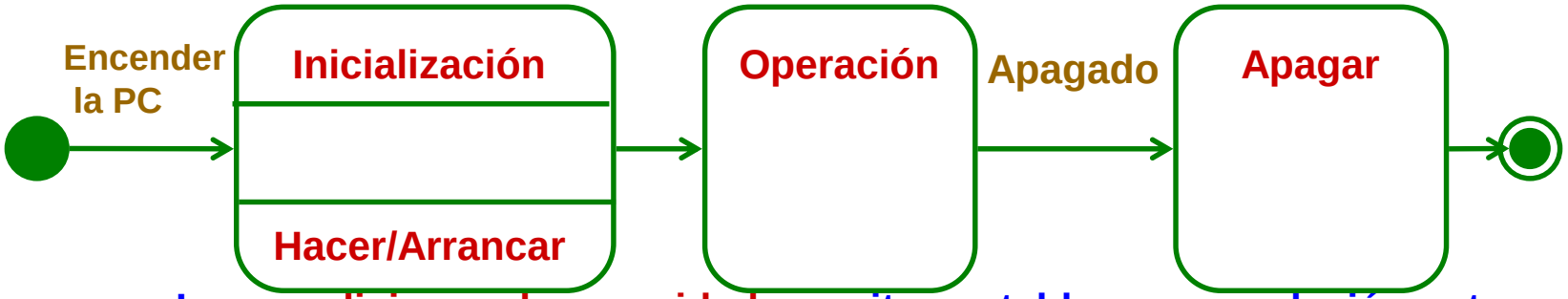
- Diagramas de estados
  - Presenta los estados en los que puede encontrarse un objeto junto con las transiciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado.
  - Los símbolos UML en un diagrama de estados son:

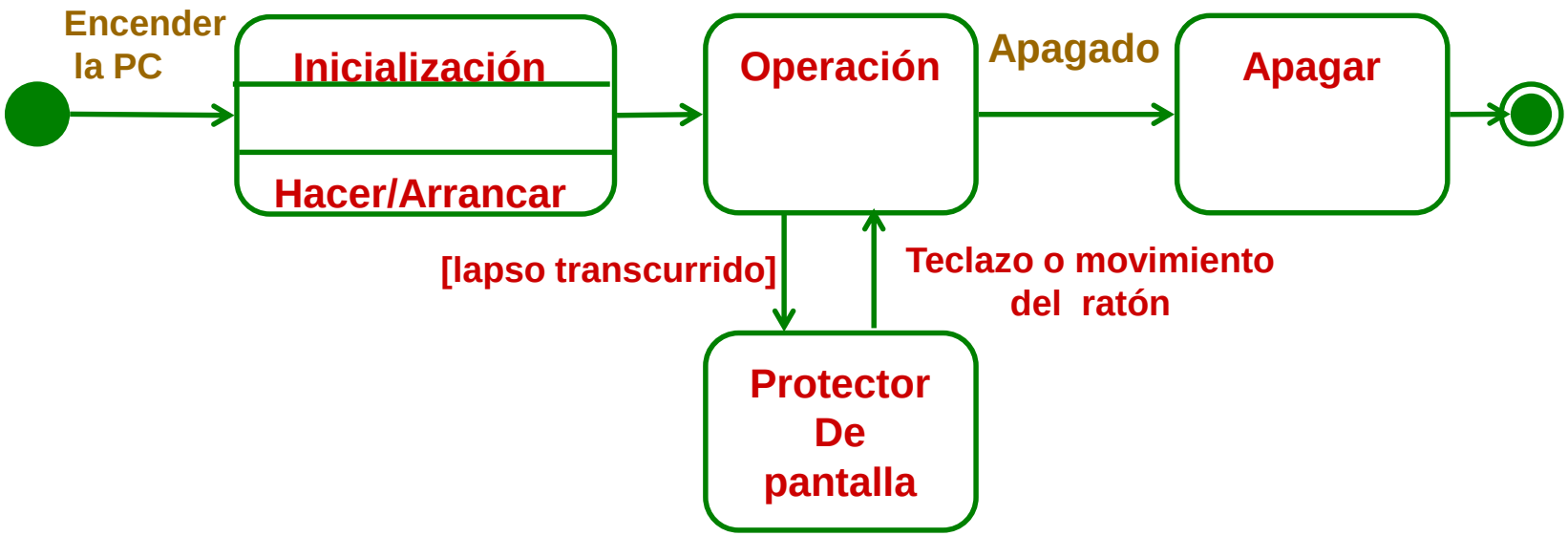
  - Las actividades cuentan con sucesos y acciones de entrada (qué sucede cuando el sistema entra al estado), salida (Qué pasa cuando el sistema sale del estado) y de hacer (que sucede cuando el sistema está en el estado)

