
Towards a System Model for UML

Manfred Broy

María Victoria Cengarle

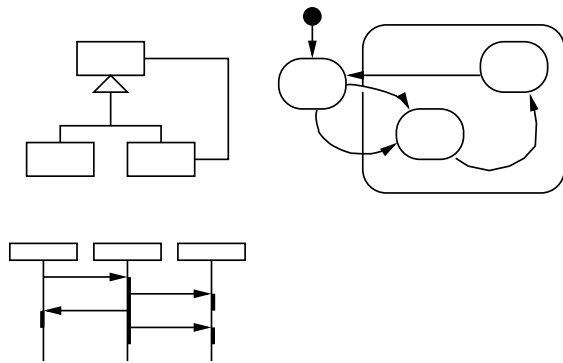
Tech. Univ. Munich

Bernhard Rumpe

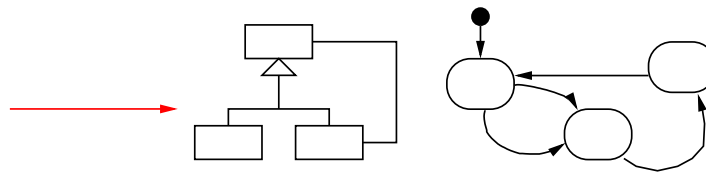
Tech. Univ. Braunschweig

Overall goal

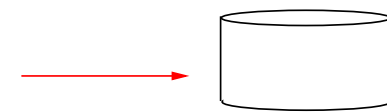
Full UML



Simplified UML



System Model



Contents

- Static part: types and values
- Dynamic part: store and control
- Timed state transition systems and [system models](#)
- Composition, interface abstraction

Static part: Universes

Universes:

- **UTYPE** of types
- **UVAL** of values

provided with a *carrier* function $\text{CAR} : \text{UTYPE} \rightarrow \wp(\text{UVAL})$

and an equivalence \approx among type expressions s. t. for $T_1, T_2 \in \text{UTYPE}$,
 $T_1 \approx T_2 \implies \text{CAR}(T_1) = \text{CAR}(T_2)$.

Static part: Basic types

Bool, Int, Void \in UTYPE

- $CAR(\text{Bool}) = \{\text{true}, \text{false}\}$ (with $\text{true} \neq \text{false}$)

- $CAR(\text{Int}) = \mathbb{Z}$

- $CAR(\text{Void}) = \{\text{void}\}$

- $\text{Bool} \neq \text{Int}$

$\text{Bool} \neq \text{Void}$

$\text{Int} \neq \text{Void}$

Static part: References

If $T \in \text{UTYPE}$, then $\text{Ref } T \in \text{UTYPE}$.

- $\text{CAR}(\text{Ref } T)$ is the set of all references (not explicitly given).
- $\text{Nil} \in \text{CAR}(\text{Ref } T)$ for each $T \in \text{UTYPE}$.
- $\text{deref}_T : \text{CAR}(\text{Ref } T) \setminus \{\text{Nil}\} \rightarrow \text{CAR}(T)$ for each $T \in \text{UTYPE}$.
- $T_1 \approx T_2 \implies \text{Ref } T_1 \approx \text{Ref } T_2$ (axiom)

Static part: Records

Universes (contd.):

- **UVAR** of variable names

If $a_1, \dots, a_n \in \text{UVAR}$ pairwise different and $T_1, \dots, T_n \in \text{UTYPE}$ ($n \in \mathbb{N}$), then $\text{Rec}((a_i : T_i)_{1 \leq i \leq n}) \in \text{UTYPE}$.

- $\text{CAR}(\text{Rec}((a_i : T_i)_{1 \leq i \leq n})) = \{(a_i \rightarrow \text{CAR}(T_i))_{1 \leq i \leq n}\}$
- $a_i = b_{\pi(i)} \wedge T_i \approx T_{\pi(i)} \implies \text{Rec}((a_i : T_i)_{1 \leq i \leq n}) \approx \text{Rec}((b_j : T_j)_{1 \leq j \leq n})$

Static part: Sets and relations

If $T \in \text{UTYPE}$, then $\text{Set}(T) \in \text{UTYPE}$.

