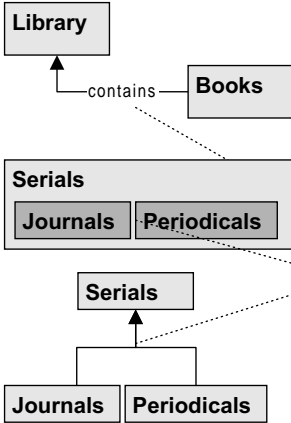
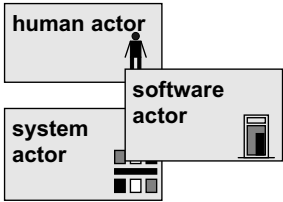


**subject name**

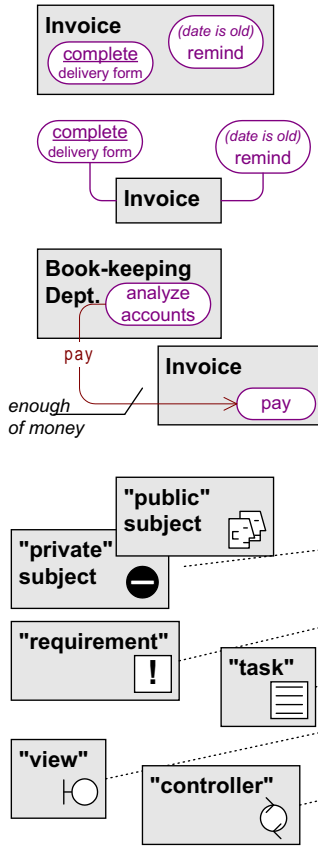
**subject**  
Subject is data concept describing problem domain entities. Any other graphical symbol may be used (e.g. pictures, clip-arts, ...) to improve the understandability of the diagram.

**actor**  
Actor is special kind of subject which expresses certain role of subjects outside the system which interacts with the system (e.g. users, software, ...)



**association**  
Association is oriented data relationship among subjects.

Association may be drawn also as subject composition. (both nested and tree form of pictures are allowed and are equivalent)



**behaviour (or script)**  
Behaviour is one of the subject's activity (with optional condition and result description), which expresses coherent sequence of desired transactions of the subject.  
Behaviours are drawn as ovals both inside and outside of the corresponding subject.

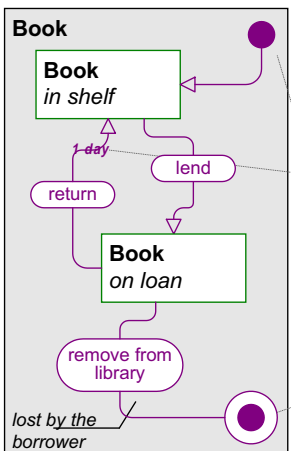
**communication**  
Communication is the behavioural relationship (with optional name and condition) among scripts or between actor and script. Communications express the control flows among objects in the system.

Icons may be used for expressing different subject roles (see actors above).

- private / public attribute expresses the accessibility of the entity outside of the modelled system.
- entity which covers a system requirement
- entity which covers a system task
- entity which covers a system interface
- entity which covers a system operation (e.g. data processing, ...)

**subject name**  
*state description*

**subject state**  
Subject state is a phase of subject lifecycle. States are drawn inside of subjects or of their superordinate states (e.g. decomposition of subject states is allowed).

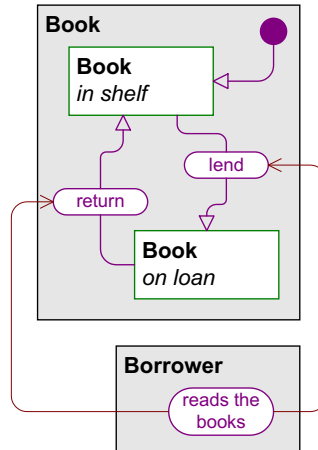


**state transition**  
State transition is the link among subject states (with optional condition and script which is responsible for change the object from one state to the other).

**timing**  
is optional description of the transition details (e.g. frequency)

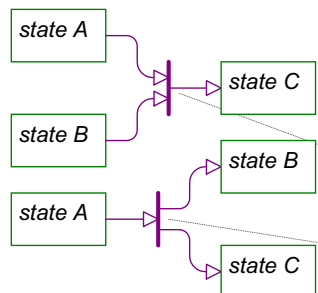
transition from the **initial state** begins the lifecycle of the whole object (or decomposed state).

transition to the **end state** ends the lifecycle of the whole object (or decomposed state).



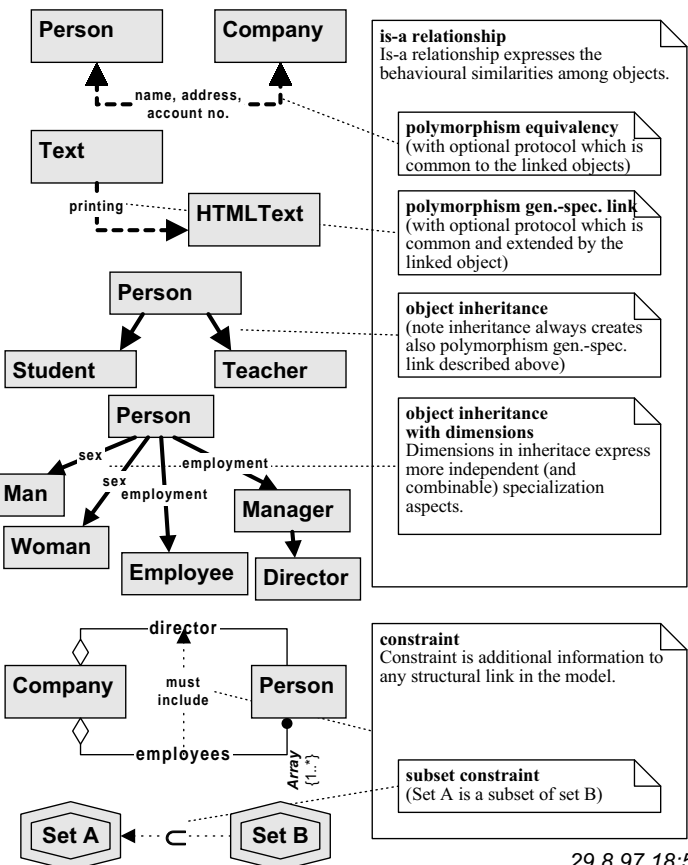
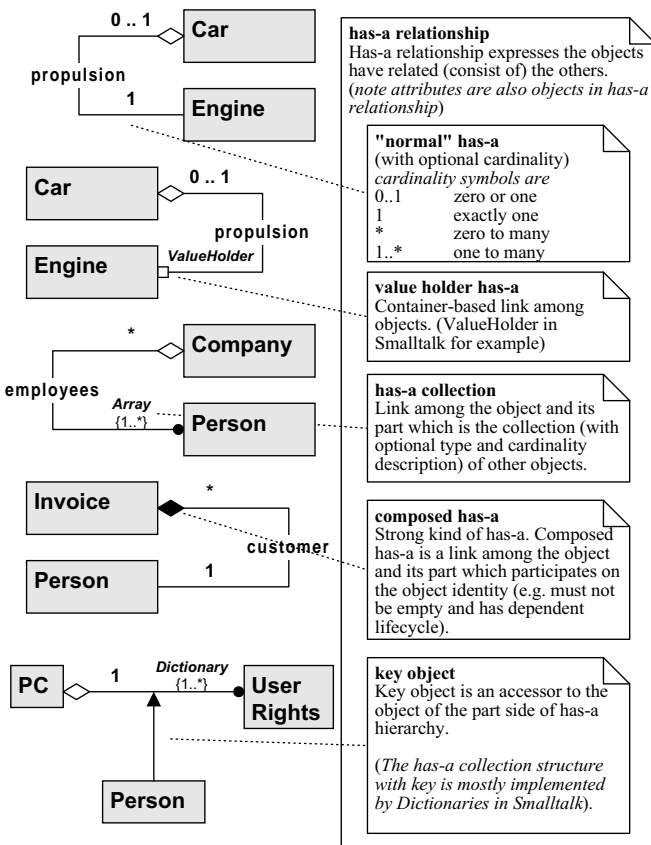
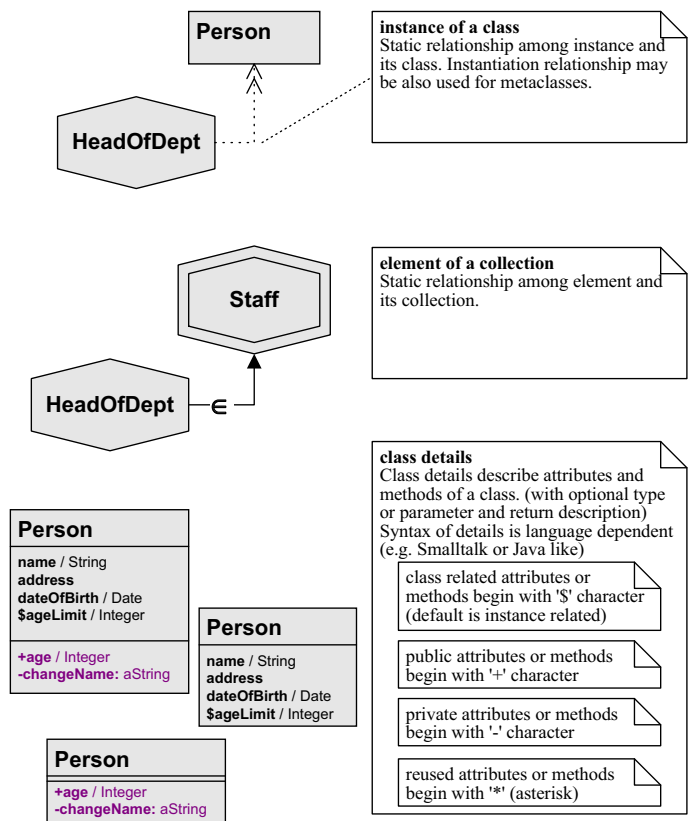
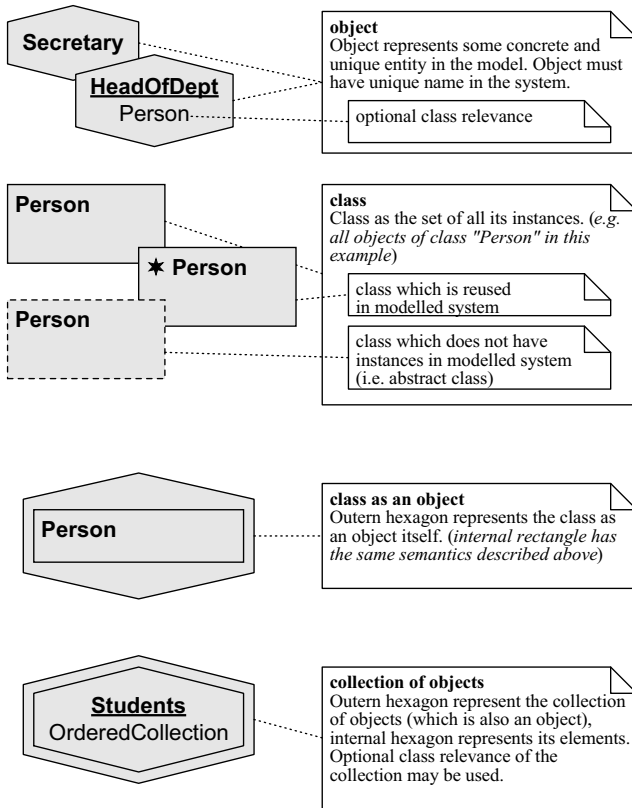
**state transitions, scripts and communications**