Diagramas de estados (cont.)

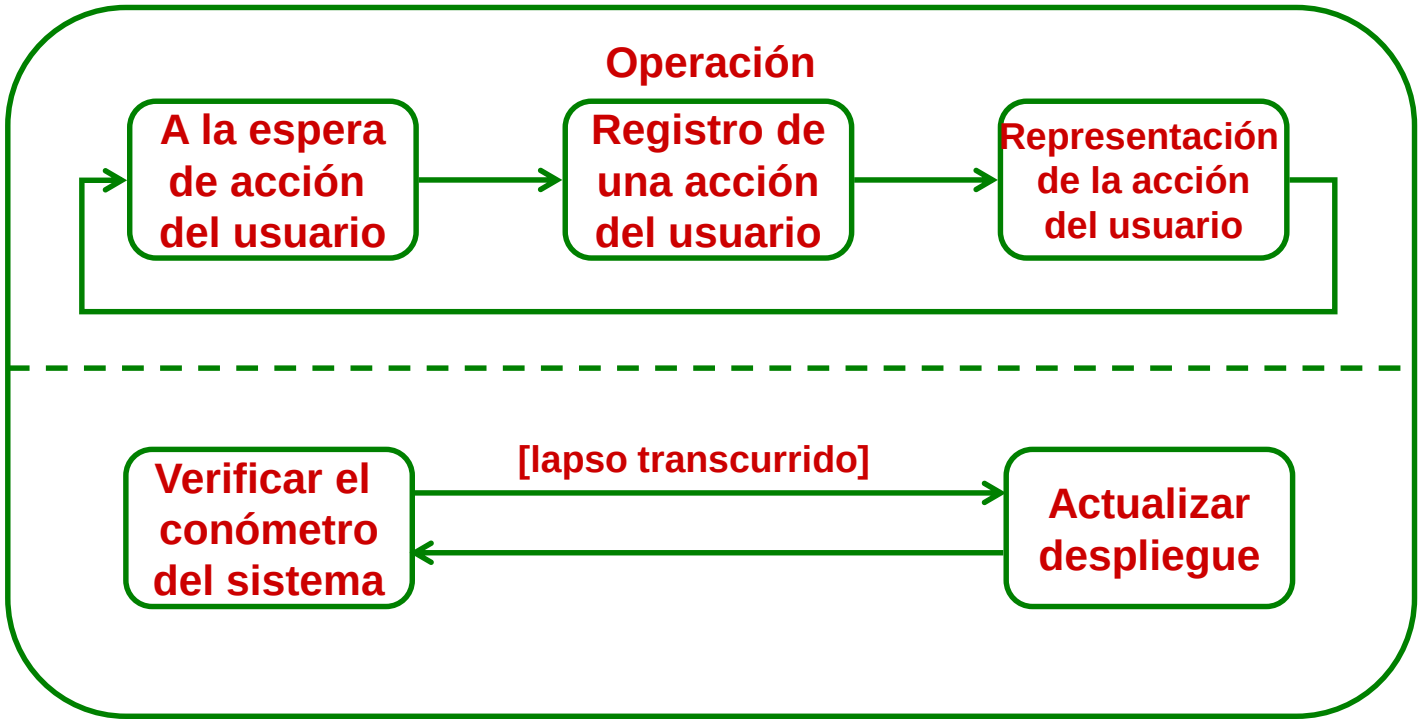
- Se puede agregar ciertos detalles a las líneas de transición, para indicar un suceso que provoca una transición (la que desencadena un suceso) y la actividad de cómputo que se ejecute y haga que suceda la modificación de estado.



