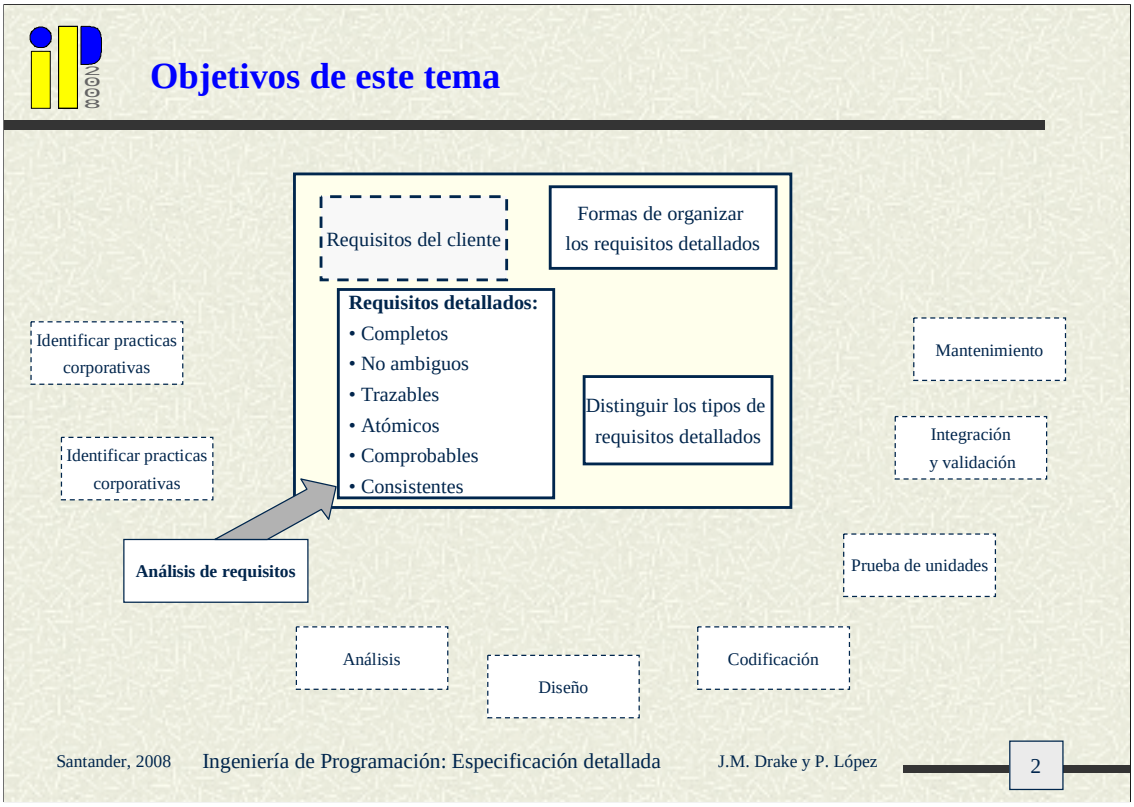
Especificación detallada de una aplicación

José M. Drake y Patricia López
Computadores y Tiempo Real

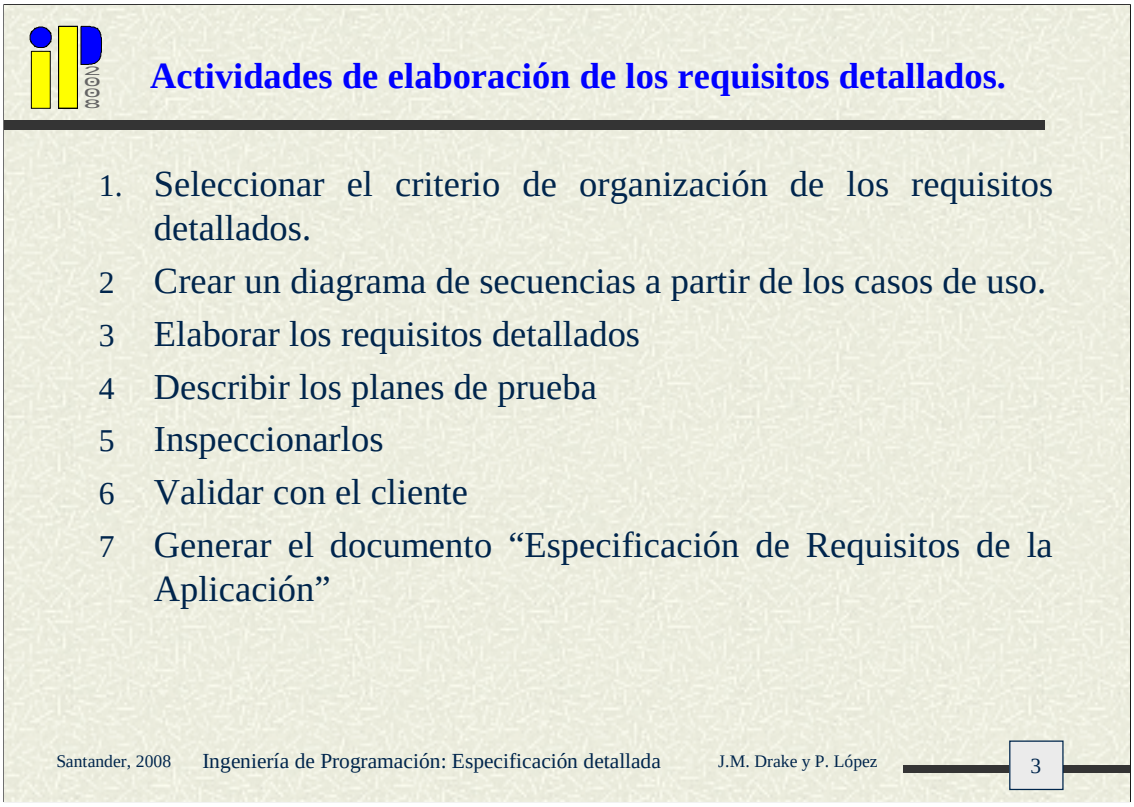
Santander, 2008

1

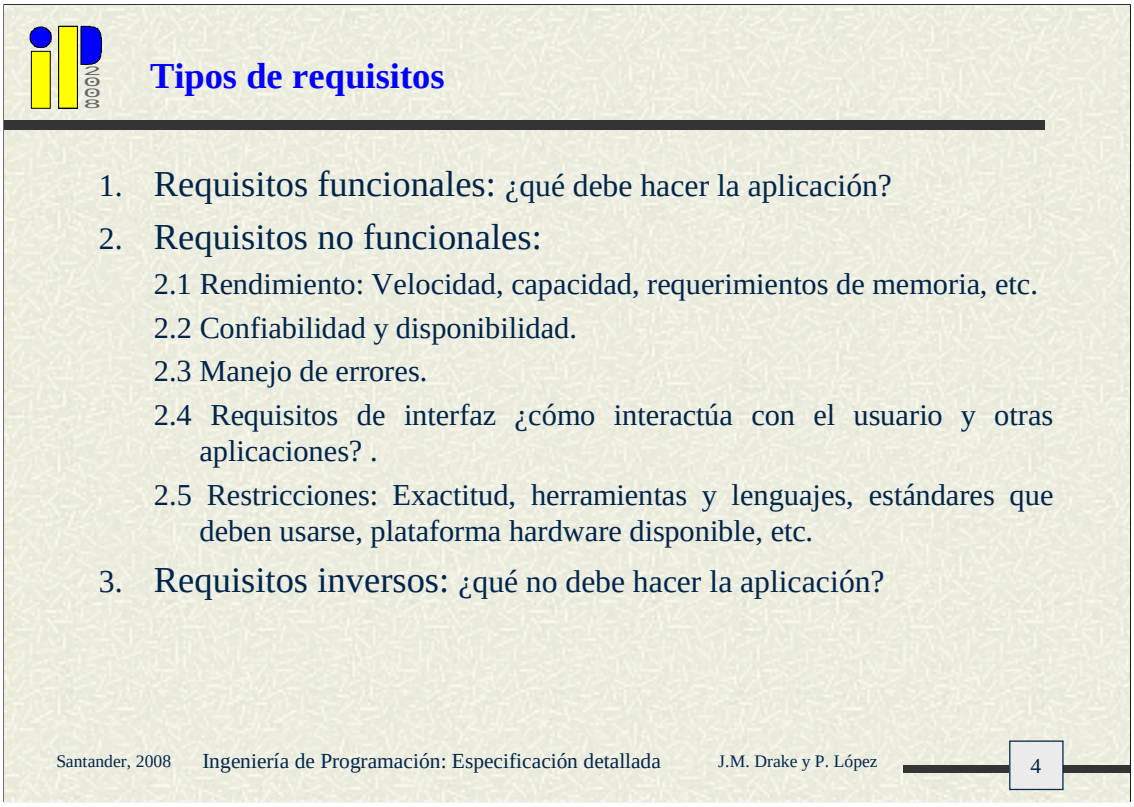


Los requisitos detallados constituyen el único lugar donde se describen la naturaleza exacta y completa de la aplicación. El nivel de detalle debe ser completo, pero se debe evitar la redundancia.
Los requisitos detallados resultan de un refinamiento de los recursos de cliente y deben ser obviamente consistentes con ellos.
Los requisitos detallados es la información sobre la que el ingeniero basa su diseño e implementación. Esta información también recibe la denominación de “requisitos específicos”, “requisitos del diseñador” o “requisitos D”. Esta constituida por una descripción completa de las características específicas y la funcionalidad que debe tener la aplicación, formulada con todos los detalles.
Se supone que los desarrolladores leerán los requisitos D.
Objetivos de este tema son:
•Disponer de recursos para organizar los requisitos detallados.
- •Por clase
- •Por caso de uso
- •Por característica.
- •Por evento.

