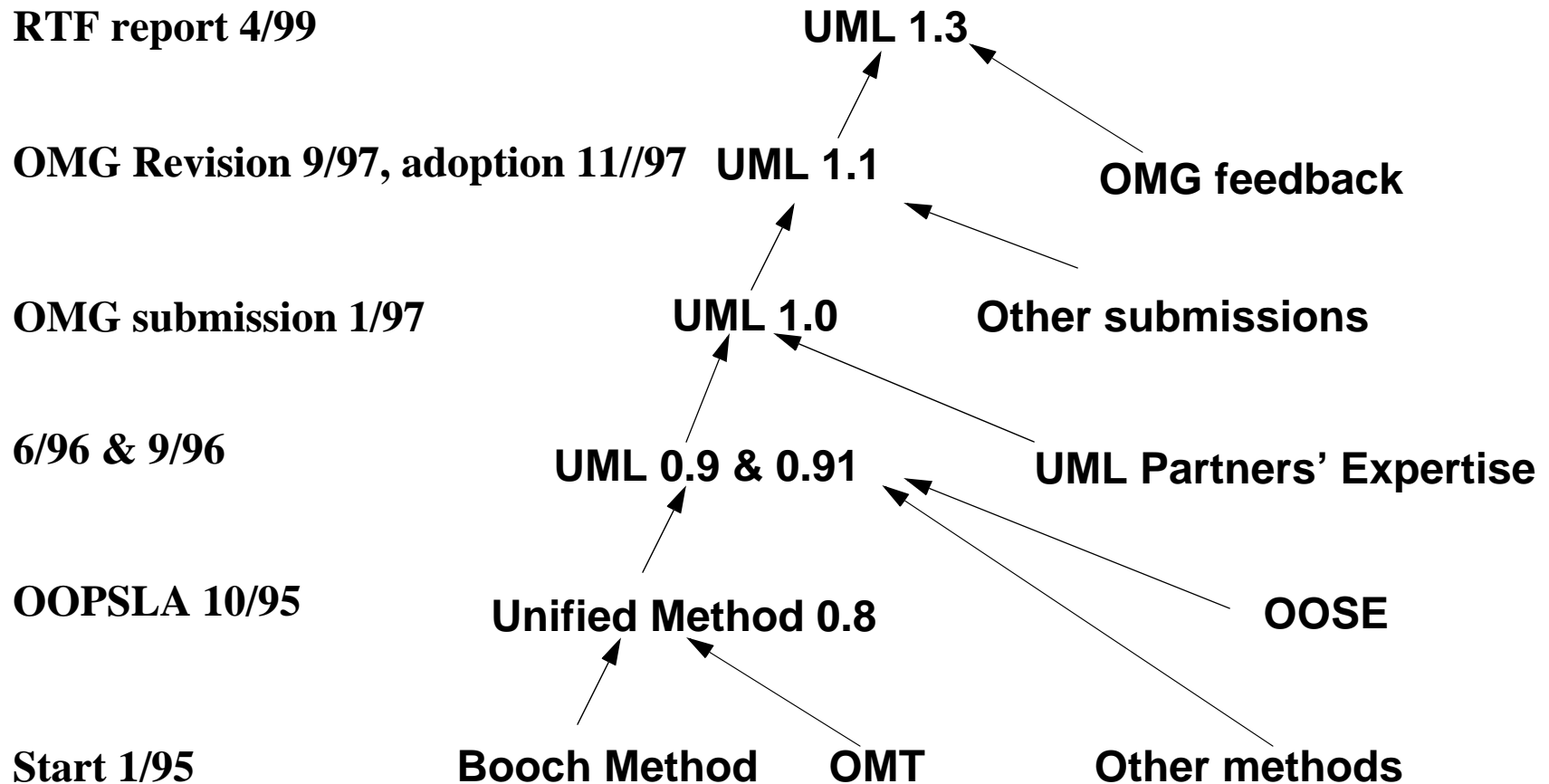

UML

The View from the Front



James Rumbaugh
9 March 1999
Rational Software Corporation

Evolution of the UML



UML Revision Goals

- **Fix typographical errors and omissions**
- **Fix technical bugs and inconsistencies**
- **Clarify ambiguities and special cases**
- **Adjust discrepancies in naming and organization**

Revision Schedule

- Formed committee of 12 persons
- Cris Kobryn was chairman
- Began work December 1997 after OMG adoption
- Completed technical changes December 1998
- Final UML version 1.3 report due April 1999
- Supplement for XMI and IDL compatibility 1 or 2 months later

Revision Process

- **Users submit comments through OMG**
- **Organize comments in a database**
- **Perform internal consistency checks on UML documents**
- **Discuss changes in telephone and OMG meetings**
- **Assign workgroups to update documents**

Workgroup Areas

- **Static structure**
- **Model management**
- **Use cases and collaborations**
- **State machines**
- **Activity graphs**
- **OCL language**
- **IDL format**
- **UML architecture and standards issues**

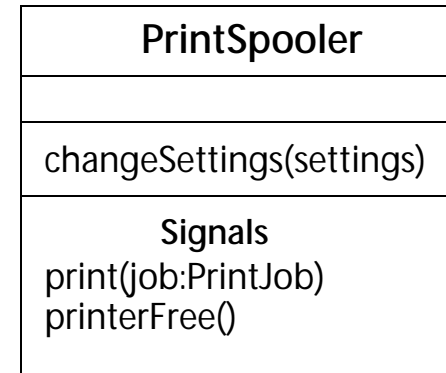
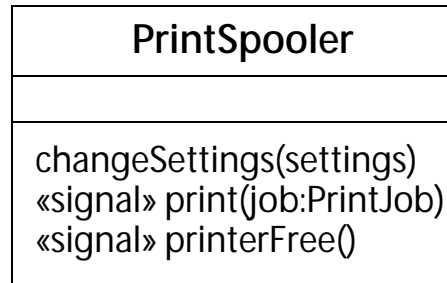
Nature of Changes

- **Over 500 comments submitted**
- **A few changes important to most users**
- **A moderate number of changes important for special areas**
- **Many clarifications of wording and interpretation**
- **Many typos fixed**
- **Lots of small internal issues of interest only to toolmakers or UML groupies**
- **Many comments were declined as incorrect or out of scope**

