

Advanced Course
16th European Summer School in Logic, Language and Information
(ESSLLI)
Nancy, 9–20 August 2004

Cognitive Modeling and Verbal Semantics

Andrea C. Schalley
University of New England, Australia
andrea.schalley@une.edu.au

What this course is about

- This advanced course deals with lexical semantics which we will approach from a software engineering point of view.
- More precisely, we will be dealing with verbal meaning.
- You will be introduced to a new way of representing verbal semantics, the *Unified Eventivity Representation* (UER), a diagrammatic representational framework designed to encode verbal meaning.
- I hope that you enjoy the course, and I welcome feedback!

What we're going to do

- We will ask ourselves what we can learn from computer science.
—> What does a third programming paradigm from computer science, namely object-orientation, have to do with semantics and conceptualization?
- We will draw conclusions from our observations.
—> We will develop the UER as an adaption of the UML.
- The presented *Unified Eventity Representation* is neither logical nor functional in nature.
—> We will look at a new paradigm within linguistic semantics.

Where we're going

- Session 1: Decompositional approaches to verbal semantics
- Session 2: Requirements for adequate semantic representations
- Session 3: Overview of the *Unified Modeling Language* (UML)
- Session 4: The *Unified Eventity Representation* (UER)
- Session 5: Application of the UER and discussion

TODAY Session 1:

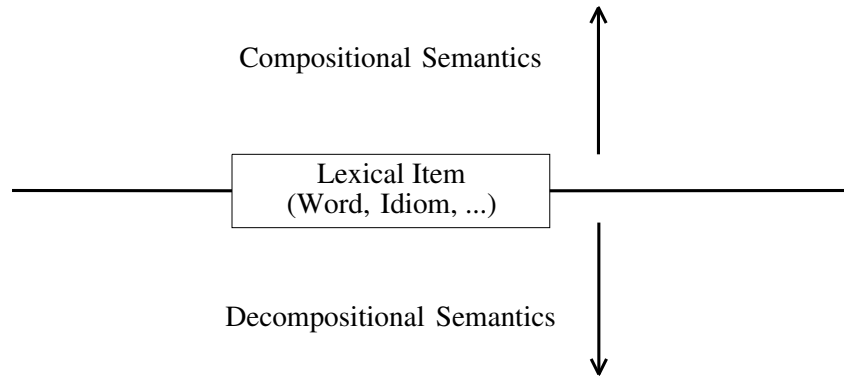
Decompositional Approaches to Verbal Semantics

- We review some of the prevailing theories:
 - Generative Semantics;
 - Jackendoff's Conceptual Semantics;
 - Wunderlich's Lexical Decomposition Grammar;
 - Pustejovsky's Generative Lexicon.
- These theories and their discussion build the basis for tomorrow's session.

What is decomposition? [1/3]

- Decompositional semantics can roughly be seen as the converse process to compositional semantics.
- *Compositional semantics* determines the semantics of a complex linguistic expression (such as a clause or sentence, e.g. *The wind woke Eleni up*) on the basis of its components (such as lexical items) and structure.
- *Decompositional semantics* tackles what is assumed to be given in compositional semantics: the aim is to represent the semantics of a lexical item (such as *wecken* 'wake up') in terms of structured semantic components inherent in its semantics.

What is decomposition? [2/3]



Decomposition is analysis into meaning components. Meanings of lexical items are considered to be complex and composed of more general components (Löbner 2002: 125), i.e., more 'basic' meanings.

What is decomposition? [3/3]

- Decompositional representations can thus be characterized as structured configurations, the components of which are 'basic' (whatever this may mean).
- They may be 'formal' or 'informal', i.e., they may use a formal language (such as Predicate Logic) or natural language. However, we concentrate on rigorous approaches.
- Decompositional approaches do not conceive of the semantics of lexical items as holistic units.
- There are many different decompositional approaches in the literature. We can only briefly look at a few of these.

What can we do with compositional analysis? [1/2]

Here are some objectives mentioned in Löbner 2002 (125f.):

- provide models for word meaning [or lexical meaning in general];
- reduce the variety of lexical meanings to a limited number of basic meanings;
- provide a means for a precise interpretation of lexical items;
- explain meaning relations within and across lexical fields;
- explain the compositional properties of lexical items;
- determine semantic relations between expressions of different languages.

What can we do with compositional analysis? [2/2]

Apart from the points mentioned by Löbner 2002 (which are genuine linguistic in nature):

Decomposition is inevitable for computational applications
—→ internal structure of lexical items must be rigorously representable, such that for instance (with appropriate tools):

- semantics of lexical items can be computationally represented;
- automatic inference becomes possible;
- lexical relations can be modeled (in particular, systematic ones).

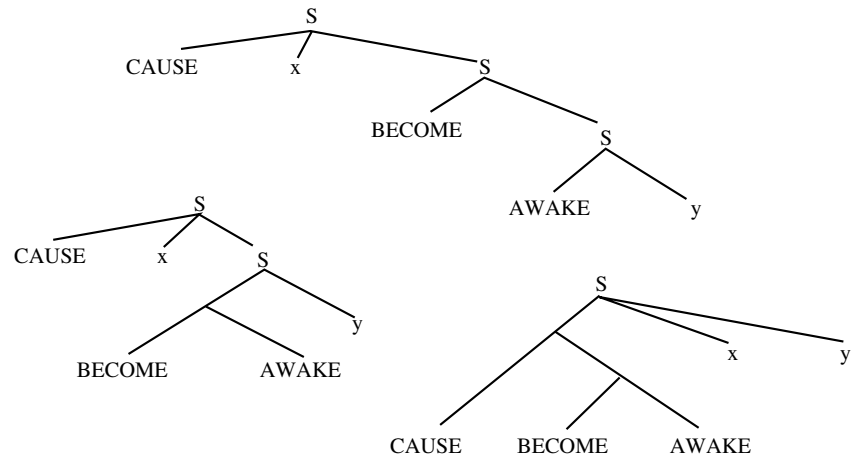
Why look at predecessing decompositional approaches?

- The rest of today will be devoted to an overview of some decompositional approaches.
- When briefly looking at the approaches, we should ask:
 - what is good;
 - which features the approach includes;
 - which problems arise.
- This will give us a 'wish-list' for our approach.

Generative Semantics – Background

- Generative Semantics was a highly-influential approach towards representing the meaning of lexical items in a structured way.
- It was developed from the 1960s by Lakoff, McCawley, Ross and others.
- Meaning is displayed as a structured complex of primitive elements.
- It implicitly includes a few points that are important.

Generative Semantics – Example



(cf. McCawley 1976)

Generative Semantics – Discussion [1/2]

- The representation is a tree-structure with predicates (such as CAUSE, BECOME) and variables for participants.
- The underlying semantic representation as given in the first tree is merely motivated by Chomsky's (1965) abstract syntax, i.e., not from a semantic point of view.
- Apart from the number of participants, only dynamic aspects of the underlying eventivity are captured.
- Semantic roles, ontological categories or generally selectional restrictions of participants are not included – i.e., static aspects of verbal meaning are neglected.

Generative Semantics – Discussion [2/2]

- The different status of the predicates is only present implicitly: whereas predicates such as CAUSE and BECOME are markers for causation and transition (change of state), the status of NOT or AWAKE as predicates is unclear.
- The prominent status of some predicates (included are CAUSE, BECOME, and later DO) can be seen as an important step towards what is today called 'basic predicates'.
- However, Generative Semantics has not systematically tackled this question (e.g., a statal predicate such as BE is missing).

Jackendoff's Conceptual Semantics – Background [1/2]

- Starting point: syntactic phrases are considered to correspond to certain structured conceptual entities. Example:

$$[_S [_{NP} \text{Bill}] [_{VP} [_V \text{went}] [_{PP} [_P \text{into}] [_{NP} \text{the house}]]]]$$

$$[_{Event} \text{GO} ([[_{Thing} \text{BILL}], [_{Path} \text{TO} ([[_{Place} \text{IN} ([[_{Thing} \text{HOUSE}]]])])])])]$$
- Basic hypothesis: there is a form of mental representation (*conceptual structure*) that is common to all natural languages
 → it serves as a 'syntax of thought' (Jackendoff 1991: 10).

Jackendoff's Conceptual Semantics – Background [2/2]

The primitives of Jackendoff's early functional semantic theory are:

- **conceptual constituents** (in square brackets)
 “Each constituent is labeled as belonging to a *major conceptual category* or ‘semantic part of speech’ – one of the kinds of entities the world is conceptualized as containing, for example Thing (or physical object), Event, State, Path (or trajectory), Place (or location), Property, Time, and Amount.” (Jackendoff 1991: 13)
- **functions** (rendered in small capitals) – mapping conceptual categories onto conceptual categories; in the example IN, TO, and GO.

Jackendoff's underlying functions [1/2]

Jackendoff relies on a motional model (‘adaptation of available machinery’); “all predicates are analysed by means of the semantic functions based on motion verbs” (Ravin 1990: 78).

1. [PLACE] → [*Place* PLACE-FUNCTION([THING])]

$$2. \text{ [PATH] } \rightarrow \left[\begin{array}{c} \left\{ \begin{array}{c} \text{TO} \\ \text{FROM} \\ \text{TOWARD} \\ \text{AWAY-FROM} \\ \text{VIA} \end{array} \right\} \left(\left[\left\{ \begin{array}{c} \text{THING} \\ \text{PLACE} \end{array} \right\} \right] \right) \\ \textit{Path} \end{array} \right]$$

Jackendoff's underlying functions [2/2]

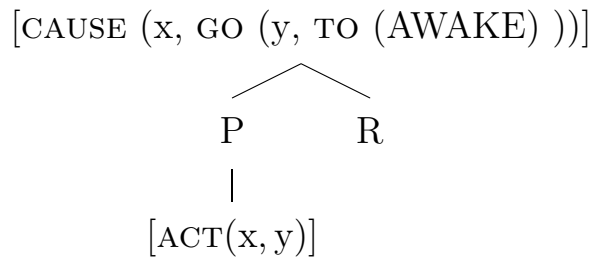
$$\begin{aligned}
 3. \quad [EVENT] &\rightarrow \left\{ \begin{array}{l} [Event \text{ GO } ([THING], [PATH])] \\ [Event \text{ STAY } ([THING], [PLACE])] \end{array} \right\} \\
 4. \quad [STATE] &\rightarrow \left\{ \begin{array}{l} [State \text{ BE } ([THING], [PLACE])] \\ [State \text{ ORIENT } ([THING], [PATH])] \\ [State \text{ EXT } ([THING], [PATH])] \end{array} \right\} \\
 5. \quad [EVENT] &\rightarrow \left[\begin{array}{c} \text{CAUSE} \left(\left[\left\{ \begin{array}{c} THING \\ EVENT \end{array} \right\} \right], [EVENT] \right) \\ Event \end{array} \right]
 \end{aligned}$$

Jackendoff's Conceptual Semantics – Refinements

Later on, Jackendoff includes in the conceptual structure:

- an **action tier**, indicating agent and patient;
- a **temporal tier**, reflecting aspectual properties ('point' and 'region' in time);
- **selectional restrictions**, limiting the range of entities permissible as arguments;
- **semantic markers**, indicating ontological categories of arguments;
- and finally, he moves towards a more **abstract, feature-based** metalanguage.

Jackendoff's Conceptual Semantics – Example



Jackendoff's Conceptual Semantics – Discussion [1/2]

- Conceptual Semantics includes many features that are important when representing verbal semantics coherently, such as ontological categories, selectional restrictions, and aspectual properties (the first two unfortunately not for agent and patient – this leaves representations as in the example too general).
- Later Conceptual Semantics includes 'point' and 'region' in time, i.e. a distinction between durative and non-durative parts of eventities.
- The different status of the components of semantic representation is formally accomplished for (different ways of display).