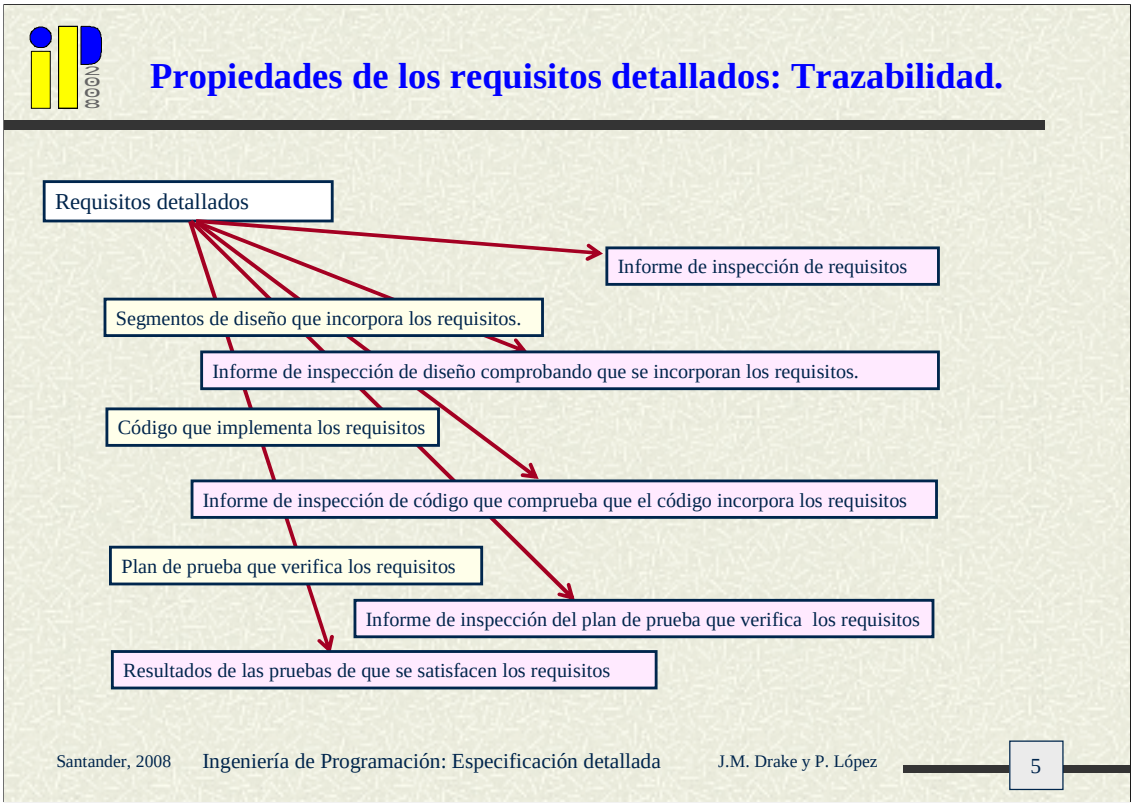
•Poder completar los requisitos:
- •Tener suficiente detalle para elaborar sobre ellos el diseño y la implementación.
- •Poder expresar los requisitos no funcionales, por ejemplo el rendimiento.



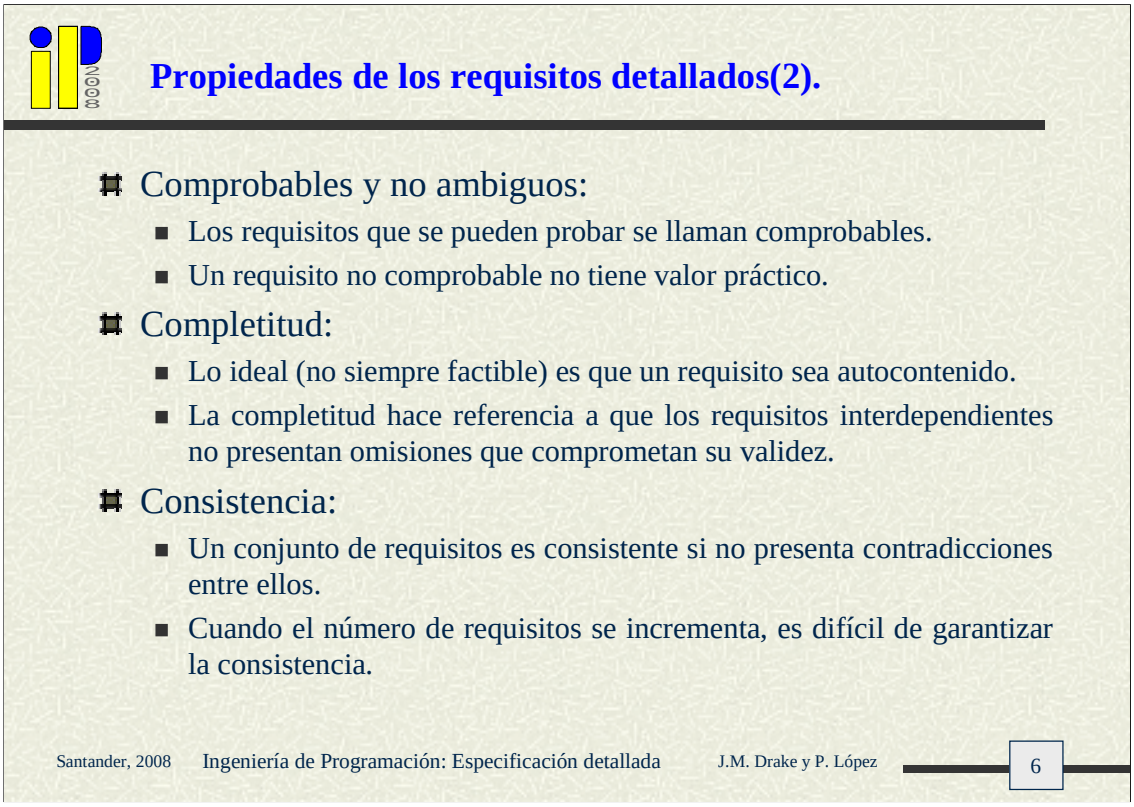
En la transparencia se muestra la secuencia natural de actividades que deben realizarse para generar los requisitos detallados.
Existen diferentes criterios para organizar los requisitos: por característica observable, por modo de operación, por caso de uso, etc.. Una lista no organizada de requisitos se vuelve inmanejable.
Los aspectos estáticos se suelen describir mediante herramientas textuales, mientras que para los aspectos dinámicos y de interacción se suelen utilizar técnicas gráficas como los diagramas de secuencias, de colaboración o de actividad.
En el caso ideal, las pruebas que deben validar los requisitos deben formularse simultáneamente con ellos.
Aunque los requisitos se escriben fundamentalmente para los desarrolladores es muy importante que sean revisados y validados por los usuarios.
El resultado del proceso de análisis de requisitos es la elaboración del documento Especificación de Requerimientos de Software (ERS)



Al nivel de los requisitos C no es tan importante distinguir los diferentes tipos de requisitos, sin embargo a este nivel si, porque va a guiar el desarrollo y el proceso de pruebas de diferente manera.



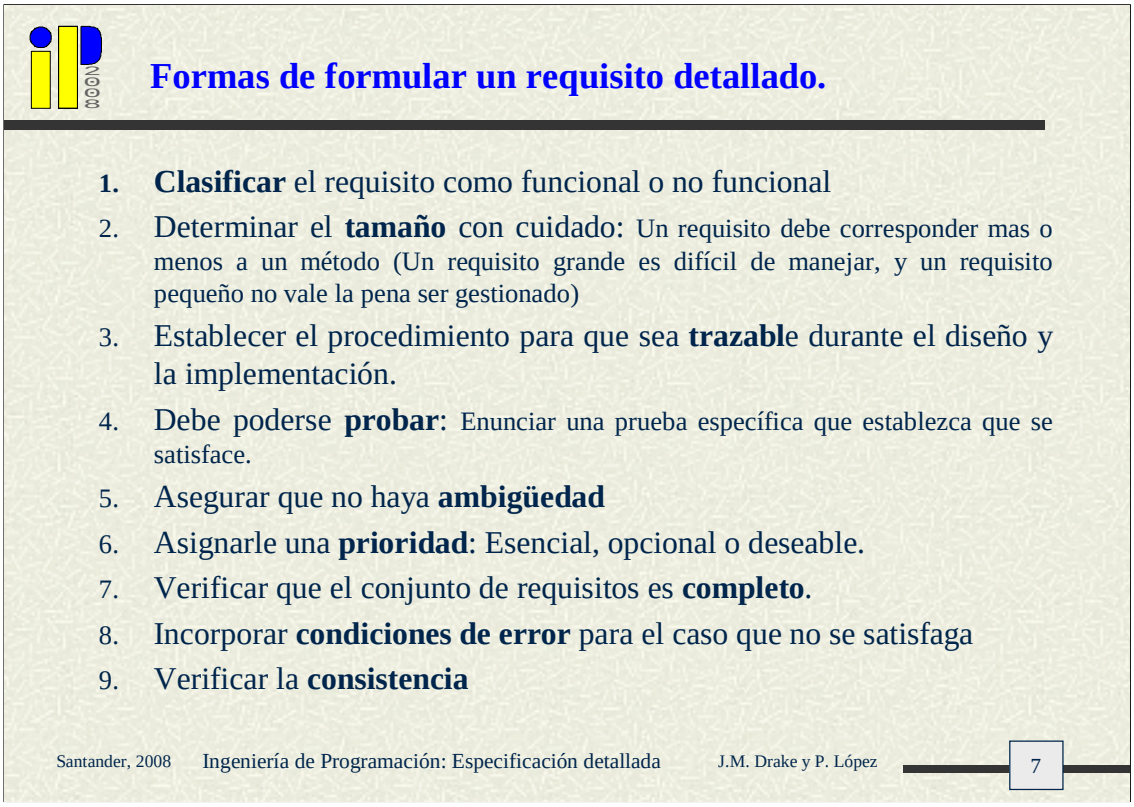
La capacidad que hace corresponder (mapear) cada requisito con las partes de la aplicación en las que se plasma se denomina **trazabilidad**. Una forma de lograr esto es mantener una correspondencia entre cada requisito y un método o función específica del código.
Conforme avanza el proyecto, el documento de requisitos debe permanecer consistente con el diseño y la implementación. Cuando es difícil seguir la traza de los requisitos en el diseño y en el código, el desarrollador tiende a evitar actualizar el documento de requisitos al hacer cambios en el mismo código. Esto conduce a un deterioro importantes de la documentación que conducen a hacer la aplicación muy difícil de mantener.
Habitualmente se necesita que los requisitos de especificación sean trazables hacia delante y hacia atrás. Esto se facilita cuando cada requisito funcional se corresponde con una parte concreta del código, lo cual es habitual en las metodologías orientadas a objetos.