Static Structure Revisions

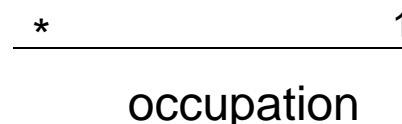
- Allowed associations *from* Classes *to* Interfaces
- Removed unnecessary Generalization stereotypes
 - extends, inherits, private, subclass, subtype, uses -> implementation
- Allowed Classes to declare Signals
- Reorganized and renamed Dependencies

Classes Declaring Signals



Relationships

- Association



- Generalization

- Flow

- become, copy

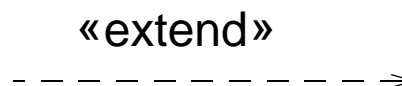
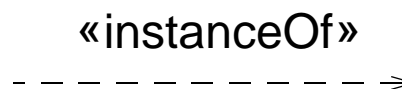
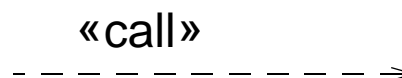
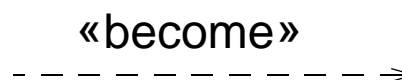
- Dependency

- Metarelationship

- instance, powertype

- Use Case

- extend, include



Dependencies

■ Abstraction

- derivation, realization, refinement, trace

■ Binding

■ Permission

- access, friend, import

■ Usage

- call, create, instantiate, send

«derive»

----->

-----▷

«refine»

----->

«trace»

----->

«bind»(args)

----->

«access»

----->

«friend»

----->

«import»

----->

«call»

----->

«create»

----->

«instantiate»

----->

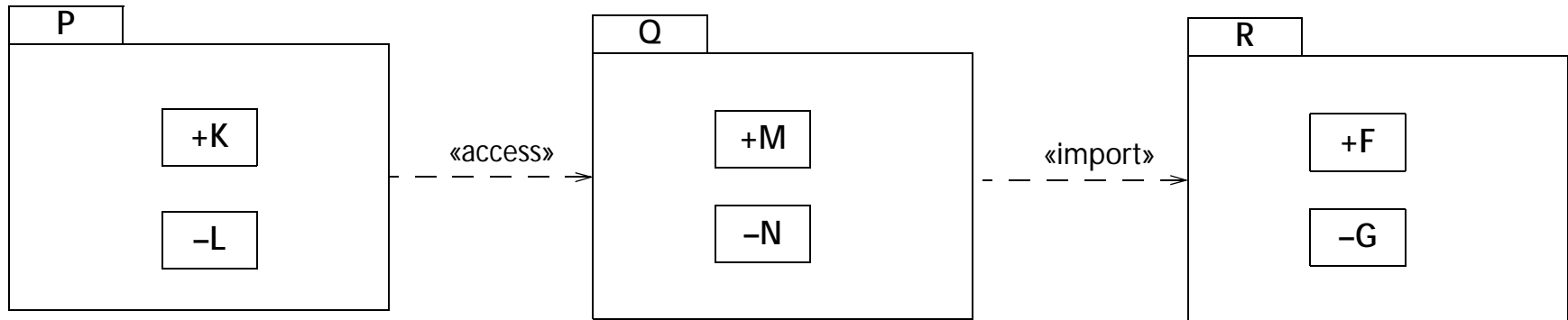
«send»

----->

Model Management Revisions

- **Distinguished «access» Permission from «import»**
 - **access: permit access**
 - **import: load namespace**
- **Defined Subsystem with multiple viewpoints**
 - **specification**
 - **implementation**
 - **others possible**