- $\text{CAR}(\text{Set}(T)) = \wp_f(\text{CAR}(T))$
- $T_1 \approx T_2 \implies \text{Set}(T_1) \approx \text{Set}(T_2)$

If $T_1, \dots, T_n \in \text{UTYPE}$ ($n \in \mathbb{N}$), then $\text{Rel}((T_i)_{1 \leq i \leq n}) \in \text{UTYPE}$.

- $\text{CAR}(\text{Rel}((T_i)_{1 \leq i \leq n})) = \wp_f(\text{CAR}(T_1)) \times \dots \times \wp_f(\text{CAR}(T_n))$
- $T_i \approx T'_i$ ($1 \leq i \leq n$) $\implies \text{Rel}((T_i)_{1 \leq i \leq n}) \approx \text{Rel}((T'_i)_{1 \leq i \leq n})$

Static part: Recursion

If $T \in \text{UTYPE}$, then $T^* \in \text{UTYPE}$.

- $\text{CAR}(T^*) = \text{CAR}(T)^*$
- $T_1 \approx T_2 \implies T_1^* \approx T_2^*$

Type names can be associated with type expressions.

Type names can be used in place of type expressions;
in this way, (mutual) recursion is allowed.

Static part: Classes

Universes (contd.):

- **UOID** of object identifiers

If $a_1, \dots, a_n \in \text{UVAR}$ pairwise different and $T_1, \dots, T_n \in \text{UTYPE}$ ($n \in \mathbb{N}$), then $\text{Class}((a_i : T_i)_{1 \leq i \leq n}) \in \text{UTYPE}$.

- $\text{CAR}(\text{Class}((a_i : T_i)_{1 \leq i \leq n})) = (\text{UOID} \hookrightarrow \text{CAR}(\text{Rec}((a_i : T_i)_{1 \leq i \leq n})))$
- \approx is the identity on class types.
- The carriers of non-equivalent class types have disjoint domains.

Static part: Classes (contd.)

Inheritance is a preorder (i.e., a reflexive and transitive binary relation) among class types denoted by \preceq and defined by the least preorder including \approx and satisfying:

$$\text{Class}((a_i : T_i)_{1 \leq i \leq n}) \preceq \text{Class}((b_j : T'_j)_{1 \leq j \leq m})$$

$$\text{if } \text{attrs}(\text{Class}((a_i : T_i)_{1 \leq i \leq n})) \preceq \text{attrs}(\text{Class}((b_j : T'_j)_{1 \leq j \leq m}))$$

where $\text{attrs}(\text{Class}((a_i : T_i)_{1 \leq i \leq n})) \stackrel{\text{def}}{=} \{a_1 : T_1, \dots, a_n : T_n\}$

and $\{a_1 : T_1, \dots, a_n : T_n\} \preceq \{b_1 : T'_1, \dots, b_m : T'_m\}$

if for each i there exists T''_i s. t. $a_i : T''_i \in \{b_1 : T'_1, \dots, b_m : T'_m\}$

with $T_i \preceq T''_i$ ($1 \leq i \leq n$).

(Any declared inheritance relation is a preorder subset of \preceq .)

Dynamic part: Store

Universes (contd.):

- ULOC of locations

A store maps locations onto values: $\text{Store} : \text{ULOC} \leftrightarrow \text{UVAL}$

with $\text{dom}(\text{Store}) \subseteq \text{ULOC}$ the set of locations assigned by Store.

STORE denotes the set of all possible stores, and

$\text{Instance}(\text{Store})$ denotes the set

$$\left\{ \text{val} \in \text{UVAL} \mid \begin{array}{l} \exists \text{loc} \in \text{ULOC} . \text{loc} \in \text{dom}(\text{Store}) \\ \wedge \text{Store}(\text{loc}) \in \text{CAR}(\text{Class}(\dots)) \\ \wedge \text{val} \in \text{cod}(\text{CAR}(\text{Class}(\dots))) \end{array} \right\}$$

Dynamic part: Control

(Control is not defined here.)

Let **CONTROL** be the set of control states.

