FOAM
A Lightweight Method for Verification of Use-Cases
(submitted to SEAA)

http://d3s.mff.cuni.cz

Viliam Šimko
Tomáš Bureš
Petr Hnětynka
František Plášil
{simko, bures, hnetynka, plasil}@d3s.mff.cuni.cz

Plzeň, May 16, 2012
Goals

- Requirements specification in a changing environment
- Specification of **temporal dependencies** among use-cases.
  - using **temporal formulae** encapsulated as annotations
- Verification on the specification level
  - using **NuSMV** model checker
Motivation: **temporal dependency**

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

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

**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 GPM.
2. The GPM displays information about the item.
3. The buyer reviews item information.
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 GPM that he/she wants to place a bid.
3. The GPM shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the GPM.
5. The GPM shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the GPM.

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 GPM.
2. The GPM displays information about the item.
3. The buyer reviews item information.

Variation:
2a. The item is not valid
2a1. The GPM 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 GPM that he/she wants to place a bid.
3. The GPM shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the GPM.
5. The GPM shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the GPM.

Extension:
1a. The use-case “Buyer Reviews Item Information” was aborted.
1a1. The GPM 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 GPM.
2. The GPM displays information about the item.
3. The buyer reviews item information.

Variation:
2a. The item is not valid
2a1. The GPM displays a message describing invalid item.
2a2. Use-case aborted.
Case Study: Donald Firesmith, Global personal marketplace system requirements specification, 2003, Link
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 GPM that he/she wants to place a bid.
3. The GPM shall respond by requesting the details about bids from the buyer.
4. The buyer sends a submit bid request to the GPM.  
   \#use:item
5. The GPM shall respond by sending a notification to the buyer.
6. The buyer sends a notification acknowledgement to the GPM.

Extension:
1a. The use-case “Buyer Reviews Item Information” was aborted.  
   \#guard:u2aborted
1a1. The GPM 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 GPM.
2. The GPM displays information about the item.  
   \#create:item
3. The buyer reviews item information.

Variation:
2a. The item is not valid
2a1. The GPM displays a message describing invalid item.
2a2. Use-case aborted.  \#abort  
   \#mark:u2aborted

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

- Temporal annotations (to be checked)
  - create, use, ... (defined in TADL)
Temporal Annotation Definition Language

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

TADL Template:

Annotations: create, use

- \(\text{CTL AG}(\text{create} \rightarrow \text{EF(use)})\)

- \(\text{CTL AG}(\text{create} \rightarrow \text{AX(AG(!create))})\)

- \(\text{CTL A}[\! \text{use } \text{create} \mid \! \text{EF(use)}]\)

A set of temporal formulae to be checked:

\[
\begin{align*}
\text{CTL AG}(\text{create}_{\text{city}} & \rightarrow \text{EF(use}_{\text{city}})) \\
\text{CTL AG}(\text{create}_{\text{city}} & \rightarrow \text{AX(AG(!create}_{\text{city}})) ) \\
\text{CTL A}[\! \text{use } \text{create}_{\text{city}} & \mid \! \text{EF(use}_{\text{city}})] \\
\end{align*}
\]

\[
\begin{align*}
\text{CTL AG}(\text{create}_{\text{map}} & \rightarrow \text{EF(use}_{\text{map}})) \\
\text{CTL AG}(\text{create}_{\text{map}} & \rightarrow \text{AX(AG(!create}_{\text{map}})) ) \\
\text{CTL A}[\! \text{use } \text{create}_{\text{map}} & \mid \! \text{EF(use}_{\text{map}})] \\
\end{align*}
\]
More TADL Examples

Annotations: create, use

\[
\begin{align*}
\text{CTL } & \text{AG( create } \rightarrow \text{ EF(use) ) "Branch with use required after create"} \\
\text{CTL } & \text{AG( create } \rightarrow \text{ AX(A}[\text{!create}]) "Only one create"} \\
\text{CTL } & \text{A}[\text{!use U create | !EF(use)}] "First create then use" \\
\end{align*}
\]

