“It Depends”
Heuristics for “Common Enough” Requirements Practice

Sarah Crary Gregory
Senior Partner
Intel Emergent Systems

Gothenburg, Sweden, March 14-17, 2016
REFSQ
22nd Intl. Working Conference on Requirements Engineering: Foundation for Software Quality
2016

Intel Emergent Systems

“Emergent Systems”

- Agile, RE, Change Agency, Technical Practices (Test-Driven Development), Lean, complex systems research, etc. in a small group (n=~15) of senior practitioners
- Different perspective on the work – we don’t teach “the right” techniques as much as we focus on *organizational transformation*.

We exist to take on Wicked Problems

“A wicked problem is a problem that is difficult or impossible to solve because of incomplete, contradictory, and changing requirements that are often difficult to recognize.” - Wikipedia

There is generally NO ‘one solution’ for wicked problems.

Copyright © 2016 Intel Corporation. All rights reserved.

Before the Answer – revisiting the Question

How common is common enough?

What requirements process do we use?

We're Agile, so we don't need to do requirements, right?

Where can I learn the right way to do RE?

How long will it take to get our requirements done?

How long do I have to get our requirements done?

Who will write our requirements?

How will we know that we're done?

Do we have to use the same tool?

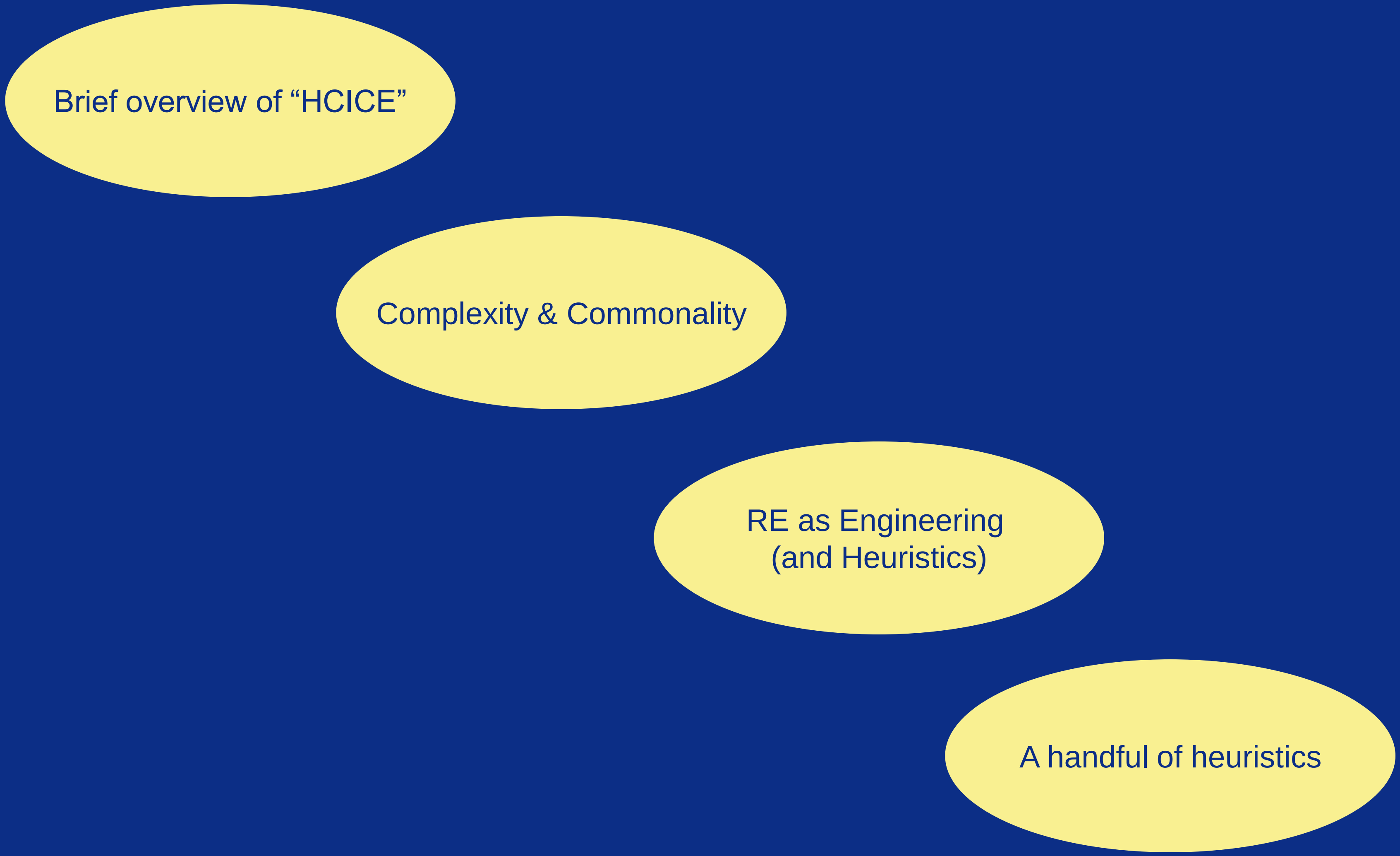
… and so much more…

No easy answers…
Invitation to a discussion

The wholly unsatisfying answer to these questions: "It depends…"

Copyright © 2016 Intel Corporation. All rights reserved.

The path we’ll take



Brief overview of “HCICE”

Complexity & Commonality

RE as Engineering
(and Heuristics)

A handful of heuristics

Copyright © 2016 Intel Corporation. All rights reserved.

Definitions: “Common” and “Enough”

Common *(ˈkämən)*

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

“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)

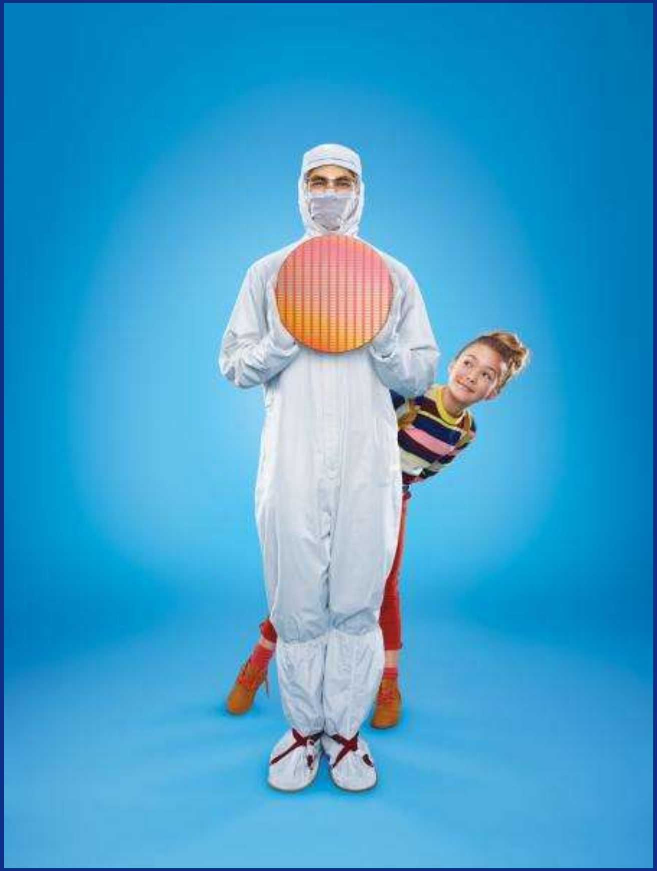
“***Enough*** participants joined the user acceptance testing to confirm that the interface met usability targets.”

Both commonality and sufficiency (“enoughness”)

Copyright © 2014 Intel Corporation. All rights reserved.

Redux: 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?

Copyright © 2016 Intel Corporation. All rights reserved.

