From Textual Specification to Formal Verification
(thesis defence)
Viliam Simko
2013-09-24, Prague
Outline

• Requirements engineering - problems
  – Early phases of a project:
    • Dynamic structure
    • Static structure

• My contribution
Context

requirements specification
  • Initial requirements analysis
  • Changing environment
  • We want to feel confident early

dynamic structure
  textual use-cases
  verification of actions

static structure
  domain model
  elicitation from text

FOAM method

Prediction Framework

Part I

Part II
Part I

Verification of textual use-cases
Textual use-cases (example)

**Use-Case U₁:** Buyer Places Bid On Item

**Main success scenario:**
1. Include use-case “Buyer Reviews Item Information”.
2. The buyer notifies the System that he/she wants to place a bid.
3. The System shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the System.
5. The System shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the System.

**Use-Case U₂:** Buyer Reviews Item Information

**Main success scenario:**
1. The buyer uses the web page to send a review item information request to the System.
2. The System displays information about the item.
3. The buyer reviews item information.

**Variation:**
2a. The item is not valid
   2a1. The System displays a message describing invalid item.
   2a2. Use-case aborted.
Motivation: temporal dependency

Use-Case $U_1$: Buyer Places Bid On Item

Main success scenario:
1. **Include** use-case “Buyer Reviews Item Information”.
2. The buyer notifies the System that he/she wants to place a bid.
3. The System shall respond by requesting the details about bids from the buyer.
4. The buyer **sends a submit bid request** to the System.
5. The System shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the System.

Use-Case $U_2$: Buyer Reviews Item Information

Main success scenario:
1. The buyer uses the web page to send a review item information request to the System.
2. The System **displays information about the item**.
3. The buyer reviews item information.

**Note:**
Steps must follow in a given order even if spread across multiple components.
Use-Case $U_1$: Buyer Places Bid On Item

Main success scenario:
1. Include use-case “Buyer Reviews Item Information”.
2. The buyer notifies the System that he/she wants to place a bid.
3. The System shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the System.
5. The System shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the System.

Use-Case $U_2$: Buyer Reviews Item Information

Main success scenario:
1. The buyer uses the web page to send a review item information request to the System.
2. The System displays information about the item.
3. The buyer reviews item information.

Variation:
2a. The item is not valid
   2a1. The System displays a message describing invalid item.
   2a2. Use-case aborted.

Counter-example
Motivation: temporal dependency

Use-Case $U_1$: Buyer Places Bid On Item

Main success scenario:
1. Include use-case “Buyer Reviews Item Information”.
2. The buyer notifies the System that he/she wants to place a bid.
3. The System shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the System.
5. The System shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the System.

Extension:
1a. The use-case “Buyer Reviews Item Information” was aborted.
1a1. The System displays a message “Bid cannot be placed”
1a2. Use-case aborted.

Use-Case $U_2$: Buyer Reviews Item Information

Main success scenario:
1. The buyer uses the web page to send a review item information request to the System.
2. The System displays information about the item.
3. The buyer reviews item information.

Variation:
2a. The item is not valid
2a1. The System displays a message describing invalid item.
2a2. Use-case aborted.
How to formalize this?

Use-Case $U_1$: Buyer Places Bid On Item

Main success scenario:
1. Include use-case “Buyer Reviews Item Information”. #include:u2
2. The buyer notifies the System that he/she wants to place a bid.
3. The System shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the System. #use:item
5. The System shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the System.

Extension:
1a. The use-case “Buyer Reviews Item Information” was aborted. #guard:abort
1a1. The System displays a message “Bid cannot be placed”
1a2. Use-case aborted. #abort

Use-Case $U_2$: Buyer Reviews Item Information

Main success scenario:
1. The buyer uses the web page to send a review item information request to the System.
2. The System displays information about the item. #create:item
3. The buyer reviews item information.

Variation:
2a. The item is not valid
2a1. The System displays a message describing invalid item.
2a2. Use-case aborted. #abort

Flow annotations (describing structure)
- abort, include, goto, mark, guard

Temporal annotations (to be checked)
- create, use, ... (semantics defined in TADL)
UML: use-case model example

