how and when they will take place.

Practices

Practices enable activities, and comprise the *applied* skills and techniques brought to bear on a particular project or program in order to achieve a particular end.

RE practices are the actions we employ in order to fulfill the activities. Each activity contains several possible practices – which will we choose for the project we are working on now?

Practices

Too Much Commonality: This practice worked well before, so it'll work again, and always, on all projects.

Not Enough Commonality: Each team member chooses how she will do her work

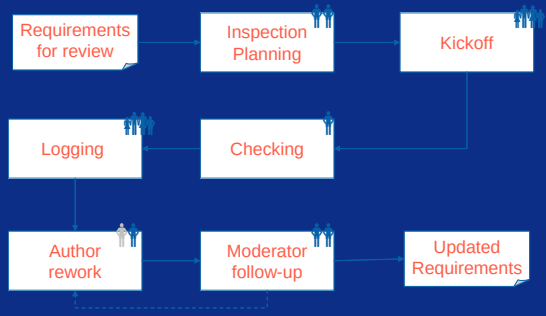
Processes

Processes constrain the **practices** with specific tasks that define roles, responsibilities, and often the format and cadence of specific output.

Too much commonality: A single process description exists, and policies mandate that it be applied to all projects and programs, without *appropriate* tailoring

Not enough commonality: Process descriptions do not exist, or are written but widely disregarded. “Send me any feedback by Friday at 5.”

Processes



Tools & Data

Tools and Data - the repositories for and output of the processes, respectively.

Questions of commonality often arise about **tools**. Must an organization settle on "one tool to rule them all," or is a heterogenous tool environment acceptable?

Questions about the commonality of **data** are often resolved through use of specific syntax.

Tools

Data

Many, many tools…

While streaming video, when a call comes in, the system shall pause the video and prompt the user to answer the call.

Boundary issues

"Common enough" invokes the many memberships that are often silently at work in product development.

Boundaries between: Organizations, groups, individuals, disciplines, phases, methodologies, conceptual models, data sets, activities, languages, *and many others*.

What is common within a category may not be common when one crosses across a boundary into a different category.

"Does this word mean the same thing to me as it does to you?"

We need to find ways to work across all these boundaries

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?

Boundaries

Our discipline and the areas in which we work, as well as the ways we work themselves, are subject to ever more *complexity*.

The emphasis on increasingly complex systems emphasizing cross-cutting concerns results in factors that are very different from those such as *weight* or *reliability*.

- Where is ***user experience*** located?
- It is easy to find the battery in a system, but where is ***battery life*** found within the system?

More complexity often leads to demands for more commonality.

Two Threats to Finding "Common Enough"

Dogma

- (n) **Dogma** is a principle or set of principles laid down by an authority as incontrovertibly true. It serves as part of the primary basis of an ideology, nationalism or belief system, and it cannot be changed or discarded without affecting the very system's paradigm, or the ideology itself.

Corruption

- (n) In philosophical, theological, or moral discussions, **corruption** is spiritual or moral impurity or deviation from an ideal. [...] The word *corrupt* when used as an adjective literally means "utterly broken."

Dogma and corruption are not necessarily opposite concepts. Might dogmatic application of a particular method or practice result in a project that is "utterly broken?"

Challenge: find the spot that is "common enough" – and have the courage to change it as needed. Complex adaptive systems demand it.

The Role of RE in a Complex World

How might we define “common enough” RE for Codename: Blue?

Framework: Acknowledge that different segments of the project will workdifferently. Establish glossary, name boundaries.

Activities: Each part of the program agrees on the model of 5 RE activities.

Practices: Actual means of implementing the activities may differ (Scrum for SW dev, inspections for hardware architecture specs, etc.)

Tools: VERY heterogenous environment. Common language.

Codename: Blue

Headcount: ~160

Life-of-program turnover: ~15%

Sites: 7

Countries: 5

100% overlap hours/week: 0

Strategic importance: High

Is It Common Enough? Some closing thoughts.

No bright line rule for “common enough.” (Sorry.)

- Requirements statements will likely never reach the point where “too common” is an issue.
- Requirements can be “not common enough” – insufficient commonality is a sign of false consensus, especially in a very heterogeneous environment.
- Imposed commonality on methods may lead to very poor results in innovation, with deleterious effects in a complex environment.

No upper limits to creativity! (Usually…)

- Know the many boundaries your program must traverse, and ensure that protocols are in place to communicate across them.
- Within any given group, innovation and creative means of sensing and responding to complexity can yield good results.

Finally

"All models are wrong, but some of them are useful." George Box

Finding **common enough** requires fluency in a wide range of "wrong but useful models."