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

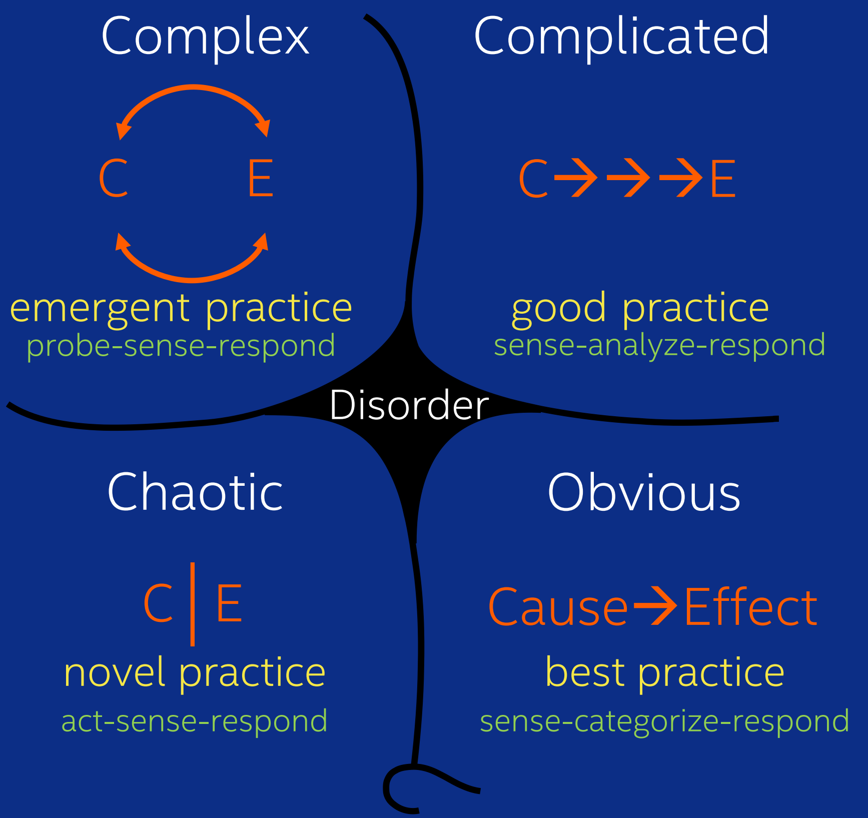
"How common is common enough?"

Copyright © 2014 Intel Corporation. All rights reserved.

Getting comfortable with complexity

This is where things are interesting…

And here as well… heuristics can move us to a more predictable place.



Complex

C E

emergent practice

probe-sense-respond

Complicated

C→→→E

good practice

sense-analyze-respond

Disorder

Chaotic

C | E

novel practice

act-sense-respond

Obvious

Cause→Effect

best practice

sense-categorize-respond

The experts provide the requirements

Requirements reuse, spec-driven requirements

HCICE – are the risky requirements in the complex quadrant?

The Cognitive Edge method is ©2012 Cognitive Edge (USA) Inc., used under a Creative Commons Attribution-Noncommercial-Noderivs license: http://creativecommons.org/licenses/by-nc-nd/3.0/.

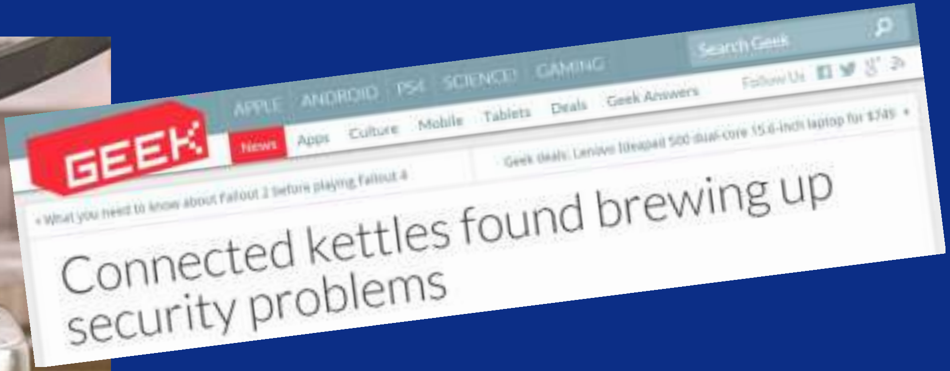
Copyright © 2016 Intel Corporation. All rights reserved.

Emergence and Cross-Cutting Concerns

Security, for example, is considered an *emergent behavior* of a system, and as such can never be fully specified.

- Emergent behaviors cannot be predicted by looking at the parts of the system

Because emergent behaviors involve many components, teams, and business groups, they also are called *cross-cutting concerns*



« What you need to know about Fallout 3 before playing Fallout 4

Geek deals: Lenovo Ideapad 500 dual-core 15.6-inch laptop for $749 »

Connected kettles found brewing up security problems

http://www.geek.com/news/connected-kettles-found-brewing-up-security-problems-1637249/

Copyright © 2016 Intel Corporation. All rights reserved.