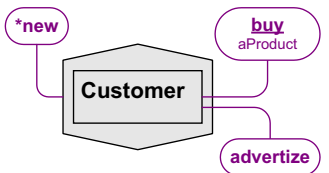
Objects mutually interact via communications. Communications among objects are also responsible for performing state transitions in the business object model.



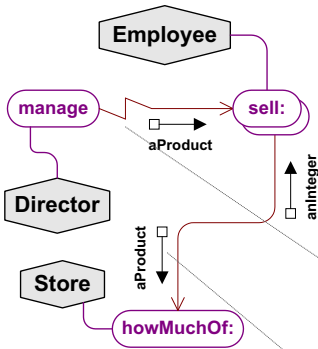
**state concurrency**  
Subject may be in more states in the same time. In this case particular states express different aspects of the subject behaviour.

- subject must be concurrently in both state A and state B for transition to the state C.
- transition from state A to two concurrent states B and B





**method**  
Methods are drawn in the same way as scripts of subjects in business modeling. Methods may have optional result or parameter description. Reused (e.g. inherited) methods have name with asterisk.



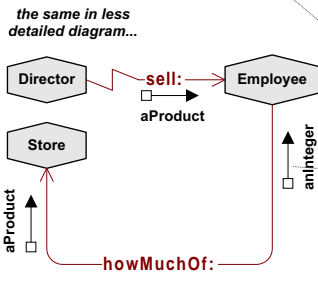
**message**  
Message is a control-flow link between methods. Message have optional parameters or return value.

**time sequence**  
Time sequence describes the order of execution (at first receive the message sell:, then send a message howMuchOf:)

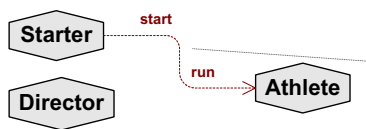
**fork message**  
Fork (or asynchronous) message makes the sender does not have to wait until message is processed by particular method of the receiver object.

**parameter**  
Parameters are objects which make the data flow in the same direction the message was sent.

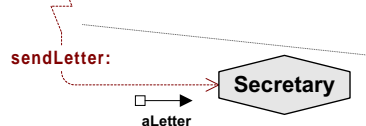
**return**  
Returns are objects which make the data flow in the reversed direction the message was sent.



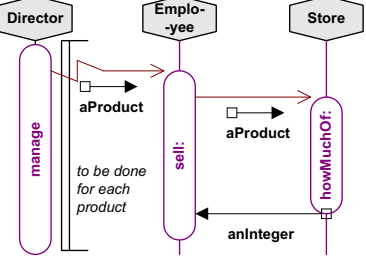
BORM - Business Object Relation Modelling



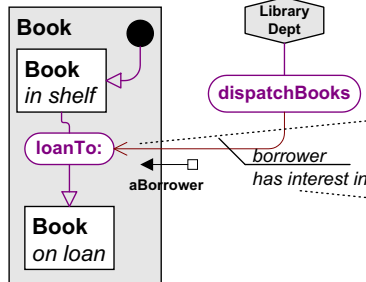
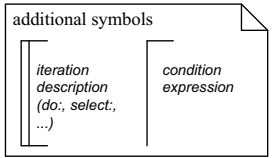
**dependency**  
This link expresses the behavioural dependency of one object on the other. (in example the execution of athlete's method "run" is dependent on execution of starter's method "start")



**delegation**  
This link expresses the delegation of message execution from one object to the other.



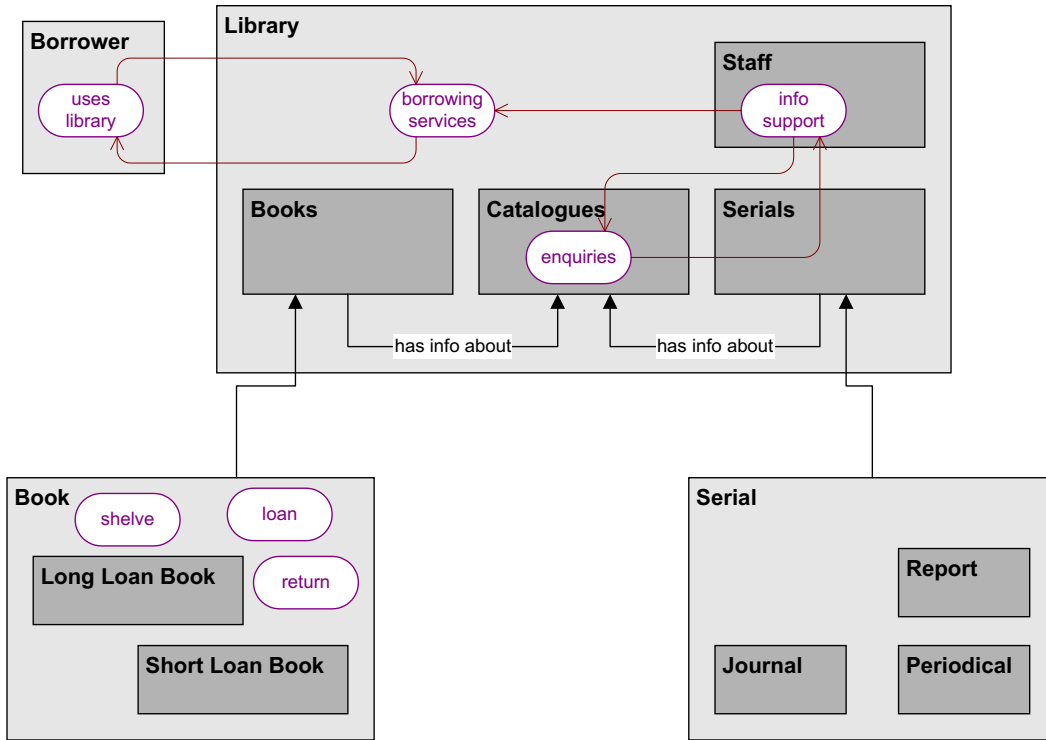
**message flow**  
Message flow style of diagram is more focused to time dependency. (note that use the same notation symbols)