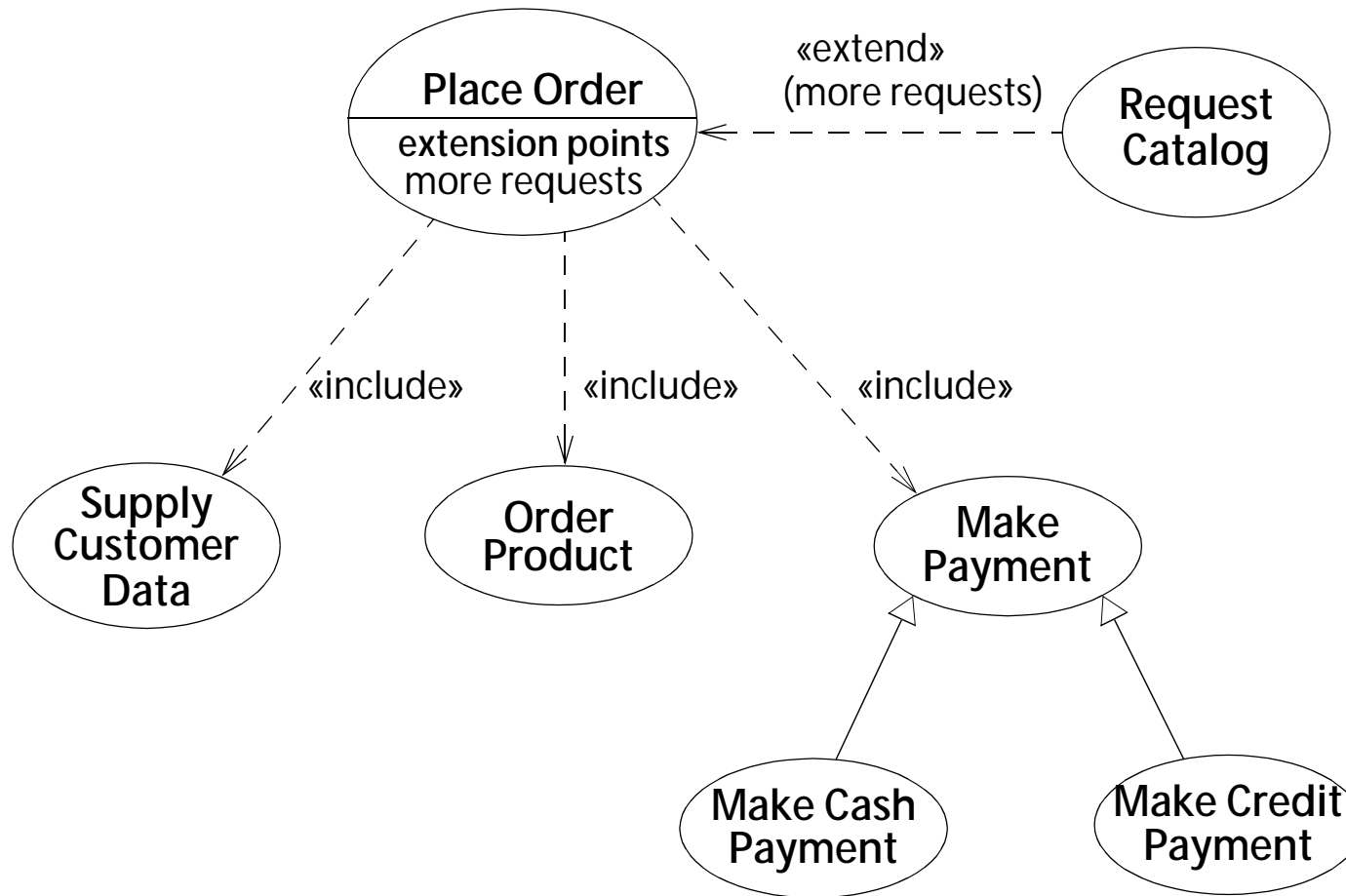
Permissions



Use Case Revisions

- Replaced «uses» relationship with Generalization and «include» dependency
- Reclassified «extend» relationship as a Dependency
- Defined format of «extend» relationship and extension points

