A Study on User-Friendly Formal Specification Languages for Requirements Formalization

Cheng Pang, Antti Pakonen, Igor Buzhinsky, Valeriy Vyatkin

igor.buzhinskii@aalto.fi

Aalto University
School of Electrical Engineering

ITMO UNIVERSITY

VTT

LULEÅ UNIVERSITY OF TECHNOLOGY

INDIN 2016, Futuroscope-Poitiers, France

July 19, 2016
Motivation

Formal specifications – why?

- They are unambiguous and precise
- In some cases, they can be validated and checked for consistency
- In some cases, they can be checked by formal methods such as model checking
- These benefits are especially important for mission-critical systems, for which testing and simulation may not be sufficient

Open issues:

1. Industry practitioners commonly lack knowledge of formal methods and related training – one of the main reasons why formal methods are not yet popular in industry
2. The problem of safety demonstration for stakeholders and authorities – requires specifications to be represented intuitively
3. Focus on requirements, not techniques to formalize them
Formal specification languages

- Formal specification language (FSL): **unambiguous and precise** requirement representation technique
- Mathematical equations? – Yes
- Temporal logics? – Yes
- Product requirements document? – No

- This work investigates how to make FSLs more **user-friendly**
Templates and boilerplates

- Pre-formatted textual representations of semi-formal requirements
- Used in the CESAR framework
- Boilerplate example: **C-BP16**: `<system> shall <action>`
  `<system>` and `<action>` are **placeholders**, `shall` is a **keyword**
- Simpler templates and boilerplates can be further combined
- Tools: DOTD, CNL editor
- Problems: difficult to manage as their number increases, semi-formal
Patterns (Dwyer et al., 1998)

- Proven formal textual representations used to specify recurring requirements
- Rigorous semantics, which explicitly prescribes its applicability and ensures its consistent interpretation
- More complex than templates and boilerplates
- Tool: Properties Editor
- Problems: selection (for a particular domain) and organization
### Pattern example

<table>
<thead>
<tr>
<th>Pattern</th>
<th>always condition [on component]</th>
</tr>
</thead>
</table>

This pattern describes an invariant condition that has to be valid at the specified component, or, if there has no component been defined, in general.

**Natural Language Requirement:**

The lock shall be designed to operate with the pressures specified below:

- Maximum supply pressure: 300 bar
- Minimum supply pressure 100 bar

**Pattern based RSL:**

```plaintext
always ((supplyPressure<300) && (supplyPressure>100))
```

(from http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R2.3_M3_v1.000_PU.pdf)
Domain ontology

- Collection of pre-agreed concepts, terms and axioms for a specific domain
- The combination of a domain ontology and a set of requirements templates forms a simple controlled natural language (Tommila et al., 2013)
- If the requirements are written using a controlled natural language whose vocabulary comes from a domain ontology, the requirements can be reasoned to check their consistency
Ontology example

Visual formalisms

- Pictorial information is much easier processed and understood by human brains compared to pure texts (Myers et al., 1988)
- Graphical formal methods (GFM): formal specifications = textual symbols + graphics
- Goals: hide the mathematical complexity, improve readability of formal specifications
- A number of methods and notations, but tool support and availability are limited
UML-based visual formalisms

- UML-B (Snook & Butler, 2006): UML integrated with B notations

<table>
<thead>
<tr>
<th>operation</th>
<th>guard</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>numbεran($pbook)</td>
<td>pbook:=numb</td>
</tr>
<tr>
<td>remove</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>lookup</td>
<td>-</td>
<td>numb:=pbook</td>
</tr>
</tbody>
</table>

UML-B model of a telephone book

**$L_R$ temporal logic**

- First level: experts of formal methods define a set of $L_R$ constructs and related patterns to express properties for a particular domain.
- Second level: domain users follow the given patterns to specify concrete system properties as directed acyclic graphs.
- Graph nodes can represent predicates, logical connectives, quantified temporal operators and modal operators.


![Diagram](image-url)
Visual formalisms based on timing diagrams

- Timing diagrams are familiar to engineers
- Constraint Diagrams (Dietz, 1996), Real-Time Graphical Interval Logic (Moser et al., 1997), Timing Diagram Logic (Fisler, 1999), ...

Constraint diagram: “If process P has been idle for ten seconds, then within one second alarm A will be triggered”

Visualizing executable formal specifications via animation (Özcan et al., 1998)

Issues in designing user-friendly FSLs

Issues by Leveson et al. (1999):

1. Bridge the semantic gap between the users’ mental model and the model composed using the FSL → use syntaxes, notations and design principles based on the users’ domain knowledge

2. Assist the users to construct black-box requirement models (which describe the externally visible behavior of a system) → mechanisms and features must be introduced in the FSL

3. Prevention of using error-prone features in FSLs, such as internal broadcast events → provide proper guidance if such features are used

4. Increase the reusability of existing formal specifications → support functions, macros, etc.

5. The FSL must facilitate the inspection of incorrect and incomplete requirement specifications → checklists of formal criteria, language syntax to enforce such constraints
Conclusions

- The issue of making formal specifications user-friendly was investigated
- Reviewed known FSLs, graphical notations, issues of FSL design
- User-friendly approaches are compulsory if we wish formal methods to be widely applied in industry
- Formal specifications should be formulated using domain concepts
- Preferably, visually
- Tool support is required, but current tool availability is limited despite the large volume of approaches
Future work

- Requirement analysis specifically for nuclear power plant engineering domain
- Propose a visual FSL for this domain
- Final goal: integrated framework supporting transparent safety demonstration
Acknowledgements

- SAUNA project, funded by the Finnish Nuclear Waste Management Fund VYR as a part of research program SAFIR2018
- Government of Russian Federation, Grant 074-U01
Thank you for your attention!
Questions?

A Study on User-Friendly Formal Specification Languages for Requirements Formalization

Cheng Pang, Antti Pakonen, Igor Buzhinsky, Valeriy Vyatkin

igor.buzhinskii@aalto.fi