Comprobación y no ambigüedad: Debe ser posible validar un requisito cuando se prueba que se ha implementado de manera apropiada. Los requisitos que se pueden probar se llaman comprobables. Los requisitos no comprobables tienen poco valor práctico.
A menos que un requisito detallado se formule con claridad y sin ambigüedad, no será posible determinar si se ha implementado correctamente.

Completitud: Habitualmente se hacen esfuerzos para conseguir que cada requisito sea autocontenido, pero pocas veces es posible conseguirlo en la práctica, ya que los requisitos hacen referencia a otros requisitos. Que un conjunto de requisitos esté completo asegura que no hay omisiones que comprometan los requisitos establecidos.

Consistencia: Un conjunto de requisitos detallados es consistente si no presentan contradicciones entre ellos. Cuando el número de requisitos crece se hace muy difícil lograr la completa consistencia.
La organización orientada a objetos ayuda a evitar las inconsistencias, ya que al agrupar los requisitos por clases, su descomposición se hace mas sencilla, se formulan en puntos próximos del documento.



La mayor parte de los pasos que se proponen para formular un requisito detallado se han descrito como características deseables para los mismos.

• Debe establecerse un criterio de organización de los requisitos. Una lista anárquica es inmanejable.
• Evaluar si un requisito es trazable significa proponer un diseño e imaginar como tendría el diseño que satisfacer los requisitos.
• Proponer una prueba de cada requisito que se propone no sólo aclara el requisito, sino que también garantiza su verificabilidad.
• Muchos requisitos dependen de datos específicos, y conviene especificar como debe operar el requisito en caso de que el dato esté equivocado o sea inconsistente. Para requisitos críticos, esto debe incluir errores explícitos.

Herramientas gráficas UML para requisitos detallados

- Estáticas: Recurso de organización
  - **Casos de usos**
- Dinámicas: Recurso para mostrar el comportamiento dinámico:

  Describen escenarios (casos particulares)
  - **Diagramas de secuencias**
  - **Diagramas de colaboración.**

  Describen la dinámica completa:
  - **Diagramas de actividad (modelo operativo)**

Santander, 2008 Ingeniería de Programación: Especificación detallada J.M. Drake y P. López 8

UML ofrece muchos recursos para expresar la funcionalidad de los elementos estructurales. Estos recursos pueden utilizarse bien para especificar la funcionalidad, o bien para describir detalladamente esa funcionalidad. En el primer caso se puede considerar como una herramienta de ayuda al análisis o diseño, en el segundo caso, puede llegar incluso a constituir un lenguaje de programación, del que se puede generar automáticamente el código.

Elementos para la descripción de elementos estructurales y su funcionalidad:

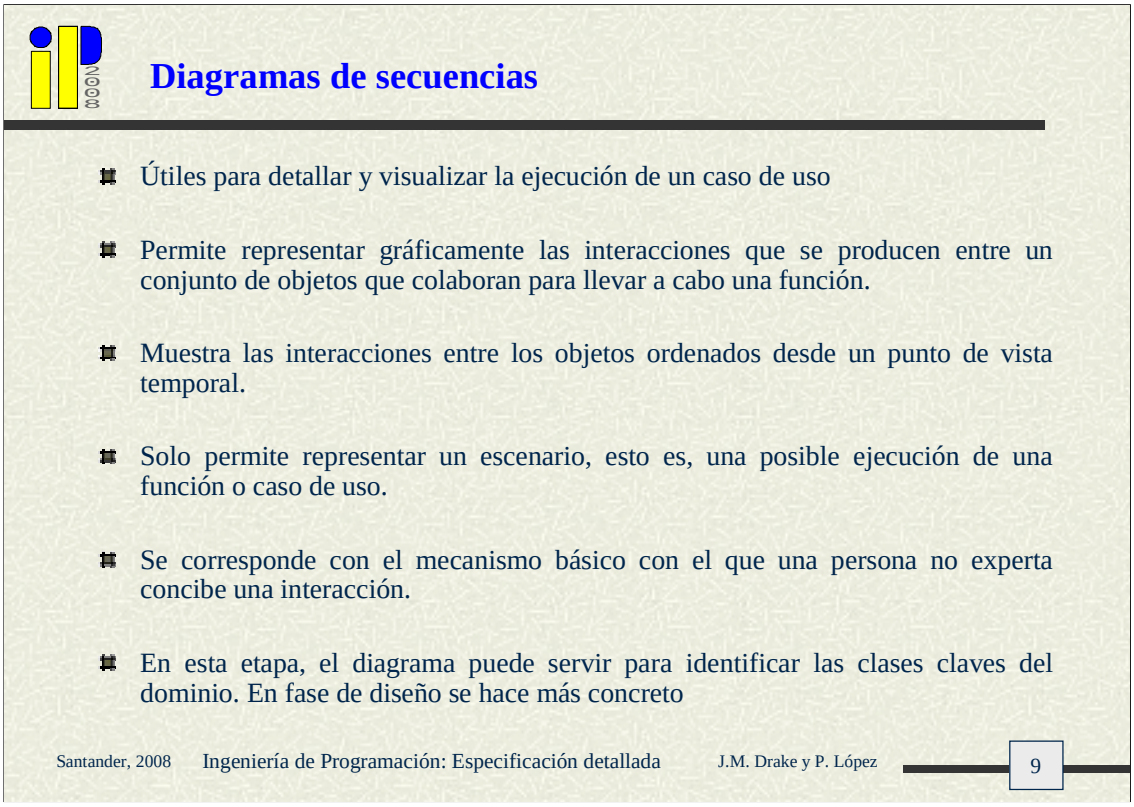
•**Diagrama de estado**: sirve para describir un elemento estructural a través de una máquina de estados finitos.

•**Diagrama de actividad**: es un diagrama de estado con una semántica específica formulado como forma de expresar algoritmos complejos a través de diagramas de flujo.

Elementos para la descripción de mecanismos de operación de interacción entre grupos de objetos del sistema:

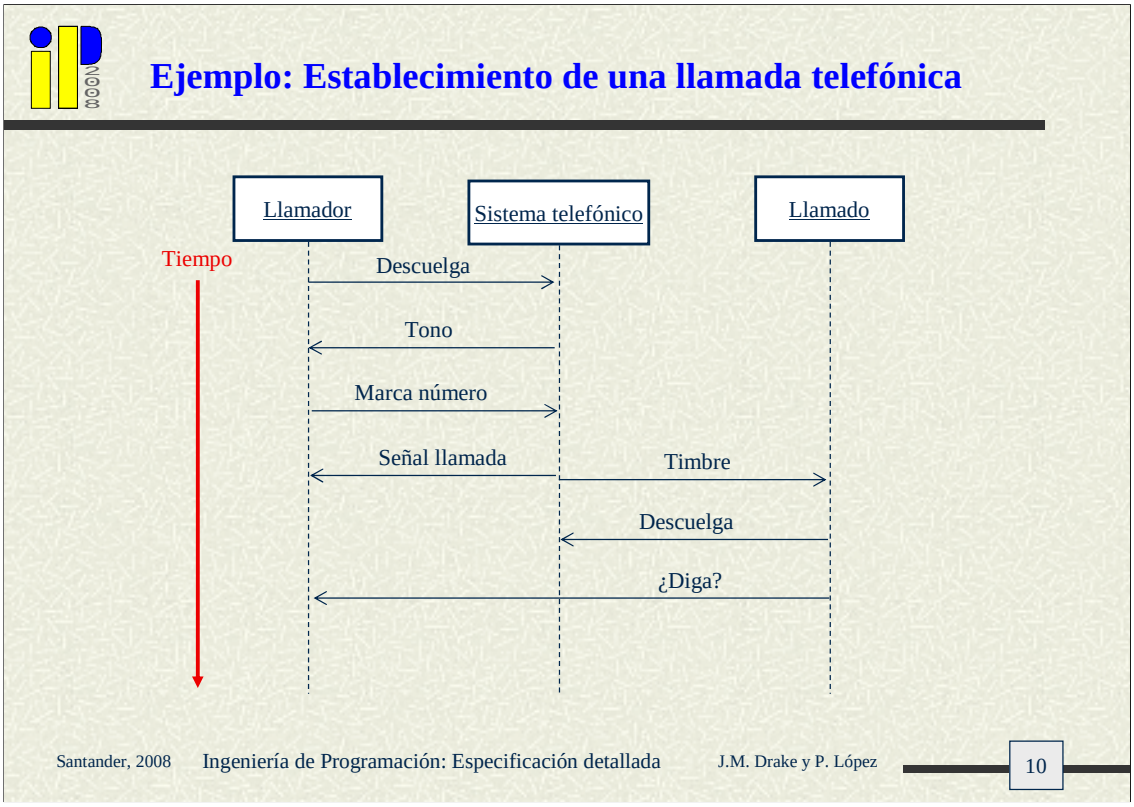
•**Diagrama de secuencias**: Permite representar a través de diagramas de transferencia de mensajes entre objetos organizados en el tiempo cualquier escenario de intercolaboración entre ellos.

