Extending a General Theory of Software to Engineering

Robert Lagerström, Mathias Ekstedt
KTH Royal Institute of Technology
Sweden
Main messages

Suggest a structure/formalism for developing a GTSE

Describe how our previous work could be a piece of a GTSE puzzle

Roughly outline an idea for future work on how to do a GTSE
- …extending to engineering
Preliminaries

Previous GTSE workshops discussions and conclusions:

A GTSE should be able to predict software project
-  Quality
-  Cost
-  Time
General assumptions of the SE context

Software Engineering

Time  Cost

Software

Quality
- Modifiability
- Reliability
- Performance
- Interoperability
- ...

KTH
VÄRDERAD GOD KOMIT


Determinants / Cause and effect

- Programmer competence
- Organizational structure
- Requirements elicitation competence
- Used tools
- ...

E.g.

- Programming language
- Coupling
- Cohesion
- Redundancy
- Used compiler
- ...

Software Engineering

Time  Cost  Quality
Determinants / Cause and effect

Simplified example

Cause and effect properties that we want to capture in the theory

- Strength and direction of impact
- Conditional dependencies
- Uncertainty
Our Theory: the Multi-Attribute Prediction (MAP) language

A formalism for predicting quality attributes of software system architectures.

Written in the Predictive, Probabilistic Architecture Modeling Framework (P2AMF), which is a combination of:
- UML
- OCL
- Monte-Carlo simulations
Multi-Attribute Prediction (MAP) language

Combining sub theories for:
- Service availability
- Application modifiability
- Data accuracy
- Application usage
- Interoperability
- Operational cost
- Requirements utility

In summary, an approximate high-level probabilistic theory
Service availability

Theory based on Fault Tree Analysis

(Some OCL excerpts: )
Application Service Modifiability

- Component Complexity (McCabe)
- Component Size (NLOC)
- Gearing factor (prog. lang.)
- Internal coupling
- External coupling

(some OCL excerpts: )

```ocaml
self.collaboration -> iterate( ai : ApplicationCollaboration ; max : Real=0 | 
  if ai.couplingInPair() > max 
  then ai.couplingInPair() 
  else max 
  endif 
) |
if self.collaboration -> notEmpty() then 
  (self.collaboration -> iterate( ai : ApplicationCollaboration ; sum : Real=0 | sum+ ai.couplingInPair() ) ) / (self.collaboration -> size()) 
else 
  0 
endif
```
The MAP language as a GTSE..?

- Multiple, but incomplete, qualities
- Simplified theory

Missing...

Software Engineering

Time

Cost

Software

Quality
- Modifiability
- Reliability
- Performance
- Interoperability
- ...
Suggested extension

Use SEMAT

Use MAP language

Software Engineering

Software
Challenges

Unify formalisms
- MAP (P2AMF)
  - Classes
  - Attributes
  - Attribute and structural dependencies
  - Probabilistic inference
- SEMAT
  - Alphas
  - Alpha states
  - Structural dependencies
Suggested future work

Implement SEMAT in P2AMF (UML/OCL)
- Extend SEMAT with attributes

Elaborate on properties for engineering focused Alphas
- Way of working
- Work
- Team

Incorporate existing sub-theories that complements SEMAT.
E.g. COCOMO and Brooks’ law.
COCOMO II

COCOMO II output:
- Person-months, calendar months, project cost

COCOMO II input → SEMAT extensions (as attributes, sub classes, and/or class relations):
- Team alpha
  - Team cohesion
  - Analyst capability
  - Programmer capability
  - Personnel continuity
  - Language and toolset experience
- Way-of-working alpha
  - Use of software tools
  - Process maturity
- Work alpha
  - % design modified
  - Development flexibility

Prediction mechanism/theory already explicit → re-coded in OCL
Brooks’ law

Brooks’ law output:
- Calendar months

Brooks’ law input → SEMAT extensions (as attributes, sub classes, and/or class relations):
- Team alpha
  - Size
  - Developer prior knowledge
- Work alpha
  - Parallelism
  - Sequential ordering
  - Coordination need

Prediction mechanism/theory not exact
→ need to be interpreted and coded in OCL
Idea in summary

Add various, preferably well-accepted, theories

Use SEMAT

Common formalism – P2AMF (UML/OCL)

Add various, preferably well-accepted, theories

Use MAP language

→ Quite a General Theory of Software Engineering
Thanks/Questions