ECE750-Topic11: Component-Based Software

Software Component Quality Models

Ladan Tahvildari
Assistant Professor
Dept. of Elect. & Comp. Eng.
University of Waterloo
Outline

- Introduction
- Measurement and Metrics for Software Components
- Multi-Criteria Evaluation
- Exploding the Myth of Component Evaluation
- Covering Some Research Papers
Introduction

❖ **Components:**
  - Independently deployed software implementations.

❖ **Assemblies:**
  - Aggregations of components that provide integrated behavior.

❖ **Selection:**
  - Choosing one component over another involves selection.

❖ **Evaluation:**
  - Formalized process of quantifying human judgment by assigning value to choices.
If…
- The quality of software components determines of the quality of the composed system,

Then…
- CBSE must provide techniques to reliably and repeatedly select high quality components.

And it follows from this that …
- Component evaluation is a distinguished CBSE activity, with distinguished workflows and techniques.
Evaluation Attributes

- select
  - vendor
    - health
    - reputation
  - component
    - function
    - usability

composed attribute

basic attribute

qualitative dependency
Evaluation Attributes

select

vendor
  - health (judgment)
  - reputation (judgment)

component
  - usability (usability index)
  - functionality (#menu items)

- non-repeatable judgment
- essential judgment
Genus: Preference Structure-Based Evaluation

- **A preference structure**
  - This is the model of the decision.
  - A preference structure emerges when we express preference relations in terms of attributes.

- **An aggregation technique**
  - This is the tool that generates interpretations of the model.
Preference Relation

- **P(x, y), strict preference:**
  - States that x is strictly preferred to y.

- **I(x, y), indifference:**
  - States that neither x nor y is preferred.

- **R(x, y), incomparability:**
  - States that x and y are incomparable.

- For example, we might define a preference relation:
  \[ S(x, y) = P(x, y) \lor I(x, y) \]
Species: Multi-attribute Utility Evaluation

- The species can be seen through its formulaic expression, in which each evaluation attribute $g_k \in G$ is defined as the triple:

$$<w_k, u_k, g_k>$$

$$U_x = \sum w_k \cdot u_k(g_k(x))$$

- $U_x$ denotes the **overall utility of component x**

- $u_k$ denotes **a transform function** that maps the scale of attribute measure $g_k$ to a universal utility scale $u_k$

- $w_k$ denotes **the substitution rate for $g_k$**
The preference structure most frequently associated with multi-attribute utility is:

\[ S(x, y, g) = P(x, y, g) \cup I(x, y, g) \]

\[ P(x, y, g) \succ U_x > U_y \]

\[ I(x, y, g) \equiv U_x = U_y \]

Which states that \( x \) is preferred to \( y \) if it has a higher utility, and \( x \) and \( y \) are indifferent if they have the same utility.
Simple Utility Transform

Functions Usually

- $u(\text{usability})$
- $u(\text{functionality})$

Graphs showing:
- utility vs. usability index
- utility vs. number of menu items
Exploding the Myth of Component Evaluation

- An Assembly:
  - Reflects the convenience in representing the composition of commercial components as systems, subsystems, sub-subsystems, and so forth.
  - The scope of a system, or its relative position in a hierarchy of systems is not material to what follows, we will use the term ‘assembly’ in place of ‘system’.

- That is, commercial components are assembled into assemblies.
Assemblies and Components

- Assemblies, once they exist, and commercial components will exhibit a variety of properties:
  - functionality
  - reliability
  - usability
  - and so forth

- The properties of an assembly are determined, in some way, by the properties of the components themselves
Assembly Properties

Assembly Properties Determined by Component Properties

\[ P_A = \mathcal{D}(P_1, P_2) \]
Satisfaction of Normative Abstract Interface

\[ P_A = D(P_{E1}, P_{E2}) \]
Accommodating Variance

\[ P_A = \mathcal{D}(P_{E1} - P_1, P_{E2} - P_2) \]
The Inevitability of Hidden Properties

\[ P_A = \mathcal{D}(P_{H1}, P_{H2}, P_{E1} - P_1, P_{E2} - P_2) \]

\[ P_{E1} \cap P_{H1} = \emptyset \]