- Las condiciones de seguridad permiten establecer una relación entre estados que dependen de que se cumpla dicha condición.



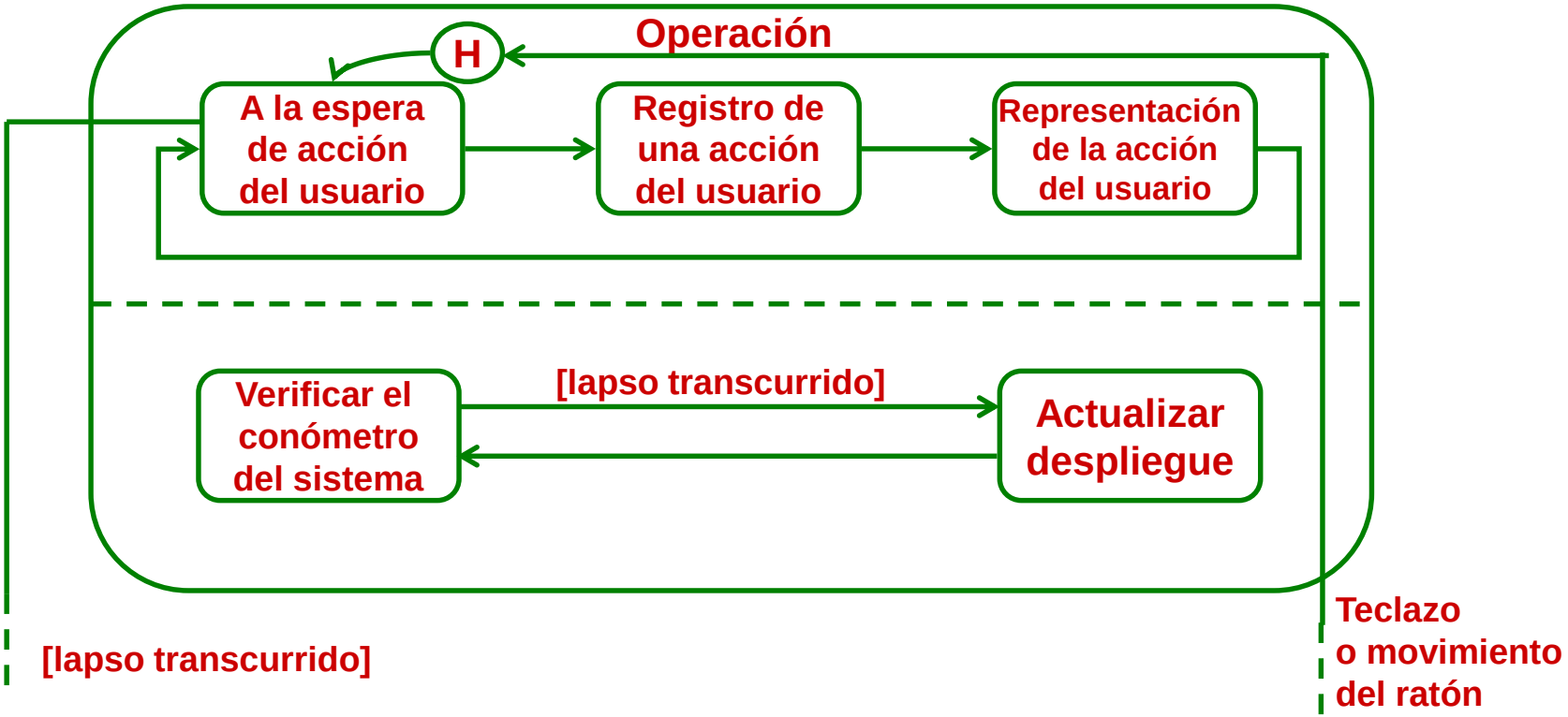
- **Diagramas de estados (cont.)**
  - **Subestados. Cuando un estado se encuentra dentro de otro estado, se conocen como subestados.**
    - **Se dice que pueden suceder en forma secuencial cuando suceden uno tras de otro y se representan dentro del cuadro de estado original, ligados secuencialmente.**
    - **También hay subestados concurrentes cuando pueden ocurrir al mismo tiempo. Se representan dentro del estado original, separados por línea punteada.**



Diagramas de estados (cont.)