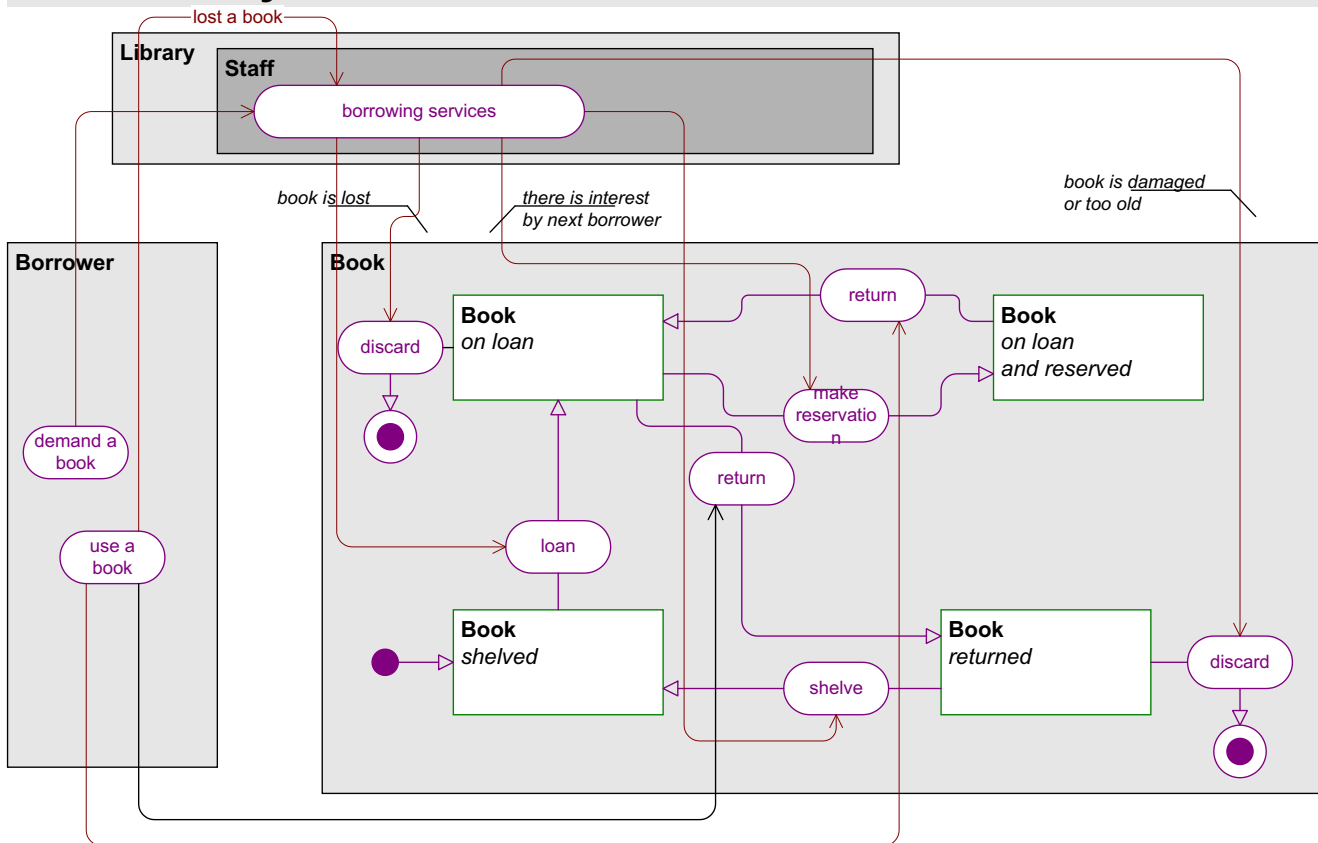
**states and transitions**  
The analogous syntax and semantics as of business modelling is used.

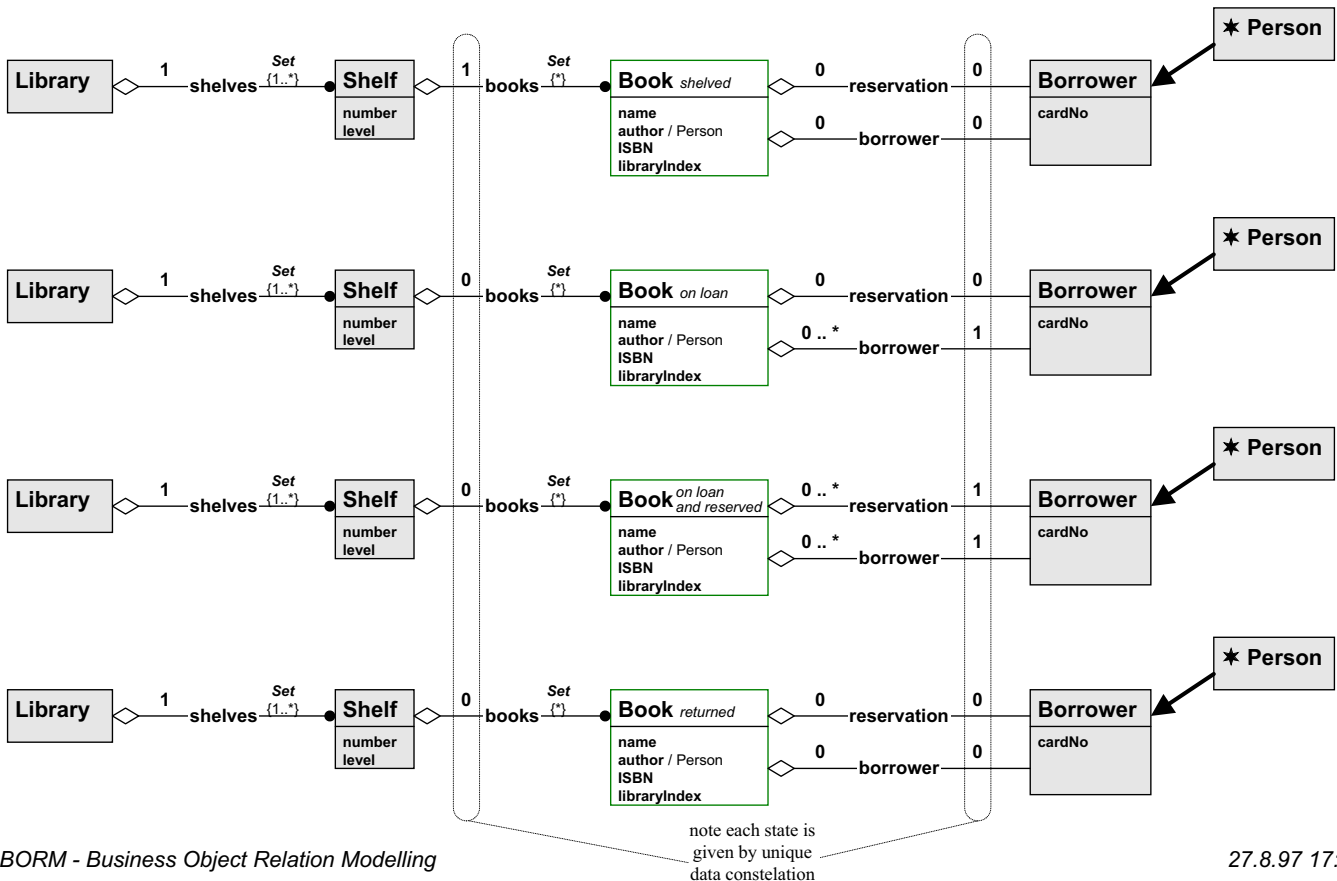
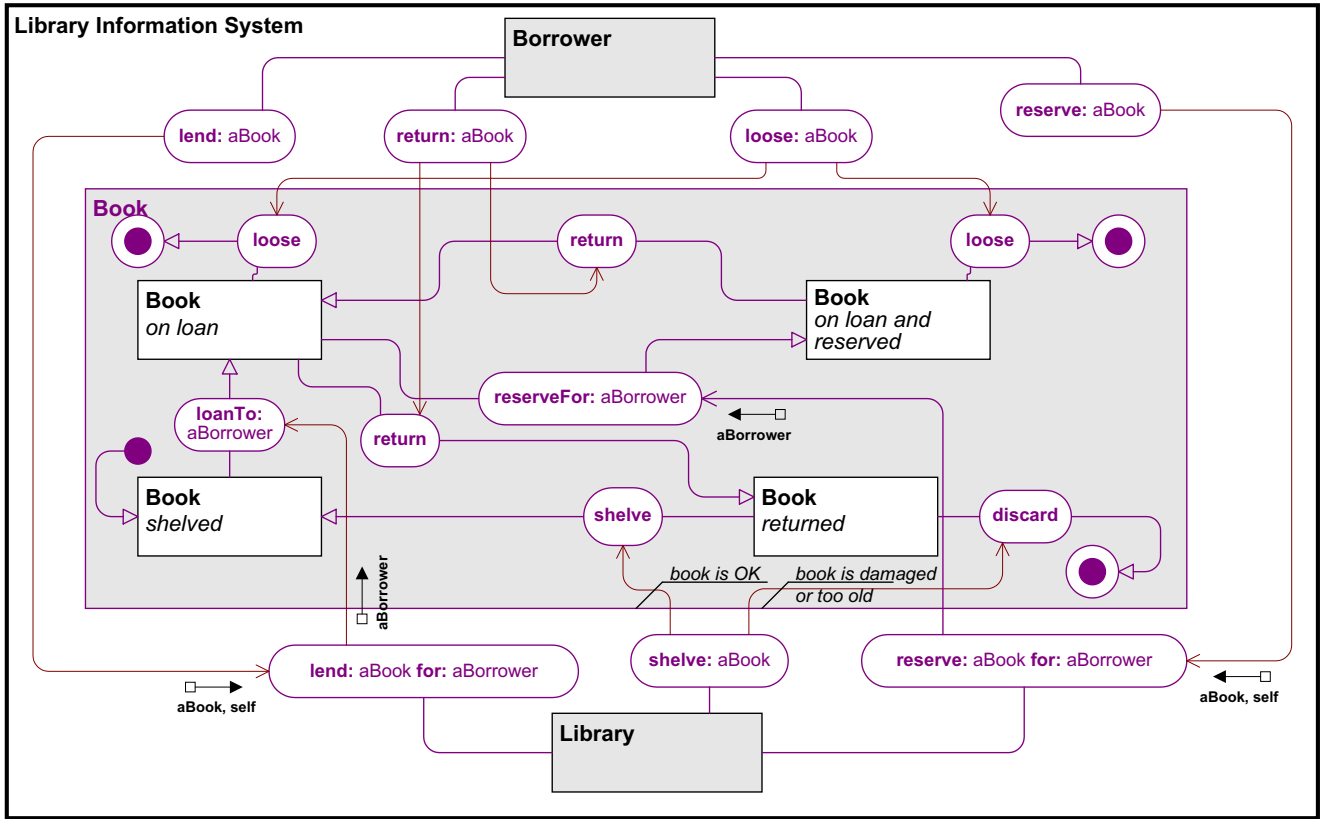
object method initiated by some message performs the transition