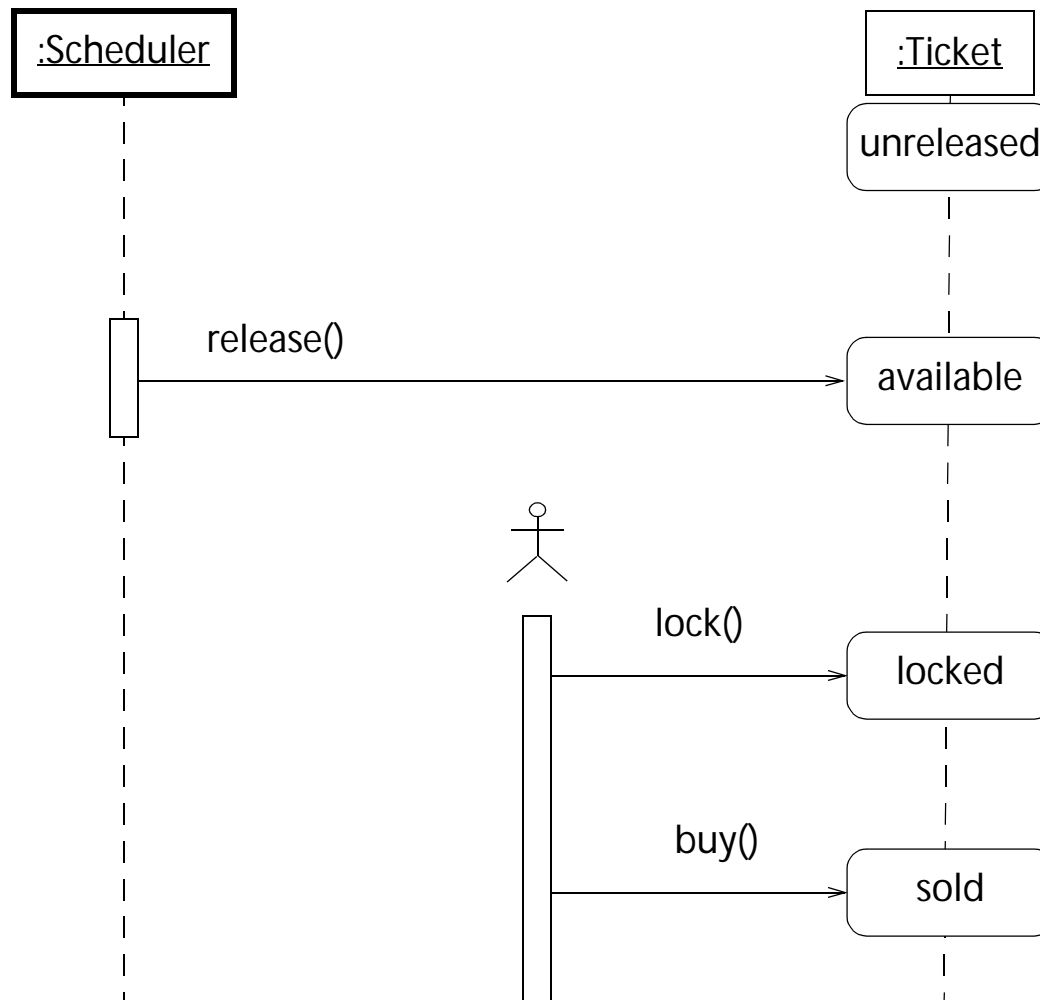
Use Case Relationships



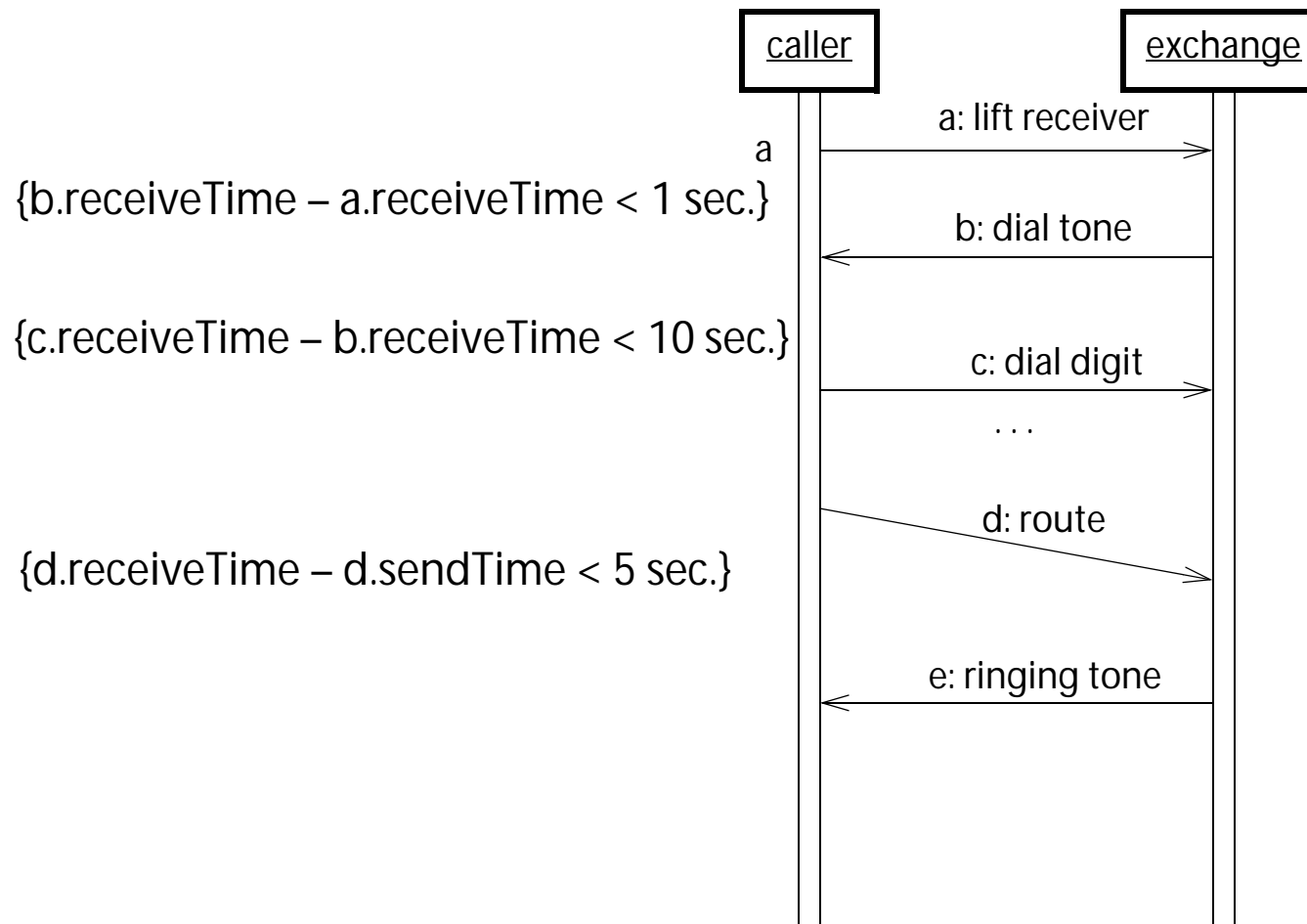
Sequence Diagram Revisions

- **Allow states on lifelines to show change of state during execution**
- **Replaced timing marks by functions on message names**
- **Show either objects or roles in an interaction**