Estados históricos.

- Se dice de los estados compuestos que pueden recordar su subestado activo cuando el objeto trasciende fuera de tal estado compuesto.
- Se representa con una H y en caso de subestados anidados se dice que es histórico profundo cuando alcanza a recordar varios niveles (H*) en caso contrario se dice que es superficial



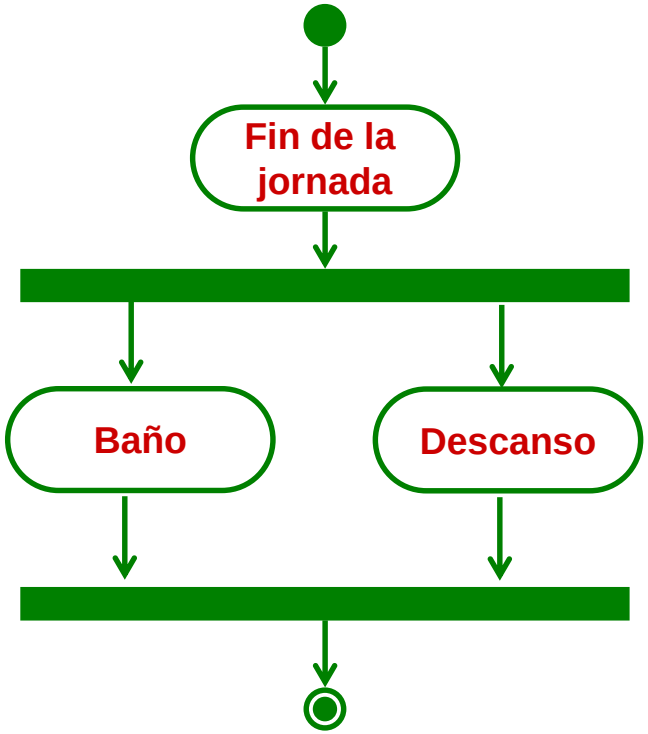
- **Diagramas de estados (cont.)**

  - Cuando utilizar los diagramas de estados:
    - Se tendría que hacer uno por cada clase del sistema, pero se sugiere hacerlos solo para aquellos que presenten un estado interesante y cuando la construcción de tales diagramas ayude a aclarar lo que sucede.
    - Algunos sugieren usarlos en los objetos de interfaz de usuario y de control, debido a que tienen el tipo de comportamiento que es útil describir mediante diagramas de estado.
    - En caso de que se desee representar las secuencias general de acciones de vario objetos y casos de uso, es mejor utilizar el diagrama de actividades.