Jackendoff's Conceptual Semantics – Discussion [2/2]

- A typization/ontology is entailed via the 'conceptual constituents'.
- Jackendoff's approach is a predicate dependent approach (semantic roles are reduced to structural configurations in conceptual structure), i.e., semantic roles are not explicated.
- Dynamic and static aspects of the eventivity are captured, but could be distinguished more explicitly.
- Relations between participants cannot be expressed.
- Yet, the refinements point out important components that need to be included in a sensible semantic representation framework.

Wunderlich's Lexical Decomposition Grammar – Background [1/2]

- LDG is based on the Two-Level-Semantics (cf. Bierwisch 1982, 1983, Bierwisch and Lang 1987).
- The Two-Level-Semantics distinguishes between two levels of representation: *semantic form* (SF, computational level of grammar) and *conceptual structure* (CF, level of reasoning).
- The SF is a "level at which complex verbs are minimally decomposed into more basic predicates" (Wunderlich 1997: 27).
→ Lexical decomposition does not need to go beyond a certain degree of granularity.

Wunderlich's Lexical Decomposition Grammar – Background [2/2]

- The LDG works on the assumption that SF is a predicate-argument representation of the invariant semantic components of lexical items.
- The components are part of a representation which may be compared to the logical structure or logical form [used in compositional semantics] extended to sublexical structures (Wunderlich 1997: 29).
- The LDG deals primarily with the SF; there is no elaborated theory of CS representations so far (which is to include implicit arguments, sortal restrictions, thematic roles, temporal/clausal structure, subevents, and the like, cf. Wunderlich 2000: 254).

Wunderlich's Lexical Decomposition Grammar – Example

Example 1 is an 'explicit' causative verb, Example 2 an 'implicit' causative verb:

"It is important to notice that a distinction is being made between verbs that only express the result of an activity and those that in addition express a particular kind of activity. The former are represented explicitly with CAUSE, whereas the latter encode the causal interpretation implicitly." (Wunderlich 1997: 57–58)

1. CAUSE (x, BECOME (AWAKE(y)))(s)
2. {SHAKE(x) & BECOME (AWAKE(y))}(s)

Wunderlich's Lexical Decomposition Grammar – Discussion [1/3]

- The CS includes many important points for semantic representations, but they are not considered to be part of the 'invariant semantic components of lexical items'.
→ This is problematic for concise semantic representations (such as those we are aiming at).
- The lack of a clear conception of the CS furthermore does not allow to deduce the SF from the "rich conceptual structure" (Wunderlich 1997: 30) of the CS.

Wunderlich's Lexical Decomposition Grammar – Discussion [2/3]

- Apart from the lack of the components that are deferred to the CS in the representation, temporal and causal structure cannot be distinguished and appropriately represented by LDG structures:
- For the correct interpretation of Example 2 an 'add-on' to the semantics of the predicate-argument structure is necessary, the *general coherence constraint*: a "lexical SF conjunction is contemporaneously or causally interpreted" (Wunderlich 1997: 36).
- This brings other formal problems with it, but in any case (even if we overlook those formal details):

Wunderlich's Lexical Decomposition Grammar – Discussion [3/3]

- The two different representations in the examples do not match our conceptualization of *wecken* 'wake up' and *wachrütteln* 'wake up by shaking':
the latter also specifies the activity of *x*, which the former does not, i.e., the representations should differ only with respect of the specification of the activity, but not in their general structure.
- That is, LDG representations are not as intuitive and consistent as desirable.

Pustejovsky's Generative Lexicon – Background [1/3]

- "[T]he meanings of words should somehow reflect the deeper conceptual structures in the cognitive system, and the domain it operates in. This is tantamount to stating that the semantics of natural language should be the image of nonlinguistic conceptual organizing principles, whatever their structure." (Pustejovsky 1995: 6)
- The lexicon is highly organized and equipped with a remarkable degree of systematicity and generativity.
- The Generative Lexicon approach is designed to e.g. dynamically capture systematic polysemy.

Pustejovsky's Generative Lexicon – Background [2/3]

- Pustejovsky aims at avoiding 'sense enumeration lexicons'.
- Lexical entries are multi-level representations with the levels **argument structure** (specification of the number and types of the logical arguments), **event structure** (definition of event type, subeventities and their temporal ordering), **qualia structure** (there are four roles to explain generativity and that are responsible for the relational force of the lexical item), and **lexical inheritance structure** (identification how a lexical structure is related to other lexical structures).

Pustejovsky's Generative Lexicon – Background [3/3]

- **Argument structure**: list that includes some ontological categories/classifications.
- **Event structure**: list attributing event types to subeventities (following Pustejovsky [1991: 55–56], any verb [and subeventity] in natural language can be characterized as belonging to one of the basic event types state, process, or transition), plus ordering restrictions on subeventities.
- **Qualia structure**: list outlining those qualia that are important to e.g. explain the compositional properties of the lexical item.

Pustejovsky's Generative Lexicon – Qualia

- CONSTITUTIVE: the relation between an object and its constituent parts;
- FORMAL: that which distinguishes it within a larger domain;
- TELIC: its purpose and function;
- AGENTIVE: factors involved in its origin or 'bringing it about'.

(Pustejovsky 1995: 76)

Pustejovsky's Generative Lexicon – Example

$$\left[\begin{array}{l} \text{wecken} \\ \text{EVENTSTR} = \left[\begin{array}{l} e_1 = e_1 : \text{process} \\ e_2 = e_2 : \text{state} \\ \text{RESTR} = <_{\alpha} \\ \text{HEAD} = e_2 \end{array} \right] \\ \text{ARGSTR} = \left[\begin{array}{l} \text{ARG}_1 = \boxed{1} \left[\begin{array}{l} \text{entity} \\ \text{FORMAL} = \text{physobj/kausobj} [??] \end{array} \right] \\ \text{ARG}_2 = \boxed{2} \left[\begin{array}{l} \text{animate_individual} \\ \text{FORMAL} = \text{physobj} \end{array} \right] \end{array} \right] \\ \text{QUALIA} = \left[\begin{array}{l} \text{natural_state-lcp} \\ \text{FORMAL} = \text{awake}(e_2, \boxed{2}) \\ \text{AGENTIVE} = \text{act}(e_1, \boxed{1}, \boxed{2}) \end{array} \right] \end{array} \right]$$

Pustejovsky's Generative Lexicon – Discussion [1/3]

- Semantic representations include participants, their selectional restrictions or ontological categories, subeventities, and their basic eventity types.
- The Generative Lexicon includes the temporal ordering of eventities, although this is (conceptually) a complex mechanism which needs to know whether there is a sequence, an overlap, or an inclusion.
- Cross-references between arguments can be built into the representations, i.e., relations between participants can be indicated (but not explicitly modeled).

Pustejovsky's Generative Lexicon – Discussion [2/3]

- The system entails basic event types (state, process, transition) as structural primitives.
- The qualia structure in representations of verbal meaning relates event structure and argument structure of the modeled verb → this way participants are assigned to subeventities; it is explicated which participant takes part in which subevent
- The qualia structure relates dynamic and static aspects of verbal semantics, which are kept apart in Pustejovsky's representation.

Pustejovsky's Generative Lexicon – Discussion [3/3]

- And: the qualia structure fills the structural frame of the dynamics of the eventivity (present in the event structure) with content
→ structure and content are kept apart with respect to the dynamic modeling.
- It is not clear why Pustejovsky assumes that any verb (and hence the underlying eventivity) can be characterized as belonging to one of the basic event types.
- Gradual changes cannot be modeled (Rapp 1997: 29); and semantic roles are not included in the representation.

TODAY Session 2: Requirements for Adequate Semantic Representations

- We summarize and give an overview of the different theories' features we have seen yesterday.
- Thereby, we put together our 'wish-list'. What do we want of a representational framework for verbal semantics?
- We look at a different representation of *wecken* 'wake up', which is object-oriented in nature and based on the UML.

Summary of the different theories [1/3]

	Generative Semantics	Jackendoff	Wunderlich	Pustejovsky
Representation distinctions	✗	✓	✗	✓
Static aspects (apart participant #)	✗	✓	✗	✓
Dynamic aspects	✓	✓	✓	✓
Distinction static-dynamic	✗	✗	✗	✓
Structure vs. content	✗	✗	✗	✓

Summary of the different theories [2/3]

	Generative Semantics	Jackendoff	Wunderlich	Pustejovsky
Subeventuality structure	✗	✗	✗	✓
Basic eventuality types	✗	✗	✗	✓
Temporal modeling	✗	✓	✓	✓
Causal modeling	✓	✓	✓	✓
Durativity	✗	✗	✗	✗

Summary of the different theories [3/3]

	Generative Semantics	Jackendoff	Wunderlich	Pustejovsky
Semantic roles	X	X	X	X
Selectional restrictions	X	✓	X	✓
Ontological categories	X	✓	X	✓
Participant relations	X	X	X	X
Inheritance	X	X	X	X

What have we learned so far?

- Many different details play a role in verbal meaning (static and dynamic ones, relations are important – our ‘wish-list’ is expressed by the points in the three tables we’ve just seen).
- Different components should be represented differently.
- Semantic descriptions need to be rich, highly-structured configurations.
- Representations should be cognitively adequate, i.e., what belongs conceptually together should be modeled together.

Which criteria should we impose on a compositional approach?

- **COGNITIVE ADEQUACY** – it should be intuitive in order to be accepted as representing human conceptualization (this is what we must aim at if we want to find viable models of meaning → mental representations have to be modeled).
- **UNIVERSALITY**– it should be able to account for all phenomena related to verbal meaning in any natural language.
- **EXPRESSIVENESS** – any verbal meaning should be representable.
- **FORMALITY** – it should be precisely grounded, not only for computational approaches but also for rigorous semantic analysis.

What do these criteria mean for our own approach? [1/3]

FORMALITY

- We need a rigorous representational framework, i.e., a framework whose syntax and semantics is precisely specified.
- In other words, we need a clearly and concisely specified metalanguage for our semantic representations.
- Given the fact that software engineering necessitates cognitive modeling (in the design and specification of software requirements), we might find a well-elaborated modeling language ready to adapt in computer science.

⇒ WE BASE OUR FRAMEWORK ON THE
UNIFIED MODELING LANGUAGE (UML)!

What do these criteria mean for our own approach? [2/3]

UNIVERSALITY / EXPRESSIVENESS

- Our framework should be founded language-independently in cognition, i.e., it should model cognitive structures.
- The framework should contain as many different conceptual components as we need to display the information coherently.
- General extension mechanisms to account for yet undiscovered or unexpected phenomena are desirable.
- A distinction between structural features and specific content draws an intuitive line: structural characteristics are more likely to be universal and in principle stable, because they incorporate basic human experience.

What do these criteria mean for our own approach? [3/3]

COGNITIVE ADEQUACY

Components of representations should be arrangeable in a way they are conceptually arranged:

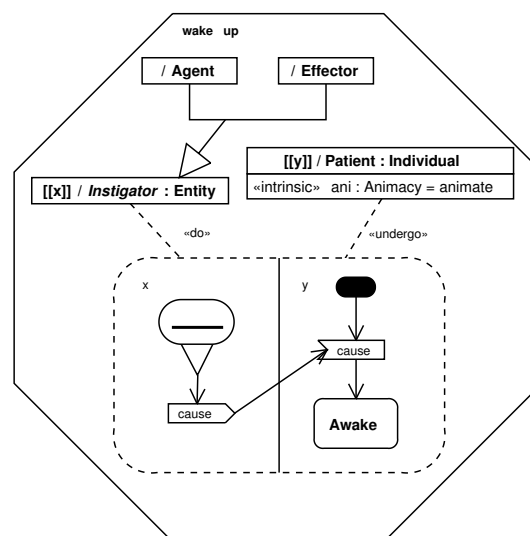
- Characteristics or properties of modeled entities must be expressible within or attached to the representations of these entities.
- Relations must be expressible, even better so if relations are expressed as relations/connections.
- Different kinds of concepts (e.g., relation types) should be represented differently, i.e., structural 'meta-concepts' should be distinguished.
- Nesting should be possible in the modeling to represent cognitive nesting.

