PROCESSES AND THREADS

- A process is a heavyweight flow that can execute concurrently with other processes.
- A thread is a lightweight flow that can execute concurrently with other threads within the same process.
- An active object is an object that owns a process or thread and can initiate control activity.
- An active class is a class whose instances are active objects.
- Graphically, an active class is rendered as a rectangle with thick lines. Processes and threads are rendered as stereotyped active classes. Figure 1:Active class

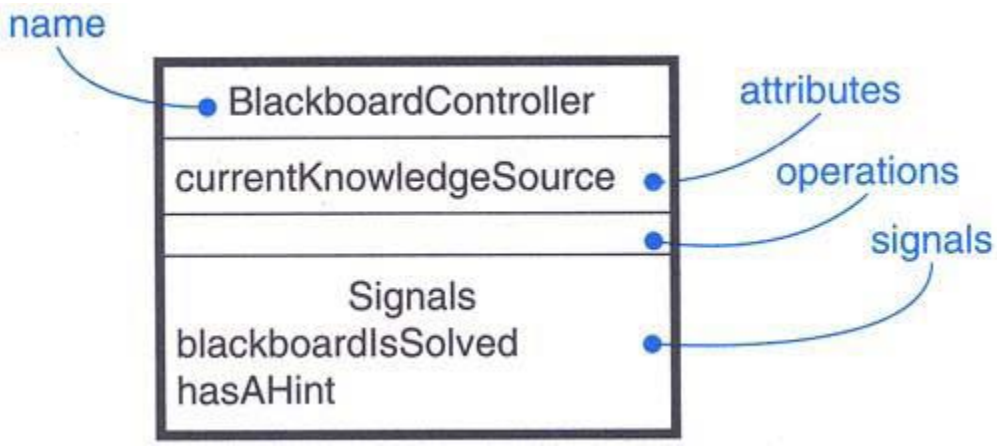

Figure 1: Active Class

Flow of Control

In a sequential system, there is a single flow of control. i.e, one thing, and one thing only, can take place at a time.

In a concurrent system, there is multiple simultaneous flow of control i.e, more than one thing can take place at a time.

Classes and Events

- Active classes are just classes which represents an independent flow of control
- Active classes share the same properties as all other classes.
- When an active object is created, the associated flow of control is started; when the active object is destroyed, the associated flow of control is terminated

- *two standard stereotypes* that apply to active classes are, ***<<process>>*** – Specifies a heavyweight flow that can execute concurrently with other processes. (heavyweight means, a thing known to the OS itself and runs in an independent address space) ***<<thread>>*** – Specifies a lightweight flow that can execute concurrently with other threads within the same process (lightweight means, known to the OS itself.)
- All the threads that live in the context of a process are peers of one another

Communication

- In a system with both active and passive objects, there are *four possible combinations of interaction*
- *First*, a message may be passed from one passive object to another
- *Second*, a message may be passed from one active object to another
- In *inter-process communication* there are two possible styles of communication. *First*, one active object might synchronously call an operation of another. *Second*, one active object might asynchronously send a signal or call an operation of another object
- a synchronous message is rendered as a full arrow and an asynchronous message is rendered as a half arrow
- Figure 2: shows Communication
- *Third*, a message may be passed from an active object to a passive object
- *Fourth*, a message may be passed from a passive object to an active one

