An Empirical Approach to a General Theory of Software (Engineering)

Mathias Ekstedt
KTH Royal Institute of Technology
• Main message: If we are just a little bit pragmatic building a GTSE is feasible.
  - Predictive theory
  - If it is to be a collaborative effort, a common formalism needs to be chosen

• Agenda
  - Ideas on how to build a GTSE
  - An example: A theory for security of operational software systems
Theory Properties

- Universality
- Internal precision
- External precision
- Formalization
- Corroboration

(Johnson 2012, GTSE-1)
Theory types

• Using Gregor’s categorization
  - “what is”
  - “how” and “why” - post fact explanation
  - “what will be” – prediction

• Prediction is what makes the famous theories from other disciplines so useful
A GTSE requirements spec

• We can't have everything at once...

• My prioritization
  - Universality - High
  - Internal precision - Low
  - External precision - Low
  - Formalization – Low (High)
  - Corroboration – Medium

  - “What will be” theory

• Iterative development
Approaches

• “Top-down”
  - Start big and ambitious in scope
  - Refine to get more detail and precision
  - Practical and empirical

• “Bottom-up” (puzzle-building)
  - Devise or reuse/adapt sub-theories (puzzle pieces)
  - Integrate

• I will argue for a middle way...
A middle-way approach

- Scope the theory
- Find Causality
- Iterate, Integrate and Keep Consistent
Scope the theory

• Software Engineering Success!?

• Success definition/dimensions
  - Project on time
  - Project on budget
  - According to requirements spec
    • Functionality
    • Non-functionality
      – Performance
      – Availability
      – Flexibility
      – Security
      – ...

Find Causality

• So, which are the factors impacting [modifiability]?

• Set up hypotheses between independently defined variables, and test.
  - Also quite abstract phenomena can be operationalized (and thus tested)

• Prediction precision is probably low – uncertainty in relations
  - Many other things are also impacting

• We don’t necessarily need to know why causality exist
  - Complexity is troublesome for our limited heads
Iterate, Integrate and Keep Consistent

• Theory refinement – reduce “Other” factors and dig deeper causality

• Integration – the moment of bottom-up

- Choice of theory encoding formalism becomes key when integrating theories
- Without a unified theory encoding formalism - no consistency and no general theory
The Example

• The Cyber Security Modeling Language (CySeMoL)
  - A domain-specific language implemented using:
    • Probabilistic Relational Models
      (Class+Object models + Bayesian Networks)
    • Defense/Attack graphs

  - A Fairly General Theory of Software System Security!
Defense graphs

- Attack steps (likelihood to succeed)
- Countermeasures (likelihood they are working properly)
- Their dependencies (conditional probabilities)
CySeMoL scope
CySeMoL full version

Including attacks and countermeasures

Defense graphs are generated by the instantiation of the class diagram
The prediction capability of the theory

- How difficult is it to succeed with different attack steps?
- We have gathered data from
  - Previous research studies
  - Asked “security experts”
  - Logical necessities
E.g.:
Connect to the service

| The attacker can obtain access to a host allowed through the firewall | NO | NO | NO | NO | NO | NO | NO | NO |
| The attacker can obtain physical access to the network | NO | NO | NO | NO | NO | NO | NO | NO |
| Network logs are reviewed on a regular basis | YES | YES | YES | YES | NO | NO | NO | NO |
| Security audits are performed on the network on a regular basis | YES | YES | NO | NO | YES | YES | NO | NO |
| Administrators have defined a formal change management process | YES | NO | YES | NO | YES | NO | YES | NO |
| Low estimate (5 %) | 18 | 22 | 24 | 26 | 24 | 25 | 31 | 33 |
| Medium estimate (50%) | 44 | 44 | 47 | 46 | 45 | 46 | 51 | 55 |
| High estimate (95%) | 67 | 67 | 67 | 67 | 66 | 67 | 70 | 75 |
| Mean | 43 | 44 | 46 | 46 | 45 | 46 | 51 | 54 |
The full scenario

$1.00 \times 0.24 \times 1.00 \times 0.51 \times 1.00 = 0.1224 = 12\%$ chance of success
Characteristics of CySeMoL as theory

- Universality – High (for Security)
- Internal precision – Medium/Low
- External precision – Medium/Low (Explicit on uncertainty)
- Formalization – High
- Corroboration – Medium/Low
My conclusion

• It is possible to build a predictive theory of software (engineering):
  - Starting with a large scope
  - Include and take into account the uncertainties this brings (by using an appropriate formalism)
  - Using already existing knowledge/theories as an input
  - Refine and evolve it over time with more studies

  - It would be possible to do it collectively if the formalism and “sub-theory interfaces” are agreed upon.
Thanks/Questions