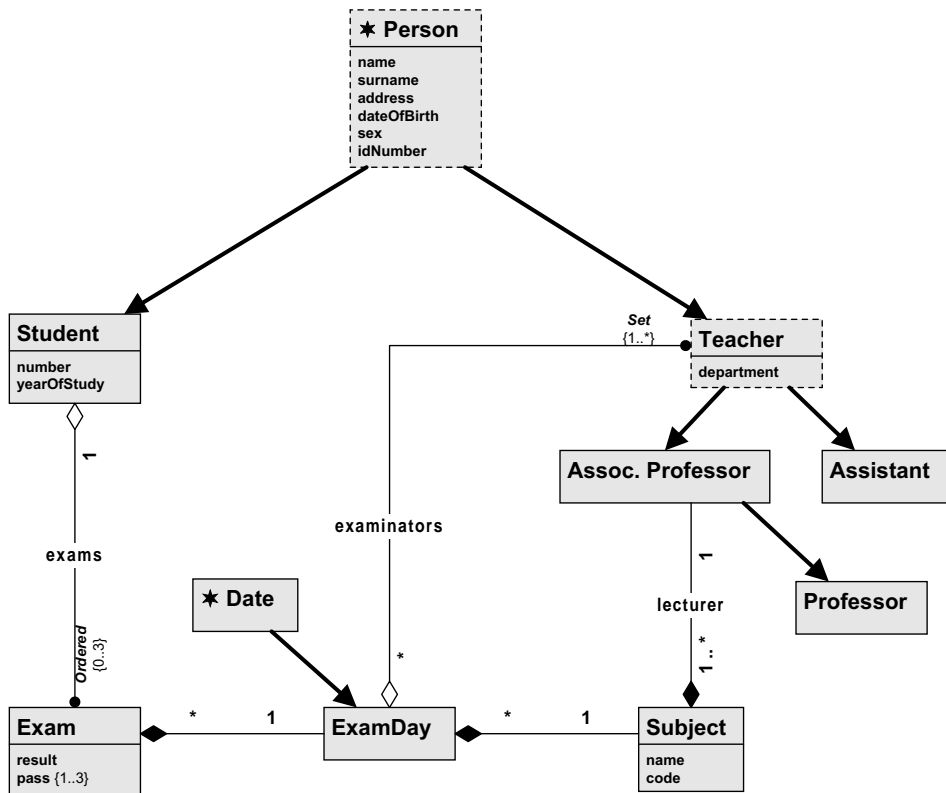
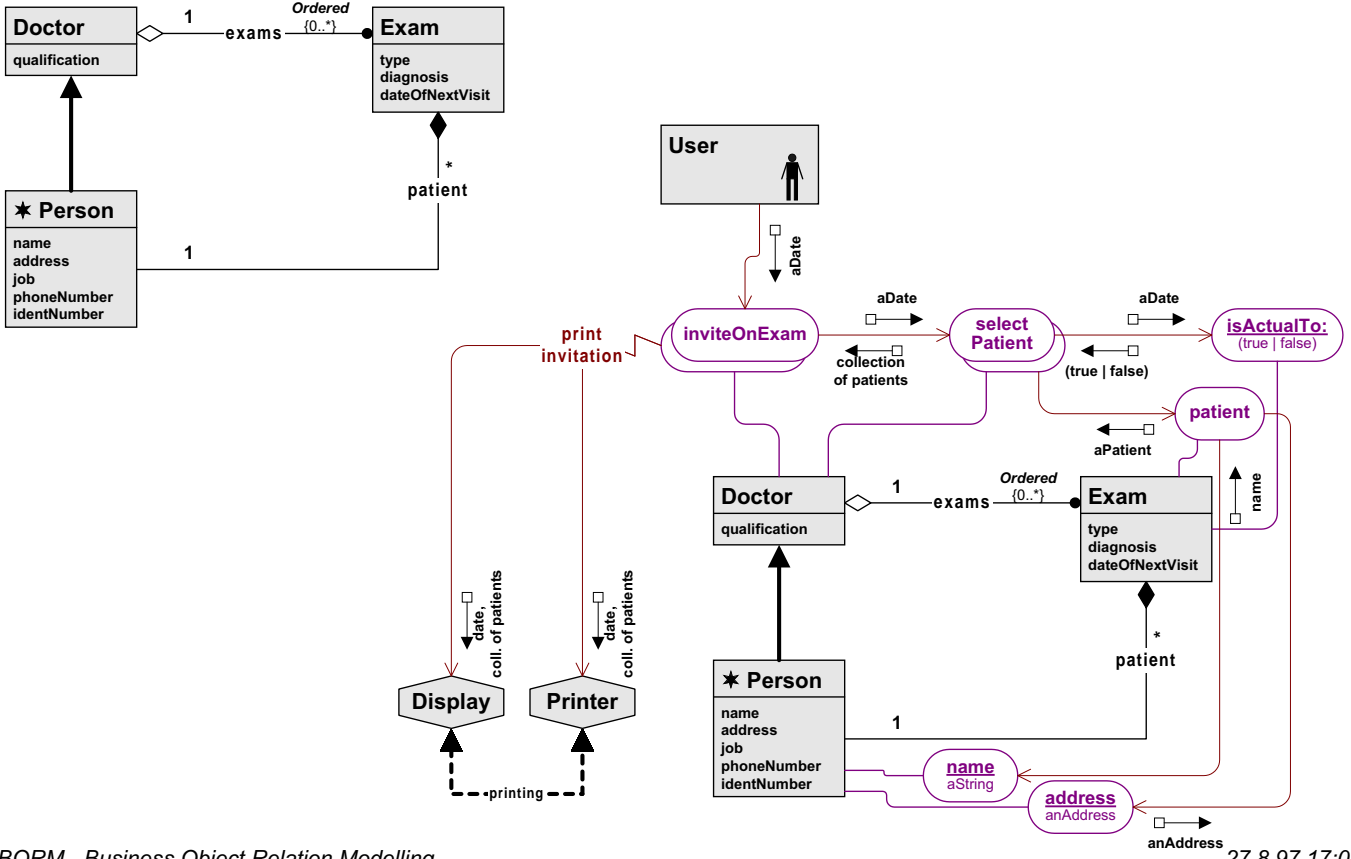
message send may be restricted by the condition