Timed state transition system

A timed state transition system is a tuple $(\text{STATE}, \Delta, I, O, \text{Init})$ with

STATE a set of states,

I and O the input resp. output channel sets,

$\text{Init} \subseteq \text{STATE}$, and

$$\Delta : (\text{STATE} \times T(I)) \rightarrow \wp(\text{STATE} \times T(O))$$

where $T(C)$ is the set of possible channel traces for the channel set C

$$T(C) = \{x : C \rightarrow \text{IM}^* \mid \text{IM} \subseteq \text{UVAL} \text{ is the universe of messages}\}$$

System model

A system model is a timed state transition system with

$\text{STATE} \in \text{STORE} \times \text{CONTROL}$

Timed state transition system (contd.)

Timed state transition systems behave as a Moore machine,⁽¹⁾ that is, the output depends on the state and not on the input:

$$(\sigma', y) \in \Delta(\sigma, x) \Rightarrow \forall x'. \exists \sigma''. (\sigma'', y) \in \Delta(\sigma, x')$$

The state transition function is furthermore total:

$$\Delta(s, i) \neq \emptyset \quad \text{for any } s \in \text{STATE}, i \in T(I)$$

(1) A finite state machine that produces an output for each state.

Timed state transition system: Composition

Two timed state transition systems

$$\begin{aligned} & (\text{STATE}_k, \Delta_k, I_k, O_k, \text{Init}_k) \quad (k = 1, 2) \\ & \text{with } \tau(O_1) \cap \tau(O_2) = \emptyset \end{aligned}$$

can be composed into $(\text{STATE}, \Delta, I, O, \text{Init})$

$$\begin{aligned} \text{where } \text{STATE} &= \text{STATE}_1 \times \text{STATE}_2 & L_1 &= I_1 \cap O_2 \\ I &= (I_1 \cup I_2) \setminus L & L_2 &= I_2 \cap O_1 \\ O &= (O_1 \cup O_2) \setminus L & L &= L_1 \cup L_2 \\ \Delta((s_1, s_2), x) &= \{ ((s'_1, s'_2), z | I) : \exists z \in \tau(L \cup I \cup O) . \\ & \quad (s'_k, z | O_k) \in \Delta_k(s_k, z | I_k) \text{ for } k = 1, 2 \} \end{aligned}$$

The composed state transition function Δ is denoted by $\Delta_1 \otimes \Delta_2$.

Timed state transition system: Interface abstraction

Interface view abstracting away from the local encapsulated state.

Given a timed state transition system $(\text{STATE}, \Delta, I, O, \text{Init})$, the state transition function induces an interface function

$$B[\Delta] : \text{STATE} \rightarrow (\vec{I} \rightarrow \wp(\vec{O}))$$

defined by

$$B[\Delta](\sigma)(\langle z \rangle \hat{x}) = \{ \langle r \rangle \hat{y} : \exists \sigma' \in \text{STATE} . \\ (\sigma', r) \in \Delta(\sigma, z) \wedge y \in B[\Delta](\sigma').x \}$$

where \vec{C} is the set of all valuations of channel C by streams.

Timed state transition system: Composition revisited

An interface function returns an I/O-behaviour. Two such functions

$$F_k : \vec{I}_k \rightarrow \wp(\vec{O}_k) \quad (k = 1, 2)$$

with $O_1 \cap O_2 = \emptyset$

can be composed into $F : \vec{I} \rightarrow \wp(\vec{O})$ defined by

$$(F_1 \otimes F_2)(x) = \left\{ \begin{array}{l} y|O : y \in I_1 \cup I_2 \cup O_1 \cup O_2 \\ \quad \wedge y|I = x|I \\ \quad \wedge y|O_1 \in F_1(y|I_1) \\ \quad \wedge y|O_2 \in F_2(y|I_2) \end{array} \right\}$$

where $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$
 $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2)$

Timed state transition system: Interface abstraction revisited

Proposition. Interface abstraction and composition commute:
given timed state transition systems $(\text{STATE}_k, \Delta_k, I_k, O_k, \text{Init}_k)$ ($k = 1, 2$),
given states $\sigma_1 \in \text{STATE}_1$ and $\sigma_2 \in \text{STATE}_2$,

$$\mathbb{B}[\Delta_1 \otimes \Delta_2](\sigma_1, \sigma_2) = \mathbb{B}[\Delta_1](\sigma_1) \otimes \mathbb{B}[\Delta_2](\sigma_2)$$