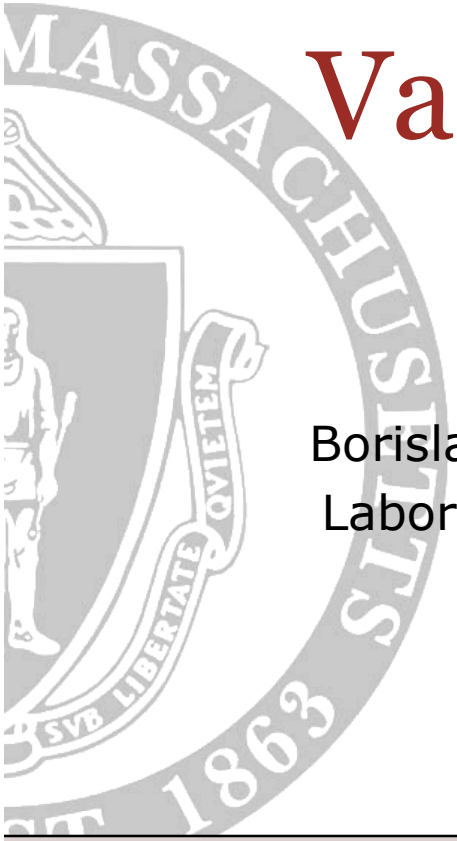
UMassAmherst

Representing Process Variation by Means of a Process Family

Borislava I. Simidchieva, Leon J. Osterweil, Lori A. Clarke
Laboratory for Advanced Software Engineering Research
University of Massachusetts, Amherst



Motivation

- Interested in creating formal process definitions to model real world processes
- Variation is inherent in real world processes
- Need to consider what constructs are needed to capture variation
- Abundance of work in variation in software but not processes
  - Should similar approaches be applied to process languages and process development environments?

Issues to be Addressed

We need:

- To accommodate variation in actual processes
- To model variation effectively
- To communicate the process and its variants in a clear and concise way
- To be able to recognize variation during elicitation and modeling

Conjecture

- A suitable process modeling language and process development environment should be able to accommodate variation and model it effectively
  - Certain language features or tools within the development environment can address certain kinds of variation

Some Informal Definitions

- Variation: a difference in the way two real-world processes handle an identical or almost identical task
- Variant: a specific process, which implements specific variations
- Process family: a collection of variants

Experimental Approach

- Consider actual real-world processes in which variation is apparent
- Develop techniques to accommodate the variation and model it effectively as a collection of variants
- Evaluate how well the techniques represent the variation

Process Modeling Language

We use Little-JIL to model variation by creating variants because it provides:

- Separation of concerns
  - Coordination specification
  - Agents
  - Artifacts
- Visual representation
  - Hierarchical decomposition of steps
- An experimental platform that allows modifying the language to include more features for creating variants

Techniques for Generating Variants

- Reusable components from each of the three parts of a process definition—coordination diagram, agent specification, artifact specification
- Combining different components based on user specifications will generate new processes
  - Step elaboration
  - Agent behavior
  - Artifact structure