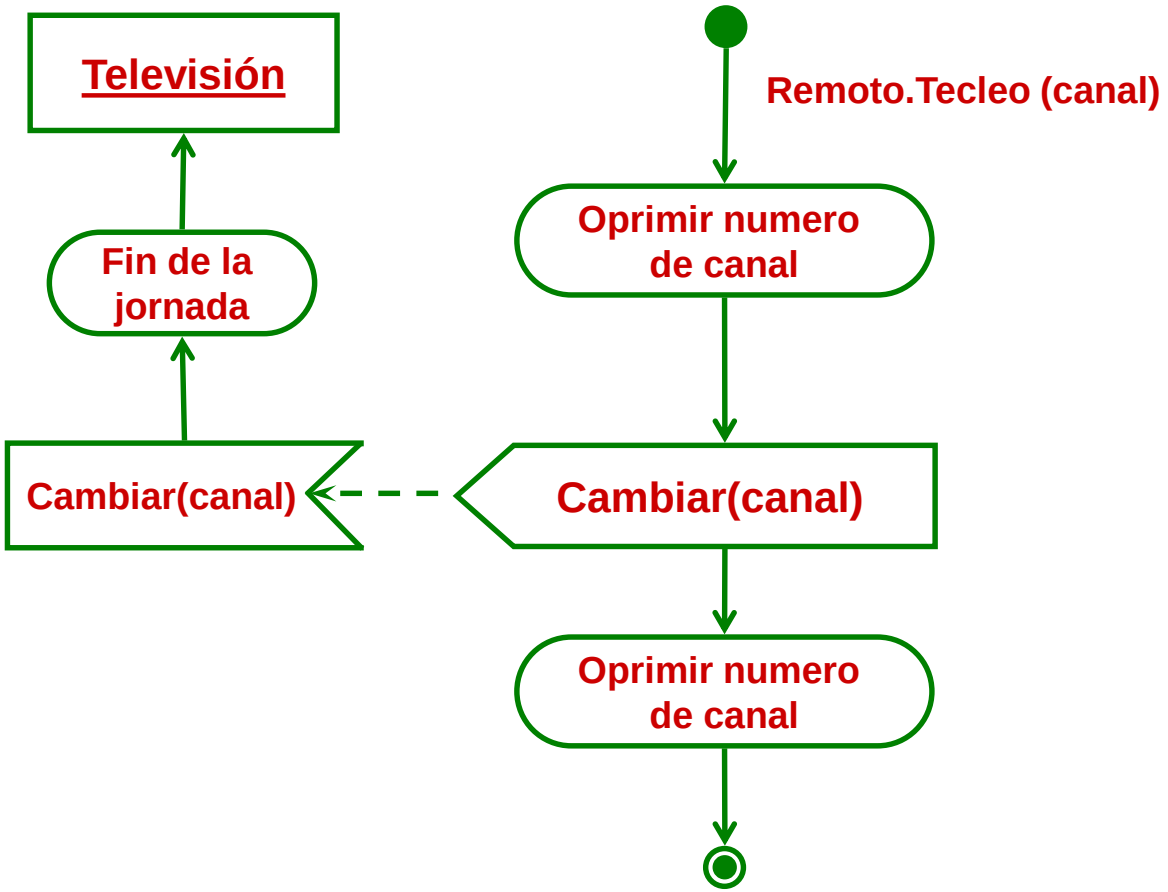
- **Diagramas de actividades**
  - Describen como se coordinan las actividades, muestran como puede ser implementada una operación que debe realizar muchas tareas diferentes y se desea mostrar cuales son las dependencias esenciales entre ellas.
  - Elementos de un diagrama de actividades:
    - La actividad se muestra como una caja con nombre con las esquinas muy redondeadas, representa cuando la actividad ha terminado

    - La transición se muestra como una flecha, normalmente no se les pone etiqueta, a menos que se tenga una condición.

- **Diagramas de actividades (cont.)**
  - **Barras de sincronización, indica coordinación de actividades y no se puede pasar de la barra hasta que todas las actividades previas a la barra han sido terminadas. Se utilizan para la ejecución de actividades en paralelo.**