RE as Engineering

Engineering is the application of heuristics under uncertainty to cause the best possible change within the available resources (Billy Vaughn Koen*)

A heuristic is anything that provides a plausible aid or direction in the solution of a problem

- Use of heuristics does not guarantee a solution to a problem, and heuristics *can offer conflicting advice* for any given situation

  "Do what worked last time."
  "Take the first solution that presents itself."
  "Always leave a path for retreat."

- Heuristics are "unjustified, incapable of justification, and potentially fallible."

RULE OF THUMB
If your extended thumb is too small to block your view of the hazmat incident, you're not far enough away.

*See Koen, *A Discussion of the Method*

Copyright © 2016 Intel Corporation. All rights reserved.

A few “common enough” heuristic definitions

Requirements Engineering is the use of heuristics for discovering, documenting, and maintaining a set of requirements for a system or service

A Requirement is a statement of one of the following:

1. What a system must do (Functional requirement)
2. How well the system must do what it does (Quality and Performance requirement)
3. A known resource or design limitation (Constraint)

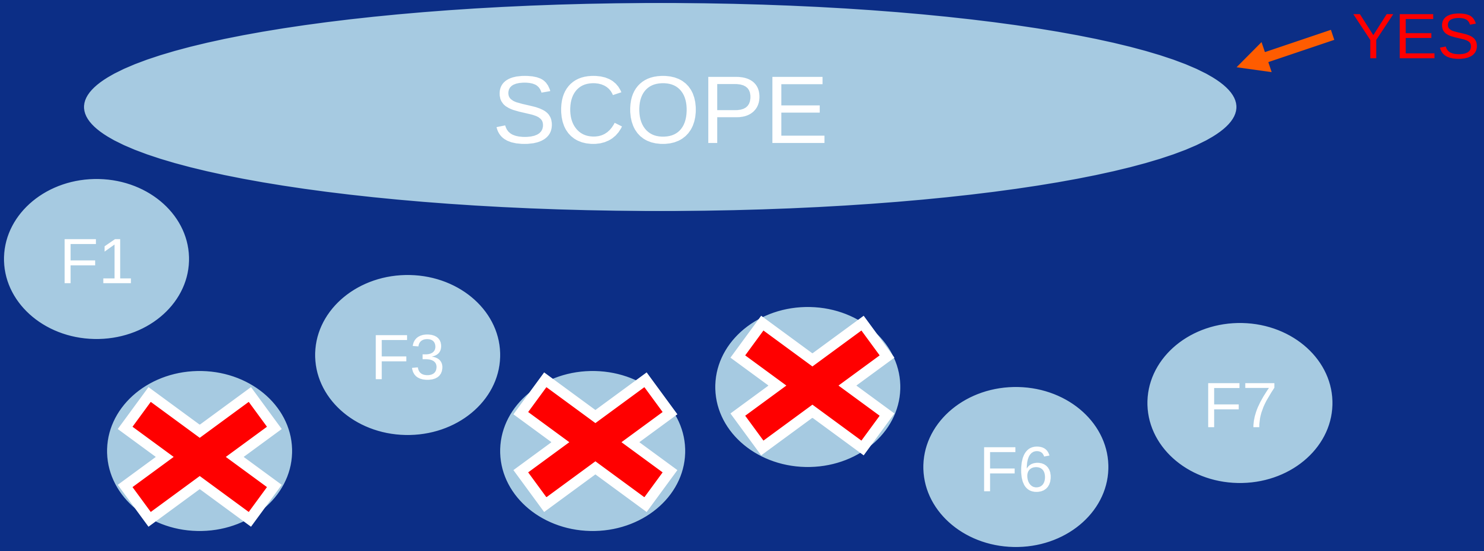
Elicitation	Analysis & Validation	Specification	Verification	Management
Gathering Requirements from stakeholders	*Assessing, negotiating, and ensuring correctness of requirements*	*Creating the written requirements specification*	*Assessing requirements for quality*	*Maintaining the integrity and accuracy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

All RE activities consist of choices among heuristics.

Copyright © 2016 Intel Corporation. All rights reserved.

Heuristic 1 of 5

Start by specifying the scope of the system at a high, broad level, with minimal details.



Make a conscious decision of what you are NOT going to write.

Do we really need to write "all the requirements"?

Copyright © 2016 Intel Corporation. All rights reserved.

Heuristic 2 of 5

Prioritize writing the risky, complex requirements.

