Software Design, Modelling and Analysis in UML

Lecture 13: Core State Machines III

2016-12-15

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Recall: **Transitions** of UML State Machines
  - **discard** event,
  - **dispatch** event,
  - **continue** RTC,
  - **environment** interaction,
  - **error** condition.
- **Example Revisited**
- **Initial States**
- **Rhapsody Demo III**: **Model Animation**
- **Create** and **Destroy** Transformers

Recall: Transition Relation

From Core State Machines to LTS

Definition. Let $\mathscr{S}_0 = (\mathscr{T}_0, \mathscr{C}_0, V_0, atr_0, \mathscr{E})$ be a signature with signals (all classes in $\mathscr{C}_0$ **active**), $\mathscr{D}_0$ a structure of $\mathscr{S}_0$, and $(Eth, ready, \oplus, \ominus, [\,\cdot\,])$ an ether over $\mathscr{S}_0$ and $\mathscr{D}_0$. Assume there is one core state machine M_C per class $C \in \mathscr{C}$.

We say, the state machines **induce** the following labelled transition relation on states $S := (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times Eth) \,\dot\cup\, \{\#\}$ with labels $A := 2^{\mathscr{D}(\mathscr{E})} \times 2^{(\mathscr{D}(\mathscr{E}) \,\dot\cup\, \{*,+\}) \times \mathscr{D}(\mathscr{C})} \times \mathscr{D}(\mathscr{C})$:

- $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$

  if and only if

  (i) an event with destination u is **discarded**,

  (ii) an event is **dispatched** to u, i.e. stable object processes an event, or

  (iii) run-to-completion processing by u **continues**, i.e. object u is not stable and continues to process an event,

  (iv) the **environment** interacts with object u,

- $s \xrightarrow{(cons, \emptyset)} \#$

  if and only if

  (v) an **error condition** occurs during consumption of $cons$, or $s = \#$ and $cons = \emptyset$.

(i) Discarding An Event

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- an E-event (instance of signal E) is ready in ε for object u of a class $\mathscr{C}$, i.e. if

$$u \in \text{dom}(\sigma) \cap \mathscr{D}(C) \wedge \exists\, u_E \in \mathscr{D}(E) : u_E \in ready(\varepsilon, u)$$

- u is stable and in state machine state s, i.e. $\sigma(u)(stable) = 1$ and $\sigma(u)(st) = s$,
- but there is no corresponding transition enabled (all transitions incident with current state of u either have other triggers or the guard is not satisfied)

$$\forall\, (s, F, expr, act, s') \in \to (\mathcal{SM}_C) : F \neq E \vee I[\![expr]\!](\sigma, u) = 0$$

and

- in the system configuration, stability may change, u_E goes away, i.e.

$$\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$$

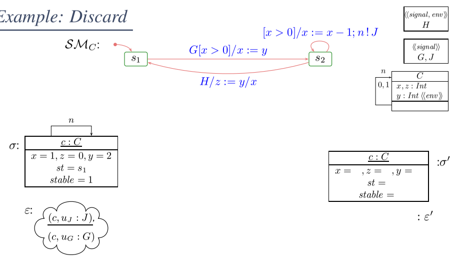
  where $b = 0$ if and only if there is a transition **with trigger '_'** enabled for u in (σ', ε').
- the event u_E is removed from the ether, i.e.

$$\varepsilon' = \varepsilon \ominus u_E,$$

- consumption of u_E is observed, i.e.

$$cons = \{u_E\}, \quad Snd = \emptyset.$$

Example: Discard



- $u \in \text{dom}(\sigma) \cap \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$
- $\forall\, (s, F, expr, act, s') \in \to (\mathcal{SM}_C)$:
  $F \neq E \vee I[\![expr]\!](\sigma, u) = 0$
- $\sigma(u)(stable) = 1, \sigma(u)(st) = s,$
- $\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$
- $\varepsilon' = \varepsilon \ominus u_E$
- $cons = \{u_E\}, \quad Snd = \emptyset$

(ii) Dispatch

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C) \wedge \exists\, u_E \in \mathscr{D}(E) : u_E \in ready(\varepsilon, u)$
- u is stable and in state machine state s, i.e. $\sigma(u)(stable) = 1$ and $\sigma(u)(st) = s$,
- a transition is **enabled**, i.e.

$$\exists\, (s, F, expr, act, s') \in \rightarrow (\mathcal{SM}_C) : F = E \wedge I[\![expr]\!](\tilde{\sigma}, u) = 1$$

where $\tilde{\sigma} = \sigma[u.params_E \mapsto u_E]$.

and

- (σ', ε') results from applying t_{act} to (σ, ε) and removing u_E from the ether, i.e.