•**Diagrama de colaboración**: Permite representar a través de diagramas gráficos los escenarios de colaboración que se establecen entre grupos de objetos.



Los diagramas de secuencias son representaciones gráficas del flujo de control y son útiles en particular para visualizar la ejecución de los casos de uso. Además de utilizarlo como una herramienta en el análisis de requisitos también es muy útil como herramienta de análisis, como se verá en el próximo tema.
Los diagramas de secuencias requiere que se piense en términos de objetos. En un diagrama de secuencia, la vida de cada objeto se muestra como una línea continua vertical con el nombre del objeto y su clase en la parte superior.
Cada interacción entre objetos se muestra como una flecha horizontal desde el objeto que la inicia hasta el objeto que la recibe.
El orden de envío de los mensajes viene dado por la posición sobre el eje vertical.

En esta etapa, el diagrama sirve para identificar las clases claves del dominio de la aplicación. En la fase de diseño se modificará para hacerlo mas preciso.

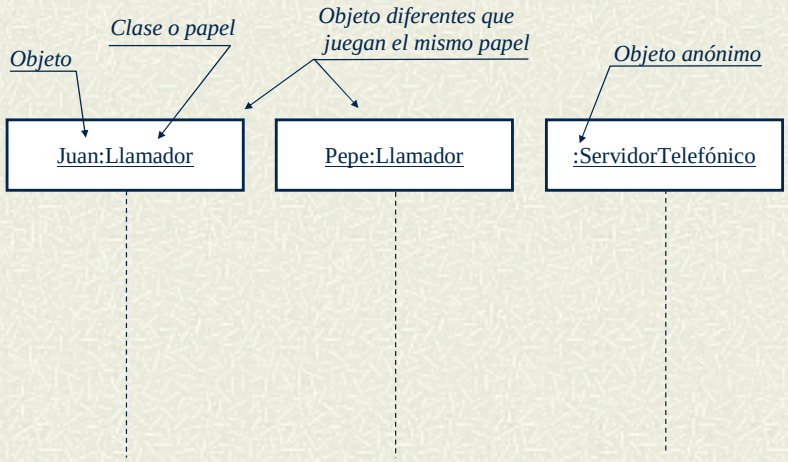


En la figura se muestra un ejemplo de diagrama de secuencias que describe el proceso de comunicación de dos personas (Llamador y llamado) a través de una línea telefónica.

La interacción entre objetos se conciben como mensajes que se intercambian entre objetos.

El diagrama de secuencias no describe la totalidad de las posibilidades que se pueden plantear, sino que únicamente describe una secuencia posible. Por ejemplo, en este caso no se contempla la posibilidad de que el objeto Llamado esté comunicando, y en consecuencia falle la conexión. Esta situación diferente debería formularse con otro diagrama de secuencia.

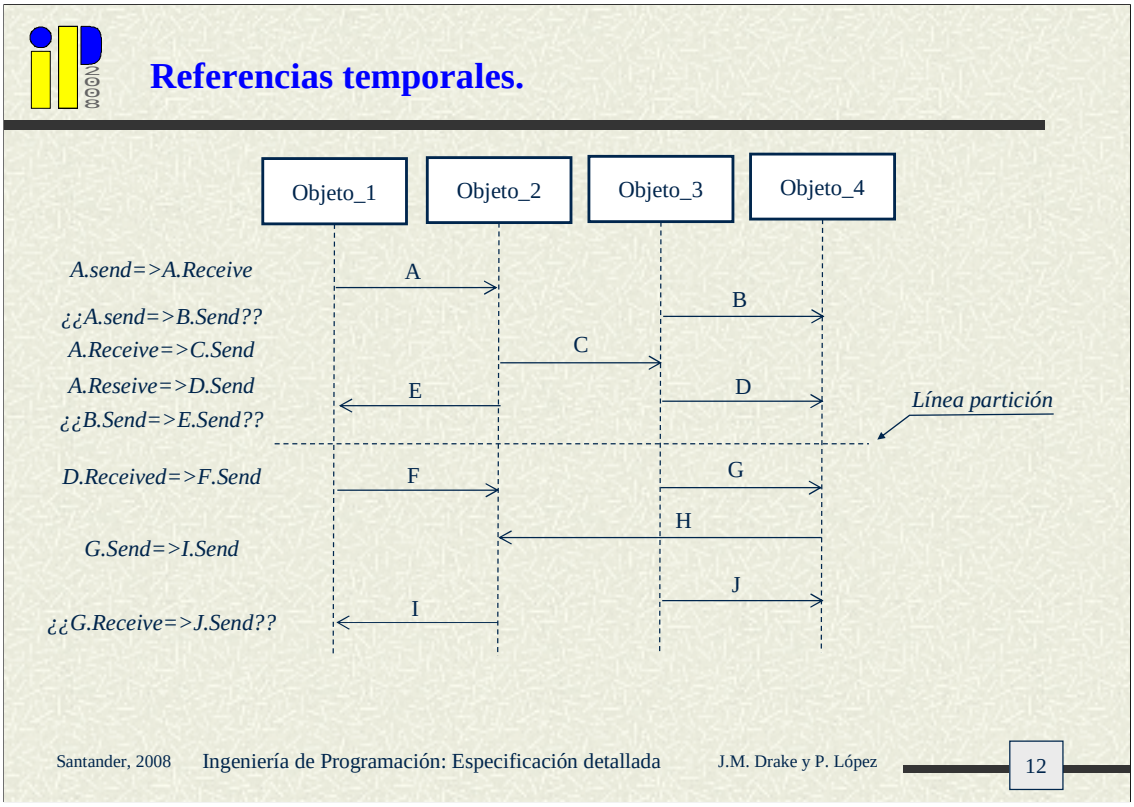
Las interacciones son entre objetos



Santander, 2008 Ingeniería de Programación: Especificación detallada J.M. Drake y P. López 11

Los participantes en un diagrama de secuencias son objetos concretos. Su denominación consta del identificador del objeto y de la clase o papel al que pertenecen.

Puede designarse un objeto como representativo de una clase haciendo únicamente referencia a la clases.



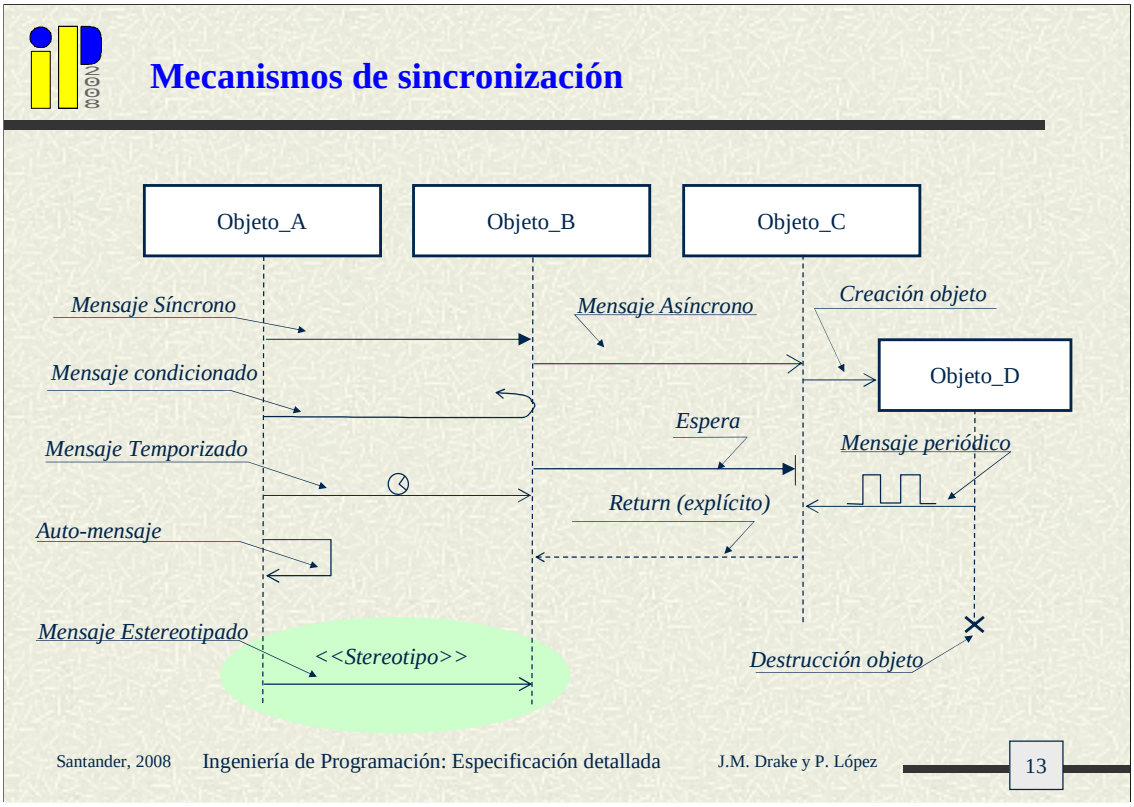
Aunque la dirección vertical representa la evolución del tiempo, solo son comparables los instantes de las acciones si están referenciadas a la actividad de un mismo objeto.

• La acción de enviar un evento es siempre anterior a su recepción. Así, A.Send =>A.Receive (la acción A.Send precede a la acción A.Receive)

• Para dos acciones que no tengan referencia con un objeto común no son ordenables:
A.Send puede preceder o seguir a B.Send.