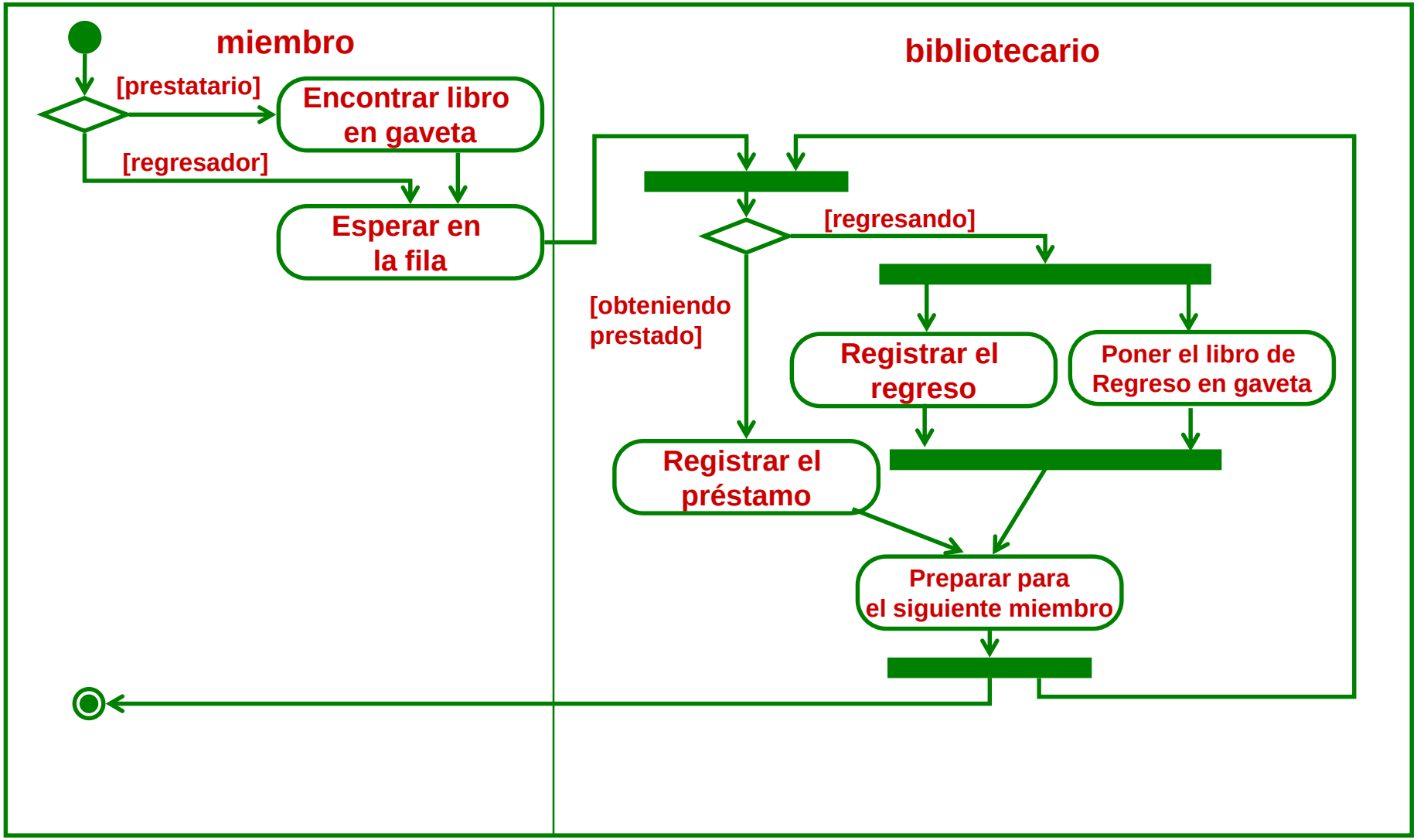
- **Diagramas de actividades (cont.)**
  - Las indicaciones, permiten que se ejecuten otras actividades, usando un pentágono convexo para el envío del un evento y uno cóncavo para la recepción del evento.



- **Diagramas de actividades (cont.)**
  - **Diamantes de decisión se usan para mostrar decisiones como una alternativa a las condiciones, para separar transiciones dejando el mismo estado.**
  - **Marcas de inicio y fin se usan para indicar donde empieza el diagrama y donde termina.**
  - **Particiones y líneas de responsabilidad, Al poner muchas actividades relacionadas entre sí, se pueden colocar de acuerdo al objeto o al actor que las ejecuta, o a cuál caso de uso pertenecen**
- **Las principales diferencias entre los diagramas de estado y los diagramas de actividades son:**
  - **Los diagramas de actividades normalmente NO incluyen eventos, porque los únicos eventos de interés es la terminación de las actividades.**
  - **Las actividades se pretende que se continúen a lo largo del diagrama sin quedarse estancadas.**

Diagramas de actividades (cont.)

- Ejemplo de un diagrama de actividades en una biblioteca.



