CS 428
"The Five Orders of Ignorance" (Armour)

Fall 2018

Bruce F. Webster

The true nature of software

◈ Software is a knowledge-storage medium

◈ Developing software is a knowledge-acquiring activity

◈ We seldom have perfect knowledge ahead of time

  ◈ If we do, then why are we doing this? The problem has already been solved.

◈ Therefore, software development is an inherently inefficient exploration and discovery process

  ◈ We determine what works

  ◈ We determine what does not work (for this particular system)

◈ Observations and experiences?

What can happen along the way

◈ The problem of late discovery

  ◇ Blind alleys, significant backtracking

  ◇ Webster: Do not Defer the Difficult in IT Projects (2013)

◈ Two kinds of knowledge

  ◇ Again, what doesn't work and what does

  ◇ Sometimes, what doesn't work is just as valuable, but is often discarded

◈ "Corrupted" knowledge

  ◇ The code often contains artifacts or remnants of our knowledge exploration during development – these may be misleading or obscuring

◈ Observations/experiences?

What the discovery process looks like

Exhibit 6. A path through the woods.

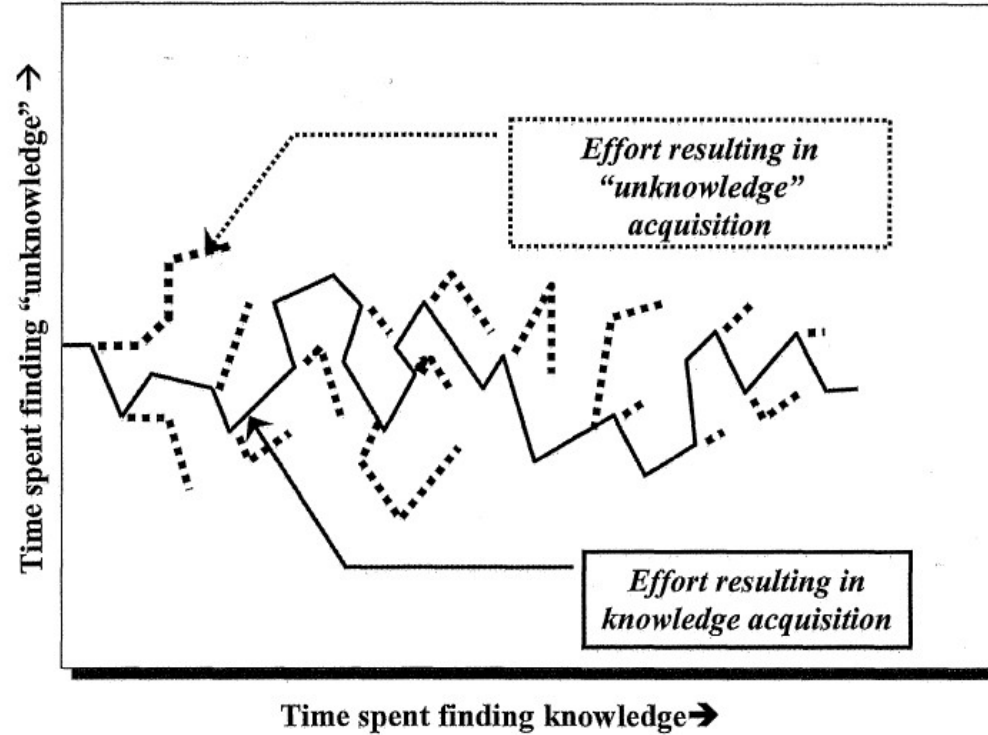# What the discovery process yields

Exhibit 4. Hacking project: Full build.

Labels in figure: Time spent finding "unknowledge" →, Time spent finding knowledge →, Effort resulting in "unknowledge" acquisition, Effort resulting in knowledge acquisition

The five orders of ignorance

◈ Zeroth Order: Lack of Ignorance
  ◇ I know something, I know that I know it, and I can demonstrate I know it

◈ First Order: Lack of Knowledge
  ◇ I don't know something, and I know I don't know it

◈ Second Order: Lack of Awareness
  ◇ I don't know something, and I don't know that I don't know it

◈ Third Order: Lack of Process
  ◇ I lack a process by which I can discover that I don't know that I don't know something

◈ Fourth Order: Meta Ignorance
  ◇ I don't know about the Five Orders of Ignorance

Applied to software development

- 0th Order: I know how to complete the system

- 1st Order: I know what I need to know to complete the system

- 2nd Order: I don't know yet what I will need to know to complete the system

- 3rd Order: I don't know how to discover what I need to know to complete the system