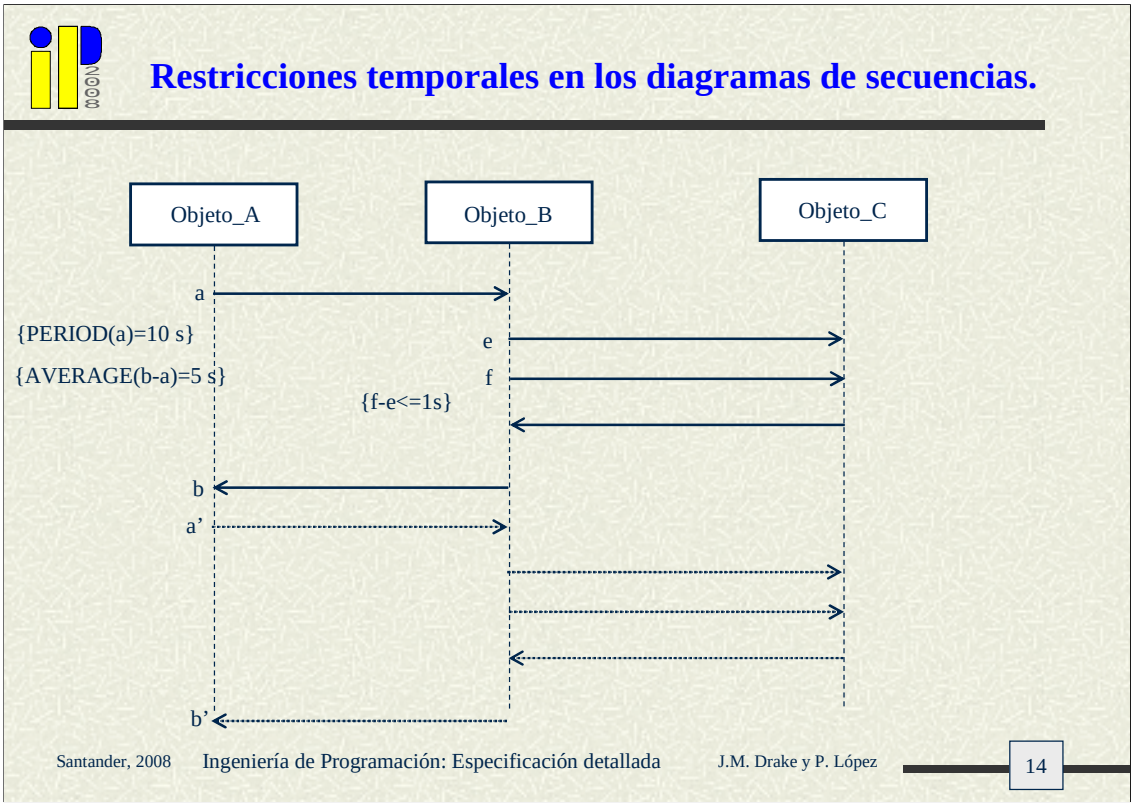
• Cuando existe una secuencia de referencias comunes, puede establecerse la precedencia.
G.Send=>G.Receive=>H.Send=>H.Receive=>I.Send=>I.Receive

• A través de una línea de partición horizontal, se pueden establecer precedencias explícitas:
E.Send=>E.Receive=>G.Send=>G.Receive



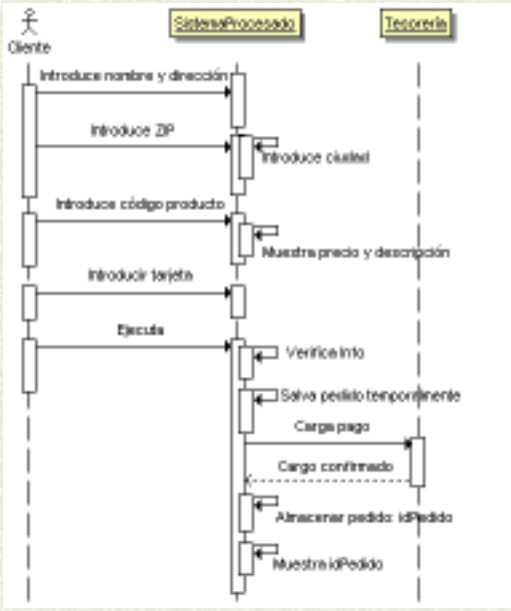
Mensaje Síncrono: El flujo en el objeto que envía el mensaje se suspende hasta que es recibido por el objeto receptor.
Mensaje Asíncrono: El emisor transfiere el mensaje y continua sin esperar a que el mensaje sea recibido.
Mensaje Condicionado(Balking): El mensaje es transferido si el receptor está disponible para aceptarlo de forma inmediata, en caso contrario, no se transfiere.
Mensaje Temporizado: El mensaje es transferido con un tiempo de guarda. Si transcurre el tiempo de timeout y el receptor no lo acepta, el mensaje no se emite.
Espera: Un objeto se suspende hasta que otro objeto alcanza un cierto estado.
Return: Habitualmente el retorno de una invocación es implícita. Algunas veces interesa manifestar su ocurrencia.
Los patrones de generación de los mensajes pueden explicitarse mediante iconos.

Todos estos iconos están definidos en el estándar UML, pero no suelen estar soportados por las herramientas CASE. Por ello, se suele utilizar el método equivalente de la inclusión del estereotipo explícito.

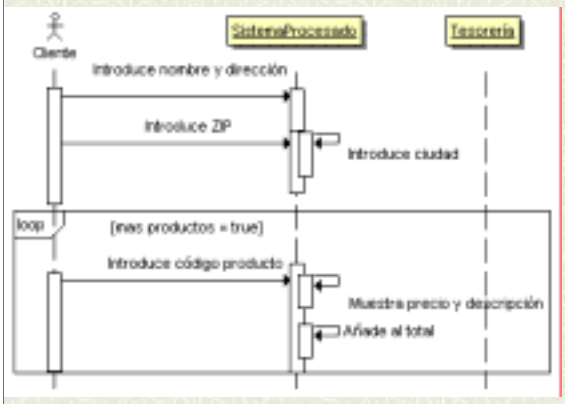


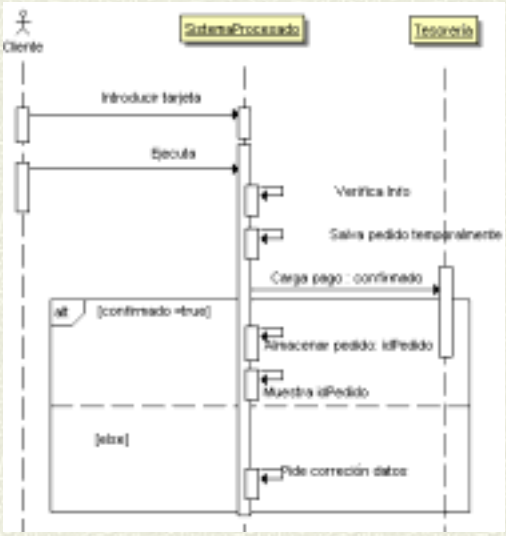
Los instantes en que se producen las acciones se pueden designar mediante etiquetas. Y haciendo uso de ellas se pueden formular múltiples restricciones sobre las respuestas temporales de los sistemas.