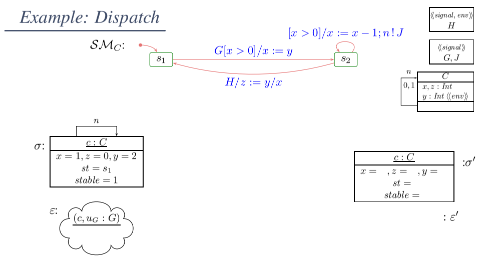
$$(\sigma'', \varepsilon') \in t_{act}[u](\tilde{\sigma}, \varepsilon \ominus u_E),$$
$$\sigma' = (\sigma''[u.st \mapsto s', u.stable \mapsto b, u.params_E \mapsto \emptyset])|_{\mathscr{D}(\mathscr{C}) \setminus \{u_E\}}$$

where b **depends** (see (i))

- Consumption of u_E and the side effects of the action are observed, i.e.

$$cons = \{u_E\}, \quad Snd = Obs_{t_{act}}[u](\tilde{\sigma}, \varepsilon \ominus u_E).$$

Example: Dispatch



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$
- $\exists\, (s, F, expr, act, s') \in \rightarrow (\mathcal{SM}_C)$:
  $F = E \wedge I[\![expr]\!](\tilde{\sigma}, u) = 1$
- $\tilde{\sigma} = \sigma[u.params_E \mapsto u_E]$.
- $\sigma(u)(stable) = 1, \sigma(u)(st) = s,$
- $(\sigma'', \varepsilon') = t_{act}(\tilde{\sigma}, \varepsilon \ominus u_E)$
- $\sigma' = (\sigma''[u.st \mapsto s', u.stable \mapsto b, u.params_E \mapsto \emptyset])|_{\mathscr{D}(\mathscr{C}) \setminus \{u_E\}}$
- $cons = \{u_E\}, \quad Snd = Obs_{t_{act}}[u](\tilde{\sigma}, \varepsilon \ominus u_E)$

(iii) Continue Run-to-Completion

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- there is an unstable object u of a class $\mathscr{C}$, i.e.

$$u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C) \wedge \sigma(u)(stable) = 0$$

and

- there is a transition without trigger enabled from the current state $s = \sigma(u)(st)$, i.e.

$$\exists\, (s, \_, expr, act, s') \in \rightarrow (\mathcal{SM}_C) : I[\![expr]\!](\sigma, u) = 1$$

  and

  - (σ', ε') results from applying t_{act} to (σ, ε), i.e.

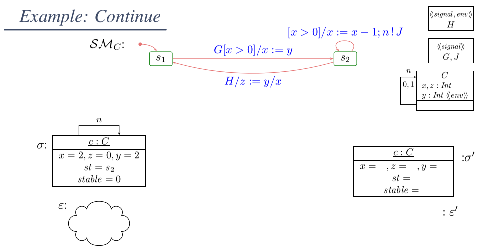
$$(\sigma'', \varepsilon') \in t_{act}[u](\sigma, \varepsilon), \quad \sigma' = \sigma''[u.st \mapsto s', u.stable \mapsto b]$$

  where b **depends** as before.

  - Only the side effects of the action are observed, i.e. $cons = \emptyset, \quad Snd = Obs_{t_{act}}[u](\sigma, \varepsilon)$.

- $\sigma' = \sigma[u.stable \mapsto 1], \varepsilon' = \varepsilon, cons = \emptyset, Snd = \emptyset$, otherwise.

Example: Continue



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C), \sigma(u)(stable) = 0$
- $\exists\, (s, \_, expr, act, s') \in \rightarrow (\mathcal{SM}_C)$:
  $I[\![expr]\!](\sigma, u) = 1$
- $\sigma(u)(st) = s,$
- $(\sigma'', \varepsilon') = t_{act}(\sigma, \varepsilon),$
- $\sigma' = \sigma''[u.st \mapsto s', u.stable \mapsto b]$
- $cons = \emptyset, \quad Snd = Obs_{t_{act}}(\sigma, \varepsilon)$

(iv) Environment Interaction

Assume that a set $\mathscr{E}_{env} \subseteq \mathscr{E}$ is designated as **environment events** and a set of attributes $V_{env} \subseteq V$ is designated as **input attributes**.

Then

$$(\sigma, \varepsilon) \xrightarrow[env]{(cons, Snd)} (\sigma', \varepsilon')$$

if either (!)