Annotations: open, close — strict ordering of 2 phases

\[
\begin{align*}
\text{LTL } & \text{G(open } \rightarrow \text{ F(close)) "After open, close is required"} \\
\text{CTL } & \text{AG(open } \rightarrow \text{ AX(A}[\text{!open U close}]) "No multi—open"} \\
\text{CTL } & \text{AG(close } \rightarrow \text{ AX(A}[\text{!close U open | !EF(close})]) "No multi—close"} \\
\text{CTL } & \text{A}[\text{!close U open | !EF(close)}] "First open then close" \\
\end{align*}
\]

Annotations: init, process, release — strict ordering of 3 phases — init → process

\[
\begin{align*}
\text{CTL } & \text{A}[\text{process U init | !EF(process))} "First init then process" \\
\text{CTL } & \text{AG(init } \rightarrow \text{ AF(process)) "After init there should always be process"} \\
\text{CTL } & \text{AG(init } \rightarrow \text{ AX(A}[\text{!init U process}]) "No multi—init without process"} \\
\text{CTL } & \text{AG(process } \rightarrow \text{ AX(A}[\text{!process U init | !EF(process)}]) \\
& "No multi—process without init" \\
\end{align*}
\]

— process → release

\[
\begin{align*}
\text{CTL } & \text{A}[\text{!release U process | !EF(release))} "First process then release" \\
\text{CTL } & \text{AG(process } \rightarrow \text{ AF(release)) "After process, release is required"} \\
\text{CTL } & \text{AG(process } \rightarrow \text{ AX(A}[\text{!process U release}]) \\
& "No multi—process without release" \\
\text{CTL } & \text{AG(release } \rightarrow \text{ AX(A}[\text{!release U process | !EF(release})]) \\
& "No multi—release without process" \\
\end{align*}
\]
Transformation: Overview

**TADL**: Temporal Annotation Definition Language (templates)

**OBA**: Overall Behavior Automaton

- Use Case Model
- Precedence Relation
- Use-Cases $U_1 \ldots U_n$
- Flow Annotations
- Temporal Annotations
- TADL annotation templates
- CTL / LTL Formulae
- OBA LTS with guards
- SMV Model
- Counter Example
Overall Behavior Automaton (OBA)

\[ \text{init}_0 \]

\[ \text{succ}_0 \]

\[ \{ \text{done}_u \leftarrow \text{true} \} \]

\[ \text{init}_u \]

\[ \text{succ}_u \]

\[ \{ \neg \text{done}_u \land G_u^{\text{prec}} \} \]

\[ [ \text{done}_1 \land \ldots \land \text{done}_n ] \]
Construction of OBA using Inference Rules (1/2)

1. Representing steps

\[
\begin{align*}
  u \in U_M, x \in S_u \\
  x^{in} &\rightarrow x^{var} \rightarrow x^{jump} \rightarrow x^{ext} \rightarrow x^{out}
\end{align*}
\]

2. Representing scenarios

\[
\begin{align*}
  u \in U_M, w \in W_u, x_1 \leq w \ldots \leq w \ x_n \\
  (x_1^{out} \rightarrow x_2^{in}), \ldots, (x_{n-1}^{out} \rightarrow x_n^{in})
\end{align*}
\]

3. Connecting variations

\[
\begin{align*}
  u \in U_M, w \in W_u, w = \{y_1, \ldots, y_n\}, Var_u(w) = x, \\
  G_V = \{g | \langle guard:g \rangle \in Flow_u(y_1)\} \\
  x^{var} \xrightarrow{[G_V]} y_1^{in}
\end{align*}
\]

4. Connecting extensions

\[
\begin{align*}
  u \in U_M, w \in W_u, w = \{y_1, \ldots, y_n\}, Ext_u(w) = x, \\
  G_E = \{g | \langle guard:g \rangle \in Flow_u(y_1)\} \\
  x^{ext} \xrightarrow{[G_E]} y_1^{in}
\end{align*}
\]

5. Continuation from scenarios

\[
\begin{align*}
  u \in U_M, w \in W_u, x = Var_u(w) \lor x = Ext_u(w), \\
  w = \{y_1, \ldots, y_n\}, \langle abort \rangle \notin Flow_u(y_n), \\
  \forall s \in S_u \langle goto:s \rangle \notin Flow_u(y_n) \\
  y_n^{out} \rightarrow x^{out},
\end{align*}
\]

6. Handling GOTO annotations

\[
\begin{align*}
  u \in U_M, x \in S_u, \langle goto:y \rangle \in Flow_u(x) \\
  x^{out} \rightarrow y^{jump},
\end{align*}
\]