Ejemplo: Caso de uso “Realiza Pedido” detallado

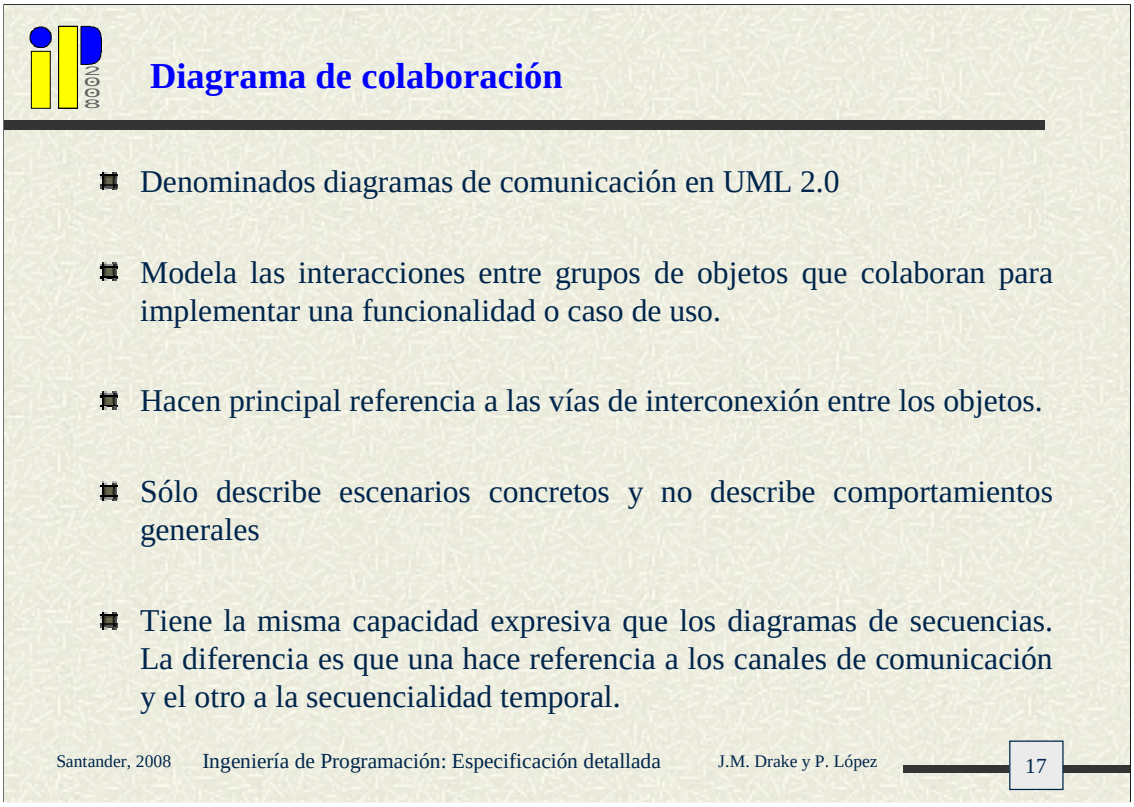


Sintaxis en diagramas de secuencias: fragmentos





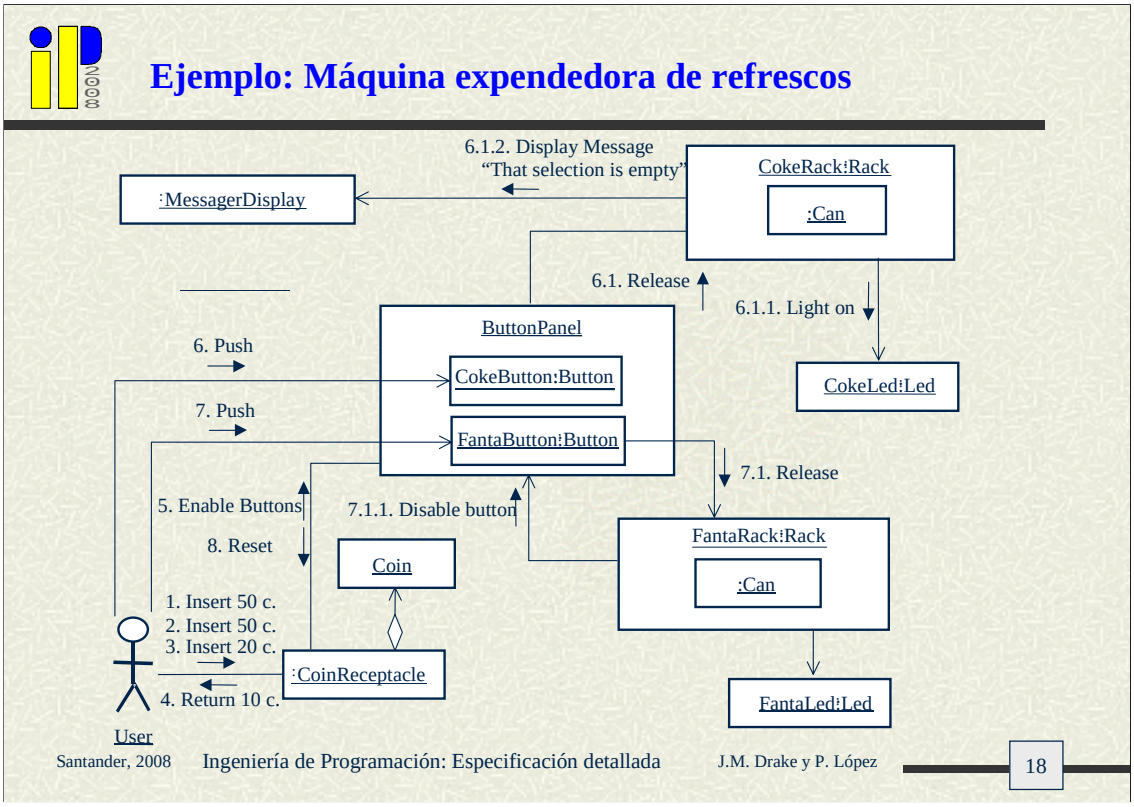
Continuación



Los diagramas de colaboración muestran interacciones entre objetos, resaltando la estructura espacial estática que permite la colaboración del grupo de objetos. Los diagramas de colaboración expresan a la vez el contexto de un grupo de objetos (a través de la representación de los objetos y de los enlaces que están establecidos entre ellos) y las interacciones entre los objetos (por la representación de envío de mensajes).

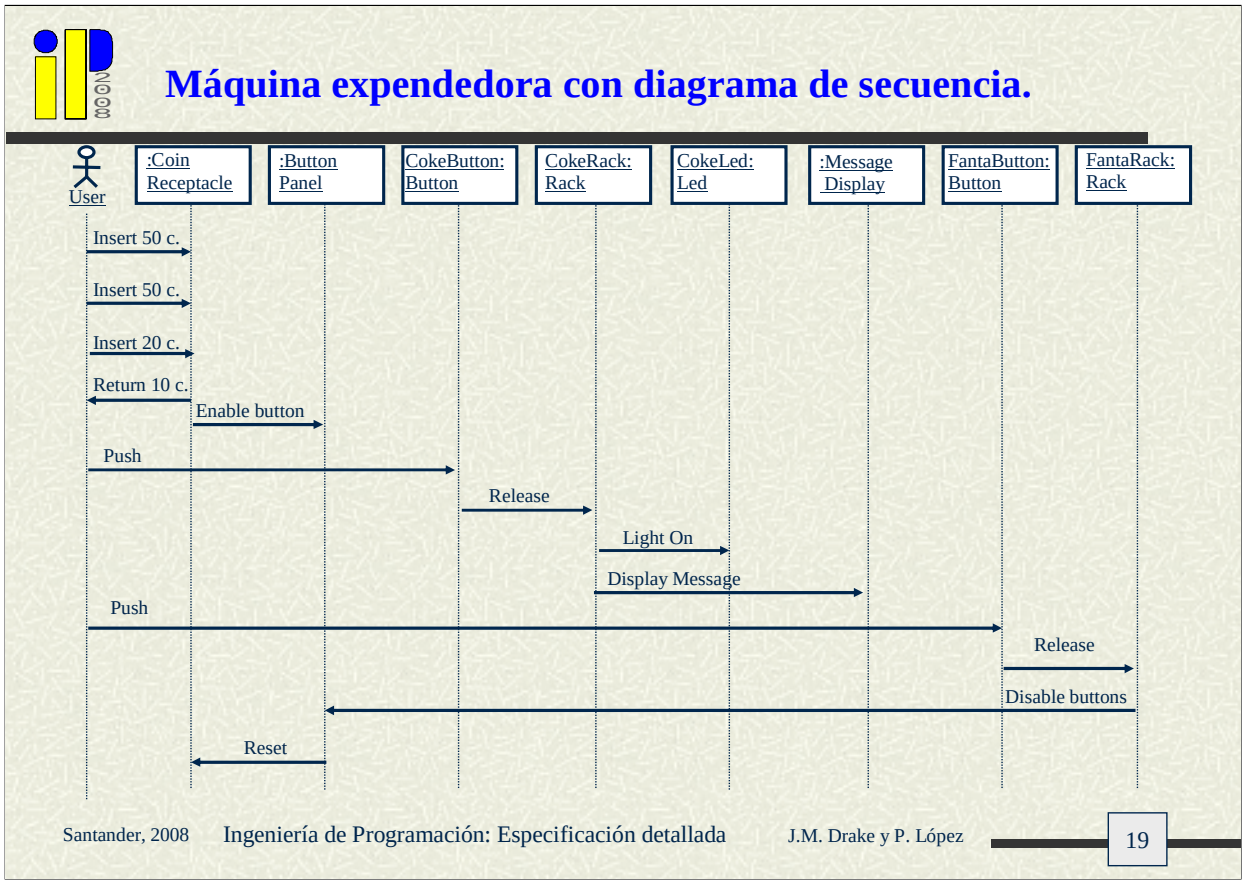
El contexto de una interacción puede comprender los argumentos, las variables locales creadas durante la ejecución, así como los enlaces entre los objetos que participan en la interacción.

Una interacción se realiza mediante un grupo de objetos que colaboran intercambiando mensajes. Los mensajes se representan a los largo de los enlaces que enlazan los objetos por medio de flechas orientadas hacia el destinatario del mensaje.

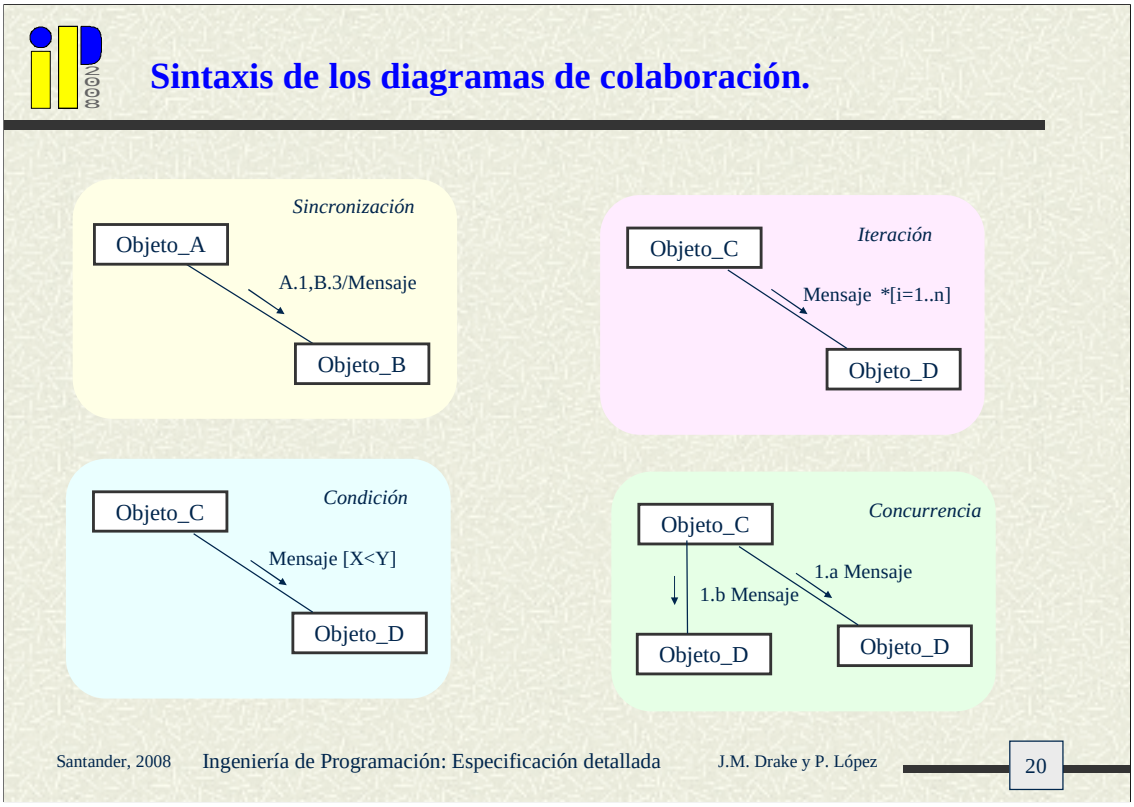


Escenario ejemplo: Drink Machine: El usuario introduce 60 céntimos como dos monedas de 50 céntimos y una moneda de 20 céntimos, y la máquina retorna una moneda de 10 céntimos. Selecciona Coke, pero la máquina no tiene este tipo de bebida y lo manifiesta a través de un LED y un mensaje. Luego elige Fanta que si está disponible, por lo que la máquina la suministra.