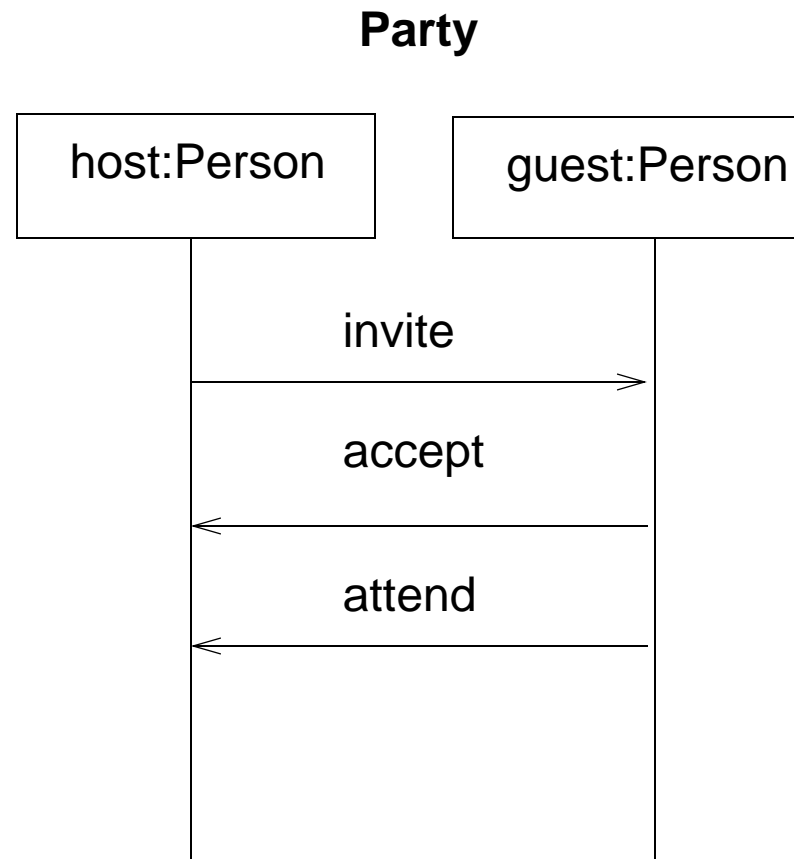
State on Lifeline



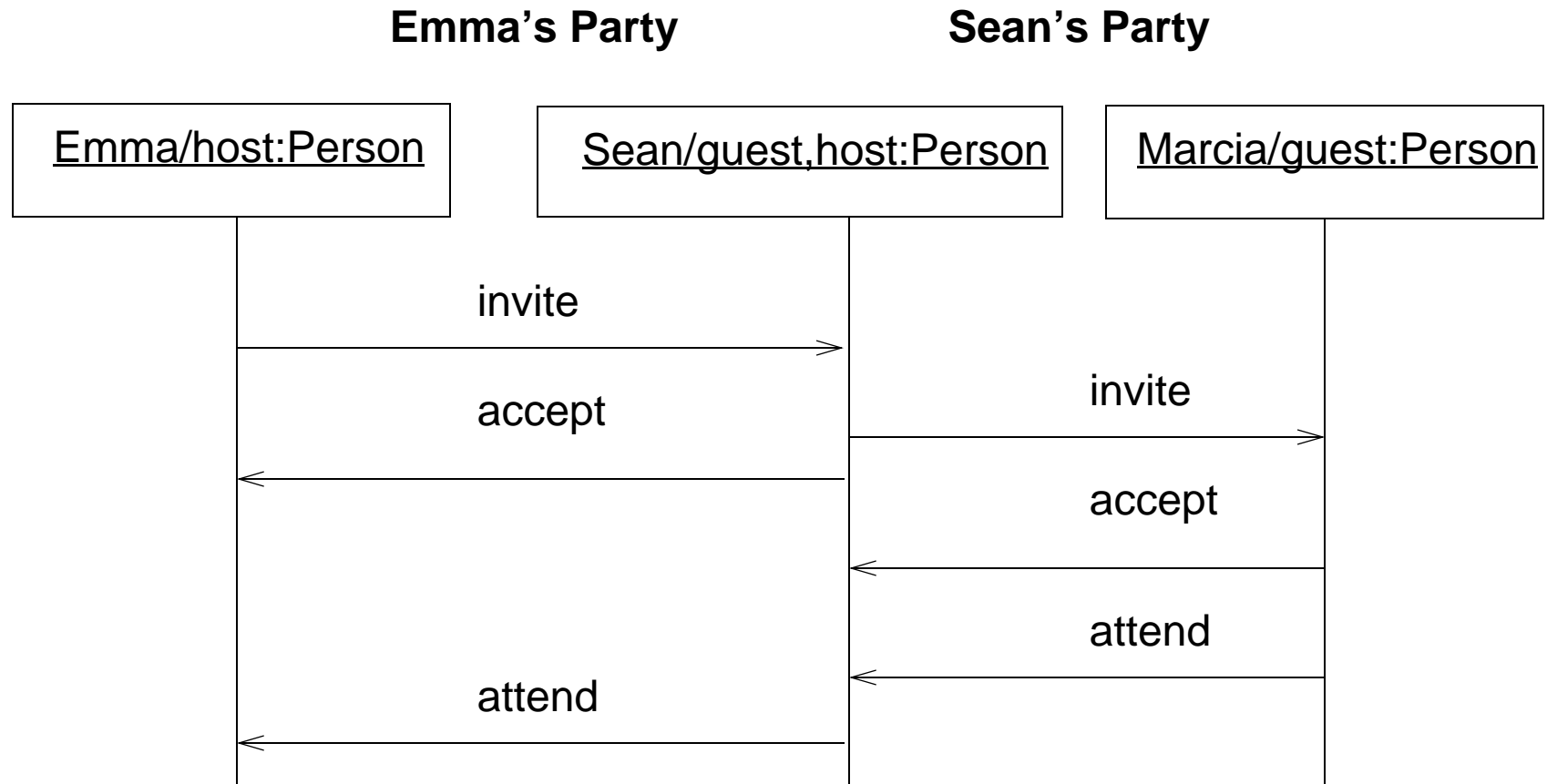
Timing Functions



Roles vs. Objects: Roles



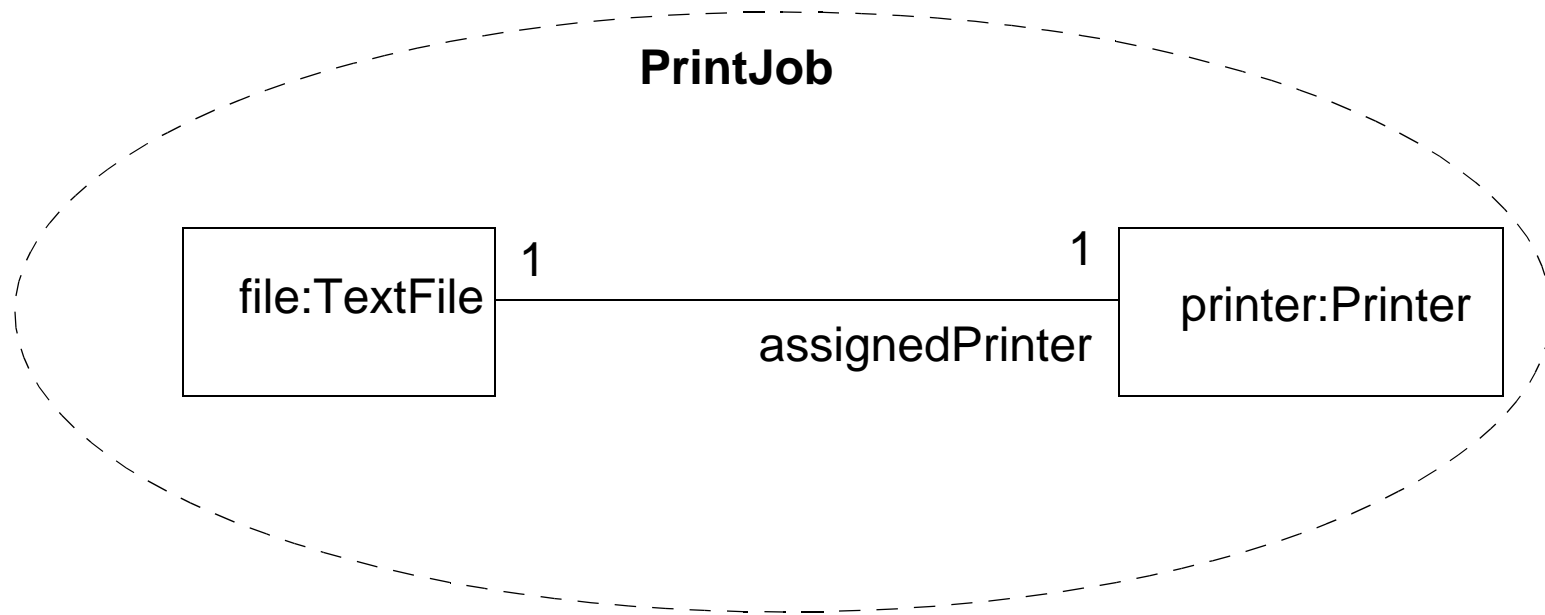
Objects



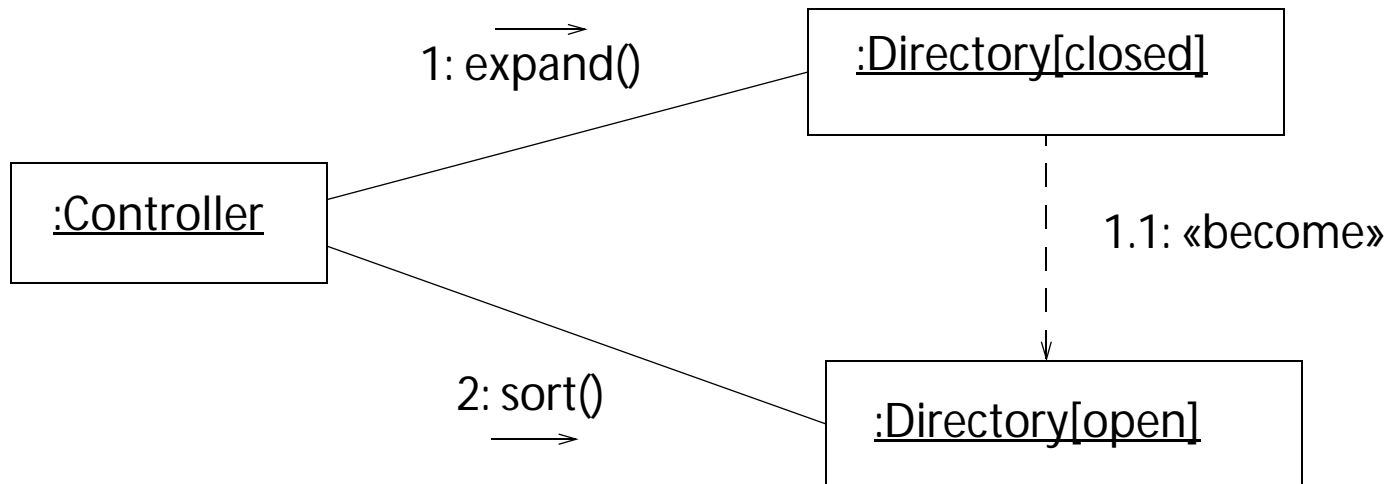
Collaboration Diagram Revisions

- Automatically generate base association if not explicit
- Allow sequence numbers on «become» and «copy»
- Apply actions concurrently to set of target objects