Basic assumptions we work on

The criteria and a cognitive approach in general lead us to the following foundation we work on:

- Meaning components are cognitive constructs.
- Eventities are composed of conceptual components (cf. also Bierwisch and Schreuder 1992: 23).
- Löbner's basic meanings should therefore be not genuine words or lexical items, but underlying conceptual elements.
- The display of the internal structure of a verb's reading is hence tantamount to the display of conceptual structure.

Unified Eventity Representation – Example



UER – Preliminary discussion

Representation distinctions

Static aspects

Dynamic aspects

Distinction static-dynamic

Structure vs. content

Subeventivity structure

Basic eventivity types

Temporal modeling

Causal modeling

Durativity

Semantic roles

Selectional restrictions

Ontological categories

Participant relations

Inheritance

TODAY Session 3: Overview of the *Unified Modeling Language* (UML)

- We will first have a look at how cognitive adequacy is achieved through object-orientation in computer science.
- Then we will introduce the UML by examining its
 - nature as modeling language;
 - architecture as modeling system;
 - most prominent types of modeling devices.
- Finally, we will summarize our observations with respect to our criteria.

Cognitive adequacy and object-orientation [1/2]

- Requiring a system to be cognitive adequate means this system should properly reflect human conceptualization, which centers around entities and their characteristics, relations, interactions, and behavior.
- In computer science this is achieved in object-oriented systems. The basic idea is that an object-oriented system comprises objects or, prototypically, 'entities out there in the world', along with their attributes and communication patterns.

Cognitive adequacy and object-orientation [2/2]

- The dynamic behavior of a group of objects is represented by their communication patterns. That is, an object's behavior is described by the way it reacts to stimuli from the 'outside world' and also under which circumstances it stimulates other objects.
- Another aspect of cognition is how we deal with abstraction. Object-oriented systems feature several very useful ways for doing so. Two of them are
 - the distinction between types and instances and
 - inheritance of structure and behavior among types.

Cognitive adequacy and modeling

- A way to view human cognition is that in perceiving our surrounding reality we build internal representations for it. These small and conceptually reduced copies of reality form models.
- The creation of models as simplification and reduction to relevant aspects of reality is a valuable tool in any engineering science, including software engineering.

⇒ LOOK AT SUITABLE OBJECT-ORIENTED MODELING TOOLS!

A Unified Modeling Language (UML)

- The UML is a diagrammatic modeling tool for object-oriented systems in computer science. It is the result of a unification of three very successful object-oriented modeling approaches out of the more than 50 created over the last two decades.
- It has been widely adopted as the standard notation for modeling in software engineering, as it represents the essence and refinement of the research that has been done in this area.
- We will now have a look at the UML's nature and architecture.

UML – Nature [1/2]

- The UML is a combination of graphical (lines, shapes, icons) and textual elements arrayed in two-dimensional diagrams that form a language with:
 - a lexicon consisting of graphical representations for common object-oriented modeling elements,
 - syntax in the form of rules for notation especially for combinations of elements,
 - and semantics defined in the standardization documents through natural language description, the UML meta-model architecture, and accompanying well-formedness rules.

UML – Nature [2/2]

- Several types of diagrams allow for the representation of different aspects of the modeled group of objects, e.g. static structure and dynamic behavior.
- The graphical elements play a key role in structuring the information. Visual grouping of elements facilitates an identification of objects and parts thereof. Temporary or permanent omission of irrelevant elements allows a display of different levels of granularity. The different diagrams emphasize different aspects of the model. This is the foundation for the object-oriented nature and cognitive adequacy of the UML.

UML – Architecture [1/2]

The UML is built in four layers, each of which consists of instances of elements from the layer above it:

- The *instance model* describes a set of concrete objects ('instances') which form an actual part of the modelled reality or system, e.g. a desk.
- The *user model* conceptually represents abstractions, common properties, and behaviour of sets of objects ('classes') and their relations among each other, e.g. the class of all desks. The act of building a user model is commonly understood as modeling in the UML.

UML – Architecture [2/2]

- The *meta-model* contains the definition of the UML in that it describes abstractions and common properties of modeling elements available in the user model ('meta-classes'), e.g. the concept of a class itself and similar concepts like attributes of a class.
- There even is a *meta-meta-model* which can be instantiated in order to formally define meta-models like the one for the UML. At a later point in time it may be used to create a formal definition of a meta-model for the UER.

UML – Diagrams

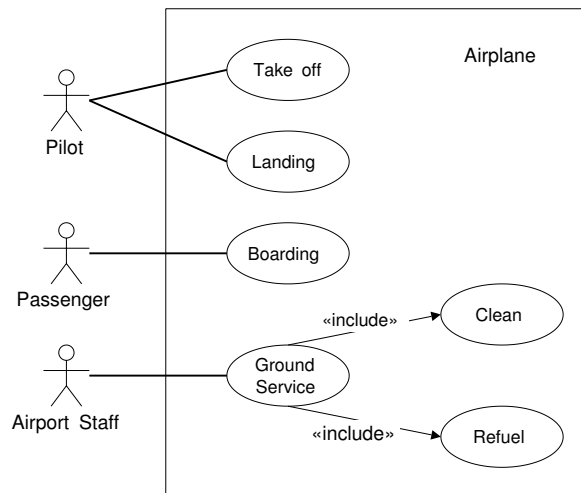
A diagram is a view onto a model. In order to be able to emphasize the different ‘angles’ one can view a model from, there are different kinds of diagrams:

- Static structure diagrams address questions like: Which objects does the model consist of? What is their internal structure? How do these objects relate to each other?
- Dynamic behavior diagrams address questions like: How does each object itself behave and how do the different objects interact? What is the reaction to outside stimuli?

Use case diagram

- It displays (a part of) the functionality of the modeled system in the form of coherent units called *use cases* which are accessible to the outside of the modeled system.
- Also *actors* are shown in order to represent entities which interact with the system from the outside. These actors could be anything from human users to other systems.
- Additionally, it may present relationships among use cases (inclusion or extension) and relationships among actors (e.g. generalization).

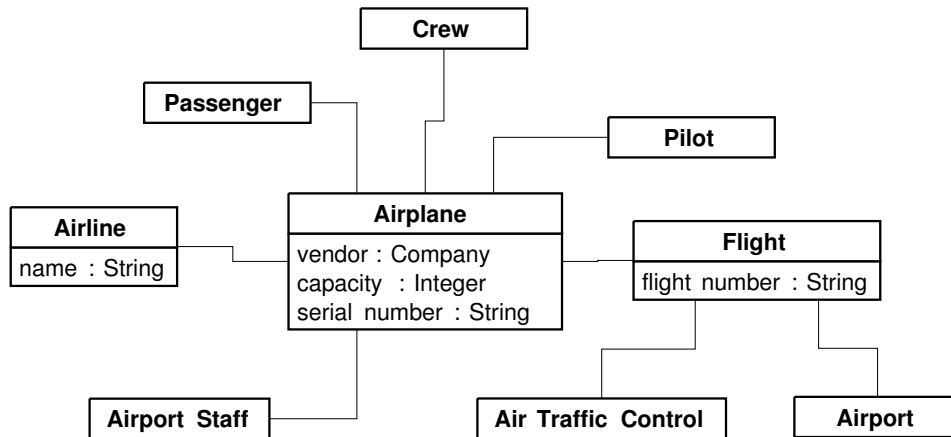
Use case diagram – Example



Class diagram

- It depicts the structure of one or more entities in the user model, i.e. classes, separated into rectangular compartments. At least the name compartment is always present and always shown.
- The displayed information of a class may also encompass the external associations and relations to other classes in the form of lines with various adornments, as well as their internal structure in the form of list compartments containing lists of typed attributes as textual strings.
- For software purposes the class diagram lists the possible operations that can be performed by instances of this class.

Class diagram – Example

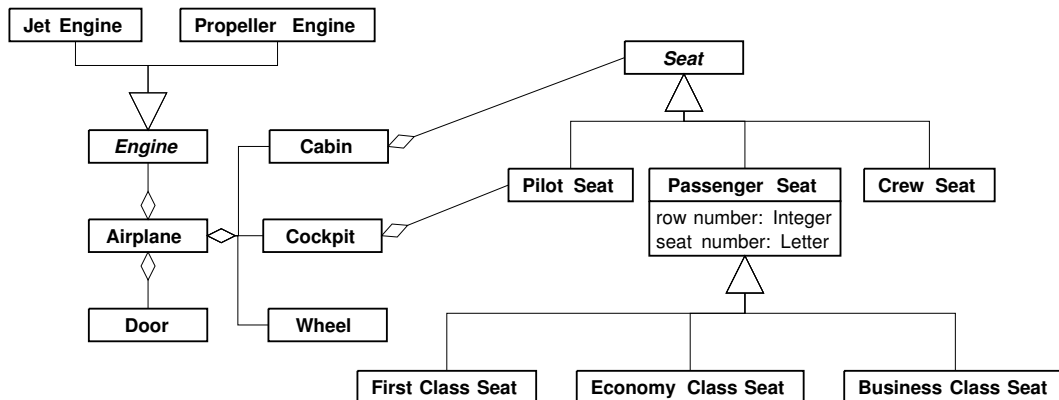


Class diagram – Aggregation and generalization

Two prominent kinds of relations between classes are:

- Aggregation, which basically represents a part-whole relationship between instances of the involved classes. It is depicted by a hollow diamond at the whole's end of the solid association line.
- Generalization, which expresses commonalities of classes in the way that all classes with a common parent share its attributes. This relation is noted by a solid line between the class and its parent, with a hollow triangle pointing at the latter.

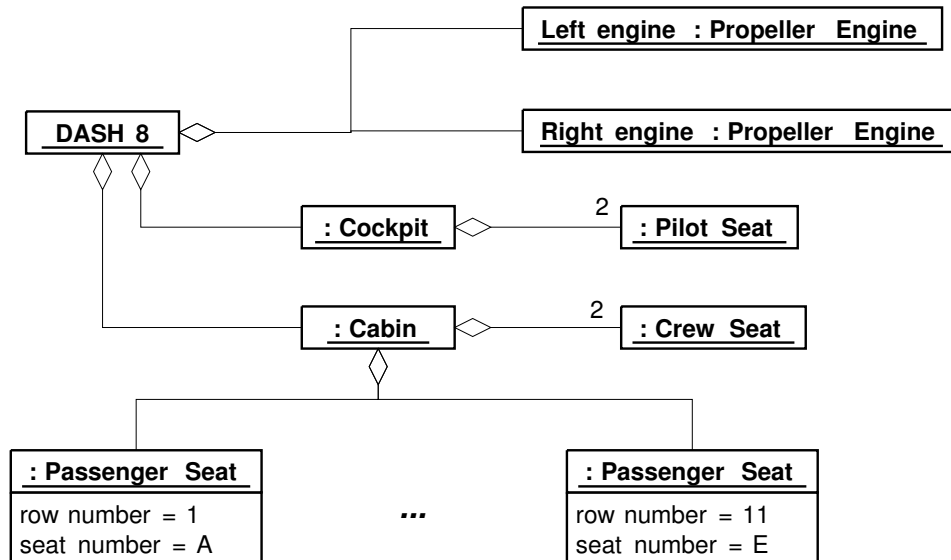
Class diagram – Aggregation and Inheritance Example



Class diagram – Instances

- The classes residing on the user model level can be instantiated to form objects on the instance model level.
- Notationally objects and classes have the same structure, but the attributes which serve as placeholders for values of certain types in classes are assigned literal values of the proper type. Classes hence represent the common structure of a group of objects.
- As instances have the type of the class they are an instance of, the name of an object has the form of an underlined 'Object Name : Class Name' string.