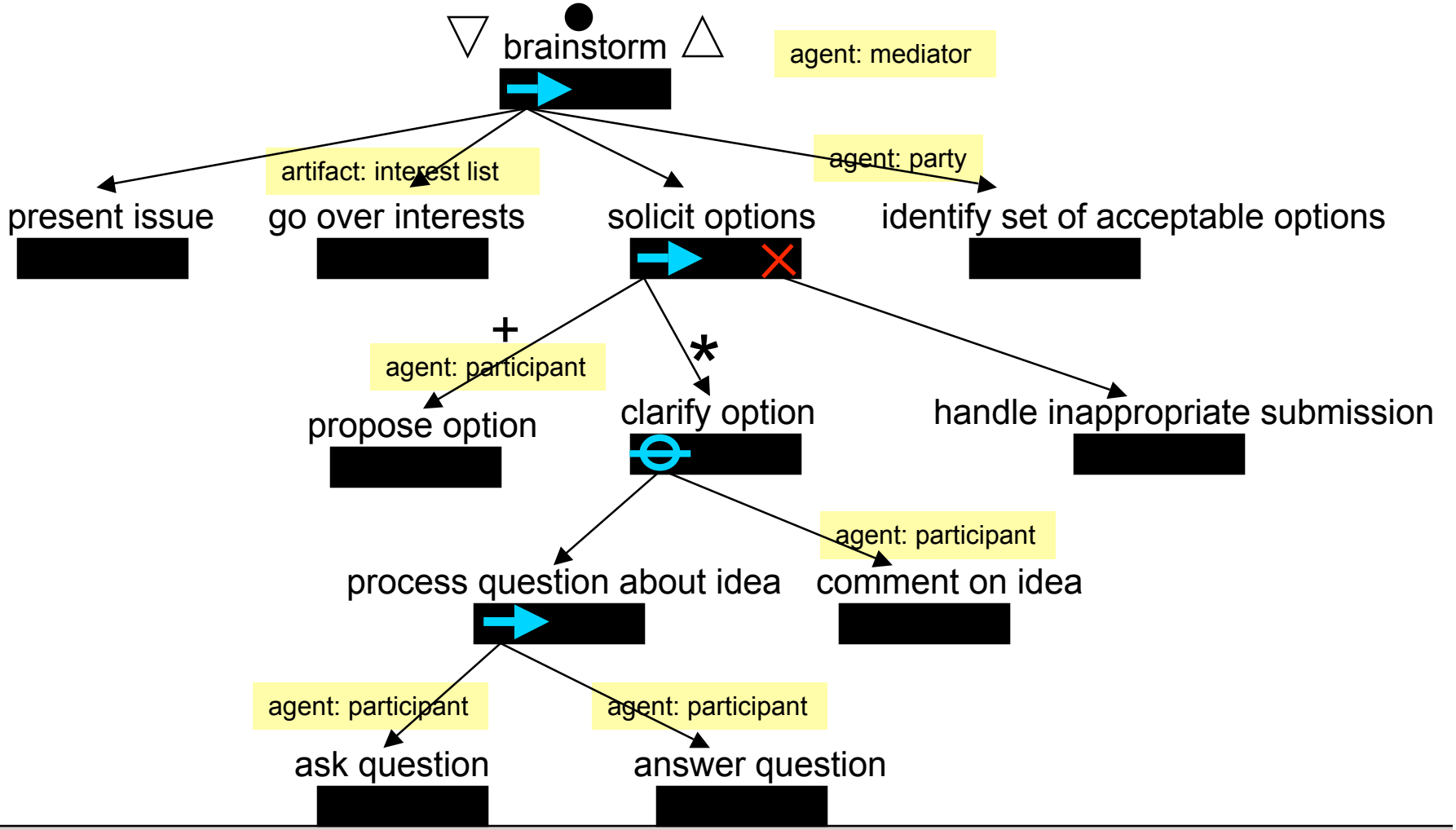
Case Study

- Collaboration with the National Mediation Board
- Goal: precisely model the process NMB uses to guide mediation
  - would enable formal analysis and potential improvements
- Problem: as we try to accurately model the process in sufficient detail, variations are discovered
  - these variants are closely related
  - vary depending on mediator preferences or style, group dynamics, and situational concerns that arise
  - mediators may wish to adapt processes dynamically
  - may not be able to capture these variations with a single process

