Categorizing and Modeling Variation in Families of Systems

Borislava I. Simidchieva, Leon J. Osterweil
Laboratory for Advanced Software Engineering Research
University of Massachusetts, Amherst
Amherst, MA 01003, USA
Introduction

- A single software product can rarely model all the variability indicated by varying requirements
  - Instead, a product line or software family is needed
- Variability should be modeled explicitly
- Variation currently not carefully categorized
  - May be beneficial to consider and model different variation relations separately, in a formal way
  - Such a taxonomy of different variation relations could lead to better guidance for accommodation
Explicit modeling of different variation relations may enable and facilitate:

1. **Analysis** of an entire software family at once
   - to prove safety and correctness properties about all software variants
2. **Generation** of new variants
   - based on defined variation relations and known requirements and architecture specification
3. **Navigation** among interrelated software families
   - to identify which variant to use in specific circumstances
Proposed Variation Taxonomy

- Functional Detail Variation
- Robustness Variation
- Performance Variation
- Service Variation
- Interaction-Based Variation
- Functional Invariance
- Goal Invariance
- Others...
Functional Detail Variation

Meaning:
- Variants differ in the amount of detail with which different functional capabilities are specified

Example:
- High-end OS variant may provide easy, elaborate built-in remote access
- Lower-end variant might only provide rudimentary functionality for remote access or less guidance
Performance Variation

Meaning:

- Variants provide the same functionality, but differ in the speed with which they execute.

Example:

- 64-bit OS variants typically offer great performance gains for native 64-bit applications and for computation-intensive, memory-hungry tasks.
- 32-bit variants offer the same functionality but often execute slower under such circumstances.
Different Variation Relations: Implementation & Management Concerns

- Functional detail variants might reasonably *share a common high-level architecture*
- Performance variants might not
- Understanding the variation relation within a family may facilitate understanding the relationship *between* families
- Interactions between different variation relation families may affect and influence variation management and implementation
Stakeholders’ Concerns with Respect to Variability

Question 1: *In your work, what are the main stakeholders and their concerns with respect to variability?*

- Designers
- Developers
- Customers
- Users
Variability with Respect to Architecture Models

Question 2: *With respect to which architectural models do you consider variability* (e.g. w.r.t. component and connector models, deployment models, etc.)?

- Requirements specification
- Component and connector model
- Architecture description model (process modeling)
Question 3: How do you integrate variability into a view-based architecture description? (e.g. variability in a view separated from the other views? Variability as a separate model in existing views? Variability integrated in an existing architectural model? Design vs run-time representation of variability?)

- Variability is considered as a first-class object, in a view separate from existing architecture models
Related Work

- **SPLE application and management** (Clements & Northrop; Pohl & Metzger; Schmid & van der Linden)
- **Architecture variation modeling** (Bachmann & Bass; Gomaa)
- **Feature models and diagrams** (Atkinson et al; Kang et al; Schobbens et al; Weiss & Lai)
- **Integrated lifecycle approaches** (Apel et al; Sinnema et al; Trujillo et al; van Ommering et al)
- **Generation and implementation approaches** (Apel et al; Batory; Batory & O’Malley; Czarnecki & Eisenecker; Kaestner et al; Kiczales et al; Knauber; Smaragdakis & Batory)
Future Work

- Do these dimensions afford for observed variation?
- How can families based on different variation relations be composed together safely?
- How would composition and intersection affect reasoning?
- How does process variation differ from product variation?
- What kind of tool support would make such a conceptual framework useful?
Conclusion

- Variability is inherent in real-world systems
- Careful treatment of variation can lead to a taxonomy of different dimensions of variation
- Establishing a disciplined way to model these different dimensions explicitly has many benefits
- If observed variation can be categorized, it may be easier to accommodate and manage
Thank you!