Diagramatic Software Specification

Adrian Rutle, Yngve Lamo (HiB)
Uwe Wolter (UiB)
Norway
Introduction

- Need for a common language for
  - Designers
  - Programmers
  - Experts of the universe
  - ...

- In order to:
  - reason about the problem
  - verify business logic

- Graphical language is well suited for this task
Why a graphical language?

- Simple
  - Easier to understand compared to textual languages
- Universal
  - Different background → same understanding
What’s a good modelling language?

- Graph-based,
- has a formal interpretation and
- sufficiently expressive (to capture all the peculiarities of the universe)

Proposal: GENERALIZED SKETCHES
State-of-the-art

For specifications in software engineering:

- Many graphical modeling languages
- Few of them have proper semantics

→ Specifications that are difficult to maintain due to:
  - ambiguous constructions
  - semantic relativism
Our contribution to the field

• Use of Generalized Sketches (GS) to:
  • compare
  • unify
  • integrate
diagrammatic specification languages

• Achieve this by:
  • Mapping different modeling languages to the GS format
  • Superimposing signatures from different graphical notations

• Graphical notations are parameterized by signatures
• *What* should be modeled must be standardized
• BNF is an analogy:
  a formal notation to describe the syntax of a given language
Generalized Sketches: history

- Ehresmann's sketches: graphical presentations of categories (1968)
- Diskin (1997) made GS:
  - more direct
  - more applicable to SE
  - complex diagram operations (operational ske.)
Formal definitions

- **Graph:** \( G = (G_0, G_1, \text{src}^G, \text{trg}^G) \)
- **Graph homomorphism**
  \[ h : G \rightarrow H \]
  \[ h_0 : G_0 \rightarrow H_0 \]
  \[ h_1 : G_1 \rightarrow H_1 \]
  \[ \forall e \in G_1 : \]
  \[ \text{src}^H(h_1(e)) = h_0(\text{src}^G(e)) \text{ and} \]
  \[ \text{trg}^H(h_1(e)) = h_0(\text{trg}^G(e)) \]
- **Diagram:** \( \delta : \text{Shape} \rightarrow G \)
- **Signature:** \( \sum = \langle P, \text{Arity}(P) \rangle \)
- **Sketch:** \( \langle \sum \rangle \text{sketch } S = \langle G_S, D_S(P) \rangle \)
  where \( \{ \delta : \text{Arity}(P) \rightarrow G_S \} \in D_S(P) \)
A sketch morphism $\mu : S_1 \rightarrow S_2$ is a graph homomorphism $\mu : G_{S_1} \rightarrow G_{S_2}$ such that $\forall \delta : \text{Arity}(P) \rightarrow G_{S_1} \in D_{S_1}$.

$\Rightarrow \delta; \mu : \text{Arity}(P) \rightarrow G_{S_2} \in D_{S_2}$
\[ \Sigma_{UML} = ( P, \text{Arity}(P) ) \]

Concept name

Value type: [String], [Int]

Total, cover and single valued partial and single valued total and single valued inclusion

\[ S = ( G_S, D_S(P) ) \]

\( G_S \): carrier graph, (d:Arity(P) \rightarrow G_S) in D_S(P)
\[ \Sigma_{UML} = ( P, \text{Arity}(P) ) \]

concept

value

total, cover and single valued

partial and single valued

inclusion

[1-1]

[cover]

[inv]

[disj]

\[ S = ( G_S, D_S(P) ) \]

\[ G_S : \text{carrier graph}, \]

\( (d : \text{Arity}(P) \rightarrow G_S) \) in \( D_S(P) \)
\[ \Sigma_{ER} = (P, \text{Arity}(P)) \]

Concept name: \textit{r}

Value:

Partial mapping

Total mapping

Inclusion

Jointly mono:
Binary relation
Ternary relation...

Disjoint union:
(inclusion)

Equivalent ways for expressing binary relations.
Categorically: product

Equivalent ways for expressing inclusion.
Categorically: co-product
\[ \Sigma_{ER} = (P, \text{Arity}(P)) \]

Inclusion

Partial mapping

Total mapping

Jointly mono:

Binary relation

Ternary relation ...

Jointly epi,

Disjoint union:

(inclusion)

\[ \sum_{ER} \text{-sketch } U = (G_U, D_U(P)) \]

\[ \exists P \in \sum_{ER} : D_U(P) = \{d : \text{Arity}(P) \rightarrow G_U\} \]

Semantic sketch of U

\[ \sum_{ER} \text{-sketch } U = (G_U, D_U(P)) \]

\[ \forall P \in \sum_{ER} : D_U(P) = \{\delta : \text{Arity}(P) \rightarrow G_U\} \]
\[ \forall P \in \Sigma_{ER} : \text{Arity}(P) \subseteq \text{Arity}(\sigma(P)) \]
Our objectives (theory)

- Formalization of:
  - ER-diagrams,
  - UML-diagrams,
  - DB-schemes and
  - Ontology languages,
  by GS

- Investigation of the
  - integration,
  - combination and
  - modularization

  of specifications within single specification formalism
  based on GS
Our objectives (technical)

• Design and development of tools supporting the application of GS in the field of software engineering:
  • drawing UML and ER diagrams based on GS formalism
  • support for mappings/translations between the two types of diagrams by mapping their signatures
  • code-generator
    • programming languages
    • specification language
      based on their graphical specifications
  • case studies to evaluate the theory in practice.
Summary

ER-Diagram

Syntax

UML-Diagram

Formal semantics
(construct signatures)

Semantic universe
Category SET

Integration of formalisms
(signatures)

\[ \Sigma_{ER} \rightarrow \sigma \rightarrow \Sigma_{UML} \]

induces

\[ \mu \]

Semantic sketch
\( \Sigma_{ER}\)-sketch U

Semantic sketch
\( \Sigma_{UML}\)-sketch U
Thank you!
What’s done and progress:

- Diskin’s program
- Ørjan Hatlands Contribution
- New Master-thesis by Stian Skjerveggen
Facts

Sketch-able spec. ↔ Formal spec.
Diskin and Kadish:

- Graphical specifications (ER diagram or UML class diagram,) can be considered as:
  - abbreviations or
  - visualizations

  of sketches for a fixed signature
  where the signature is the corresponding diagram type
Diskin and Kadish:

- A graphical notation: a **visualization** on the top of a **specification** core
  - Visualization: a presentation (a user interface) of specification,
  - Specification ”deals with” semantics of the notational constructs