- **Diagramas de actividades (cont.)**
  - Cuando utilizar diagramas de actividades:
    - Debido a que manejan y promueven el comportamiento en paralelo, son una herramienta muy útil para el modelado de flujo de trabajo y para la programación multihilos.
    - Se recomienda usarlos para:
      - El *análisis de un caso de uso*. Para comprender qué acciones deben ocurrir y cuáles son las dependencias de comportamiento. Asignando posteriormente los métodos a los objetos y mostrando tales asignaciones mediante diagramas de secuencia o colaboración.
      - La *comprensión del flujo de trabajo, a través de numerosos casos de uso*. Para representar y entender el comportamiento de las interacciones entre los casos de uso. Ayuda a aclarar situaciones dominadas por flujo de trabajo.
      - Cuando se trata de aplicaciones multihilos. Son adecuados en éste uso
    - No sirven para:
      - *Tratar de ver como colaboran los objetos*. Usar mejor diagramas de secuencia o colaboración.
      - *Para tratar de ver como se comporta un objeto durante su período de vida*. Es mejor usar un diagrama de estados.

- **Diagrama de secuencias.**
  - Muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo. Constan de objetos y representando en una línea vertical el tiempo, se indican las operaciones que ejecuta el objeto o activación se representan mediante un rectángulo cuya altura va en relación a la duración de la operación.
  - Los mensajes van de un objeto a otro se representan con líneas. Pueden ser simples (transfieren control), sincrónicos (esperan respuesta) o asincrónicos (no espera respuesta)

