# Software Design, Modelling and Analysis in UML Lecture 17: Live Sequence Charts I

The Plan

2017-01-

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg. Germany

Thu, 19, 1: Live Sequence Charts I
Firstly, State-Machines Rest, Code Generation
Toe, 24, 1: Live Sequence Charts II

Thu, 26, 1: Live Sequence Charts III

Thu, 21, 1: Model Based/Driven SW Engineering
Thu, 2, 2: Model Based/Driven SW Engineering
Mon, 6, 2: Inheritance

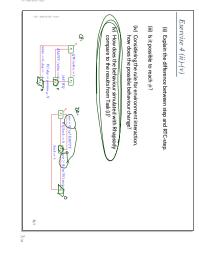
Non, 6, 2: Inheritance

February. 17th: The Exam.

Hierarchical State Machines

Hierarchical State Machines: Retrospective

4/34



Active and Passive Objects

Passive and Reactive / Rhapsody Style: Example

Active and Passive Objects: Nomenclature

A class (and thus the instances of this class) is either active or passive, as defined by the class dagram.
 An active object has (in the operating option series) an own thread: an own program counter, anown stack, etc.

rel and Gery (1997) propose the following (orthogonall) notions:

A class is either reactive or non-reactive.
 A reactive class has a (non-trivial) state machine.
 A non-reactive one hasnit.

A passive object doesn't.

Which combinations do we (not) understand yet?

reactive active passive



Passive Reactive / Rhapsody Style

- is board data, add impedial like  $A_{i+1}A_i$  and date in terminal and oblighter  $A_i$  board materials above  $A_i$  and which is expensible of gradualing event to  $A_i$ . If it is a instance of an active data then  $a_{i+1} = a_i$ . Suppose the frequency object  $A_i$  and  $A_i$  a



### What about non-Active Objects?

- Wée etill working under the assimption
  that all classes in the class diagran front this all objects) are active.
   That is, each object hast its own thread of control and is iff stable)
  at any time ready to process an event from the ether.
   steps of active objects can indectione.

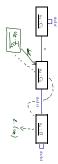
### So we have to address questions like:

- But the world doesn't consist of only active objects.

  For instance, in the Vending Machine from the exercises we could wish to have the whole system live in one thread of control.
- Can we send events to a non-active object?
   And if so, when are these events processed?
   etc.

## Passive Reactive / Rhapsody Style

- In each data addingshell left-full of addict for make each object u. Innot the action object, which is regroteful for dispatching events to u. If u is an instance of an active clock them  $u_v = u$ . Full signals with inspired association dust at use it to point to the destination object. For each signal F, have a ventor  $F_G$  with an association dust it  $G_G$ ,  $G \in G$  for alteritization  $G_G$ .



- Serding an event

  n If in u; ic) becomes

  Caste an instance, of Fc<sub>2</sub> and set u,\$ ident by

  u; = \(\sigma(u\_1)(0), \quad \text{u} \in \sigma(u\_1)(1)\)

  Serd to u, \(\sigma(u\_1)(1)(1)\)

  i.e. \(\text{if} = \varphi(0)(u\_1)(1)\)

- Cosevation: the ether only has events for active ob-icity.

  Say u., is easy in the ether for u.,

  Say u., is easy in the ether for u.,

  Then u., adds of u., J(dest) = u., to ponces u.,

  and vaits unit completion of corresponding RTC.

  u., may in particular discard event.

### Behavioural Features

And What About Methods?

In the current setting, the (local) state of objects is only modified by actions of transitions, which we abstract to transformers.
 In general, there are also methods.

 UML follows an approach to separate the interface declaration from
 the implementation.

In UML the former is called behavioural feature and can (roughly) be

In C++-lingo: distinguish declaration and definition of method.