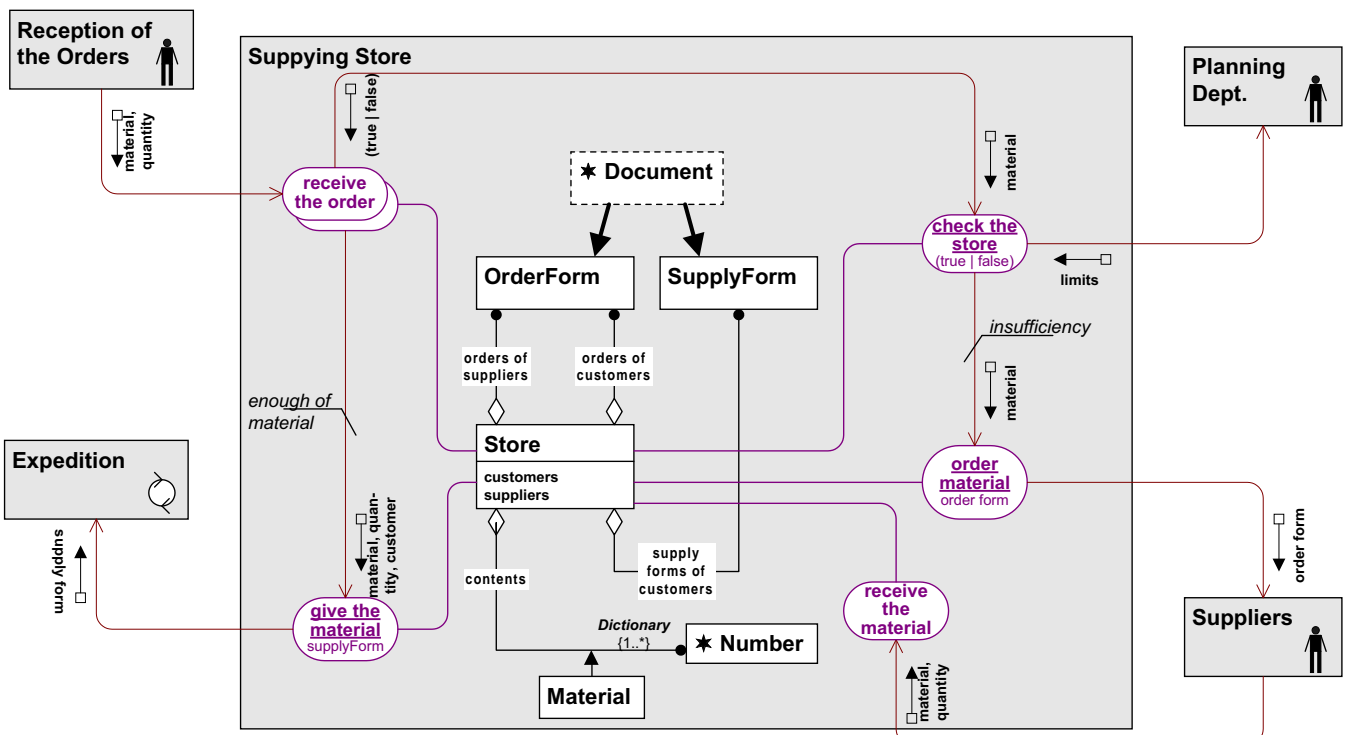
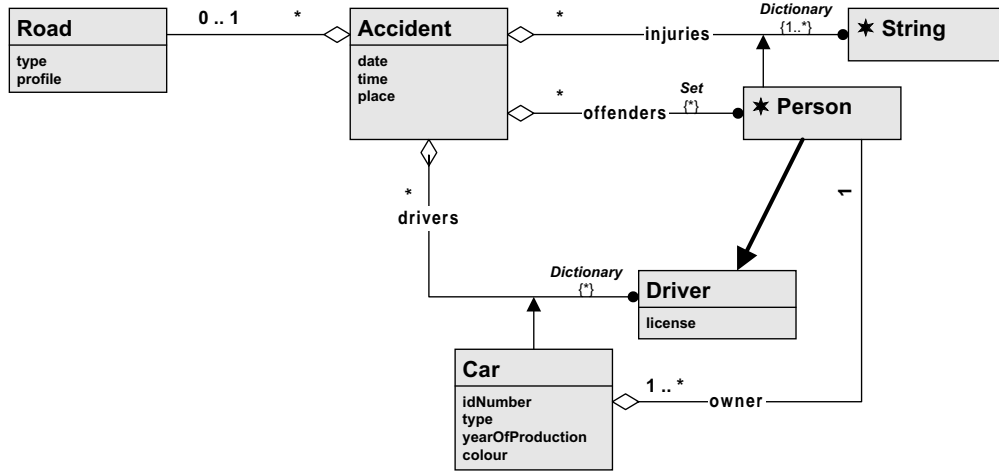


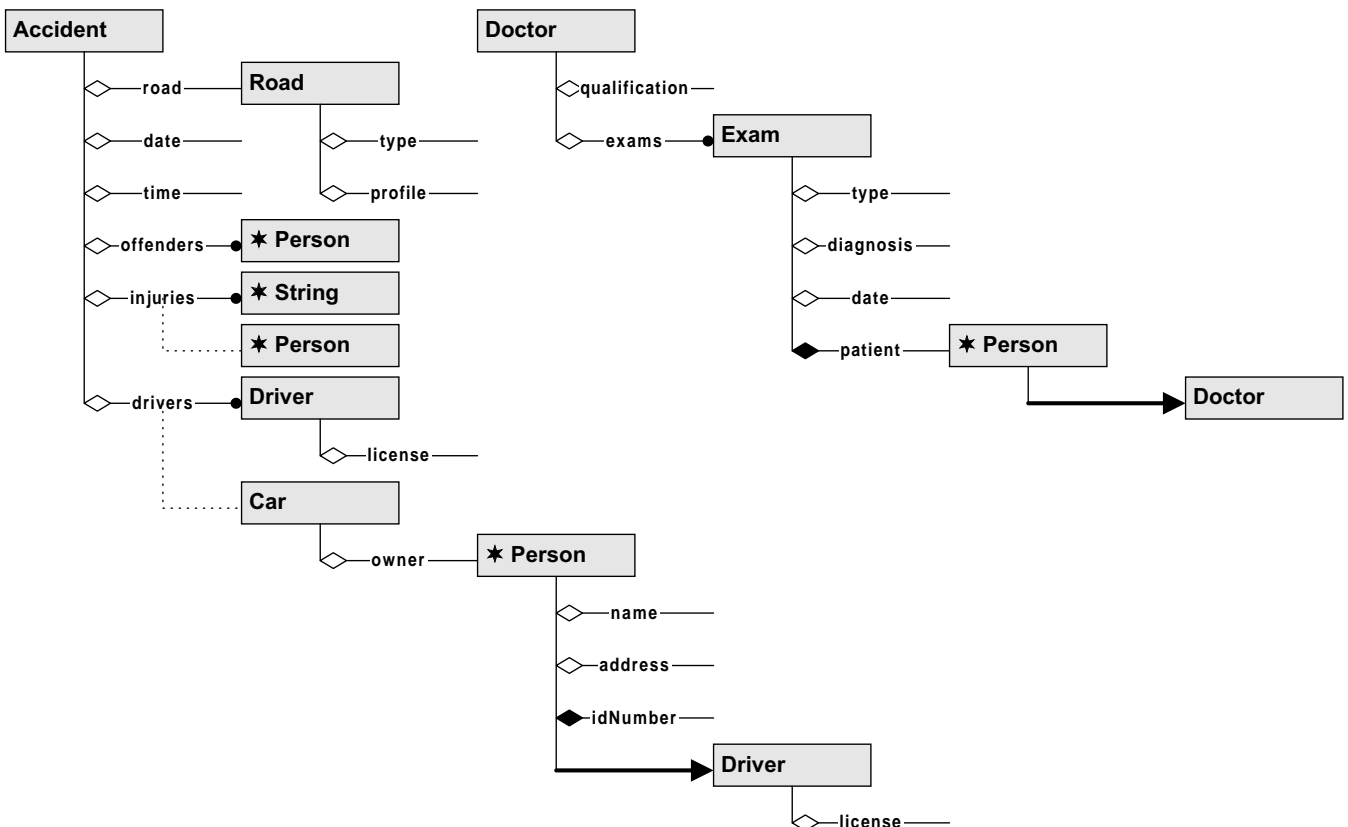
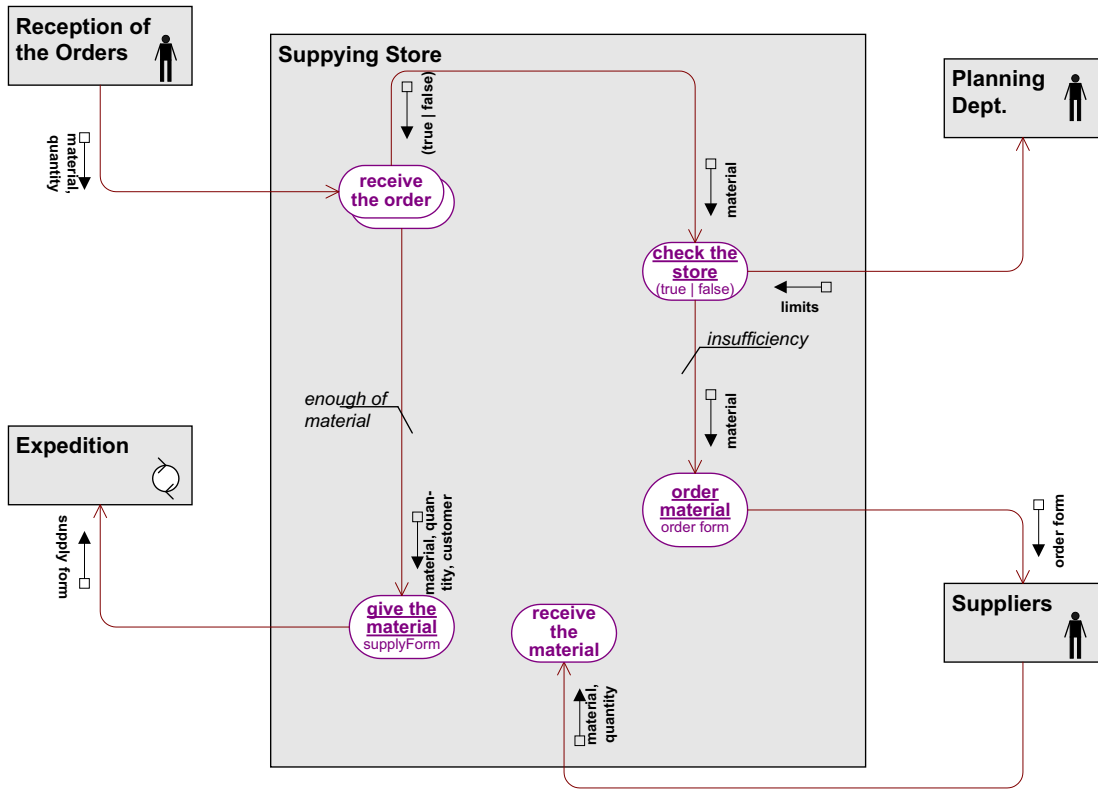
# book - lifecycle