En los diagramas de colaboración el tiempo (orden en el tiempo) se puede expresar introduciendo un índice en los mensajes.
En sistemas concurrentes en los que existen diferentes sesiones de interacción se utiliza un segundo índice de sesión.



Escenario ejemplo: Drink Machine: El usuario introduce 60 céntimos como dos monedas de 50 céntimos y una moneda de 20 céntimos, y la máquina retorna una moneda de 10 céntimos. Selecciona Coke, pero la máquina no tiene este tipo de bebida y lo manifiesta a través de un LED y un mensaje. Luego elige Fanta que si está disponible, por lo que la máquina la suministra.

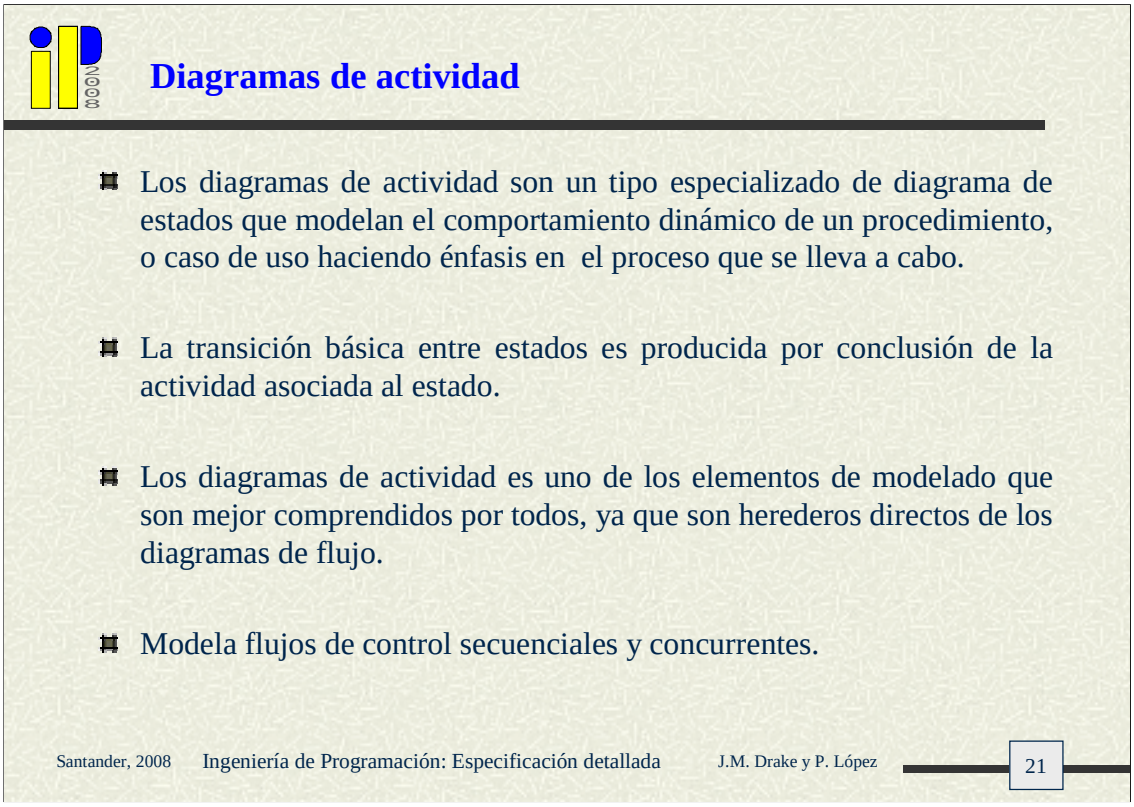


En los diagramas de colaboración se pueden formular diferentes técnicas expresivas:

Sincronización: Un mensaje se envía si previamente se han producido otras interacciones.

Condición: Un mensaje se envía si se verifica una cierta condición de entorno

Iteración: Un mensaje se envía por un número determinado de veces (FOR) o mientras se verifique una determinada condición (WHILE).



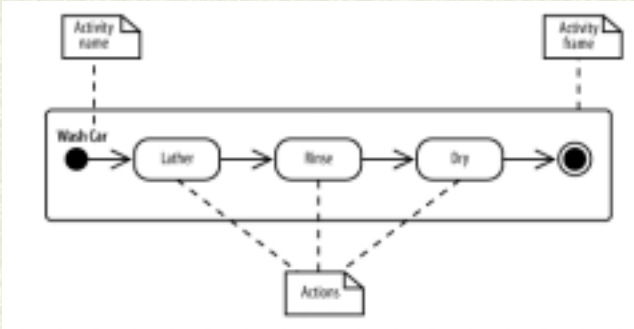
Un diagrama de actividad es una variante de un diagrama de estados, organizado principalmente respecto de actividades que fluye como consecuencia de la finalización de las mismas. Es una herramienta muy útil para la descripción de un método o función y de un caso de uso.

Un diagrama de actividad representa el estado de ejecución de un programa bajo la forma de un desarrollo de etapas agrupadas secuencialmente en ramas paralelas de flujo de control.

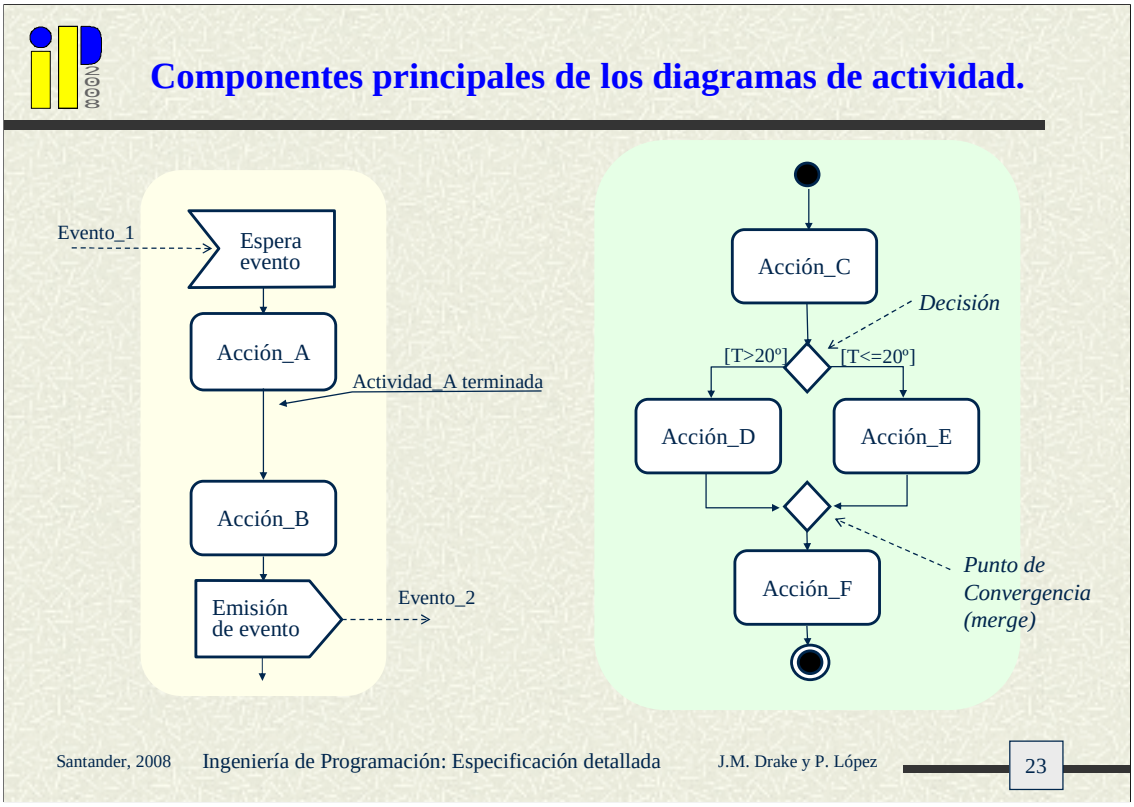
Cada actividad representa una etapa particular en la ejecución del método que describe. Las actividades se enlazan por transiciones que se ejecutan automáticamente como consecuencia de la finalización de la actividad, y que se representan por flechas. Cuando una actividad termina se desencadena la transición y empieza la actividad siguiente. Las actividades no tienen transiciones internas, ni transiciones desencadenadas por eventos.