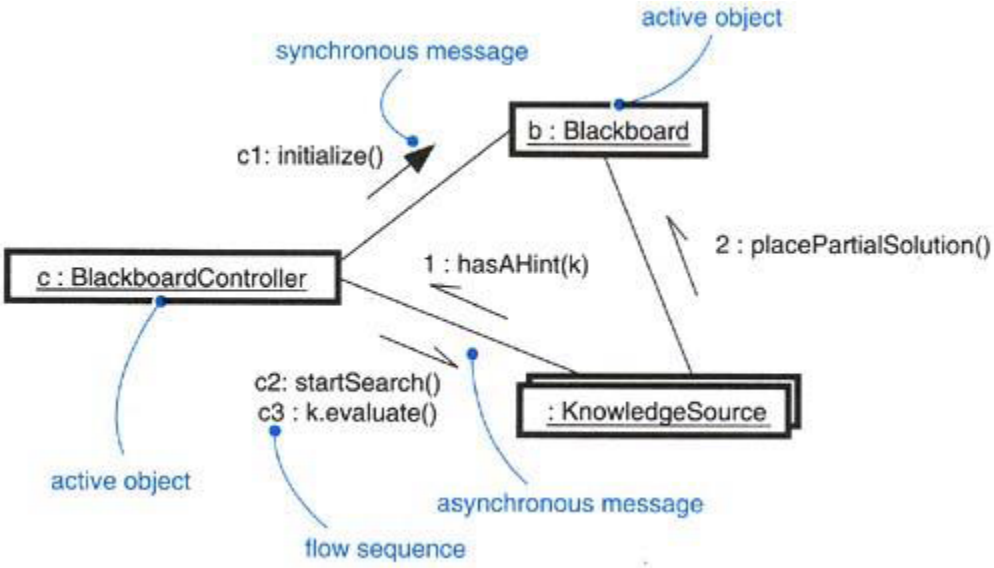

Figure 2: Communication

Synchronization

- synchronization means arranging the flow of controls of objects so that mutual exclusion will be guaranteed.
- in object-oriented systems these objects are treated as a critical region
- Figure 3 Synchronization
- *three approaches* are there to handle synchronization:
- *Sequential* – Callers must coordinate outside the object so that only one flow is in the object at a time
- *Guarded* – multiple flow of control is sequentialized with the help of object's guarded operations. in effect it becomes sequential.
- *Concurrent* – multiple flow of control is guaranteed by treating each operation as atomic
- synchronization are rendered in the operations of active classes with the help of constraints

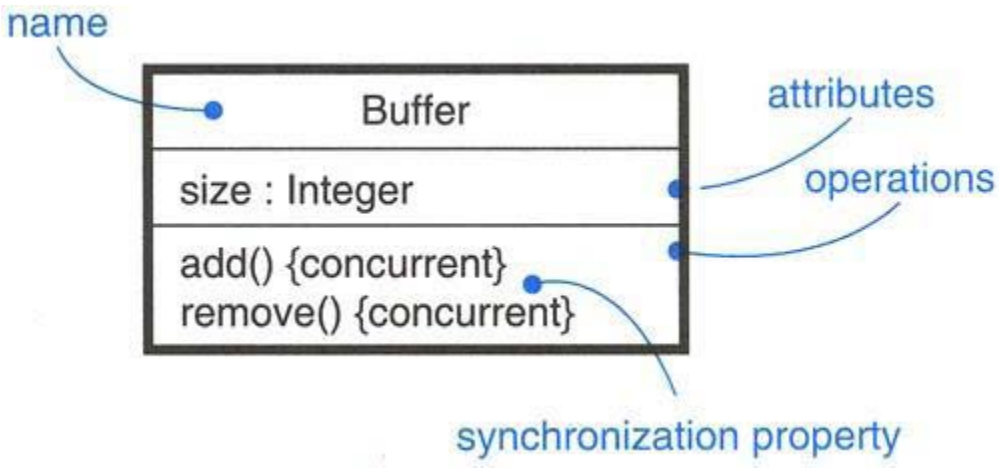

Figure 3: Synchronization

Process Views

- The process view of a system encompasses the threads and processes that form the system's concurrency and synchronization mechanisms.
- This view primarily addresses the performance, scalability, and throughput of the system.

Modeling Multiple Flows of Control

To model multiple flows of control,

- · Identify the opportunities for concurrent action and reify each flow as an active class. Generalize common sets of active objects into an active class. Be careful not to overengineer the process view of your system by introducing too much concurrency.
- · Consider a balanced distribution of responsibilities among these active classes, then examine the other active and passive classes with which each collaborates statically. Ensure that each active class is both tightly cohesive and loosely coupled relative to these neighboring classes and that each has the right set of attributes, operations, and signals.
- · Capture these static decisions in class diagrams, explicitly highlighting each active class.
- · Consider how each group of classes collaborates with one another dynamically. Capture those decisions in interaction diagrams. Explicitly show active objects as the root of such flows. Identify each related sequence by identifying it with the name of the active object.
- · Pay close attention to communication among active objects. Apply synchronous and asynchronous messaging, as appropriate.
- · Pay close attention to synchronization among these active objects and the passive objects with which they collaborate. Apply sequential, guarded, or concurrent operation semantics, as appropriate.

  Figure 4 shows part of the process view of a trading system.