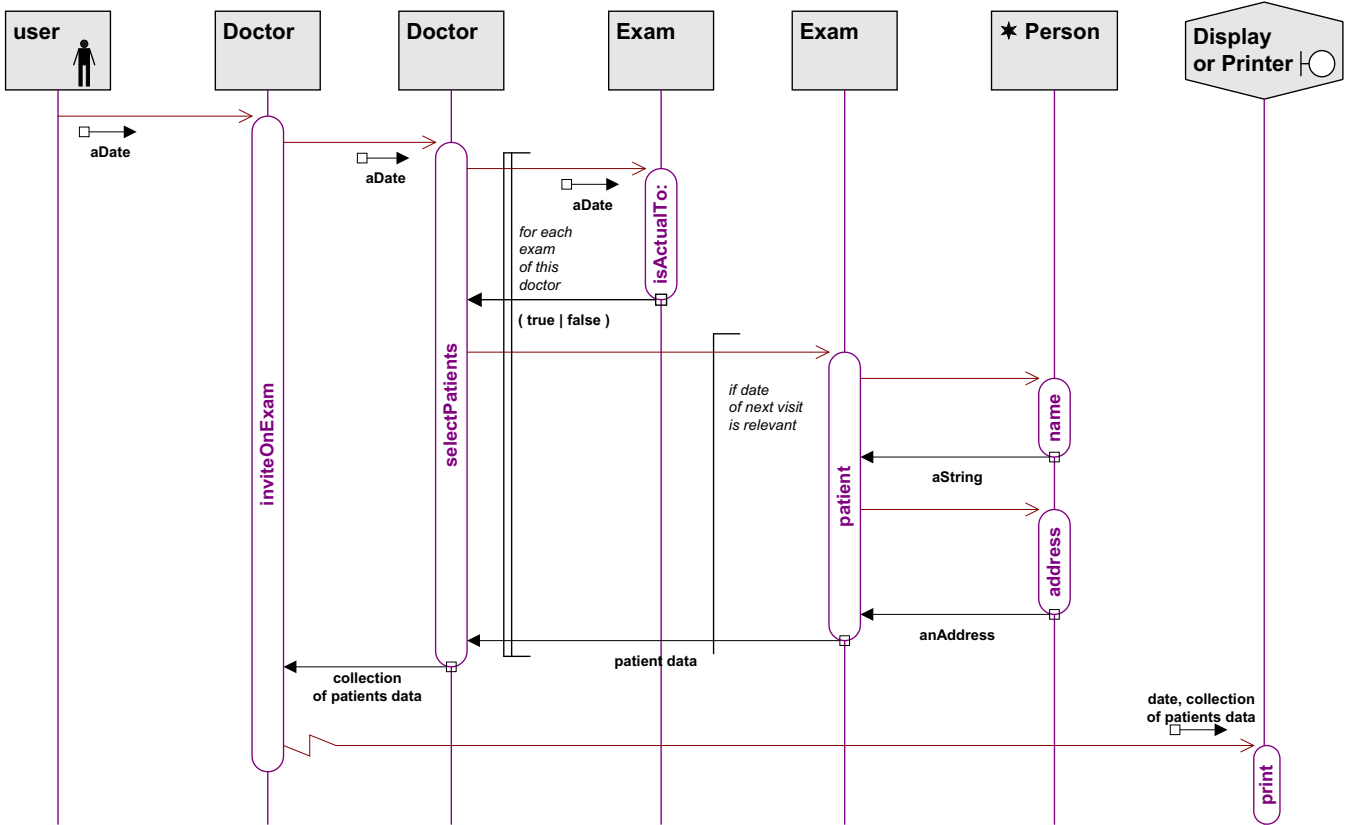




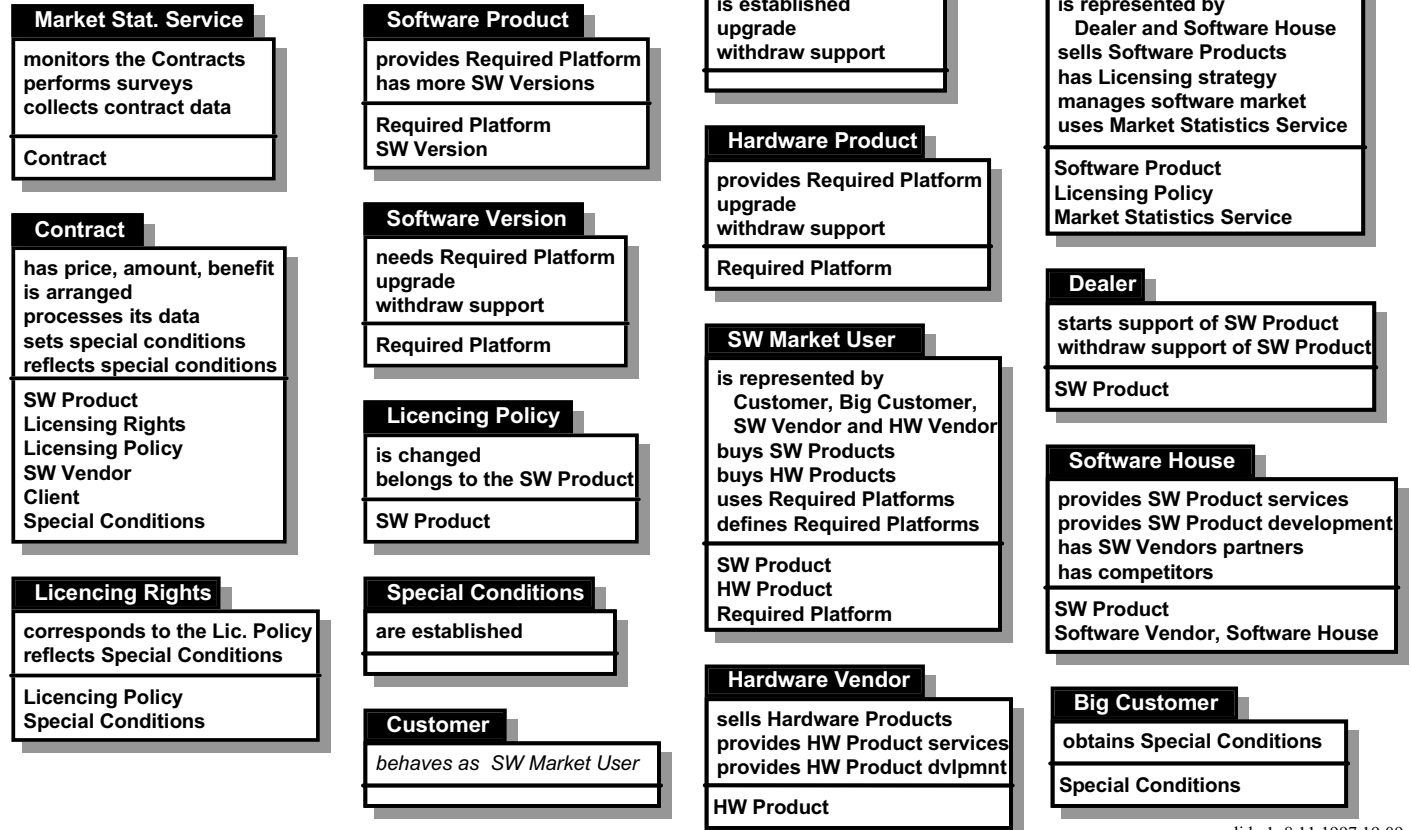








**OBA - Modelling Cards**



slide 1, 8.11.1997 19:09

**OBA - Object Relationships**

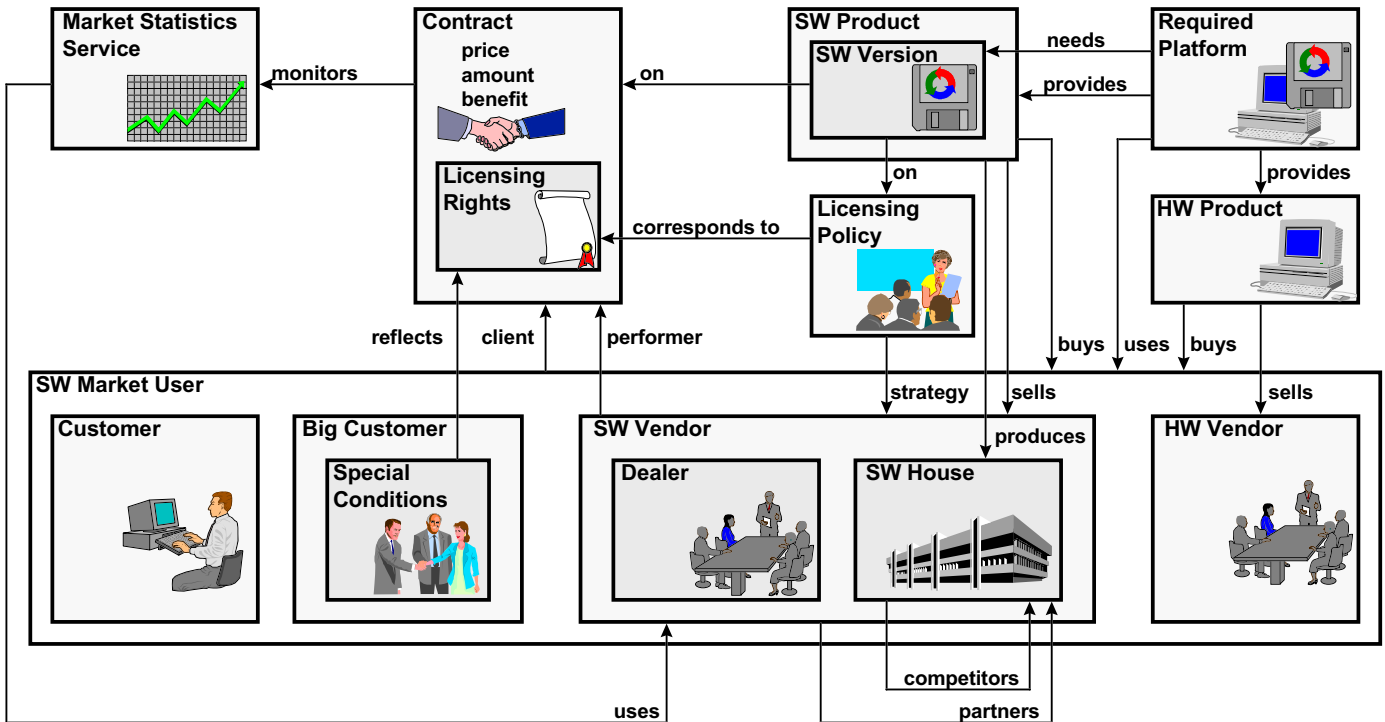
Object	Relation	Related Object
Market Statistics Service	monitors get data from	Contract Contract
Contract	is made on has associated sets up is performed by has client as sets special cond.	Software Product Licensing Rights Licensing Rights Software Vendor SW Market User Special Conditions
Licensing Rights	reflect correspond to	Special Conditions Licensing Policy
Licensing Policy	is made on	Software Product
Software Product	has creates	Software Version Required Platform
Software Version	needs upgrades withdraws support of	Required Platform Required Platform Required Platform
Special Conditions	set up	Licensing Rights
Customer	behaves as	SW Market User
Big Customer	has associated	Special Conditions
Required Platform	is created by is created by	Software Version Hardware Product