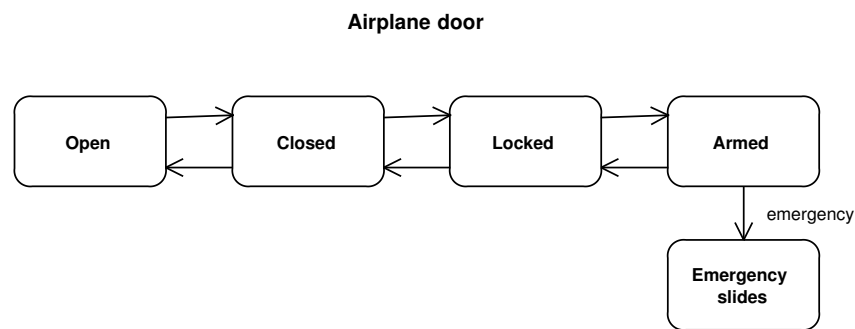
Class diagram – Instances Example



Statechart diagram

- The dynamic behavior of an object – its reaction to outside stimuli – is shown in the form of a state machine.
- Such a state machine consists of the different states an object may be in at a given time (depicted by rectangles with rounded corners) and the possible transitions (lines with arrowheads) between these states. This can also be interpreted as a life history of an object where the states represent stages in this object's lifetime.
- The transitions usually show the events which possibly cause the transition, that is, completion of an operation (completion event), elapse of time (time event), change of a watched value (change event), or an external signal (signal event).

Statechart diagram – Example

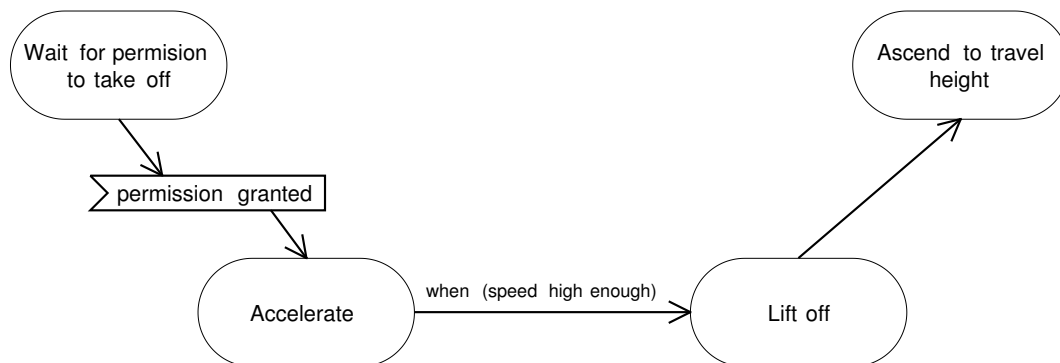


Activity diagram

- This diagram is similar to a statechart diagram, but it consists of action states instead of the states found in statechart diagrams.
- Action states represent the performing of actions like computational activities or steps in a workflow and hence have an active aspect, which the states in a statechart diagram lack.
- An activity diagram may contain *signals*, i.e. asynchronous stimuli communicated between instances and *control icons* for signal receipt and sending.

Activity diagram – Example

Airplane pilot performing take off

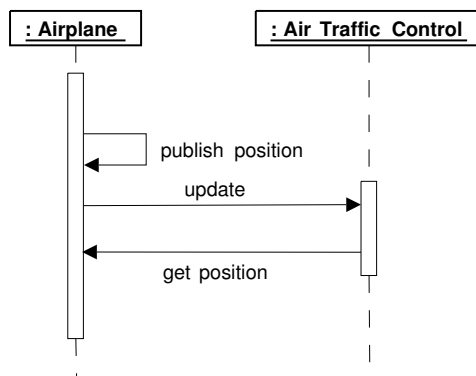


Interaction diagrams

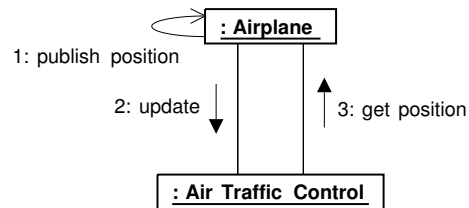
- There are two semantically equivalent notations for an interaction between a set of objects in the context of an executed operation or a result that is to be achieved.
- Both show an explicit sequence of communications between the set of objects but they emphasize different aspects of thereof.
 - In sequence diagrams the time ordering of the stimuli is one prominent dimension which makes them useful for modeling complex scenarios.
 - In collaboration diagrams the interaction of objects is organized around their structural relations and roles they play in the interaction.

Interaction diagrams – Example

Sequence Diagram

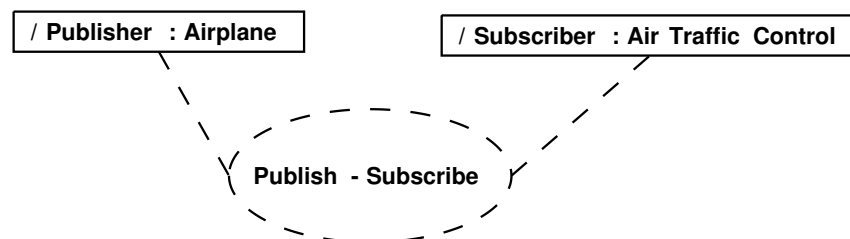


Collaboration Diagram



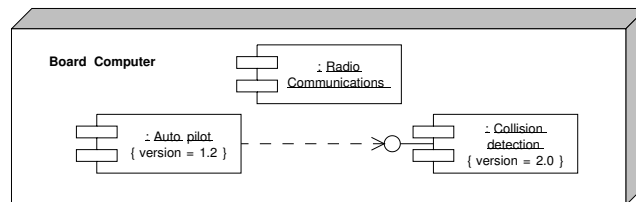
Collaboration

- In order to combine static structure and dynamic behavior, collaborations are used. The name of a collaboration is shown in an oval with a dashed outline. All classes which are relevant for a collaboration are connected to the oval with dashed lines and are assigned a role name.



Implementation diagrams

- A component diagram displays the physical realization of parts of the modeled system, their interfaces and relationships. These *components* include binary code libraries and source code files.
- Concrete run-time processing configuration of software and hardware systems is depicted in the form of nodes and component instances in an deployment diagram.



The UML – Observations [1/2]

- The UML is
 - object-oriented;
 - easily adaptable and extensible;
 - versatile;
 - graphical.
- The UML distinguishes
 - different views on a system;
 - types and instances;
 - structures and substructures (nesting);
 - concepts and their relations.

→ It actively supports cognitive adequacy.

The UML – Observations [2/2]

Furthermore,

- the UML is a rigorous modeling language;
- its extensibility and lightweight visual way of conveying information enhances expressiveness;
- the graphical approach also lends itself to a language-independent way of representing semantics and thus fosters universality;
- hence, the UML meets the criteria established yesterday (cognitive adequacy, formality, expressiveness, and universality) to a great extent.

Therefore, the obvious conclusion is to adapt it to meet our needs, and to introduce object-orientation as a new paradigm into linguistic semantics.

TODAY Session 4: The *Unified Eventity Representation* (UER)

- Having seen the basic ideas of the UML, we will introduce the UER more explicitly than we have done so far.
- We will take up the example from Day 2, English *wake up* (or German *wecken*), and revisit it in the light of what we have seen yesterday.
- In introducing the UER, we will discuss the relevance of the modeling elements within linguistic semantics and within our representational framework.

Relation between the UER and the UML [1/2]

- The UER and the UML share the architecture and the meta-metamodel.
- Although not explicitly specified so far, the UER's metamodel differs from the UML's, i.e., the UER is not just what the UML-Specification (OMG 2001) calls a 'profile' of the UML.
- The UER has the same visual relationships – connections, containment, visual attachment – and kinds of graphical constructs – icons, two-dimensional symbols, strings, and paths – as the UML.

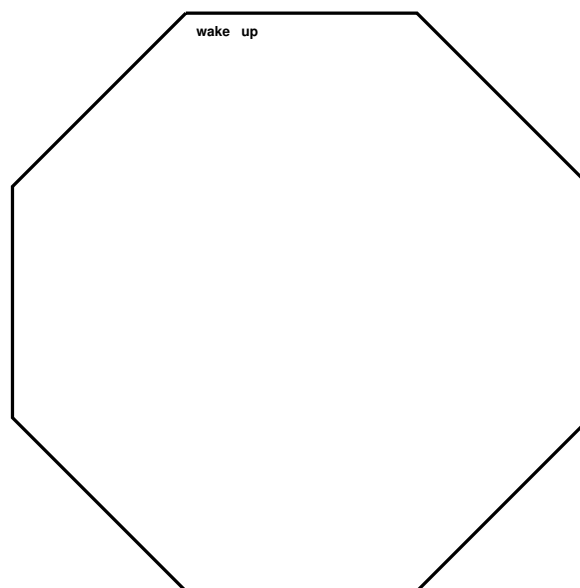
Relation between the UER and the UML [2/2]

- Many elements of the UER have been taken from the UML. However, others have been adapted or added to enhance cognitive adequacy and to accommodate our application domain.
- The UER has given up the different views onto a system as modeled in the different diagram types in the UER (such as class, state, activity, collaboration diagrams).
- This has been done because what is conceptualized by human language users as cognitive unit will presumably not exceed a certain degree of complexity and does include all those aspects in one concept.

Eventivity frame

- The modeling element 'eventivity frame' is not part of the UML and has been introduced in the UER.
- It represents eventivities, i.e., the cognitive units that are in the center of our efforts.
- It is notated in the UER as a solid-outline octagon, with its name shown in the upper left hand corner.
- It holds the information about the modeled eventivity.
- Eventivity frames are possible both on the type and on the instance level, depicting a verb reading and a proposition, for instance (such as *wake up* and *Jeff woke Diana up*).

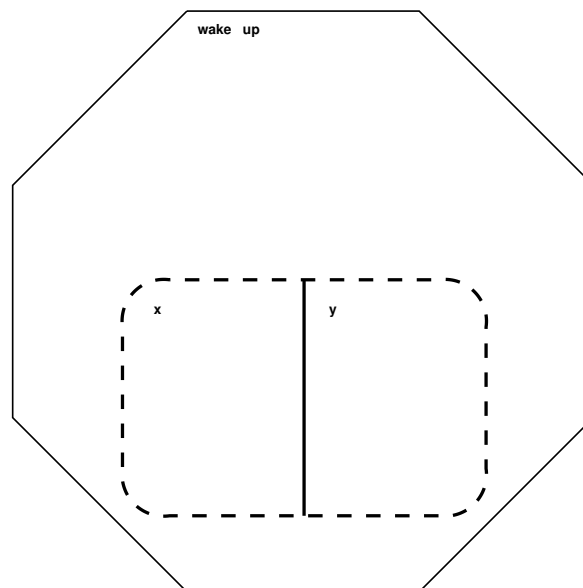
Example [1/10]



Dynamic core and static periphery

- As we have seen, an eventity includes both static and dynamic aspects. Accordingly, both are contained in an eventity frame.
- Static and dynamic aspects are kept apart in the UER, the dynamic aspects are modeled in the dynamic core.
- The dynamic core is depicted as a dashed-outline state symbol (rectangle with rounded corners).
- The dynamic core holds the state machines of the prominent participants [of those whose state-transition systems are conceptualized], each within a swimlane (vertical region).
- The static periphery is the remaining area of the eventity frame and holds the static aspects such as the participants and their relations.

