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

Sarah Crary Gregory
Senior Partner
Intel Emergent Systems
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.
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...”
The path we'll take

1. Brief overview of “HCICE”
2. Complexity & Commonality
3. RE as Engineering (and Heuristics)
4. A handful of heuristics
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  (ɪˈnəf)

- (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”)
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?
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?”
Getting comfortable with complexity

This is where things are interesting…

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

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/](http://creativecommons.org/licenses/by-nc-nd/3.0/).
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.

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.”

*See Koen, A Discussion of the Method
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)

All RE activities consist of choices among heuristics.
Heuristic 1 of 5

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

SCOPE

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

Do we really need to write “all the requirements”?

F1
F3
F6
F7

F2
F4
F5
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/
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....
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:

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

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

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!
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.
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.
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.
Thank you!

Questions?