7. Handling ABORT annotations

\[
\begin{align*}
  u \in U_M, x \in S_u, \langle abort \rangle \in Flow_u(x) \\
  x^{out} \rightarrow x^{out}
\end{align*}
\]
Construction of OBA using Inference Rules (2/2)

8. Handling INCLUDE (procedure call)
\[ u, c \in U_M, x \in S_u, \langle \text{include}:c \rangle \in \text{Flow}_u(x), \]
\[ w_c^m = \{y_1, \ldots, y_n\} \]
\[ \text{jump} \quad \{\text{incl}_u,c \leftarrow \text{true}\} \rightarrow y_1^{\text{in}}, \text{jump} \quad [\text{false}] \rightarrow x^{\text{ext}}, \]

9. Handling INCLUDE (return)
\[ u, c \in U_M, x \in S_u, \langle \text{include}:c \rangle \in \text{Flow}_u(x), \]
\[ w_c^m = \{y_1, \ldots, y_n\}, \forall s \in S_c \langle \text{goto}:s \rangle \notin \text{Flow}_c(y_n) \]
\[ y_n^{\text{out}} \quad \{\text{incl}_u,c \leftarrow \text{false}\} \rightarrow x^{\text{ext}}, \]

10. Scheduler
\[ u \in U_M^P, w_u^m = \{x_1, \ldots, x_n\}, \]
\[ G_u^{\text{prec}} = \{\text{done}_v | \exists v \in U_M^P (v, u) \in \text{Prec}_M\} \]
\[ \text{init}_0 \quad [G_u^{\text{prec}}, \neg \text{done}_u] \rightarrow x_1^{\text{in}}, x_n^{\text{out}} \quad \{\text{done}_u \leftarrow \text{true}\} \rightarrow \text{init}_0 \]

11. Final state
\[ G = \{\text{done}_u | u \in U_M^P\} \]
\[ \text{init}_0 \xrightarrow{[G]} \text{succ}_0 \]

12. Atomic propositions
\[ x \in S_u, u \in U_M \]
\[ \text{Lab}(x^{\text{jump}}) = \text{Temp}_u(x) \]
OBA represented in NuSMV

MODULE main

VAR state : \{s_1, ..., s_n\} \quad \text{all states of OBA}

ASSIGN init(state) := init_0; \quad \text{initial state of OBA}

next(state) := case

\text{state}=x : \{y_1, ..., y_n\}; \quad \text{transitions } x \to y_1, ..., x \to y_n

\text{state}=y_i \& \neg(g) : x; ... \quad \text{guarded transition } x \xrightarrow{g} y_i

esac;

FAIRNESS \! \text{guardloop} \quad \text{avoids infinite loops when testing guards}

DEFINE guardloop := state \text{ in } \{x_1, ..., x_m\} \quad \text{states in guards}

VAR v : boolean; \quad \text{variable } v \text{ from OBA}

ASSIGN init(v) := FALSE; \quad \text{valuation function } Val_A

next(v) := case

\text{state} = s^v : b^v; ... \quad \text{assigns value } b^v \text{ to } v \text{ in state } s^v

\text{TRUE} : v; \quad \text{preserves the current value of } v

esac;

\quad \text{LTL/CTL formula } f \in F_A \quad \text{which uses variables } t_1, \ldots, t_j

LTLSPEC f(t_1, \ldots, t_j) \quad \text{CTLSPEC } f(t_1, \ldots, t_j)
Unfortunately the current approach does not scale well
Future work

- Reduction of the problem
  - Partial order reduction
  - Independent groups of use-cases
  - Construction of a specific OBA per annotation group
- Automatic detection of annotations from text
- Sequencing of use-cases vs parallelism
Summary

- Method for verification of textual use-cases
  - against temporal properties (CTL/LTL)
  - under all use case orderings implied by precedence relation
  - custom temporal annotations (TADL)
- Supports collaborative work and iterative development
- Paper submitted to SEAA 2012
  - 38th Euromicro Conference on Software Engineering and Advanced Applications, September 5-8, 2012, Cesme, Izmir, Turkey
Thank you for attention

See also
http://code.google.com/a/eclipselabs.org/p/reprotool/