- an environment event $E \in \mathscr{E}_{env}$ is spontaneously sent to an alive object $u \in \mathrm{dom}(\sigma)$, i.e.

$$\sigma' = \sigma \,\dot{\cup}\, \{u_E \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}, \quad \varepsilon' = \varepsilon \oplus (u, u_E)$$

  where $u_E \notin \mathrm{dom}(\sigma)$ and $atr(E) = \{v_1, \dots, v_n\}$.

- Sending of the event is observed, i.e. $cons = \emptyset, Snd = \{u_E,)\}$.

or

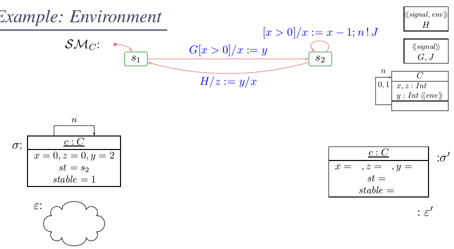
- Values of input attributes change freely in alive objects, i.e.

$$\forall\, v \in V\ \forall\, u \in \mathrm{dom}(\sigma) : \sigma'(u)(v) \neq \sigma(u)(v) \implies v \in V_{env}.$$

  and no objects appear or disappear, i.e. $\mathrm{dom}(\sigma') = \mathrm{dom}(\sigma)$.

- $\varepsilon' = \varepsilon$.

Example: Environment



- $\sigma' = \sigma \,\dot{\cup}\, \{u_E \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}$
- $\varepsilon' = \varepsilon \oplus u_E$ where $u_E \notin \mathrm{dom}(\sigma)$
  and $atr(E) = \{v_1, \dots, v_n\}$.
- $u \in \mathrm{dom}(\sigma)$
- $cons = \emptyset, Snd = \{(env, E(\vec{d}))\}$.

(v) Error Conditions

$$s \xrightarrow[u]{(cons,Snd)} \#$$

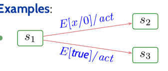
if, in (i), (ii), or (iii),

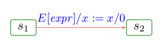
- $I[\![expr]\!]$ is not defined for σ and u, or
- $t_{act}[u]$ is not defined for (σ, ε),

and

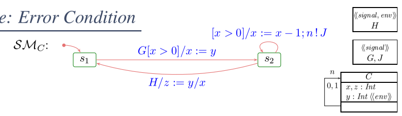
- $cons = \emptyset$, and $Snd = \emptyset$.

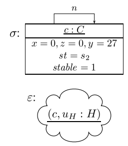
Examples:

- 

- 

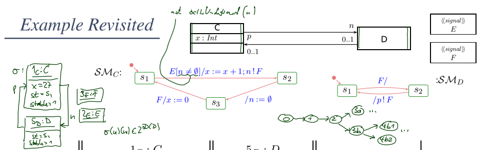
Example: Error Condition





- $I[\![expr]\!]$ not defined for σ and u, or
- $t_{act}[u]$ is not defined for (σ, ε)
- $cons = \emptyset$,
- $Snd = \emptyset$

Example Revisited



		$1_C : C$				$5_D : D$			
Nr.	x	n	st	$stable$	p	st	$stable$	ε	rule
0	27	5_D	s_1	1	1_C	s_1	1	$(3_F, 1_C).(2_E, 1_C)$	
1	27	5_D	s_1	1	1_C	s_1	1	$(2_E, 1_C)$	(i)
2	28	5_D	s_2	0	1_C	s_1	1	$(4_F, 5_D)$	(ii)
3a	28	$\emptyset$	s_3	1	1_C	s_1	1	$(4_F, 5_D)$	(iii)
					⋮				
3b	28	5_D	s_2	0	1_C	s_2	0	ϵ	(ii)
4b1	28	$\emptyset$	s_3	1	1_C	s_2	0	ϵ	(iii)
4b2	28	5_D	s_2	0	1_C	s_1	1	$(3_F, 1_C)$	(ii)
					⋮				

Transition Relation, Computation

Definition. Let A be a set of **labels** and S a (not necessarily finite) set of of **states**. We call

$$\rightarrow \; \subseteq S \times A \times S$$

a (labelled) **transition relation**.

Let $S_0 \subseteq S$ be a set of **initial states**. A (finite or infinite) sequence

$$\underbrace{s_0 \xrightarrow{a_0} s_1}_{\in\to} \overbrace{\xrightarrow{a_1} s_2}^{\in\to} \underbrace{\xrightarrow{a_2}}_{\in\to} \dots$$

with $s_i \in S$, $a_i \in A$ is called **computation** [starting at s_0] of the **labelled transition system** $(S, A, \rightarrow, S_0)$ if and only if

- [**initiation**: $s_0 \in S_0$]
- **consecution**: $(s_i, a_i, s_{i+1}) \in \rightarrow$ for $i \in \mathbb{N}_0$.