- It is easy to find the battery in a tablet, but where is battery life found within the system?
- Antivirus software is easy to find, but what about security?
- Where is user experience located?
- *Which group owns this work?*

Addressing cross-cutting concerns isn't as simple as finding "the architect" with expertise in each area.

These are social questions as much as they are technological ones.

http://www.macrumors.com/2015/09/24/iphone-6s-apple-store-lineups/

Copyright © 2016 Intel Corporation. All rights reserved.

Heuristic 3 of 5:

The requirements must guide the current activities of all team members at an acceptable level of risk.

"A requirement is complete when it contains sufficient detail for those that use it to guide their work."

Therefore, a requirement (and a specification) *can be complete at all times** during a project, even though additional detail continues to be added.

Is my specification complete? Is this requirement complete?

... It depends....

Copyright © 2016 Intel Corporation. All rights reserved.

Heuristic 4 of 5

Quantify qualitative requirements so they are verifiable.

Just because a requirement can’t be neatly tested with a Boolean script doesn’t mean it’s not able to be described with precision and yes, can be *complete*, too.

Consider a Landing Zone:

Requirement	Minimum	Target	Outstanding
Cost	$50,000	$35,000	$15,000
Passengers	6	4	2
Gas Mileage	30mpg	40mpg	n/a
0-60 speed (“fun”)	15 sec.	10 sec.	3 sec.

Each item is a separate requirement, described with a Scale (“measured against”) and Meter (“how measured”). (T. Gilb.)

There is rarely a “right value” for a quality or performance requirement.

Copyright © 2016 Intel Corporation. All rights reserved.

Heuristic 5 of 5

Rigorously verify requirements to prevent defects and maximize requirements quality.

Two initial questions:

1. How good were your last requirements?
2. How do you *know*?

Then two more questions:

1. What is the most common review method in use for most specifications?
2. What is the most common form of feedback received by authors, given that review method?

Defect prevention > defect removal.

Copyright © 2016 Intel Corporation. All rights reserved.

Every discipline has its own specialized heuristics

The system should always keep the user informed about what is going on, through appropriate feedback within a reasonable time.

SEE IT: Teens hilariously react to Windows 95 with shock and awe

BY KERI BLAKINGER / NEW YORK DAILY NEWS / Tuesday, March 8, 2016, 3:32 PM

Jakob Nielsen, 10 Usability Heuristics for User Interface Design (1995)
https://www.nngroup.com/articles/ten-usability-heuristics/

Get the right heuristics to get the right requirements –and to get the requirements right!

Copyright © 2016 Intel Corporation. All rights reserved.

The RE Pragmatist in Industry

Affirmatively choose to base your Requirements Engineering program around heuristics – or at least consider that your standard practices are heuristic – “unjustified, incapable of justification, and potentially fallible.”*

If what we consider to be “best RE practice” is in fact simply a set of heuristics, then the answer to any* RE question will probably be “It depends.”

Use heuristics to answer the inevitable follow-on question – “It depends on what?”

*Keep in mind that everything I’ve said – also unjustified, incapable of justification, and potentially fallible.

Copyright © 2016 Intel Corporation. All rights reserved.

Finally – my favorite heuristic

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."

By all means, look to see what your RE colleagues are doing as a source of new heuristics to try.

Look to orthogonal – or even completely unrelated (?) fields for additional heuristics. (We spend a good deal of time with ethnographers…)

"It depends" isn't the answer – only the beginning of the question. But "try what worked well last time" isn't always a bad place to start the answer – as long as you don't stop there automatically.

Copyright © 2016 Intel Corporation. All rights reserved.

Acknowledgements

- Erik Simmons, Ray Arell, Brian Bramlett and Intel Emergent Systems colleagues, as well as our mentees.
- Dozens of projects and hundreds of practitioners in close to 15 years' work in RE at Intel who taught me that history does ***not*** always repeat itself, despite what my undergraduate professors claimed.

  (The heuristic Koen mentions, "Do what worked last time" – has pitfalls, yes.)
- We remain ever grateful for the work of researchers as well as our industry colleagues in the broader RE and Software Quality communities. Your research is a wealth of interesting heuristics, many of which come in and inform the way we approach our own work.

Copyright © 2016 Intel Corporation. All rights reserved.

Thank you!

Questions?

Copyright © 2016 Intel Corporation. All rights reserved.