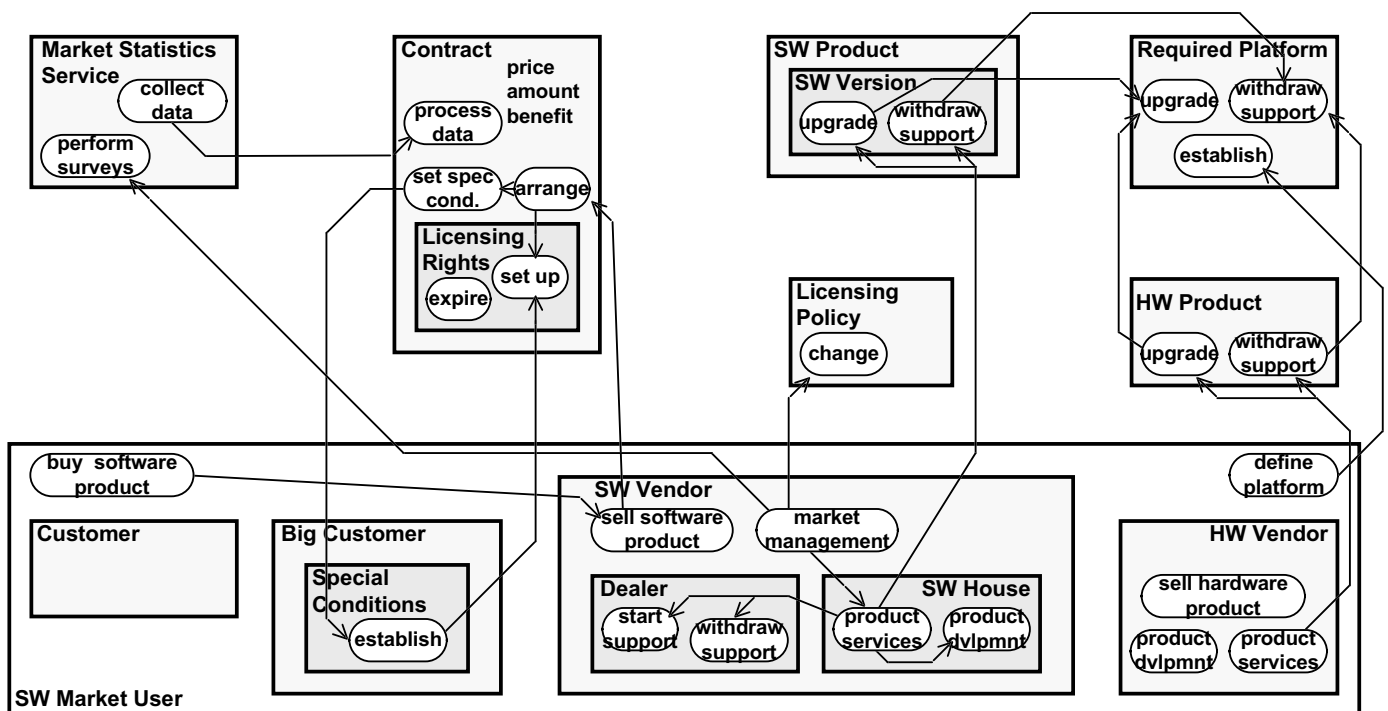
Object	Relation	Related Object
Hardware Product	creates upgrades withdraws support of	Required Platform Required Platform Required Platform
SW Market User	is represented by is represented by is represented by buys buys from buys uses defines	Customer Big Customer Software Vendor Hardware Vendor Software Product Software Vendor Hardware Product Required Platform Required Platform
Software Vendor	is represented by is represented by has strategy of sells performs changes uses	Dealer Software House Licensing Policy Software Product Contract Licencing Policy Market Stat. Service
Dealer	behaves as	Software Vendor
Software House	produces upgrades withdraws support of has partners of has competitors of	Software Product Software Product Software Product Software Vendor Software House
Hardware Vendor	sells upgrades withdraws support of	Hardware Product Hardware Product Hardware Product

