Software Architecture Viewpoint - Analysis & Visualisation

CS4099 INTERIM DEMONSTRATION

YUXUAN WANG

[Final Report] Context Survey (background)

- Initiation: the importance of software architectural design
  - System design documentation (Introduction of component & connector)
  - Stakeholder communication encouragement (Introduction of viewpoint)
- Architecture viewpoint: perspective
  - Restrict the amount of information within the concerns of target stakeholder(s)
  - View: viewpoint instance with a typical concern (e.g. logical, process, development, physical)
- Architect consideration (from Ian Sommerville)
  - What views are useful when designing typical systems (What is a well-designed architecture?)
  - What notations should be used when designing typical systems (How should architecture be presented?)
  - IMO: a set of architectural design decisions to provide the ideal performance, robustness, etc.

[Final Report] Context Survey (problem)

- Multiple views lead to multiplied workloads (DOER)
  - Scratching (static) vs. Professional tools (dynamic)

- Multiple viewpoints & connections (DOER)
  - The relationship between view, view connection, component & connector

- Concurrency (new idea)
  - How to reveal the constraints/concurrency problems between views/components...

[Documentation] DOER Update

2.1 **Primary objectives**: `SA = { Elements, Forms, Rationale/Constraints }`

- [x] 1. Implement an architecture component design tool that provides predefined components for user to build customised architecture components. (progress: single-level architecture component)
- [x] 2. Support hierarchical features on architecture components. (progress: hierarchical architecture component)
- [x] 3. Provide visualisation on architecture components.
- [x] 4. Implement an architecture connector design tool that provides interactions among architectural components. (revealing the relationship by component definitions and supporting manually connection) (progress: hierarchical architecture component with intra-components connector)
- [ ] 5. Support hierarchical features on connectors. (progress: a single architecture view - hierarchical architecture component with inter-components connector)
- [x] 6. Provide visualisation on connectors.
- [x] 7. Implement a design tool that is able to edit and connect multiple views. (progress: composite views with view-view connections)
- [ ] 8. Support intra-components and inter-components connection. (progress: customised model: composite views with hierarchical view connections)
- [ ] 9. Based on connections, the tool is able to reveal constraints between different components and views. (progress: core function)
- [ ] 10. Conduct a user evaluation on core design tool, evaluate the design functions and the visualisation design.

5 & 8: connector still debugging

10. Start once connector works as expected

9. Concurrency & constraint

- Inconsistency: views assert design decisions that cannot be simultaneously true

Direct Inconsistencies. These inconsistencies occur when two views assert directly contradictory propositions such as "the system runs on two hosts" and "the system runs on three hosts." These inconsistencies can often be detected by automatic mechanisms that employ appropriate constraints and rules.

Refinement Inconsistencies. These inconsistencies occur when two views of the same system at different levels of detail assert contradictory propositions. For example, a "top level" structural view contains a component that is absent from a structural view that includes subarchitectures. These inconsistencies can also be automatically detected with appropriate consistency rules, provided both the top-level and refined views contain enough information to understand the relationship between the two.

Static Aspects versus Dynamic Aspects. In this inconsistency, a view of a static aspect of a system conflicts with a view of a dynamic aspect. For example, a message sequence chart view might depict the handling of messages by a component that is not contained in the structural view. These inconsistencies can be somewhat harder to automatically detect, depending on how explicit the dynamic aspect's specification is.

Dynamic Aspects. In this inconsistency, two views of dynamic aspects of the system conflict. For example, a message sequence chart depicts a specific interaction between components that is not allowed by the behavioral specifications contained in those components' statecharts. These inconsistencies are often extremely difficult to detect automatically because it would require extensive state exploration or simulations.