Collaborations



Become and Copy



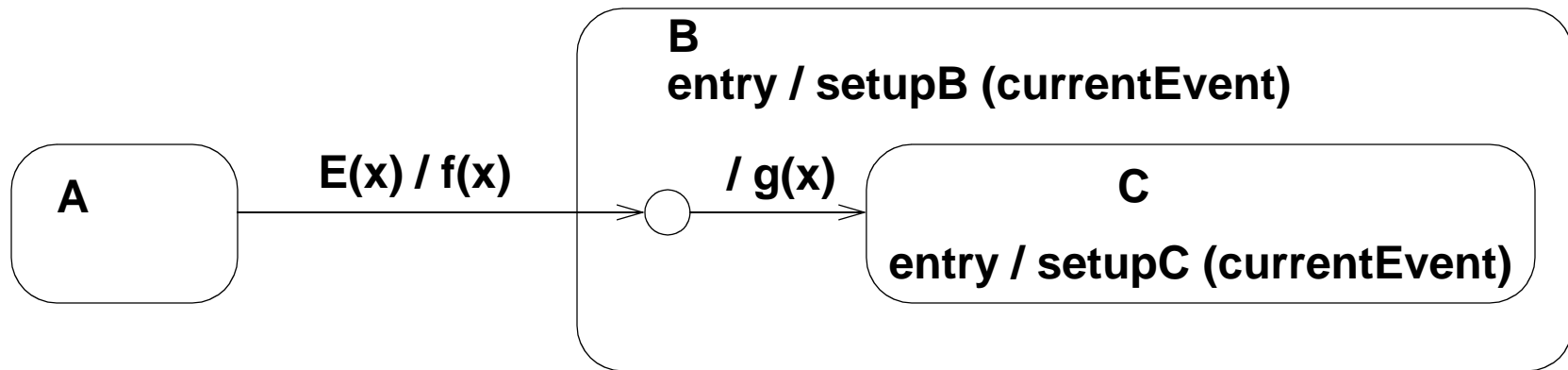
Multiple Targets of Action

targets . op (arguments)

State Modeling Revisions

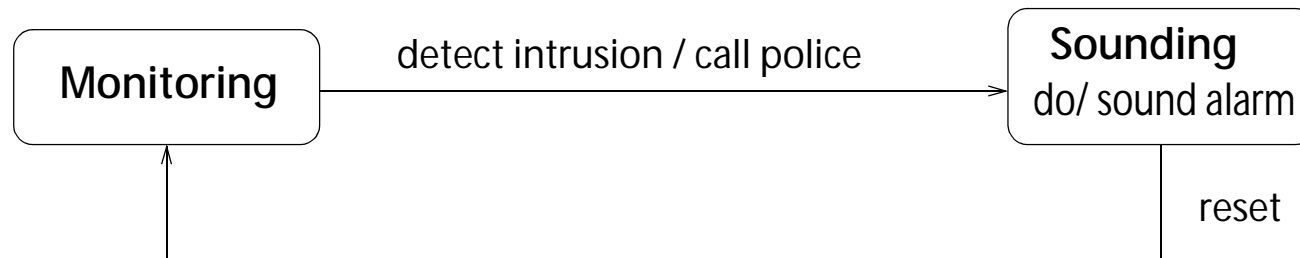
- Added currentEvent to handled chained transitions
- Dropped special syntax (^) for sending signals
- Added continuous activity to states
- Allowed call events as alternative implementation of operation calls