slide 2, 8.11.1997 19:09

ORD - Initial Analysis - subjects, associations

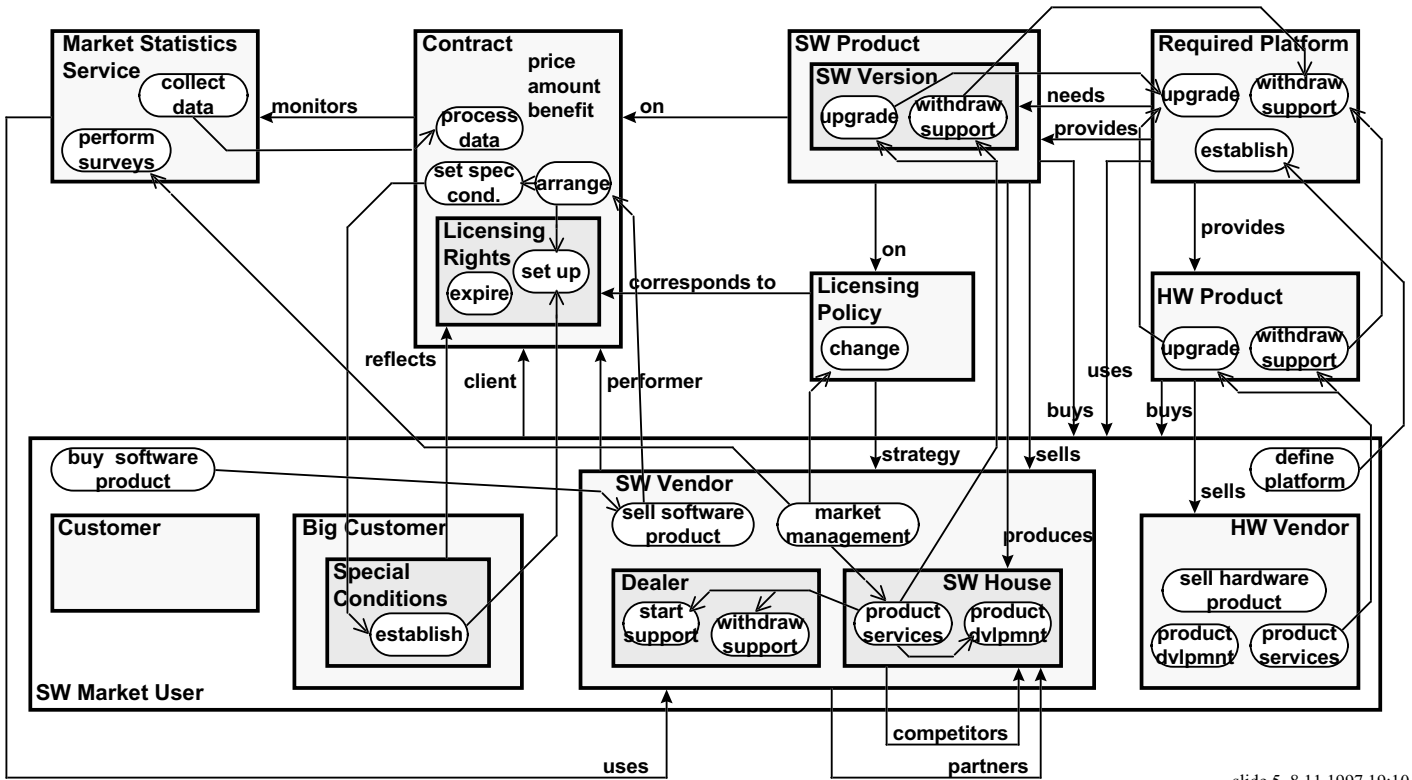


ORD - Initial Analysis - subjects, behaviours and communications

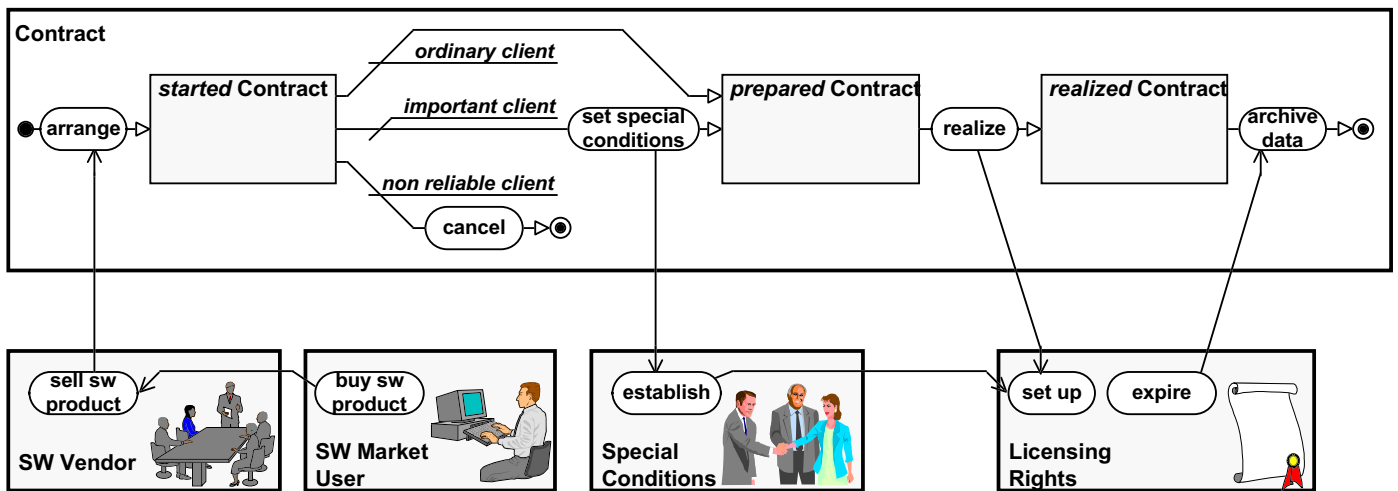


ORD - Initial Analysis

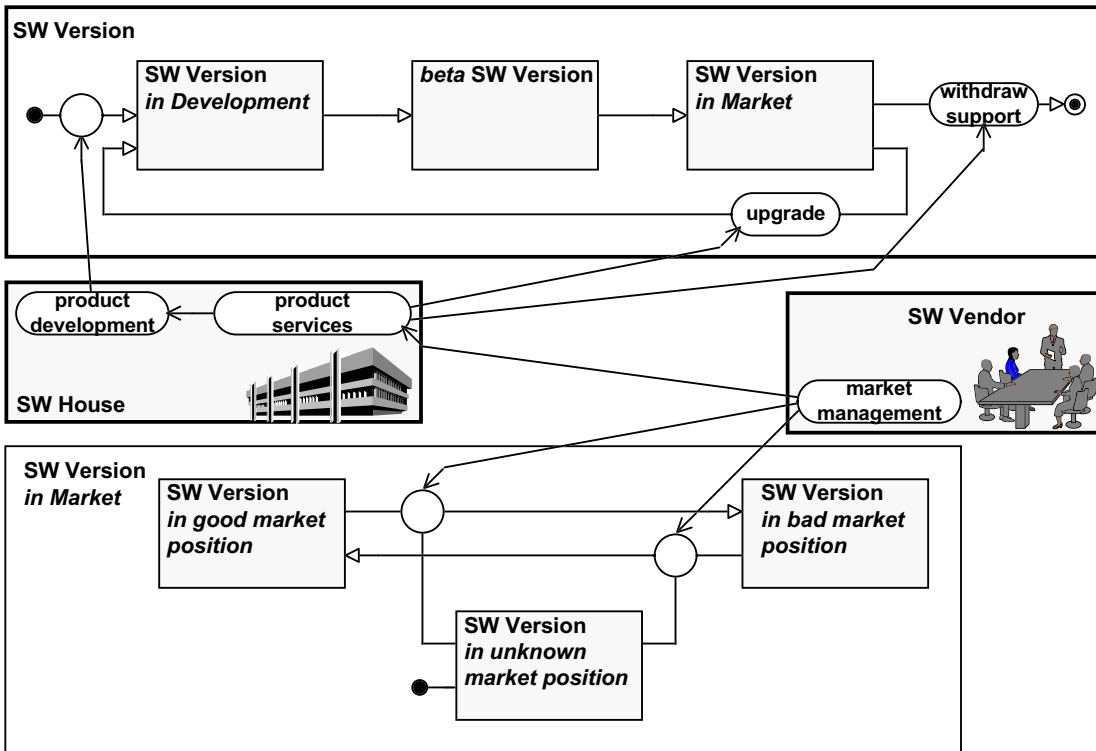
- subjects, associations, behaviours and communications



ORD - Initial Analysis, Contract Life-Cycle

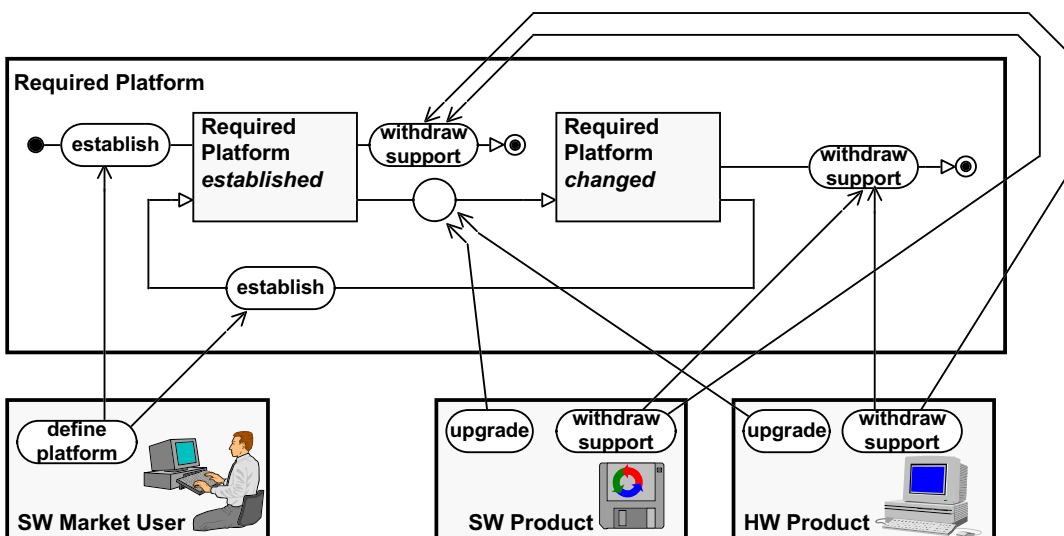


ORD - Initial Analysis, SW Version Life-Cycle



slide 7, 8.11.1997 19:10

ORD - Initial Analysis, Required Platform Life-Cycle



slide 8, 8.11.1997 19:10