Example [2/10]



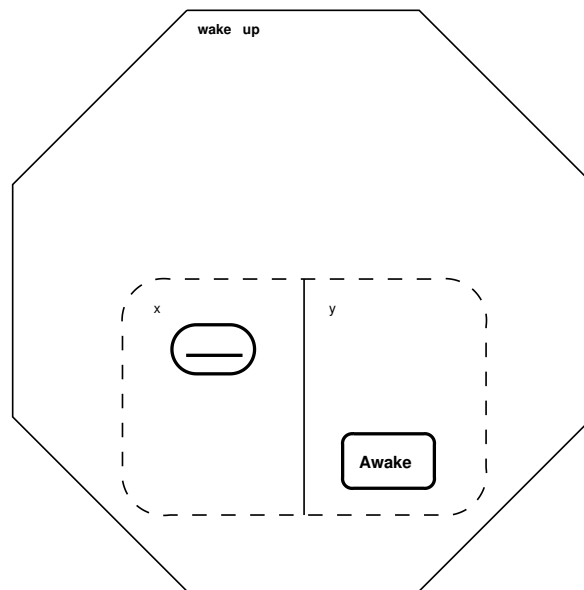
Simple state [1/2]

- Simple states constitute one of the basic eventuality types in the UER (together with transitions), representing a conceptual state in which an object/entity reacts to incoming stimuli or acts in the same way.
- They are adapted from the UML. The UER includes the basic distinction between 'passive' and 'active' simple states, where the passive states depict the object/entity's undergoing or enduring of a state and active simple states the object/entity's actions.
- The semantics has been adapted and adjusted to the basic aspectual differences found in semantic structure.

Simple state [2/2]

- Passive simple states are shown as UML states, active simple states are shown as UML action states.
- Both passive and active simple states can in principle be left unspecified in a modeling (in which case the name in the corresponding state symbol is substituted by an underscore). That is, the structure is still present, but the specification or statal content is unknown.
- Active simple states are further differentiated in activities (with duration) and acts (without duration), distinguished by additional icons (infinity and dot) depicted in the lower right corner.

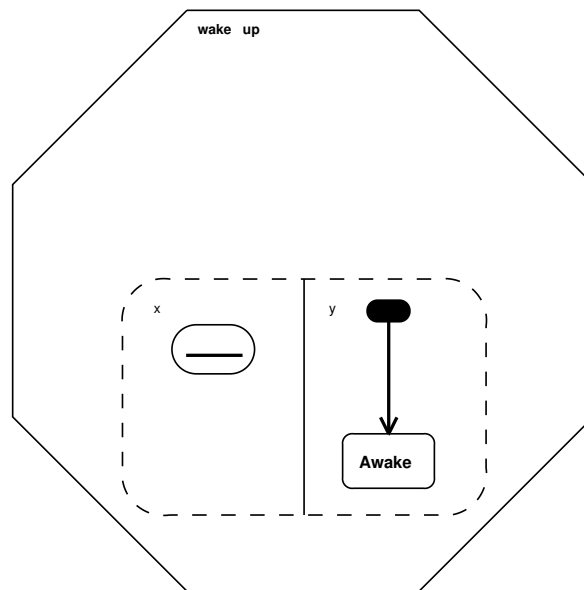
Example [3/10]



Transition

- Transitions constitute another basic event type in the UER, representing a conceptual change of state.
- They are adopted from the UML, and also depicted accordingly.
- The only addition to UML transitions is the introduction of a gradual transition, i.e., of a durative transition. This way, a transition can be explicitly marked as durative.
- A UER transition consists of a source state, a target state, and a transition arrow.
- Source and target states can be underspecified. In case there is no information about content *and* structure, newly introduced 'unspecified source and target state' symbols are deployed.

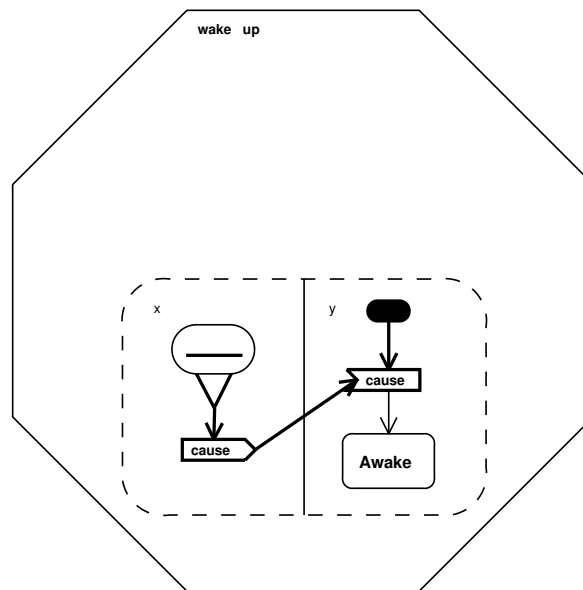
Example [4/10]



Event / Signal

- The UER has the same event types as the UML: completion event, change event, time event, and signal event (the signal receipt).
- All of these events trigger transitions, i.e., represent triggers of conceptual changes of state.
- Signals are particularly important because they model interactions: signals come in from other objects/entities and have an effect on the receiving object/entity.
- In the UER, signals are depicted with two pentagons, showing signal sending and signal receipt. They are connected by an arrow that crosses the swimlane border.
- A continuous signal as a durative trigger was introduced in the UER.

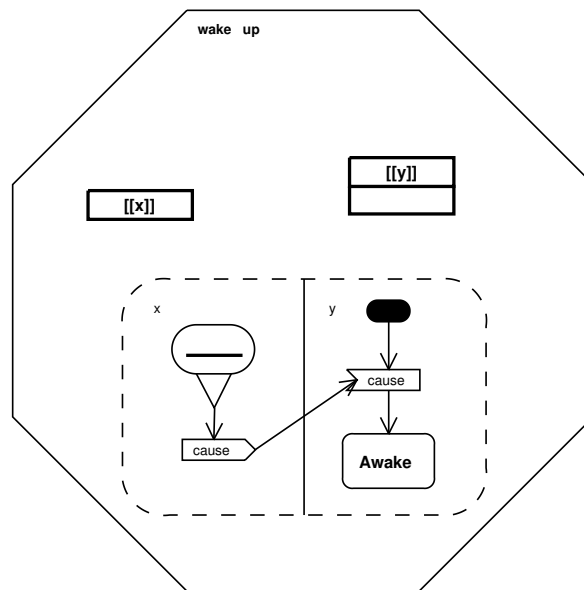
Example [5/10]



Participant class

- A participant in an eventity is represented by a 'participant class', which is a special kind of class.
- A participant class is depicted as a solid-outline rectangle and contains a name and may contain a list compartment.
- The name compartment specification is taken from UML collaborations, because participant classes conceptually specify the entities participating ('collaborating') in the eventity.
- The first component of the name compartment specification differs from the UML: it includes a participant representative $[[x]]$ standing for any potential participant.
- Participant classes do not form fixed classes, but are eventity-related.

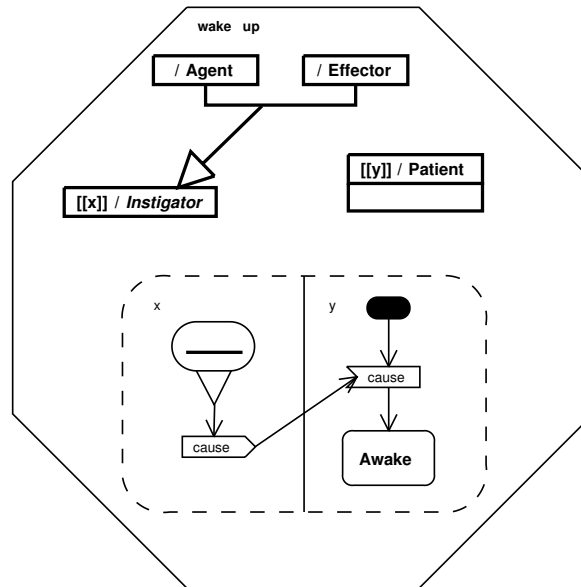
Example [6/10]



Participant role

- The second component of the name compartment specification in participant classes specifies roles.
- Participant roles are added as a string of the form '/ Role' after the participant representative.
- The role string specifies the role the participant plays within the event and hence the semantic role the entity holds within the event.
- The UER as a representational framework does not predefine the number and nature of semantic roles (this is in principle left to users, although a suggestion is made).
- Participant roles can be part of an inheritance structure.

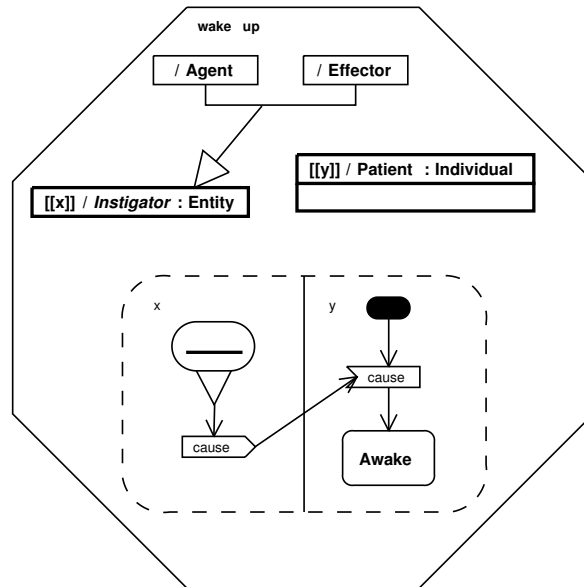
Example [7/10]



Participant type

- The third component of the name compartment specification in participant classes specifies types.
- Participant types are added as a string of the form ': Type' after the participant role.
- The type string specifies the ontological category a participant of the event must belong to (i.e. it establishes a coarse filter on entities).
- The UER as a representational framework does not predefine the ontological categories and hence possible types (this is in principle left to users, although a preliminary suggestion is made).
- Participant types are defined in an underlying ontology (which includes a participant type inheritance structure).

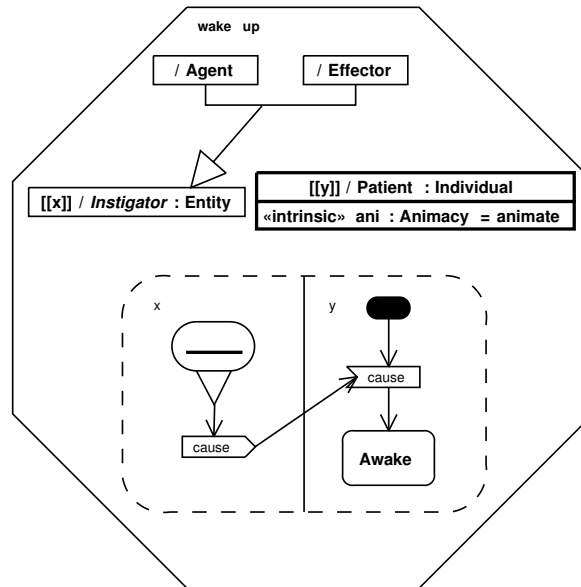
Example [8/10]



Attribute

- Attributes are adopted from the UML and notated accordingly in the list compartment and with the syntax 'name : type-expression [= value-expression]'.
- Attribute types are usually enumeration types, lists of values that are subsumed under a label (the enumeration type) which represents a recurring category.
- Within participant classes, they represent features potential participants must have if a value is given, i.e., they represent selectional restrictions.
- They can also be used to refine the role specification within a participant class (in which case the stereotype «role» is preposed).