Step and Run-to-Completion

Notions of Steps: The Step

Note: we call one evolution

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons,Snd)} (\sigma', \varepsilon')$$

a **step**.

Thus in our setting, a **step** often[1] **directly corresponds** to

one object (namely u) taking **a single transition** between regular states.

(We will extend the concept of "single transition" for hierarchical state machines.)

[1]: In case of **dispatch** and **continue** with enabled transition.

That is: We're going for an interleaving semantics without true parallelism.

Notions of Steps: The Run-to-Completion Step

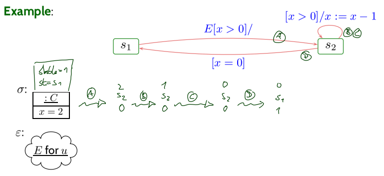
What is a **run-to-completion** step...?

- **Intuition**: a **maximal** sequence of steps of one object, where the first step is a **dispatch** step, all later steps are **continue** steps, and the last step establishes stability (or object disappears).

Note: while one step corresponds to one transition in the state machine, a run-to-completion step is in general **not syntacically definable**:

one transition may be taken multiple times during an RTC-step.

Example:



Notions of Steps: The Run-to-Completion Step Cont'd

Proposal: Let

$$(\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} \dots \xrightarrow[u_{n-1}]{(cons_{n-1}, Snd_{n-1})} (\sigma_n, \varepsilon_n), \quad n > 0,$$

be a finite (!), non-empty, **maximal**, consecutive sequence **such that**

- $(cons_0, Snd_0)$ indicates dispatching to $u := u_0$ (by Rule (ii)), i.e. $cons = \{u_E\}, u_E \in \mathrm{dom}(\sigma_0) \cap \mathscr{D}(\mathscr{E})$,
- if u becomes stable or disappears, then in the last step, i.e.

$$\forall\, i > 0 \bullet (\sigma_i(u)(stable) = 1 \vee u \notin \mathrm{dom}(\sigma_i)) \implies i = n$$

Let $0 = k_1 < k_2 < \cdots < k_N < n$ be the maximal sequence of indices such that $u_{k_i} = u$ for $1 \leq i \leq N$. Then we call the sequence

$$(\sigma_0(u) =) \quad \sigma_{k_1}(u), \sigma_{k_2}(u) \dots, \sigma_{k_N}(u), \sigma_n(u)$$

a (!) **run-to-completion step** of u (from (local) configuration $\sigma_0(u)$ to $\sigma_n(u)$).

Divergence

We say, object u **can diverge** on reception $cons_0$ from (local) configuration $\sigma_0(u)$ if and only if there is an **infinite**, consecutive sequence

$$(\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow[u_1]{(cons_1, Snd_1)} \dots$$

where $u_i = u$ for infinitely many $i \in \mathbb{N}_0$ and $\sigma_i(u)(stable) = 0, i > 0$, i.e. u does not become stable again.

Run-to-Completion Step: Discussion.

Our definition of RTC-step takes a **global** and **non-compositional** view, that is:

- In the projection onto a single object we still **see** the effect of interaction with other objects.
- Adding classes (or even objects) may change the divergence behaviour of existing ones.
- Compositional would be:

  the behaviour of a set of objects is determined by the behaviour of each object "in isolation".

  Our semantics and notion of RTC-step doesn't have this (often desired) property.

Run-to-Completion Step: Discussion.

Our definition of RTC-step takes a **global** and **non-compositional** view, that is:

- In the projection onto a single object we still **see** the effect of interaction with other objects.
- Adding classes (or even objects) may change the divergence behaviour of existing ones.
- Compositional would be:

  the behaviour of a set of objects is determined by the behaviour of each object "in isolation".

  Our semantics and notion of RTC-step doesn't have this (often desired) property.

Can we give (syntactical) criteria such that any (global) run-to-completion step is an interleaving of local ones?

Run-to-Completion Step: Discussion.

Our definition of RTC-step takes a **global** and **non-compositional** view, that is:

- In the projection onto a single object we still **see** the effect of interaction with other objects.
- Adding classes (or even objects) may change the divergence behaviour of existing ones.
- Compositional would be:

  the behaviour of a set of objects is determined by the behaviour of each object "in isolation".

  Our semantics and notion of RTC-step doesn't have this (often desired) property.

Can we give (syntactical) criteria such that any (global) run-to-completion step is an interleaving of local ones?

Maybe: **Strict interfaces**. *(Proof left as exercise...)*

- **(A)**: Refer to private features only via "self".