12 use-cases from our case study
Verification overview

Transformation pipeline

UCM → rUCM* → UCBA (LTS with guards) → NuSMV Model → Counter Example

```
MODULE main --- NuSMV code gens
FAIRNESS (guard loop): --- prove
DEFINE guardloop := s in {init_0, s

VAR exec_1 : boolean;
ASSIGN
init(exec_1) := FALSE;
next(exec_1) := case
s=succ_1 : TRUE;
TRUE : exec_1;
esac;
VAR exec_2 : boolean;

VAR s : {init_0, init_1, init_2, init_3}
ASSIGN
init(s) := init_0;
next(s) := case
s=succ_0 : succ_0, init_1, if
s=succ_0 & !exec_1 & exec
s=succ_0 : succ_0, --- fit
--- UCT
s = init_1 & !exec_1 ; init
s = init_1 : u1 & e0 ; init

Trace:

UCT AG (create -> EF u1)
```

Viliam Šimko - From textual specification to formal verification - 2013
Results of the evaluation of scalability

\((t \times u \times m \times bc \times bl \times a)\)

- \(u\) = number of use-cases
- \(m\) = size of the main scenario
- \(bc\) = number of branches
- \(bl\) = size of each branch
- \(a\) = number of annotations

1 minute deadline

**CPU:** Core2 Duo 2.53GHz
**RAM:** 4GiB RAM
**OS:** 64bit Linux
Part II

Elicitation of a domain model from text
What would a human analyst do?

- reading the text
- sketching the domain model
- again and again and again ...

Our goal is to automate this

Textual specification

Domain model sketched
Ambiguity inherent to natural language

3.1 User administration

The user administration contains a user account for each user which contains all user data.

A user is able to register at the system with his user number, to manage his user account, and to extend the media's rental period.

A password is not necessary because the user number on the identification card is read with a bar code scanner.

3.2 Media administration

The media administration contains an entry for each medium in the library. Several instances of each medium may be available, and they may have different locations.

A user is able to search for a medium by specifying one or more features of the media. User can choose and reserve a medium. To this end, the user number on the identification card is scanned. If a user borrows an instance, it is added to his account and will only be deleted when it is returned.
The media administration contains an entry for each medium in the library. Several instances of each medium may be available, and they may have different locations.

A librarian is able to add a new media instance to the media administration, change the status of an instance, and remove an instance from the media administration.

A user is able to search for a medium by specifying one or more features of the medium. User can choose and reserve a found instance. To this end, the user number on the identification card is scanned. If a user borrows an instance, it is added to his account and will only be deleted when it is returned.
Overview of phases

Specification Documents

Preprocessing

Feature Selection
- Scientist

Training
- Domain Expert

Elicitation
- Analyst

Domain Model
Results of the evaluation

**Elicitation phase:** 4 seconds
- Library System specification (853 words)

**Preprocessing (linguistic analysis):** 2 minutes

**Training all 3 models:** 4 seconds
- 178 annotated phrases
- 40 unique entities/relations

**Feature selection phase (cross-fold validation):**
- **linktype:** 150s (19/21 features selected)
- **roleInLink:** 140s (20/21 features selected)
- **relcl:** 15s (5/6 features selected)

CPU: Core2 Duo 2.53GHz
RAM: 4GiB RAM
OS: 64bit Linux
Conclusion

Goals of the thesis fulfilled

- Formulate behavior of textual use-cases
  - Flow/temporal annotations in text
  - User-defined temporal semantics
  - Transformation to LTS

- Design a method for verification of use-cases
  - Model-checking using NuSMV
  - HTML report with counter-examples + traces to the original use-cases
  - Evaluation of speed + learning curve

- Combine linguistic and software engineering artefacts
  - NLP pipeline (Stanford CoreNLP)
  - Specification model (Eclipse Modelling Framework)
  - Maximum entropy classifiers (Apache OpenNLP) + evaluation
Future work

FOAM
- Free structure of textual use-cases
- Multiple model-checking back-ends (e.g. SPIN)
- Suggesting annotations

Elicitation
- More training data
- More types of features
- Predicting more details (aggregation, generalization, ...)
- Clean-up after elicitation
- Improved feature selection
- Compare other classifiers (SVM, CRF, Bayes, ...)
The D3S team:

People from other departments/institutions:
M. Holeňa, Z. Žabokrtský, Ch. Manning
My publications (only reviewed listed)

FOAM: A lightweight method for verification of use-cases
V. Simko, P. Hnetynka, T. Bures, F. Plasil.
In: Postproc. of SEAA’12, 2012, pp. 228-232, DOI: 10.1109/SEAA.2012.15

Verifying temporal properties of use-cases in natural language
V. Simko, D. Hauzar, T. Bures, P. Hnetynka, F. Plasil.
In: Postproc. of FACS’11. LNCS, Springer, 2011, DOI: 10.1007/978-3-642-35743-5_21

From textual use-cases to component-based applications
V. Simko, P. Hnetynka, T. Bures.
ISBN: 978- 3-642-13264-3, DOI: 10.1007/978-3-642-13265-0

Open internet gateways to archives of media art

Accessing Libraries of Media Art through Metadata

OASIS Archive - Open Archiving System with Internet Sharing

Long-term digital preservation of a new media performance: "Can we re-perform it in 100 years?"
V. Simko, M. Masa, D. Giaretta.