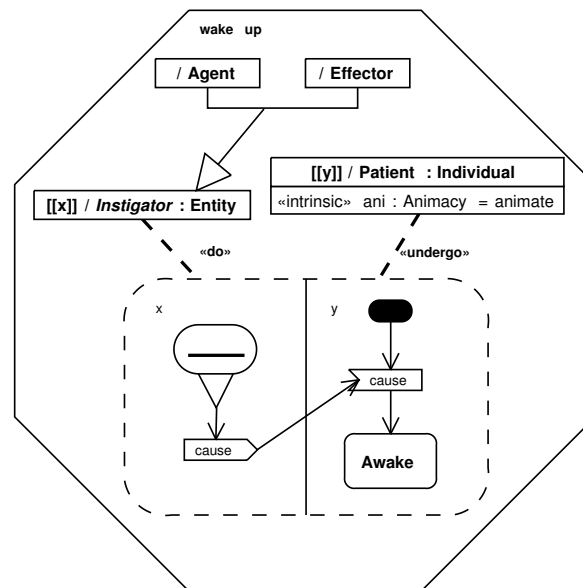
Example [9/10]



Participate association

- The connecting element between static and dynamic aspects in the modeling is the 'participate association' that is not part of the UML.
- Participate associations are shown as dashed lines connecting the participant class rectangles with the border of the dynamic core.
- They explicitly indicate which elements of the static periphery count as participants of the eventuality (we have a rather broad understanding of 'participant', including entities that are instruments, goals etc.).
- Up to two participants are marked via the stereotypes «do» and «undergo» that are attached to their participate associations. These participants are conceptualized as being more prominent: their state machines are depicted explicitly in the dynamic core.

Example [10/10]



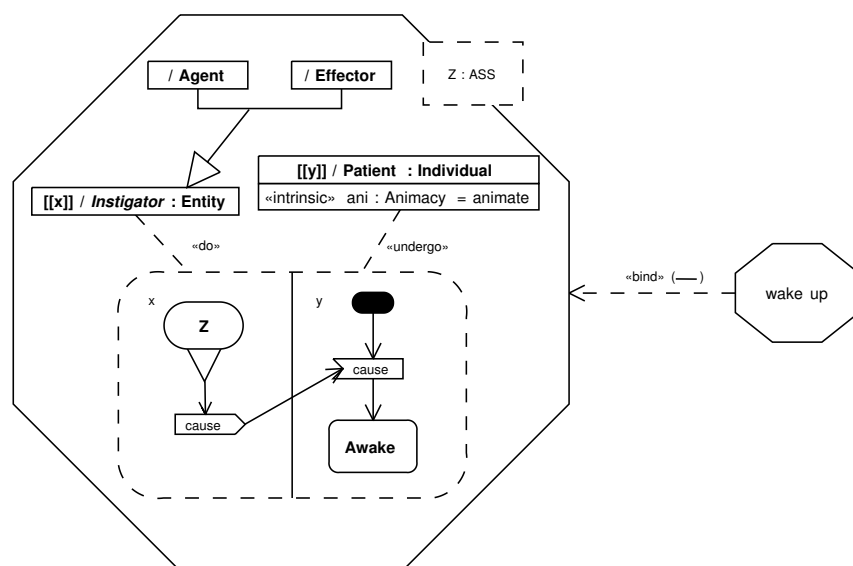
Template [1/2]

- Templates are adopted from the UML. They are parameterized model elements, i.e., they are not 'ordinary' model elements.
- A template defines a family of ordinary model elements, where each member of the family is specified by binding all the parameters to values.
- An eventivity frame template does hence not represent an eventivity but a family or class of eventivities.
- A dashed rectangle is superimposed on the upper right-hand corner of the symbol for the parameterized model element. It contains a list of parameters and their types.

Template [2/2]

- The relationship between the template and its bound element is usually shown by a dashed arrow connecting the template and bound element symbols, with the keyword «bind» and the values for the parameters as arguments in brackets attached to the arrow.
- Templates are a mechanism for abstraction away from content. They are an important structuring device, a tool for analyzing structures and modeling systematic relations between or within languages and their lexicons.
- We will see an application of templates tomorrow.

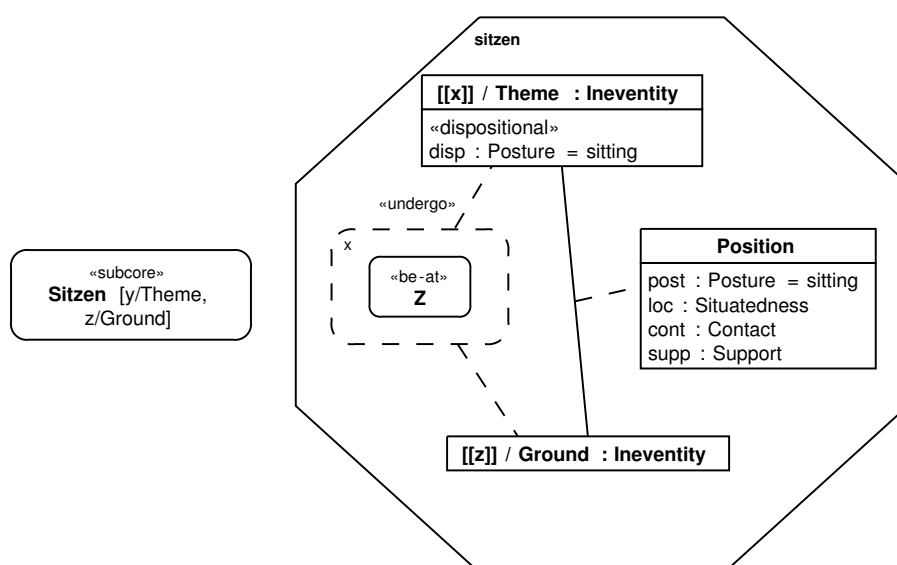
Template – Example



Subcore state

- A 'subcore state' represents a subeventivity (a conceptual macro) of the modeled eventivity.
- It is a conceptual structuring device.
- It is a new concept that is an interim solution for the representation of nesting of subeventivities within eventivities.
- It can be considered to reference the relationships between participants and the dynamic core of another eventivity frame.
- It is shown as a state symbol supplemented by the keyword «subcore». The name string cites the referenced eventivity frame's name and indicates which roles the participants of the containing eventivity frame play within the referenced eventivity frame.

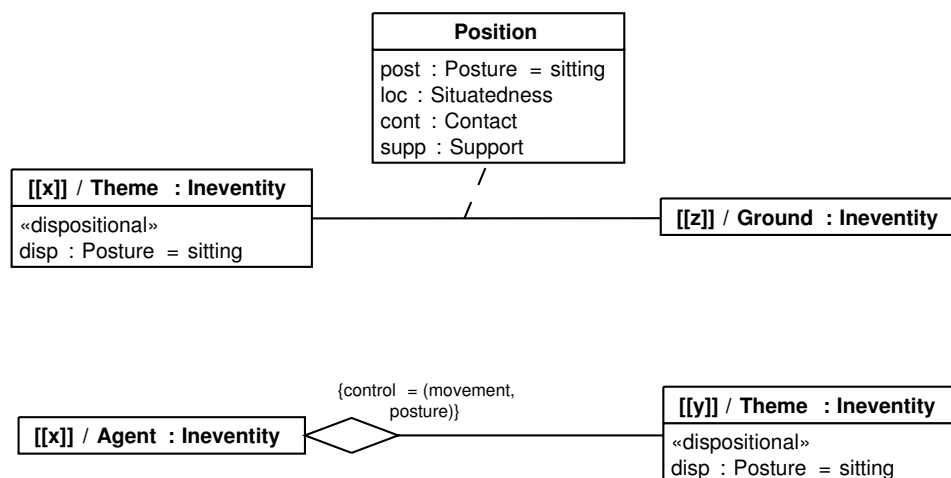
Subcore state – Example



Association/Aggregation

- Association and aggregation are taken from the UML, although the semantics of aggregation is different in the UER.
- Both are accordingly shown as in the UML (with additions for aggregation).
- Associations represent conceptual relationships between classes.
- They can be shown as 'association classes' as well: a class symbol is attached by a dashed line to the association path. The class can hold attributes which specify features of the association.
- Aggregations represent particular conceptual relationships, in which one participant propagates its characteristics or controls particular characteristics of the related participant.

Association/Aggregation – Example



TODAY Session 5:

Application of the UER and discussion

- We will look at two applications today, one descriptive linguistic and one theoretical linguistic one:
 - **Application 1:** The polysemy of German *setzen*
 - **Application 2:** Proposal for an eventy classification
- Both use the UER as analytic tool.
- We will briefly summarize and discuss what might be done with the UER in future research.
- I look forward to your feedback!

Application 1:

The polysemy of German *setzen*

We will:

- look at examples taken from the COSMAS Corpora of the IDS Mannheim;
- model the semantics of different readings of *setzen*, i.e. the underlying eventities, with the UER;
- determine the changes from one reading to another;
- and compile a semantic network.

[Note that the readings treated here are only a subset of the actual readings of *setzen*.]

Examples 1

- *Da das Wetter heute noch sommerlich schön war, zog Delia ihm ein niedliches kariertes Hemd zu Jeans an, die mit lustigen bunten Flickern verziert waren. Dann **setzte** sie ihn in die Kinderkarre.*

'Since the weather was still summerly warm, Delia dressed him with a cute checkered shirt and jeans which were decorated with gaily colored patches. Then she **put** him into the pram.'

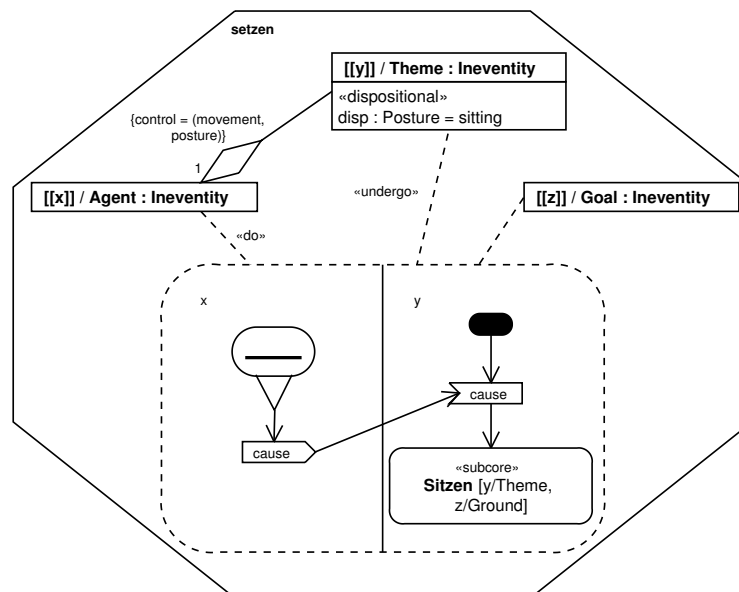
(de Groot, A.: Dein Vater wird uns lieb gewinnen. Hamburg, 1990, p. 24. [GR1])

- *Da alle Briefkörbe schon mit Verwundeten belegt waren, **setzten** wir Kobyella gegen einen Korb, ...*

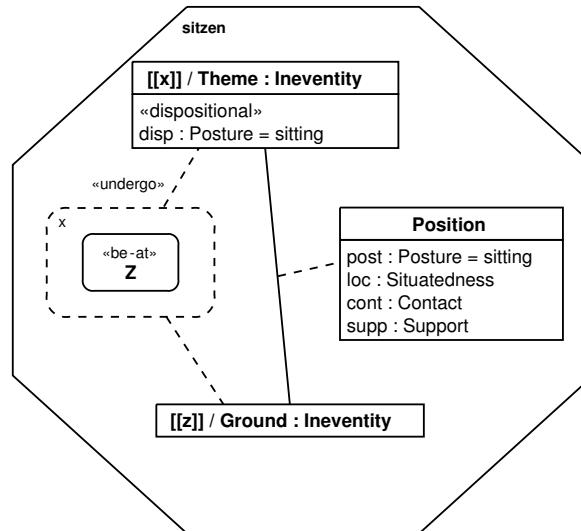
'As all letter baskets were already occupied by casualties, we **put** Kobyella against a basket, ...'