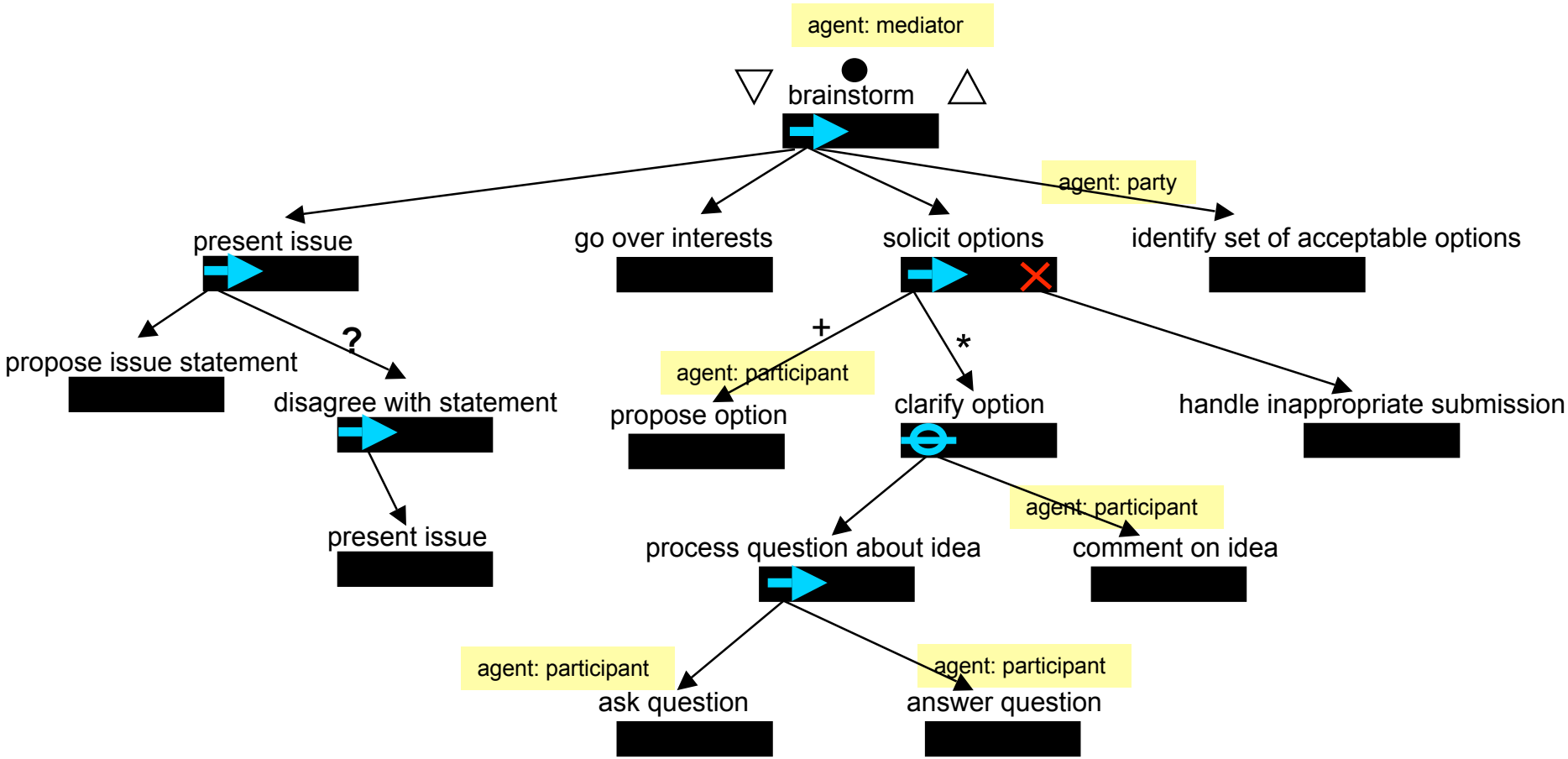
Case Study: Examples of Variations Encountered

- Restrictions on how part of the process should be executed
- Anonymity control
- Differences in mediator style