Conferences such as REFSQ are excellent opportunities to explore more models, and ask the question of whether another's experience or idea just might be common enough to work in a situation in your own environment.

Many thanks for the opportunity to ask and discuss the question, "How common is common enough?"

Questions?

Backup

Acknowledgements

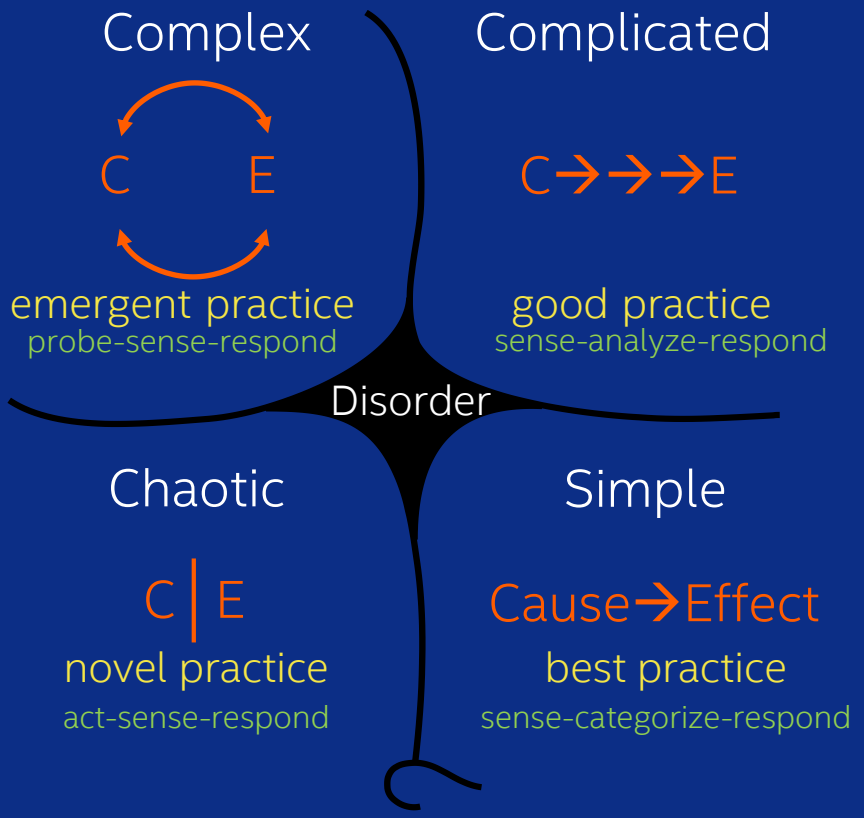
- Erik Simmons, John Terzakis, Ray Arell, Intel Emergent Systems and Coaching colleagues
- Dozens of projects and hundreds of practitioners in a decade's work in RE at Intel who taught me that history does ***not*** always repeat itself, despite what my undergraduate professors claimed.
- We remain grateful for the work of researchers as well as our industry colleagues in the broader RE and Software Quality communities. Your work challenges and inspires us to continuously reflect on and improve our practice.

Cynefin Framework

The Cynefin Framework describes five system domains: simple, complicated, complex, chaotic, and disordered

- The Cynefin Framework helps characterize the problems we face, and the solutions we construct

Systems often exhibit the characteristics of more than one domain



Cynefin is a framework developed by David Snowden/Cognitive Edge, built on the science of complex adaptive systems.
http://cognitive-edge.com

Not just for Requirements anymore: The “3 Cs”

Requirements help establish a clear, common, and coherent understanding of what the system must accomplish.

Clear: All statements are unambiguous, complete, and concise

Common: All stakeholders share the same understanding

Coherent: All statements are consistent and form a logical whole

Clarity can be judged through evaluating a statement for ambiguity, assessing its completeness for its intended purpose, and determining the absence of extraneous content.

Coherence refers to the conceptual integrity of the group of statements as a whole.

Commonality of understanding of requirements statements is simple, but a more complex question arises with questions of stakeholder definition and project management.

Intel Emergent Systems and Coaching

RE at Intel has been housed in several parts of the company over nearly 15 years.

- Quality
- Project Controls / Project Management
- Software Program Office
- Emergent Systems (since Dec 2012)

"Emergent Systems"

- Not about providing a packaged method, but about adding value and facilitating transformation.
- Agile, RE, Change Agency, Technical Practices (Test-Driven Development), Lean, complex systems research, etc. in a small group of senior practitioners and thought leaders
- Different perspective on the work, as we're not about teaching techniques as much as we are about organizational transformation.