Heuristics for systems-level architecting

CS 428

WINTER 2023

BRUCE F. WEBSTER

What are these?

- "heuristic" = from the Greek heuriko, "to discover" (hence: "Eureka! I have found it!") – something to help guide you to a solution
  - In AI, a heuristic is typically a goal-seeking algorithm or principle
  - Here: principles, laws, rules, maxims or rules of thumb to remember in creating and building a system of any kind (not just information technology)
- D = descriptive, that is, telling you what can happen
- P = prescriptive, this is, telling you what you should be doing to help avoid the problems
- Source: **The Art of Systems Architecture** (3rd ed.) by Mark W. Maier and Eberhardt Rechtin (CRC Press, 2009)
- These slides cover only a small subset from the document

Multitask Heuristics

- Performance, cost, and schedule cannot be specified independently; at least one of the three must depend upon the other
- **Efficiency is inversely proportional to universality** [the reuse problem]
- The most reliable part on an airplane is the one that isn't there
- There is no such thing as a purely technical problem

Scoping and Planning

- The beginning is the most important part of the work. (Plato)
- **In architecting a new program, all the serious mistakes are made the first day. (Spinrad)**
- Success is defined by the beholder, not by the architect.
- Four questions need to be answered as a self-consistent set if a system is to succeed economically, namely: Who benefits? Who pays? Who provides? and, as appropriate, Who loses?
- Do the hard parts first.

Interlude: Machiavelli

- "It ought to be remembered that **there is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things.** Because the innovator has for enemies all those who have done well under the old conditions, and lukewarm defenders in those who may do well under the new. This coolness arises partly from fear of the opponents, who have the laws on their side, and partly from the incredulity of men, who do not readily believe in new things until they have had a long experience of them."

— **Niccolò Machiavelli**, The Prince

Modeling

- If you can't analyze it, don't build it.
- A model is not reality.
- The map is not the territory.
- **If you can't explain it in five minutes, either you don't understand it or it doesn't work.**
- A good solution somehow looks nice.

Prioritizing (Trades, options, choices)

- In any resource-limited situation, the true value of a given service or product is determined by what one is willing to give up to obtain it.
- The choice between architectures may well depend upon what set of drawbacks the client can handle best.
- Every once in a while you have to **go back and see what the real world is telling you.**

Aggregating (“chunking”)

- Group elements that are strongly related to each other; separate elements that are unrelated.
- Choose the elements so that they are as independent as possible; that is, elements with low external complexity (low coupling) and high internal complexity (high cohesion).
- **System structure should resemble functional structure.**

Partitioning (Decompositioning)

- Design the structure with good "bones"
- It is inadequate to architect up to the boundaries or interfaces of a system; one must architect across them
- **Be prepared for reality to offer a few interfaces of its own**

Integrating

- Relationships among the elements are what give systems their added value
- The greatest leverage – and the greatest dangers – of systems architecting are at the interfaces
- Be sure to ask the question, "What is the worst thing that other elements could do to you across the interface?"

Certifying (System Integrity, Quality, and Vision)

- **As time to delivery decreases, the threat to functionality increases.**
- The least expensive and most effective place to find and fix a problem is at its source.
- Quality can't be tested in; it has to be built in.
- High-quality, reliable systems are produced by high-quality architecting, engineering, design and manufacture, not by inspection, test, and rework.

Assessing Performance, Cost, Schedule, & Risk

- **System quality is defined in terms of customer satisfaction**, not requirements satisfaction.
- If you think your design is perfect, it's only because you haven't shown it to someone else.
- "Proven" and "state of the art" are mutually exclusive qualities.

Re-Architecting, Evolving, Modifying, & Adapting

- **If you don't understand the existing system, you can't be sure you're rearchitecting a better one.**
- When implementing a change, keep some elements constant to provide an anchor point for people to cling to.
- Before the change, it is your opinion. After the change, it is your problem.