Little-JIL Features Overview

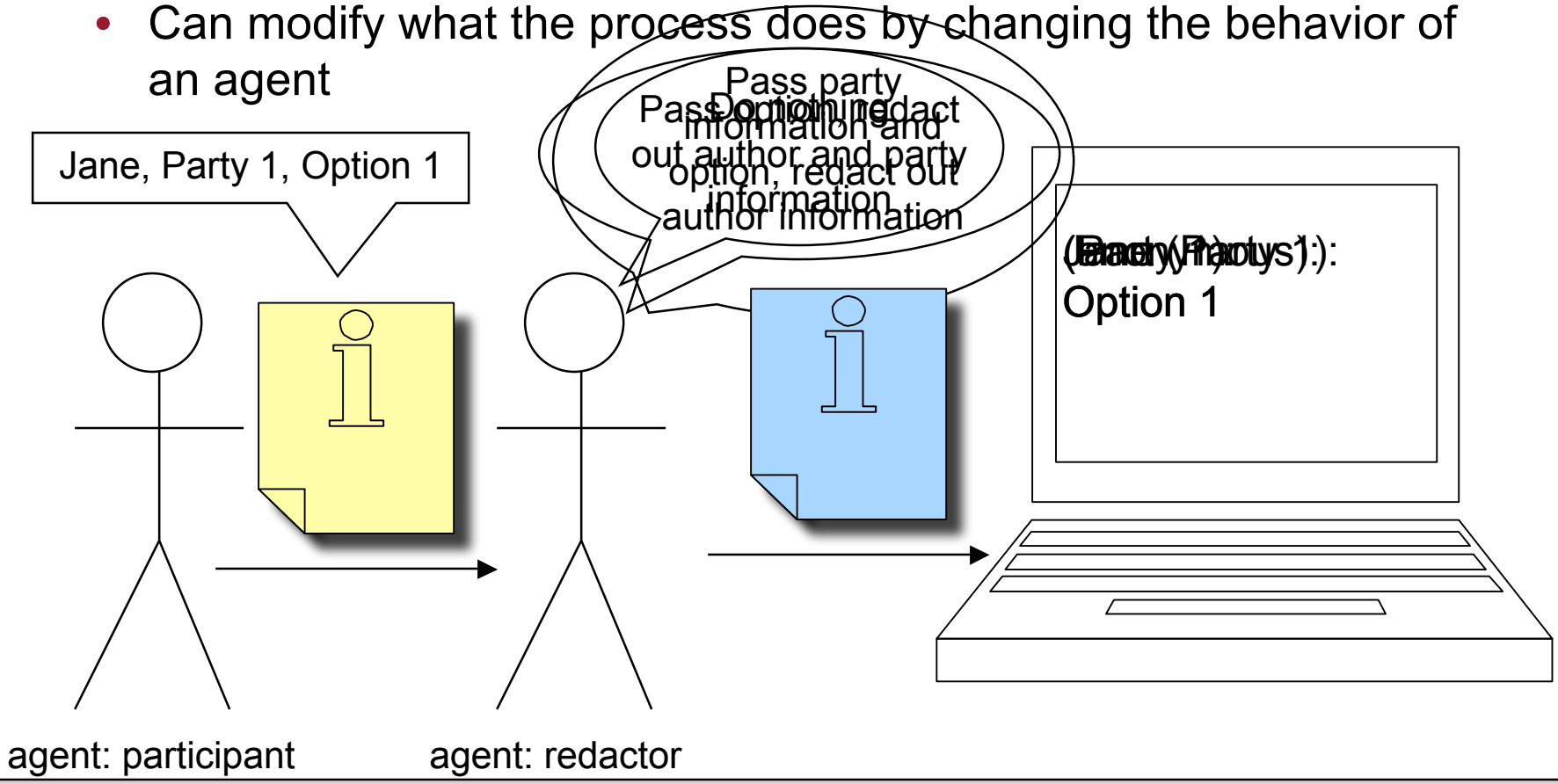


Creating Variants by Step Elaboration in Coordination Diagram



Creating Variants by Modifying Agent Behavior

- Anonymity example
  - Can modify what the process does by changing the behavior of an agent



Creating Variants by Modifying Artifact Structure

- "Interest list" example:
  - NMB specializes in interest-based bargaining
  - Before the brainstorming begins, parties have to identify their interests
  - Mediators keep a list of interests
    - divided by party
    - together

Interest List	
Party 1	Party 2
Interest 1	Interest 2
Interest 2	Interest 3
Interest 3	Interest 4
Interest 4	Interest 5
Interest 5	Interest 6
Interest 6	

Observations

Little-JIL provides some support for modeling process variation

- Outlined variant generation techniques seem to span most of the variations we have encountered
- Applying more than one technique could produce even larger families
- Can easily fine-tune large processes and switch components depending on context
- Visual representation provides a way to communicate about variants within the family

Related Work

- Software families, product lines, and variation
  - Variability realization techniques (e.g. Svahnberg et al)
  - Variability (software reuse on a common core) via inheritance, parameterization, extension points (e.g. Jacobson et al)
  - Conditional compilation/dynamic binding (e.g. Gacek et al)
- Generation approaches
  - Using component generators to support dynamically configurable components (e.g. Pavel et al)
  - Extending UML models with decision models (KobrA); each component associated with structural, behavioral and functional model, as well as a decision model (Atkinson et al)
- Collaboration and group support systems
  - Some allow implicit variation by providing dynamic configuration options, but no explicit support for variants (e.g. Facilitate.com)
  - Groupware construction tools enable the creation of process instances via explicit coding (e.g. LotusNotes)

Future Work

- Consider other processes where different kinds of variation might occur to identify additional variant generation techniques
  - E.g. variations in the medical domain
- Consider techniques from software product line research and how they may be relevant to processes

Questions?