Tools implemented

- FOAM tool
- Prediction Framework
- ReProTool
Additional slides
(thesis defence)

Viliam Simko
2013-09-24, Prague
Rewiever's comments

Measured results compared to the baseline
e.g. just using nouns as candidate entities

- We could add a new feature “isnoun” similar to our existing “pos” feature
Comparing MaxEnt to other approaches

- Other types of classifiers (SVM, CRF, Bayes)
  - Easy to plug-in to our existing framework
  - **Expected result:** Bayes < SVM < MaxEnt < CRF

- Grammar-based approaches, e.g. PCFG
  - Similar to picking the most-probable parse tree, we could pick the most probable relation between domain-model entities.
  - Significant changes to the elicitation pipeline required
Reviewer's comments

**Annotation agreement**
- Multiple versions of the same training data
- More training data always helps

**Impact of the coref. resolution on accuracy**
- Coreferences extracted in our linguistic pipeline
- Coref-related features avoided in the current implementation
- Easy to implement and end easy to benchmark in future
Rewieever's comments

Cross-domain application
- More training data needed (from multiple domains)
- Then easy to benchmark using current framework
Cleaning-up the generated domain model

- We aimed at generating a prototype domain model
- Users were expected to clean up manually
- Automated cleaning – future work
  - Duplicate elements
  - Consistency checks e.g. using Alloy
Rewieever's comments

Improved feature selection

- Current implementation:
  
  random sampling of $2^n$ points in n-dim. space
  
  $k$ samples for every $\binom{n}{m}$ subset of features

  $k$: parameter of the method
  $n$: total number of features
  $m$: size of the subset

- Future work
  
  • Chi-squared
  
  • Mutual information (Information gain)
Evaluation of learning curve

- 9 use-cases analysed by 4 people
  - sample from 34 use-cases (Admission System)
- Manual review
  - positive feedback from testers
  - 2 inconsistencies found
  - learning time negligible
  - FOAM annotations – easier to spot important places in the text
- After running FOAM
  - 6 more inconsistencies detected
  - fixed immediately

<table>
<thead>
<tr>
<th></th>
<th>Author</th>
<th>TesterA</th>
<th>TesterB</th>
<th>TesterC</th>
<th>AVG(A,B,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading</td>
<td>n/a</td>
<td>10min</td>
<td>30min</td>
<td>7min</td>
<td>16min</td>
</tr>
<tr>
<td>Precedence</td>
<td>12min</td>
<td>6min</td>
<td>5min</td>
<td>13min</td>
<td>8min</td>
</tr>
<tr>
<td>Flowannot.</td>
<td>7+3min</td>
<td>11min</td>
<td>8min</td>
<td>17min</td>
<td>12min</td>
</tr>
<tr>
<td>Temp.annot.</td>
<td>25min</td>
<td>23min</td>
<td>24min</td>
<td>23min</td>
<td>23min</td>
</tr>
<tr>
<td>TotalAnalysis</td>
<td>47min</td>
<td>50min</td>
<td>67min</td>
<td>60min</td>
<td>59min</td>
</tr>
<tr>
<td>AnalysisperUC</td>
<td>5min</td>
<td>5min</td>
<td>7min</td>
<td>6min</td>
<td>6min</td>
</tr>
<tr>
<td>Learning</td>
<td>n/a</td>
<td>13min</td>
<td>15min</td>
<td>18min</td>
<td>15min</td>
</tr>
<tr>
<td>Total</td>
<td>47min</td>
<td>63min</td>
<td>82min</td>
<td>78min</td>
<td>74min</td>
</tr>
</tbody>
</table>
Temporal Annotation Definition Language

Annotations:
#create:city
#use:city
#create:map

TADL Template:
Annotations: create, use
CTL AG( create \rightarrow EF(use) ) "Branch with use required after create"
CTL AG( create \rightarrow AX(AG(\neg create)) ) "Only one create"
CTL A[\neg use U create | \neg EF(use)] "First create then use"

