Domain globalization

- What is a DSL?
- Questions
- Case study

Chapter outline

- 1. Context
- 2. State of Art
- 3. Open Challenges
  - Integration of multiple DSLs
    - How do we integrate languages?
      - ⚠ Do we compose at level M2 or M3? — ie do we compose models or languages?
      - What are suitable "composition operators"?
      - What does it mean for languages to be composable ? and how do you check that ? — Are there some languages that are so specific that they cannot be composed ?
    - Can we view languages as components? If so, what are their interfaces?
      - Events can be fine for some purpose but not for all so is there a finite list of let's say suitable things ?
      - Kienzle Interfaces
        - usage
        - customization — eg inheritances/subclassing
        - variation — eg feature models
      - ❓ If a language is a component, what is its interface?
        - ie each sentence specifies a model
        - ie language <=> meta-model
      - Language specifies a set of models
        - **a language interface is a characterization of a subset of the models suitable for integration** — eg functions, values, statements, objects ...
        - ❓ Is there a difference between a language interface and a model interface? — language = models (semantics) + syntax + tools
      - Is it possible to generate/extract a well-founded interface of a model?
    - What are the semantics of a composed model?
      - What kind of meta-languages can describe the individual DSL semantics ?
        - For simulation it could be easier to encode SOS
        - is there difference between causal (eg assignment) or acausal (eg constraints)?
        - For verification it becomes difficult to capture any efficient theory.
      - Identify classes of DSLs whose composition can be automated
        - eg Architectural DSLs?
        - eg behavioural DSLs (timed automata, markov chains etc etc)
        - Composition of acausal models (BCa)
      - How can we support compositional reasoning across DSLs?
      - Is there a "unique"/"unified" semantics for my system ? so that I can denote the semantics of each DSLs in ?
        - How can we avoid constructing a complex unified semantics?
        - Do we need a common semantic domain to check "compatibility" (even just based on artifact interfaces)?
          - GK: they are using formula as a common base semantics for the interfaces and their compatibility. It seems it is a need ? what is the more abstract but suitable language for that ?
          - BC: traces can be used as a common base semantics
        - How do we check compatibility and consistency?
          - For Group 1!
          - What is the difference between compatibility and consistency?
            - *Components are compatible if there exists an environment in which they can work together.*
              - a precondition?
              - a postcondition?
            - *Models are consistent if there exists a common implementation.*
              - A model is an instance of a DSL, ie a sentence
              - A model is a specification? ie a requirement?
    - How can we exploit legacy tools in an integrated system?
  - Overall consistency along lifecycle of DSL instances (CB)
  - Multi-view modeling shared by multiple stakeholders (CB/ES)
  - Large scale model management (ES)

Breakout group - Dagstuhl DSL Globalization
Cedric Brun, Benoit Caillaud, Julien Deantoni, Robert France, Gabor Karsai, Oscar
Nierstrasz, Eugene Syriani

http://link.springer.com/chapter/10.1007/978-3-642-41533-3_37