Actividades y acciones

- Una acción es un paso de un proceso que tiene la sémática “run to completion” (Se inicia para ser terminado)
  - Ejemplo de acciones: Crear o eliminar un objeto, invocar un procedimiento, realizar un cálculo simple, etc
- Una actividad es un conjunto de acciones que modelan un proceso. No tiene la semántica “run to completion”. Una actividad se modela mediante un diagrama de actividad.
- Enjabonar, enjuagar o secar un coche son acciones de la actividad “Lavar un coche”



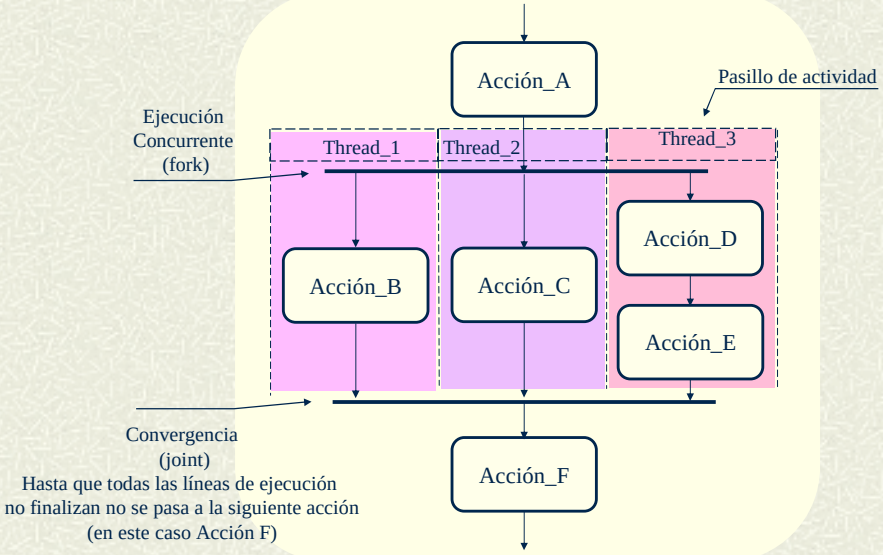
Una acción es una especificación de una primitiva ejecutable que tiene como consecuencia un cambio de estado en el modelo, y puede consistir en la transferencia de un mensaje, modificar un enlace o cambiar de valor un atributo.
Las acciones son primitivas computacionales simples que tienen la semántica de iniciarse para ser terminada (“run-to-completion”). Esto no significa que no pueda ser expulsada de su ejecución por otra de mayor prioridad, sino que aun en este caso posterior se retorna a ella para completarla. Un objeto que este ejecutando una acción, no acepta nuevos mensajes hasta que la concluya.



Acción: Tarea que es realizada dentro de procedimiento o caso de uso que se describe.
Espera de evento: El flujo de control se suspende hasta que se genera un evento o mensaje.
Emisión de un evento: Se genera un evento o mensaje que puede ser un resultado o una forma de interacción con otras secciones de la tarea.
Bifurcación condicionada (Branching): El flujo de control pasa a una u otra rama del diagrama, en función de que satisfaga o no una cierta condición. Las condiciones deben ser no ambiguas y completas (no repetir ningún caso ni dejar ningún caso sin alternativa)
Convergencia (Merge): Varia líneas alternativas se encauzan hacia una secuencia de actividades.

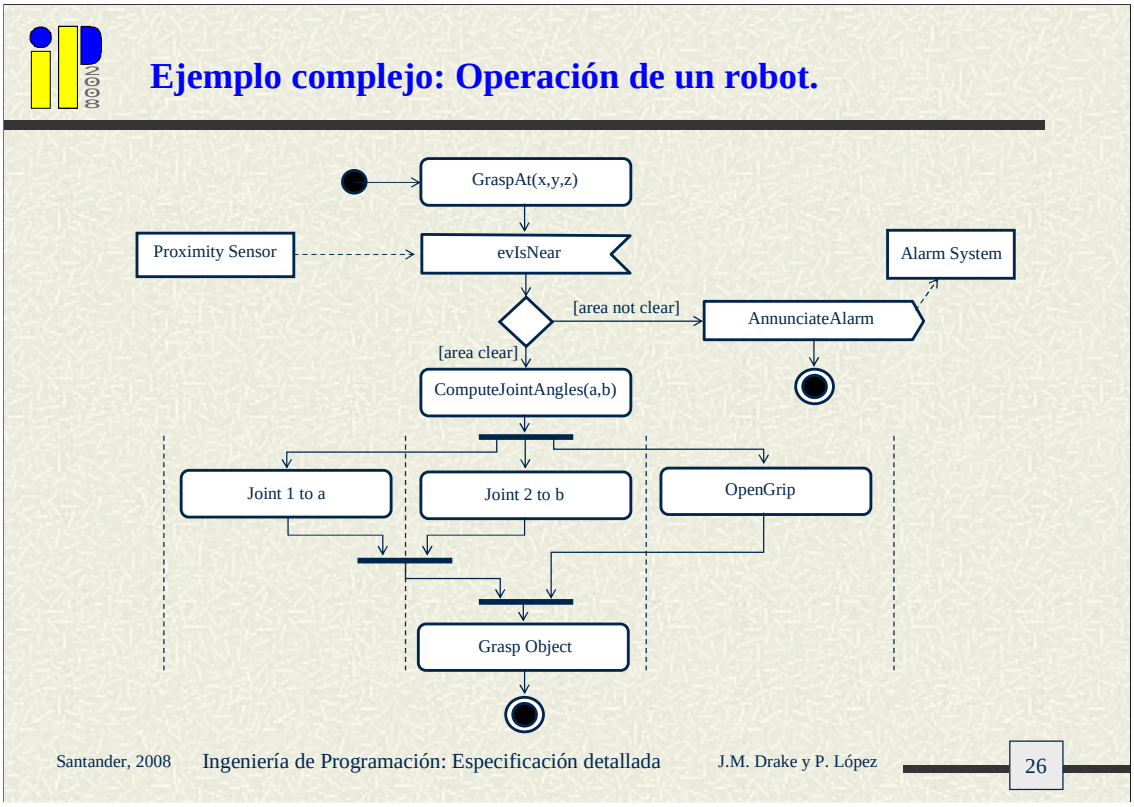
Ejemplo: Caso de uso “Busca Pedido” detallado

Representación de la concurrencia.



Santander, 2008 Ingeniería de Programación: Especificación detallada J.M. Drake y P. López 25

Los diagramas de actividad permiten formular ejecuciones concurrentes de actividades.
Barras de generación de concurrencia (Fork): El flujo de control se replica en varias líneas que se ejecutan concurrentemente.
Barras de sincronización (Joint): Las diferentes líneas de control de entrada se suspenden en ella hasta que todas la ha alcanzado. Luego se unifican en una única linea de control.
Pasillos de actividad (Swimlane): Representan a los objetos o procesos que soportan las diferentes líneas de flujo de control concurrentes.

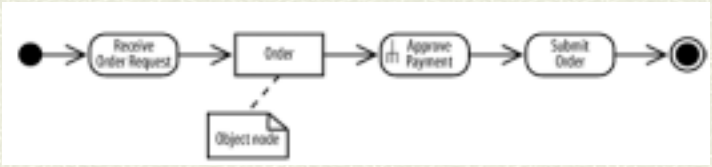


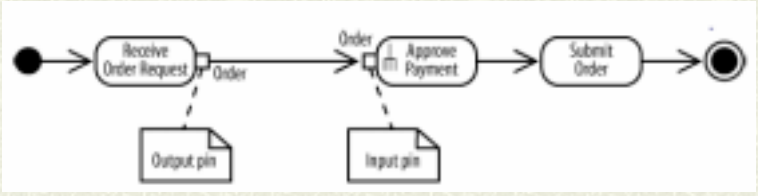
Ejemplo Operación de un robot: Modela el proceso de cogida de un objeto mediante un brazo robotizado con una pinza.

•Recibe las coordenadas del objeto que debe ser agarrado.
•Verifica que el área de desplazamiento está libre. En caso contrario genera una alarma y no la ejecuta.
•Computa la modificación de las coordenadas de las articulaciones internas en función del destino.
•Concurrentemente mueve las articulaciones internas, y abre la pinza.
•Cuando todas las anteriores han finalizado, cierra la pinza y coge el objeto.

Otros aspectos de los diagramas de actividad: objetos

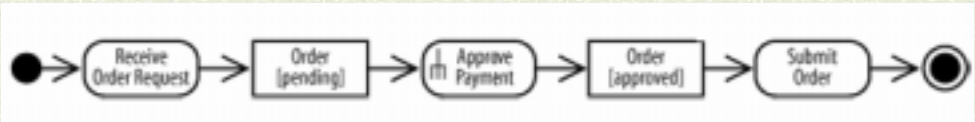
Representación de objetos





Representación de objetos como entrada o salida de acciones

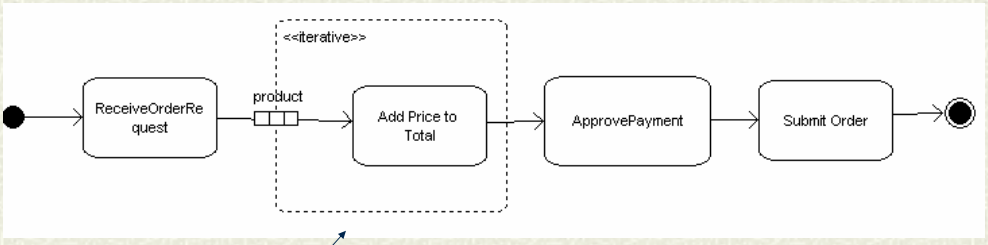
Representación de objetos cambiando de estado



Santander, 2008 Ingeniería de Programación: Especificación detallada J.M. Drake y P. López 27

En un diagrama de actividad se pueden representar los objetos de datos que se generan, se consumen o se intercambian en un proceso y que son relevantes para su descripción. Para representarlos se utilizan los denominados “Object Node”
Cuando un objeto de datos se representan como una caja, significa que esos datos existen en el punto de flujo de control en que se insertan, se intercambian entre las acciones entre las que es insertado.
Los objetos pueden mostrar los estados en que se encuentran y pueden representar los puntos de inicio y finalización de la actividad que representan.

Diagramas de actividad complejos



Regiones de expansión:
Realiza las acciones interiores para todos los elementos del pin de entrada