(Grass, Die Blechtrommel. Fischer, Frankfurt 1962 (1964), p. 193. [MK1])

The prototypical reading of *setzen* [1/2]



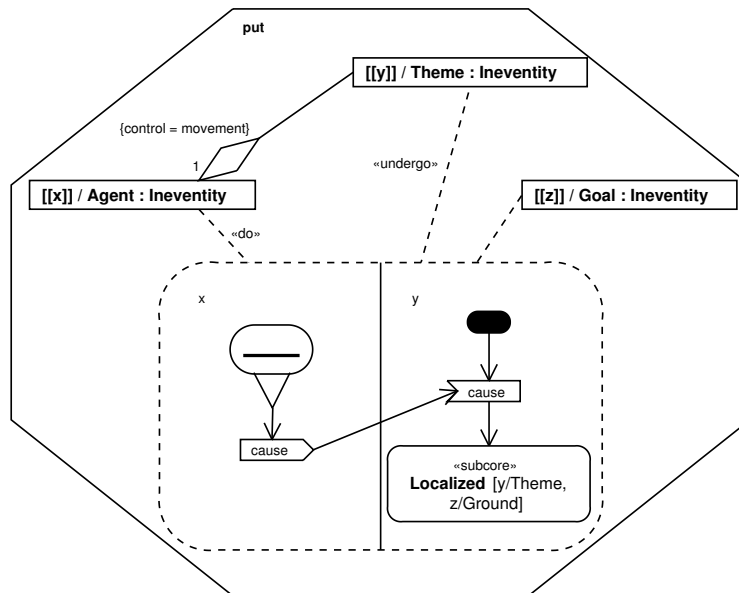
The prototypical reading of *setzen* [2/2]



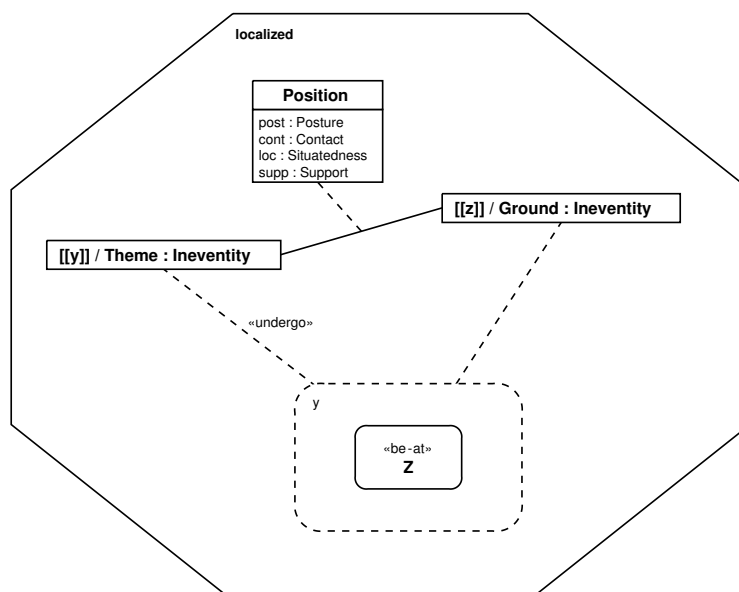
Examples 2

- *Klaus **setzte** die beiden Gläser auf den Tisch und nahm ihr gegenüber Platz.*
 'Klaus **put** both glasses onto the table and took a seat opposite her.'
 (Torwegge, C.: *Liebe hat ihre eigenen Gesetze*. Hamburg, 1990, p. 20. [GR1])
- *Er **setzte** neue Dielenstücke in den brüchigen Fußboden.*
 'He **put** new floorboards into the crumbly floor.'
 (Strittmatter, Ole Bienkopp. *Aufbau*, Westberlin, 1963, p. 67. [MK1])

PUT [1/2]



PUT [2/2]



Examples 3a

- *Er verstand aber das Zaubern, und zumal so hatt er ein Rößlein, das konnte wohl reiten und traben, damit **setzte** er in hohen Sprünge über Felsen und Risse ...*

‘But he knew how to perform magic, and especially as he had a little steed which was certainly capable of riding and trotting, with this, he **leaped** in high leaps over rocks and cracks ...’

(Eppela Gaila, p. 162. [GRI])

- *Gelassen **setzt** Bienkopp durchs Tonnentor.*

‘Keeping cool, Bienkopp **leaps** through the gate put up with barrels.’

(Strittmatter, Ole Bienkopp. Aufbau, Westberlin, 1963, p. 323. [MK1])

Examples 3b

- *368 **setzte** Kaiser Valentinian über den Rhein und verwüstete das Siedlungsgebiet dieses gefährlichen Gegners vollständig.*

‘In 368 Emperor Valentinian **crossed** the [river] Rhine and inflicted complete devastation on the settlement area of this dangerous enemy.’

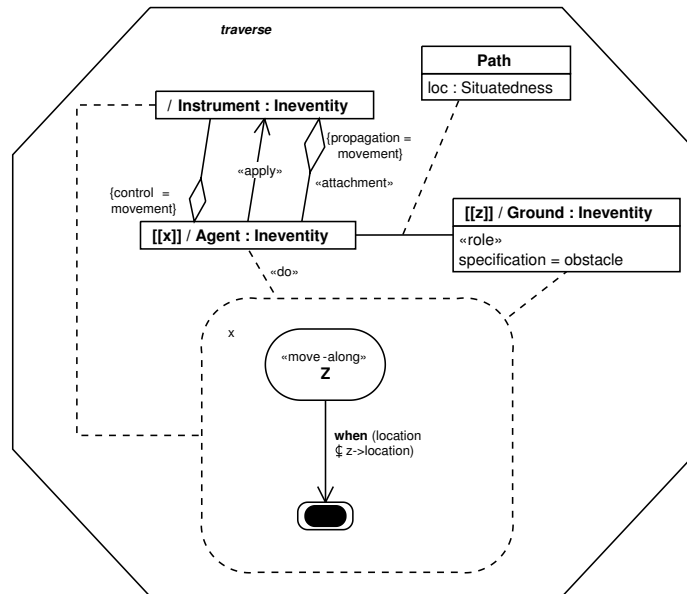
(Pörtner, Die Erben Roms. Econ, Düsseldorf, 1964 (1965), p. 124. [MK1])

- *Ihr Zug mußte aber über einen breiten Fluß **setzen**, und als die Hälfte das Heers hinüber war, geht die Sage, sei die Brücke gebrochen, so daß kein Mann zurückkehren, der hinüber war, und keiner mehr übersetzen konnte.*

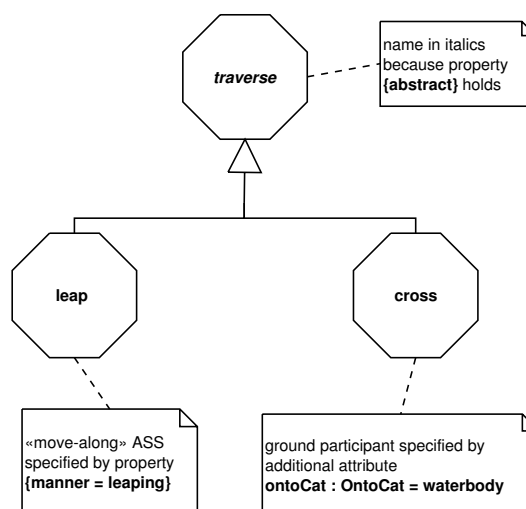
‘But their platoon had to **cross** a broad river, and when half of the army had reached the other side, the legend has it, the bridge collapsed, so that no man who was there could return hither nor could any get thither.’

(Die eingefallene Brücke, p. 346. [GRI])

TRAVERSE



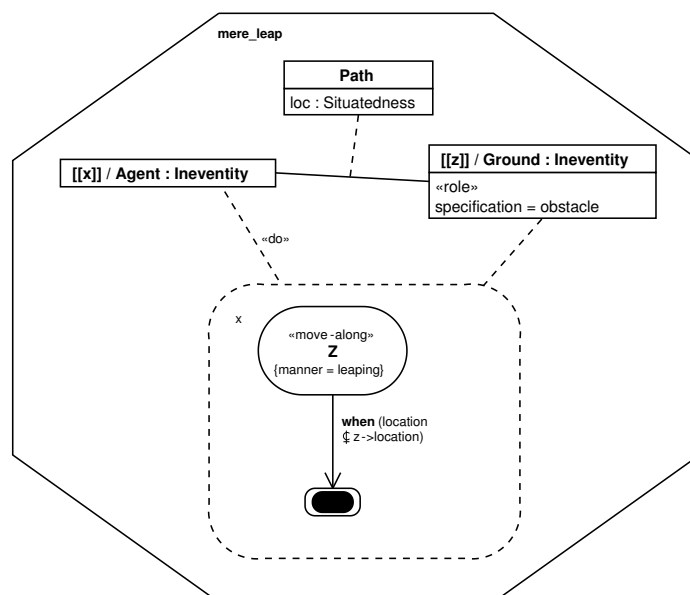
CROSS and LEAP



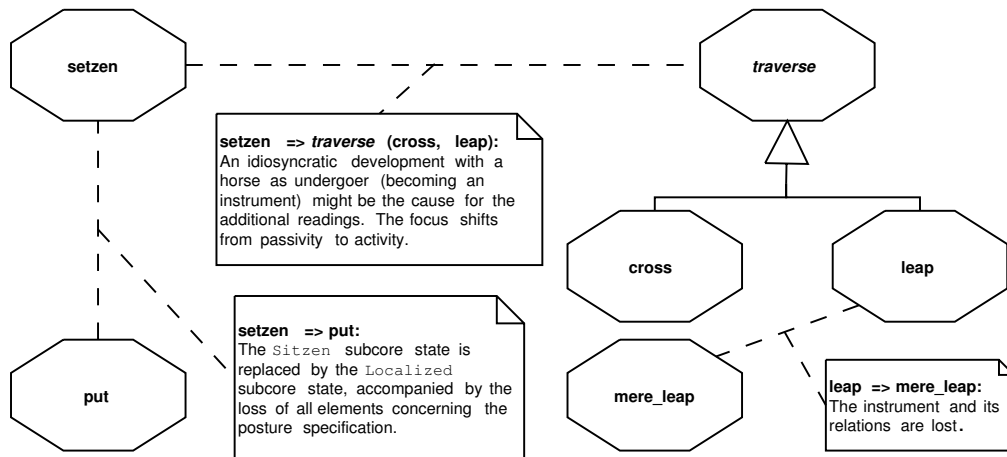
Example 4

- *Barfuß und in kurzer Hose **setzte** er durch den Schlosswald, selbst im Winter.*
 'Barefooted and in short trousers he **leaped** through the castle woods, even in winter.'
 (Süddeutsche Zeitung. Magazine, No. 26, June 28, 2002, p. 11. [SZM])

MERE_LEAP



Overview of *setzen* readings



The polysemy of German *setzen* – Observations

- The presented readings, even though only a few, are a good example of Wittgenstein's notion of family resemblance: only few features are common to all representations.
- Participant relations have been underestimated in the research on verbal meaning.
- Structural changes seem to be necessary in order for language users to conceptualize new readings.

Application 2:

Proposal for an eventivity classification

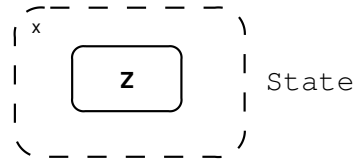
We will:

- partially propose an eventivity classification reflecting Aktionsart differences (as opposed to a verb classification, because our classification in fact classifies eventivities and not verbs);
- systematically abstract from semantic content (and hence take advantage of the fact that the UER is a graphical framework and that structure and content are easily differentiated);
- focus on the dynamic core and parameterize elements of the dynamic core; using template structures to represent eventivity classes.