  (Recall that other objects of the same class can modify private attributes.)
- **(B)**: Let objects only communicate by events, i.e. don't let them modify each other's local state via links **at all**.

Putting It All Together

Initial States

Recall: a labelled transition system is $(S, A, \to, S_0)$.
We **have**

- S: system configurations (σ, ε)
- $\to$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$.

Wanted: initial states S_0.

Initial States

Recall: a labelled transition system is $(S, A, \to, S_0)$.
We **have**

- S: system configurations (σ, ε)
- $\to$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$.

Wanted: initial states S_0.

Proposal:
Require a (finite) set of **object diagrams** $\mathscr{OD}$ as part of a UML model

$$(\mathscr{CD}, \mathscr{SM}, \mathscr{OD}).$$

And set

$$S_0 = \{(\sigma, \varepsilon) \mid \sigma \in G^{-1}(\mathcal{OD}), \quad \mathcal{OD} \in \mathscr{OD}, \quad \varepsilon \text{ empty}\}.$$

Initial States

Recall: a labelled transition system is $(S, A, \to, S_0)$.
We **have**

- S: system configurations (σ, ε)
- $\to$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$.

Wanted: initial states S_0.

Proposal:
Require a (finite) set of **object diagrams** $\mathscr{OD}$ as part of a UML model

$$(\mathscr{CD}, \mathscr{SM}, \mathscr{OD}).$$

And set

$$S_0 = \{(\sigma, \varepsilon) \mid \sigma \in G^{-1}(\mathcal{OD}), \quad \mathcal{OD} \in \mathscr{OD}, \quad \varepsilon \text{ empty}\}.$$

Other Approach: (used by Rhapsody tool) multiplicity of classes (plus initialisation code).
We can read that as an abbreviation for an object diagram.

Semantics of UML Model (So Far)

The **semantics** of the **UML model**

$$\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$$

where

- some classes in $\mathscr{CD}$ are stereotyped as 'signal' (standard),
  some signals and attributes are stereotyped as 'external' (non-standard),
- there is a 1-to-1 relation between classes and state machines,
- $\mathscr{OD}$ is a set of object diagrams over $\mathscr{CD}$,

is the **transition system** $(S, A, \to, S_0)$ constructed on the previous slide(s).

The **computations of** $\mathcal{M}$ are the computations of $(S, A, \to, S_0)$.

OCL Constraints and Behaviour

- Let $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ be a UML model.
- We call $\mathcal{M}$ **consistent** iff, for each OCL constraint $expr \in$ *Inv*$(\mathscr{CD})$,

  $\sigma \models expr$ for each "reasonable point" (σ, ε) of computations of $\mathcal{M}$.

  (Cf. tutorial for discussion of "reasonable point".)

Note: we could define *Inv*$(\mathscr{SM})$ similar to *Inv*$(\mathscr{CD})$.

OCL Constraints and Behaviour

- Let $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ be a UML model.
- We call $\mathcal{M}$ **consistent** iff, for each OCL constraint $expr \in$ *Inv*$(\mathscr{CD})$,

  $\sigma \models expr$ for each "reasonable point" (σ, ε) of computations of $\mathcal{M}$.

  (Cf. tutorial for discussion of "reasonable point".)

Note: we could define *Inv*$(\mathscr{SM})$ similar to *Inv*$(\mathscr{CD})$.

Pragmatics:

- In **UML-as-blueprint mode**, if $\mathscr{SM}$ doesn't exist yet, then providing $\mathcal{M} = (\mathscr{CD}, \emptyset, \mathscr{OD})$ is typically asking the developer to provide state machines $\mathscr{SM}$ such that $\mathcal{M}' = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ is consistent.

  If the developer makes a mistake, then $\mathcal{M}'$ is inconsistent.
- **Not so common** (but existing):
  If $\mathscr{SM}$ is given, then constraints are also considered when choosing transitions in the RTC-algorithm.
  In other words: even in presence of "mistakes", the state machines in $\mathscr{SM}$ never move to inconsistent configurations.

Rhapsody Demo III: Model Animation

Tell Them What You've Told Them...

- **State Machines** induce a **labelled transition system**.
- There are five kinds of transitions in the LTS:
  - **discard**, **dispatch**, **continue**, **environment**, **error**.
- For now, we assume that all classes are active,
  thus steps of objects may interleave.
- We distinguish **steps** and **run-to-completion step**.
- **Initial states** can be characterised using **object diagrams**.
- Missing transformers:
  - **Create**: re-use identities vs. use fresh ones.
  - **Destroy**: allow dangling references vs. clean up.

→ next time

References

References

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.