8.1.2 Consistency

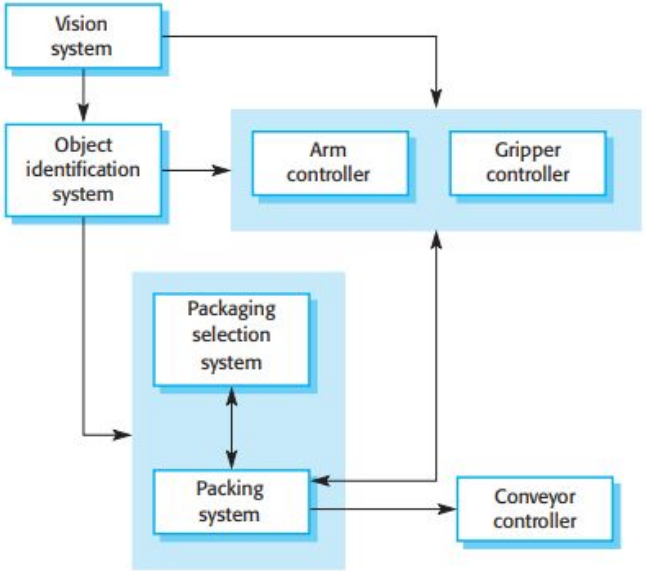
Consistency is an internal property of an architectural model, which is intended to ensure that different elements of that model do not contradict one another. The need for consistency derives from the fact that software systems, and thus their architectural models, are complex and multifaceted. As a result, even if no architectural design decisions are invalidated during the architectural design process, capturing the details of those decisions during architecture modeling may result in many inadvertently introduced inconsistencies. Examples of inconsistencies in a model include the following.

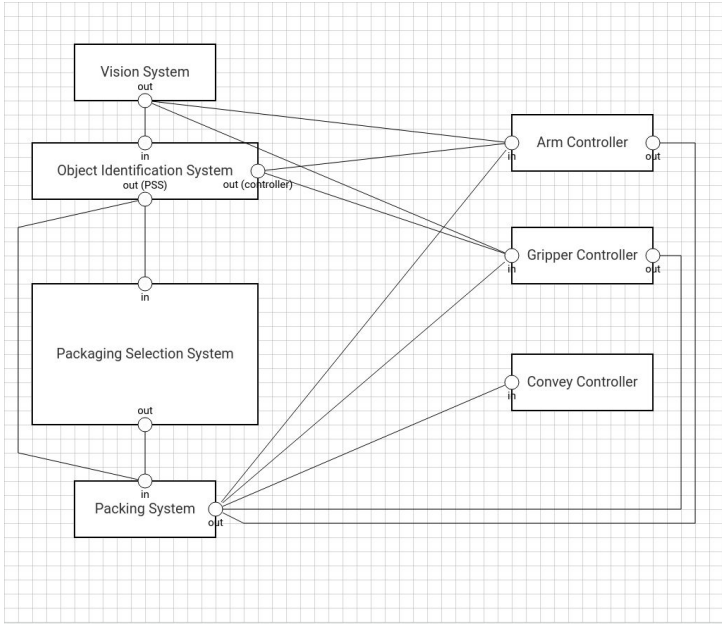
- Name inconsistencies.
- Interface inconsistencies.
- Behavioral inconsistencies.
- Interaction inconsistencies.
- Refinement inconsistencies.

[Artifact] Design decisions

- Web application
  - Client-side: Vuetify (Frontend framework), JointJS (Workbench)
  - Server-side: Node.JS (Server construction)
  - Communication: AXIOS (RESTful)
  - Data representation: HTML shapes <-> Client-side JS object <-> Server-side JSON file
- Architecture composition
  - Top-level: viewpoint(?) & connection (e.g. 4+1)
  - Hierarchical-level: component & connector

[Artifact] Demonstration





This semester

- Debug connector issues (5)
- Conduct user evaluation (10)
- Work on concurrency & multiple view connection issues (9)
- Secondary & Tertiary objectives + Final report (...)

Documentation

Github & DOER: https://github.com/El15ande/CS4099_SHProject

Design form:
https://docs.google.com/spreadsheets/d/17t339HamR7QzjQzV_GF6dRHW-bJkCdzY8Lhw6yHwgZk/edit?usp=sharing

Final report:
https://docs.google.com/document/d/10sQ_gjIIORgNqmJ4ZFjDqs-XWodzEVgzkOk5H3DjjvM/edit?usp=sharing