A set of temporal formulae to be checked

\[
\begin{align*}
\text{CTL AG( create}_{\text{city}} \rightarrow EF(use}_{\text{city}}) \\
\text{CTL AG( create}_{\text{city}} \rightarrow AX(AG(\neg create}_{\text{city}})) \\
\text{CTL A[\neg use}_{\text{city}} U create}_{\text{city}} | \neg EF(use}_{\text{city}}) \\
\text{CTL AG( create}_{\text{map}} \rightarrow EF(use}_{\text{map}}) \\
\text{CTL AG( create}_{\text{map}} \rightarrow AX(AG(\neg create}_{\text{map}})) \\
\text{CTL A[\neg use}_{\text{map}} U create}_{\text{map}} | \neg EF(use}_{\text{map}})
\end{align*}
\]
**Elicitation Phase**

- Predict Candidates for Domain Entities
- Predict Multi-Word Entities
- Derive Names for Entity Links
- Convert Entity Links to Classes
- Merge Classes by Name
- Predict Relations

**Training Phase**

- Config
- Prepare Context Generators
- Specification Model
- Extract Samples
- Speciﬁcation Model
- Extract Samples
- Trained Models
- Prepared in the Preprocessing Phase
- Train MaxEnt Models
- Prepared in the Preprocessing Phase
- Trained Models

**Feature-selection Phase**

- Feature Selection Config
- Generate Random Combinations of Feature Sets
- Feature Selection Config
- Generate Random Combinations of Feature Sets
- Prepared in the Preprocessing Phase
- Prepare Context Generator
- Speciﬁcation Model
- Extract Samples
- Prepared in the Preprocessing Phase
- Extract Samples
- k-fold cross-validation
- Evaluation Results
- [next feature set to evaluate]
Linguistic Pipeline
Tokenizer
XML Preprocessor
MaxEnt Sentence Splitter
MaxEnt POS Tagger
Lemmatizer
NER Classifier
Lexicalized Parser
Coreference Resolution

Tidy HTML Cleaner

Basic Linguistic Info
Semantic Graph (Typed Dependencies)
Coreferences
Resolution of Entity Links

Create Empty Specification
Transform to Specification Model
Load Existing Domain Model

Preprocessing Phase
Input Document

Stanford CoreNLP tools
Other tools
### NLP in requirements engineering

<table>
<thead>
<tr>
<th>Others</th>
<th>This thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguity detection</td>
<td>Extracting models</td>
</tr>
<tr>
<td>Quality checks</td>
<td></td>
</tr>
<tr>
<td>Semantic analysis</td>
<td></td>
</tr>
<tr>
<td>Elicitation of requirements</td>
<td></td>
</tr>
<tr>
<td>Classification of documents</td>
<td></td>
</tr>
<tr>
<td>Detection of requirements</td>
<td></td>
</tr>
<tr>
<td>Extracting models</td>
<td></td>
</tr>
<tr>
<td>Traceability</td>
<td></td>
</tr>
</tbody>
</table>

### Static structures

<table>
<thead>
<tr>
<th>Others</th>
<th>This thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>hand-written rules and heuristics</td>
<td>statistical approach</td>
</tr>
<tr>
<td>constituency parse trees</td>
<td>dependency trees</td>
</tr>
<tr>
<td>require hand-made glossary</td>
<td>just text</td>
</tr>
<tr>
<td>automated, semi-automated</td>
<td>automated</td>
</tr>
<tr>
<td>rely on the document structure</td>
<td>oblivious to structure</td>
</tr>
<tr>
<td>ontology for clean-up</td>
<td>manual clean-up</td>
</tr>
<tr>
<td>output: ER, UML class diagram</td>
<td>domain model (EMF)</td>
</tr>
</tbody>
</table>

### Dynamic structures

<table>
<thead>
<tr>
<th>Others</th>
<th>This thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC, activity/state/sequence diagrams, use-cases</td>
<td>input: textual use-cases</td>
</tr>
<tr>
<td>Petri-nets, MSC, LTS, FSM</td>
<td>LTS</td>
</tr>
<tr>
<td>templates, extraction from text</td>
<td>template (sentences in English)</td>
</tr>
<tr>
<td>temporal logics, predefined patterns, BP, description logic</td>
<td>LTL/CTL/PLTL as annotations (user-defined patterns)</td>
</tr>
<tr>
<td>generalization, include, extend, pre/post-conditions</td>
<td>precedence, include, extend, goto</td>
</tr>
<tr>
<td>SPIN</td>
<td>NuSMV</td>
</tr>
<tr>
<td>QoS, liveness, sequencing of actions</td>
<td>sequencing of actions</td>
</tr>
</tbody>
</table>