Note: The signal list can be seen as redundant (can be looked up in the state machine) of the class. But certainly useful for documentation (or sanity check).

 a signal name E • a callinterface  $f(T_{1_1},\ldots,T_{n_1}):T_1$ 



- The implementation of a behavioural feature can be provided by:
- An operation. In our entering we shopk assume a transforme like  $T_I$ , the three general parameters at the first e.g. dust how to shaft method calls a sations or transitions function composition of transformers (clear but colour son-termination).
- h a setting with Java as action language: operation is a method body. • The class' state-machine ("triggered operation").  $\nabla$
- Calling F with n<sub>2</sub> parameters for a stable instance of C course in an availary event F and dispatched in Papassing the ethed.
   Transition actions may fit in the etum value.
   On completion of the RTC step, the call enture.
   For a non-stable instance, the caller blocks unit stability is eached again.

## Passive Reactive / Rhapsody Style



- Sending an event:  $= n(f \cdot n u_1 : C) \text{ becomes:} \\ = n(f \cdot n u_1 : C) \text{ becomes:} \\ = 0 \text{ case an instance, u.of } R_{C,u} \text{ and set } u_1 \text{ $d$ dest to } \\ = u_1 := \sigma(u_1)(n), \\ = 0 \text{ send to } u_1 := \sigma(u_1)(n), \\ = 0 \text{ for } 0 \text{ $d$ $d$ $u_1$ $is$ $d$ $e$ $d$ $is$ $d$$

### And What About Methods?

# Behavioural Features: Visibility and Properties



# Extend typing rules to sequences of actions such that a well-typed action sequence only calls visible methods.

- Useful properties:

- concurrent is thread safe
   guarded some mechanism ensures/should ensure mutual exclusion
   sequential is not thread safe, users have to ensure mutual exclusion
- isQuery doesn't modify the state space (thus thread safe)

A Closer Look to Rhapsody Code Generation

17/34

DL D COPP Main Default Comp. cpp 門圓 CONFILER Default Composant. exc DEDT just memore from so the start by thompstank

References

Harel, D. (1997). Some thoughts on statecharts, 13 years later, in Grumberg, O., editor, CAV, volume 1254 of LNCS, pages 226–231. Springer-Nerlag. Damm, W. and Harel, D. (2001). LSCs: Breathing life into Message Sequence Charts. Formal Nethods in System Design, 19(1):45–80. Crane, M. L. and Dingel, J. (2007). UML vs. classical vs. rhapsody statecharts not all models are created equal Software and Systems Modeling. 6(4):415–435.

Harel, D. and Gery, E. (1997). Executable object modeling with statecharts. IEEE Computer, 30(7):31-42.

References

Harel, D. and Marelly, R. (2003). Come, Let's Play: Scenario-Based Programming Using LSGs and the Play-Engine. Springer-Verlag. Harel, D. and Maoz, S. (2007). Assert and negate revisited: Modal semantics for UML sequence diagrams. Software and System Modeling (SoSyM). To appear. (Early version in SCESMO6, 2006, pp. 13-20).

Klose, J. (2003). LSG: A Graphical Formalism for the Specification of Communication Behavior. PhD thesis, Carl von Ossietzky Universität Oldenburg.

OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

Storiet, H.(2003). Asset (negate and efficiented in UNL-2 intendions. In Julijens, J., Rumpe, B., France, R., and Fernandez, E. B., editors, CSDUNE, 2003, number TUN-10333 Technische Universität München.

34:ni OMG (201a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.
OMG (201b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.

33/34

Tell Them What You've Told Them...

- Rhapsody also supports non-ractive objects their instances share an event pool with an artive object.

  Behavioural Features: exist.

  Semantic Viriation Points are legion but manageable, e.g. by appropriate modelling guidelines (sitck to "the beaten track").

- nteractions can be used for reflective descriptions of behaviour,
- describe what behaviour is (un)desired, without (yet) defining how to realise it.
- One visual formalism for interactions: Live Sequence Charts
- partially ordered locations.
   instantaneous and aynchronous messages.
   conditions and local invariants

Later: pre-charts.