- 4th Order: I have no clue about any of the issues above
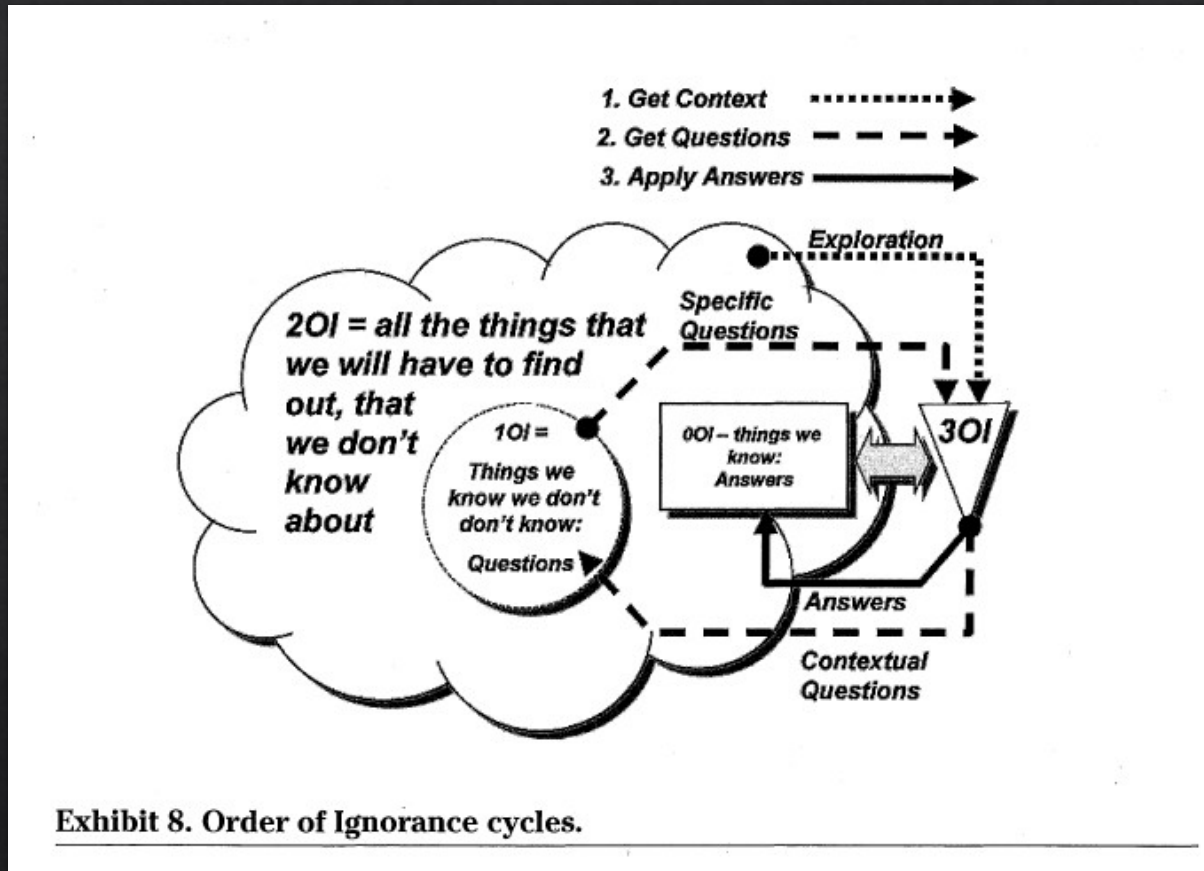
Order of Ignorance Cycles

Exhibit 8. Order of Ignorance cycles.

Real-world challenges

◈ "…the acquiring of knowledge is neither the business expectation nor the business goal in most companies. Few companies, even those that create and sell software only, count knowledge acquisition and management as their highest priority. Most operate on a modified manufacturing model that views the creation and delivery of the system to the customer as the highest priority. **It is not**, and this prevailing view has caused considerable problems to both customers and developers for decades." (p.235, emphasis added)

◈ "Our inability to actually measure knowledge means that much of our metric process is built on a foundation of sand. Compounding this is the fact that the critical measure of knowledge in software is not the measure of knowledge in software; it is the measure of the knowledge that is notin the software. This is the knowledge we have to get, not the knowledge we already have. As described earlier, **the key determinant of a software project is the 20I, which is knowledge we do not know we do not know.** So we are in a double bind. Not only can we not measure knowledge we have, what we really want to measure is knowledge we do not have." (p. 242, emphasis added)

Nature of the process

◈ To identify whether there are areas where we have ignorance (need to acquire knowledge)

◈ To identify what questions we would need to ask to resolve ignorance in these areas

◈ To obtain the answers to these questions in a form that we can usefully integrate into the system

◈ Key problems

    ◈ Acquiring knowledge also illuminates more areas of lack of knowledge

    ◈ Humans have not found a way to empirically measure knowledge

    ◈ The critical measure of knowledge in software is that of the knowledge *not* in the software

◈ Observations/thoughts?