Chained Transitions



$E(x) / f(x); \text{setupB } (E(x)); g(x); \text{setupC } (E(x))$

Continuous Activity

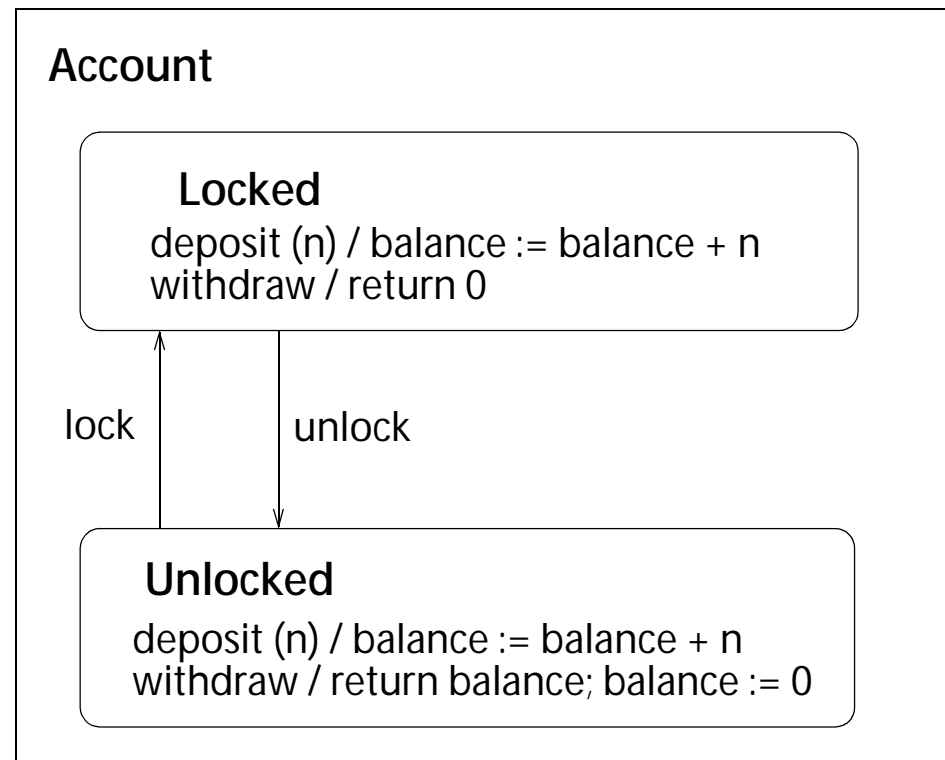


Call Events

calling procedure text

```
deposit (10);  
...  
amount := withdraw ()
```

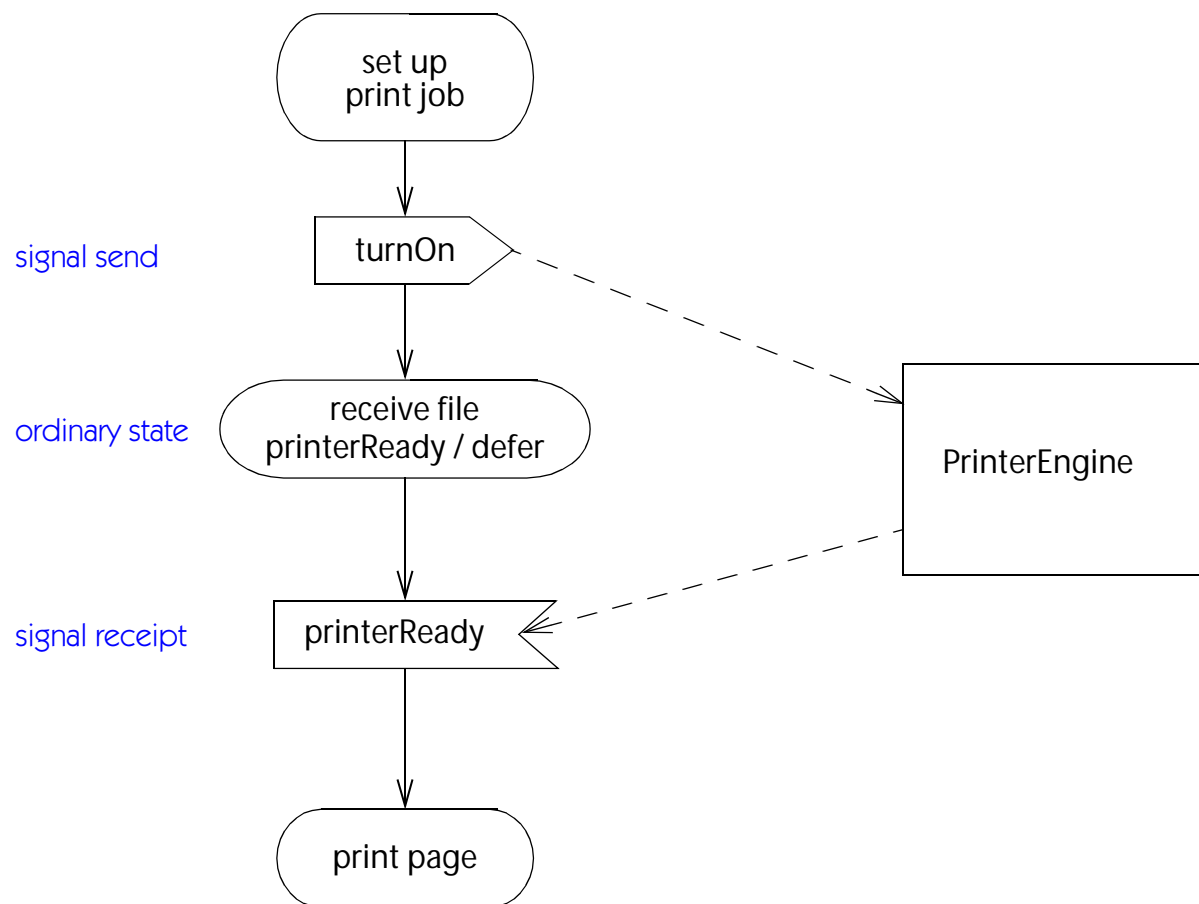
Get all or nothing
depending on whether
the account is locked.