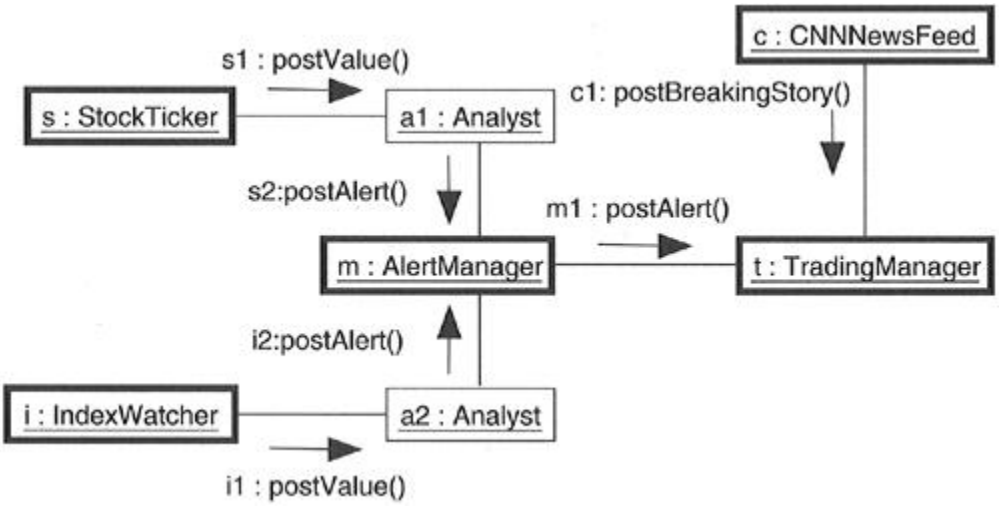

Figure 4: Modeling Flows of Control

Modeling Interprocess Communication

To model interprocess communication,

- · Model the multiple flows of control.
- · Consider which of these active objects represent processes and which represent threads. Distinguish them using the appropriate stereotype.
- · Model messaging using asynchronous communication; model remote procedure calls using synchronous communication.
- · Informally specify the underlying mechanism for communication by using notes, or more formally by using collaborations.

Figure 5 shows a distributed reservation system with processes spread across four nodes.

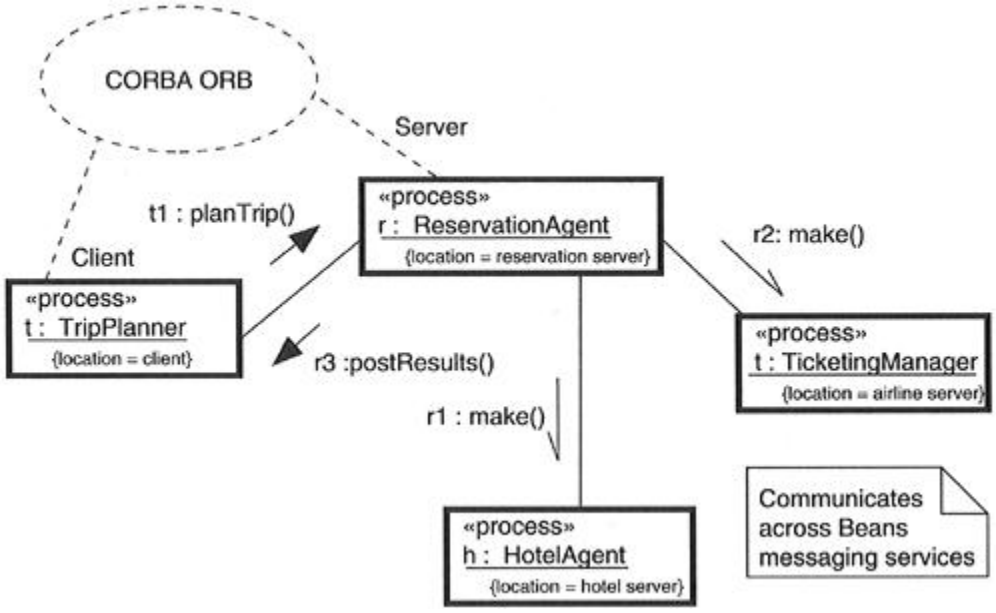

Figure 5: Modeling Interprocess Communication

Source : http://praveenthomasln.wordpress.com/2012/04/07/processes-and-threads-s8-cs/