Procedure

- We will start with the most basic element of dynamic structuring: state.
- Subsequently, we will systematically add other basic elements of dynamic modeling, in particular transition (resulting in change eventivities) and cause-signal (resulting in interactional eventivities).
- The dynamic cores that have to be well-formed become more and more complex as a result of the additions.
- This gain in complexity corresponds to a gain in conceptual complexity.

Statal eventities



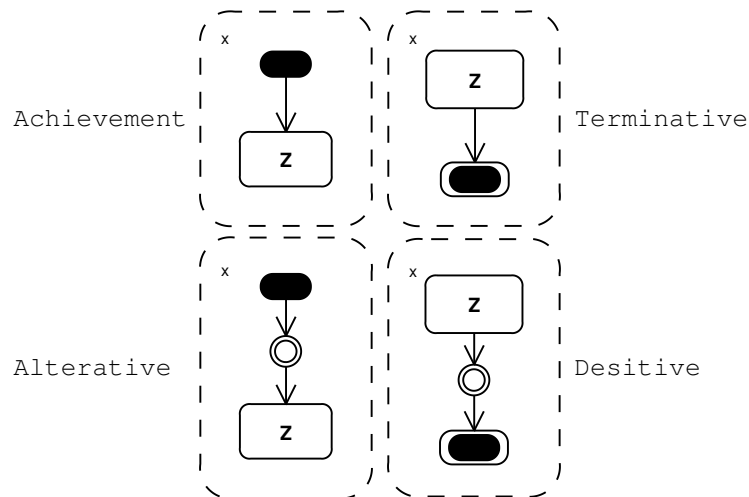
Examples

- *Helen is senior lecturer in education at University of Auckland.*
(<http://www.education.monash.edu.au/research/news/seminars/2004/helen25-2.html>, accessed on June 15, 2004)
- *Only he knows the password.*
(<http://www.subterrane.com/smimefaq.shtml>, accessed on June 15, 2004)

Change eventities [1/2]

- Transition as the second building block of dynamic modeling is defined as a relationship between two states.
- This corresponds to our conceptualization of change of state: there has to be a source and a target state within a conceptual transition.
- For a UER transition to be well-formed this implies that there needs to be a source and a target state of this transition.
- The nature of the source and target states and of the transition are the elements that can be systematically altered, resulting in – amongst others – the following configurations.

Change eventities [2/2]



Change eventities – Examples [1/2]

- **ACHIEVEMENT:**

*He **died** yesterday morning aboard the shuttle Columbia when it exploded 200,000 feet above Texas.*

(http://seattletimes.nwsources.com/news/nation-world/columbia_disaster/the_crew/anderson.html, accessed June 15, 2004)

- **TERMINATIVE:**

*Ich habe mein Passwort **vergessen!***

'I have forgotten my password!'

(<http://faq.gmx.de/faq/6.html>, accessed on March 14, 2004)

Change eventities – Examples [2/2]

- **ALTERATIVE:**

*The bottle **filled up** with tar.*

(<http://www.spinaweb.ie/showcase/2001/prim/spin1105/spinaweb/Body/Lungs/lung4.htm>, accessed on March 14, 2004)

- **DESITIVE:**

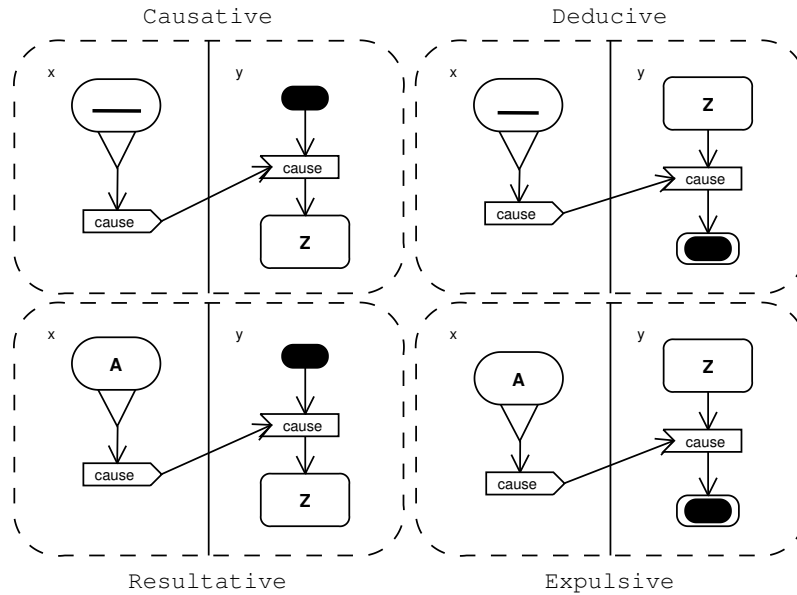
*All tyres **deflate** overtime –typically your tyres can lose up to 2psi per month.*

(<http://www.firestone.co.nz/default.asp?action=tyr.tyrecare>, accessed on March 14, 2004)

Interactional eventities [1/2]

- Cause-signal is the last building block of dynamic modeling that is added.
- It is specified as a signal emanating from an action state of one participant and triggering a transition of another participant.
- This corresponds to our conceptualization of causation: something has to happen which triggers or causes a change of state.
- For a UER modeling this implies that there has to be a state and a transition that can systematically be altered, resulting in – amongst others – the following configurations.

Interactional eventities [2/2]



Interactional Eventities – Examples [1/2]

- CAUSATIVE:

*Ivan died 3 years later after he **killed** his son in 1584.*

(<http://fsix.net/files/ivan.htm>, accessed on March 14, 2004)

- DEDUCTIVE:

*Therefore, Hades was able to bring her to the underworld and by offering her hidden grains of pomegranate he **made** her **forget** about the upper world and remain with him.*

(<http://www.archaeonia.com/religion/cults/hades.htm>, accessed on March 14, 2004)

Interactional Eventities – Examples [2/2]

- RESULTATIVE:

*Und als sich dann ein Freund zu mir gesellte, der mich
wachrüttelte,...*

‘And when a friend then joined me, who woke me up by shaking [me]...’

(<http://www.suchtzentrum.de/drugscouts/dsv3/stoff/erfahrber/erflachgas1.html>, accessed June 15, 2004)

- EXPULSIVE:

??

(Maybe there is a language in which such an eventity is lexicalized, maybe there isn't. I'd be happy for any hints...)

Proposal for an eventity classification – Observations [1/2]

- Any classification involves the abstraction of specific semantic information. For this purpose, templates are a suitable tool.
- This is supported by the graphical representation which allows the differentiation between content and structure in a straightforward way.
- Template structures as representations of eventity classes correspond perfectly to our conceptualization: they are not model elements (e.g., do not represent an eventity), but depict a family of model elements (i.e., a family of eventities with the same underlying structure, and hence eventity classes).

Proposal for an eventivity classification – Observations [2/2]

- Another point (already indicated above) demonstrating cognitive adequacy of the UER is the design of its dynamic modeling:
 - The representation of changes of states should model (i) the states themselves and (ii) the change or transition from one state to the other
→ UER transition: (i) represented by UER states and (ii) by the transition arrow.
 - The representation of causation should (i) model the action that is the trigger/cause, (ii) the implied interaction between the participants, and (iii) the change of state that leads to the result/effect of the causation
→ UER cause-signal: well-formed only if (i) the triggering action state, (ii) the cause-signal itself, and (iii) the transition are modeled.

What have we done so far? [1/2]

- We have worked our way into a new representational framework for verbal semantics (and eventities in general).
- This framework, the UER, is based on the computational modeling language UML. Both have been introduced briefly.
- We have discussed issues such as:
 - which components must be representable within a good representational framework for verbal semantics;
 - why a semantic representation should be a structural configuration rather than a linear conjunction of information;
 - why it is good to use a graphical representation.

What have we done so far? [2/2]

- We have discussed two application areas and seen the potential of the UER as a new modeling framework in both descriptive and theoretical work.
- The UER is, as the UML, a graphical, object-oriented modeling language.
- With object-orientation, we have introduced a new paradigm into linguistic semantics that is very successful in computer science.
- Of course, we should add that the relation to the UML will make the UER easily accessible for computational applications.

What can and should be done in future research?

Being a rigorous, cognitively oriented, intuitive decompositional modeling language, the UER will be worthwhile

- in scientific discovery within linguistics
(semantic relations, polysemy, semantic change, 'verb' classification, discourse representation, 'ontolinguistics' and generally the relation between concepts, their complexity, and their linguistic coding in natural languages, etc.)
- and in computational linguistics
(speech processing systems, machine translation, etc.)

References

- Bierwisch, Manfred, 1982. Formal and lexical semantics. *Linguistische Berichte* 80: 3–17.
- Bierwisch, Manfred, 1983. Semantische und konzeptuelle Repräsentation lexikalischer Einheiten. In *Untersuchungen zur Semantik*, Rudolf Ružička and Wolfgang Motsch (eds.), 61–99. Berlin: Akademie-Verlag.
- Bierwisch, Manfred, and Ewald Lang, 1987. Etwas länger – viel tiefer – immer weiter: Epilog zum Dimensionsadjektiveprojekt. In *Grammatische und konzeptuelle Aspekte von Dimensionsadjektiven*, Manfred Bierwisch and Ewald Lang (eds.), 649–699. (Studia Grammatica 26/27). Berlin: Akademie-Verlag.
- Bierwisch, Manfred, and Robert Schreuder, 1992. From concepts to lexical items. *Cognition* 42: 23–60.

- Chomsky, Noam, 1965. *Aspects of a Theory of Syntax*. (Special Technical Report of the Research Laboratory of Electronics of the Massachusetts Institute of Technology 11). Cambridge, MA: MIT Press.
- Jackendoff, Ray, 1991. Parts and boundaries. *Cognition* 41: 9–45.
- Löbner, Sebastian, 2002. *Understanding Semantics*. (Understanding Language Series). London: Arnold.
- McCawley, James D., 1976. Lexical insertion in a transformational grammar without deep structure. In *Grammar and Meaning. Papers on Syntactic and Semantic Topics*, James D. McCawley (ed.), 155–166. (Taishukan Studies in Modern Linguistics). New York/San Francisco/London: Academic Press. [First published in *Papers from the fourth regional meeting, Chicago Linguistic Society* 1968, Darden, B. J., C.-J. N. Bailey, and A. Davidson (eds.), 71–80. Chicago: Linguistics Dept. of the University of Chicago].

- OMG, 2001. OMG Unified Modeling Language Specification. Version 1.4.
[http:// www.omg.org/technology/documents/formal/uml.htm](http://www.omg.org/technology/documents/formal/uml.htm), accessed on September 15, 2002.
- Pustejovsky, James, 1991. The syntax of event structure. *Cognition* 41: 47–81.
- Pustejovsky, James, 1995. *The Generative Lexicon*. Cambridge, MA/London: MIT Press.
- Rapp, Irene, 1997. *Partizipien und semantische Struktur. Zu passivischen Konstruktionen mit dem 3. Status*. (Studien zur deutschen Grammatik 54). Tübingen: Stauffenburg.
- Ravin, Yael, 1990. *Lexical Semantics Without Thematic Roles*. Oxford: Clarendon Press.
- Rumbaugh, James, Ivar Jacobson, and Grady Booch, 1999. *The Unified Modeling Language Reference Manual*. (The Addison-Wesley Object Technology Series). Reading: Addison-Wesley.

- Schalley, Andrea C., 2004. *Cognitive Modeling and Verbal Semantics*. (Trends in Linguistics. Studies and Monographs 154). Berlin/New York: Mouton de Gruyter.
- Wunderlich, Dieter, 1997. Cause and the structure of verbs. *Linguistic Inquiry* 28 (1): 27–68.
- Wunderlich, Dieter, 2000. Predicate composition and argument extension as general options – a study in the interface of semantic and conceptual structure. In *Lexicon in Focus*, Barbara Stiebels and Dieter Wunderlich (eds.), 247–270. (Studia grammatica 45). Berlin: Akademie-Verlag.