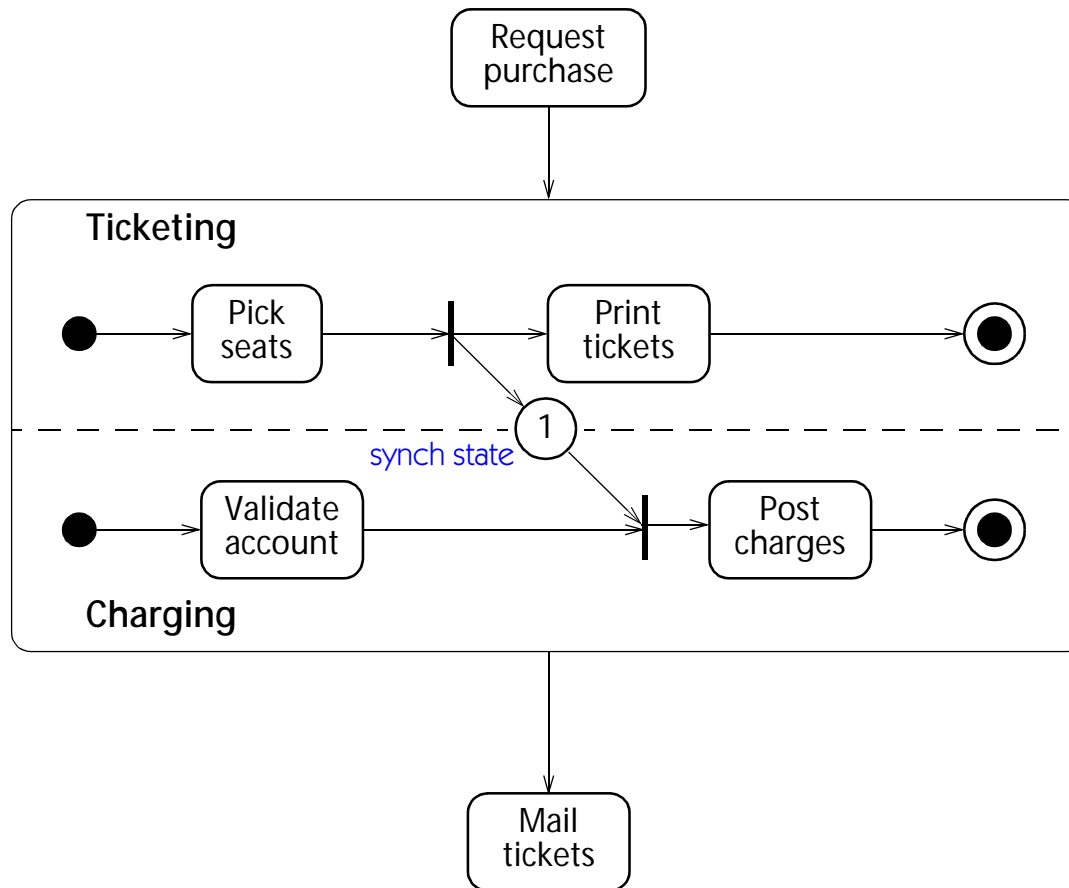
Activity Graph Revisions

- Allowed events to be deferred in states
- Provided synch states for synchronizing concurrent activities
- Added special icons for sending and receiving signals, conditional threads
- Added dynamic control of spawning parallel subactivities

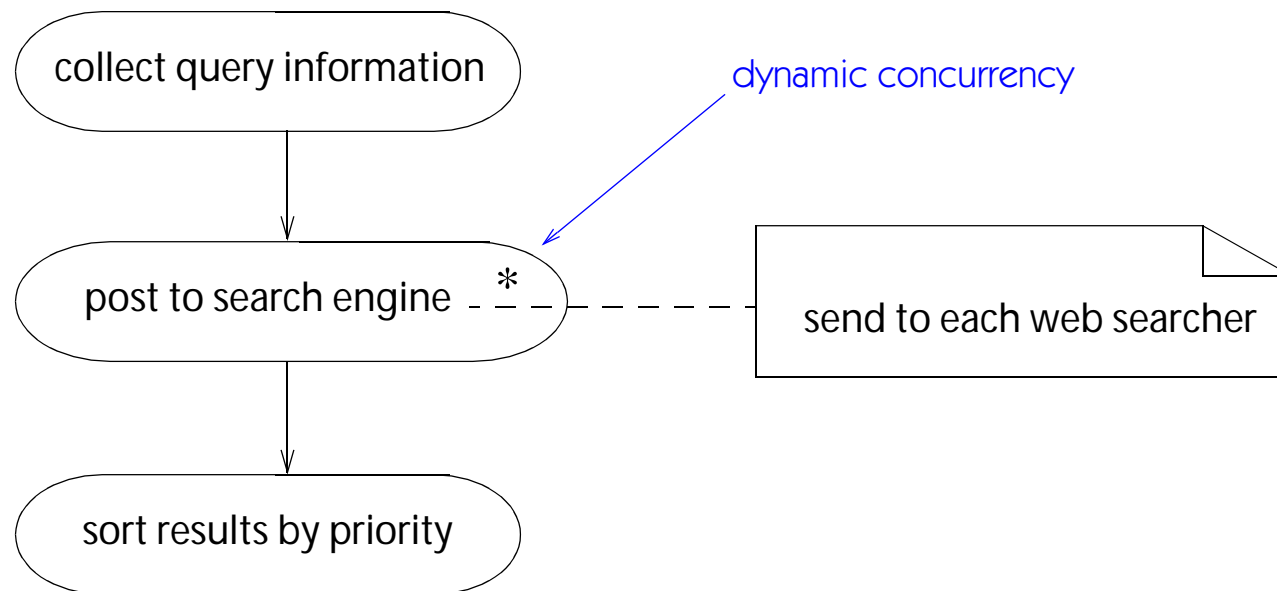
Deferred Events and Special Icons



Synch States



Dynamic Parallelism



Existing RFPs for Extensions

- **Stream-based model interchange using XML**
- **Semantics of executable models (“action language”)**
- **Profile for business enterprise modeling**
- **Real-time systems (architecture, performance, reliability)**

Other Interim Issues

- Standard way to define profiles
- Diagram interchange formats

Possible Topics of UML 2.0

- **First-class extensibility mechanism**
- **Precise specification of refinement semantics**
- **Versioning of models**
- **More permissive concurrency in activity and state graphs**
- **Associations at several levels**

UML Assessment

- **Clearly the dominant modeling language**
- **Not as clean as if one person had made it, but much better accepted**
- **Some mechanisms should be more general in the future, but a conservative approach was probably best**
- **Standard elements are uneven, probably need cleanup**
- **Pressures to apply UML widely, possibly beyond its intent**
- **Danger of incoherence if too many independent OMG RFPs push UML in different directions**
- **Vendors and users will provide pragmatic pushback to untried theory**