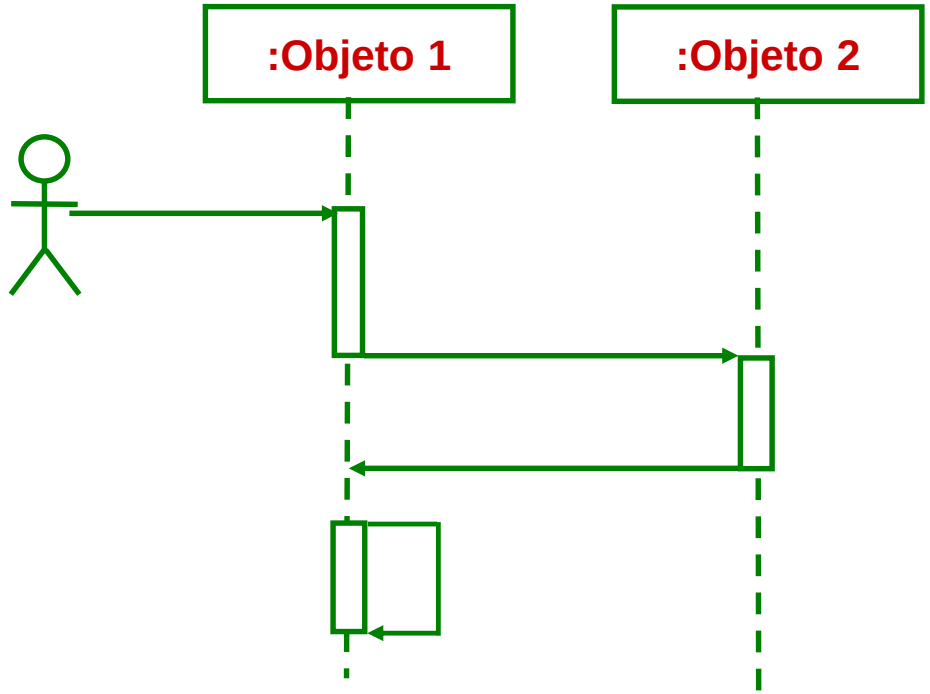


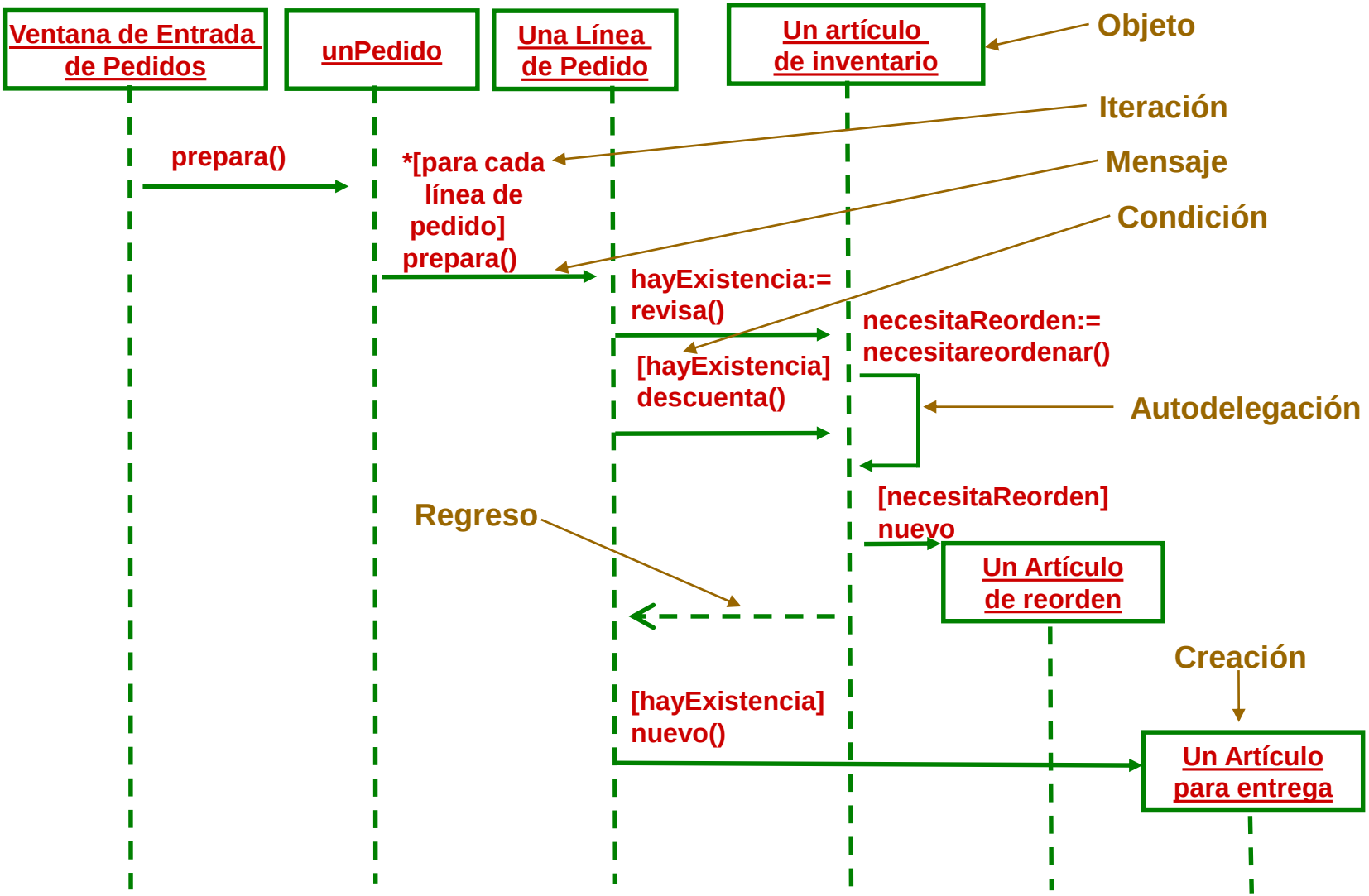
Diagrama de secuencias. (cont.)

- Para ilustrar los diagramas de secuencia se usará el siguiente caso de uso:
  - La ventana Entrada de pedido envía un mensaje “prepara” a Pedido.
  - El Pedido envía entonces un mensaje “prepara” a cada línea de pedido dentro del Pedido.
  - Cada Línea de pedido revisa el Artículo de inventario correspondiente.
    - - Si esta revisión devuelve “verdadero”, la Línea de pedido descuenta la cantidad apropiada de Artículo de inventario del almacén.
    - Si no sucede así, quiere decir que la cantidad del Artículo de inventario ha caído más abajo del nivel de reorden y entonces dicho Artículo de inventario solicita una nueva entrega.

- En el diagrama siguiente se muestra el diagrama de secuencia omitiendo las activaciones, que según Fowler, no aportan mucho a la ejecución de procedimientos y solamente sugiere usarlas en situaciones concurrentes.

Diagrama de secuencias. (cont.)

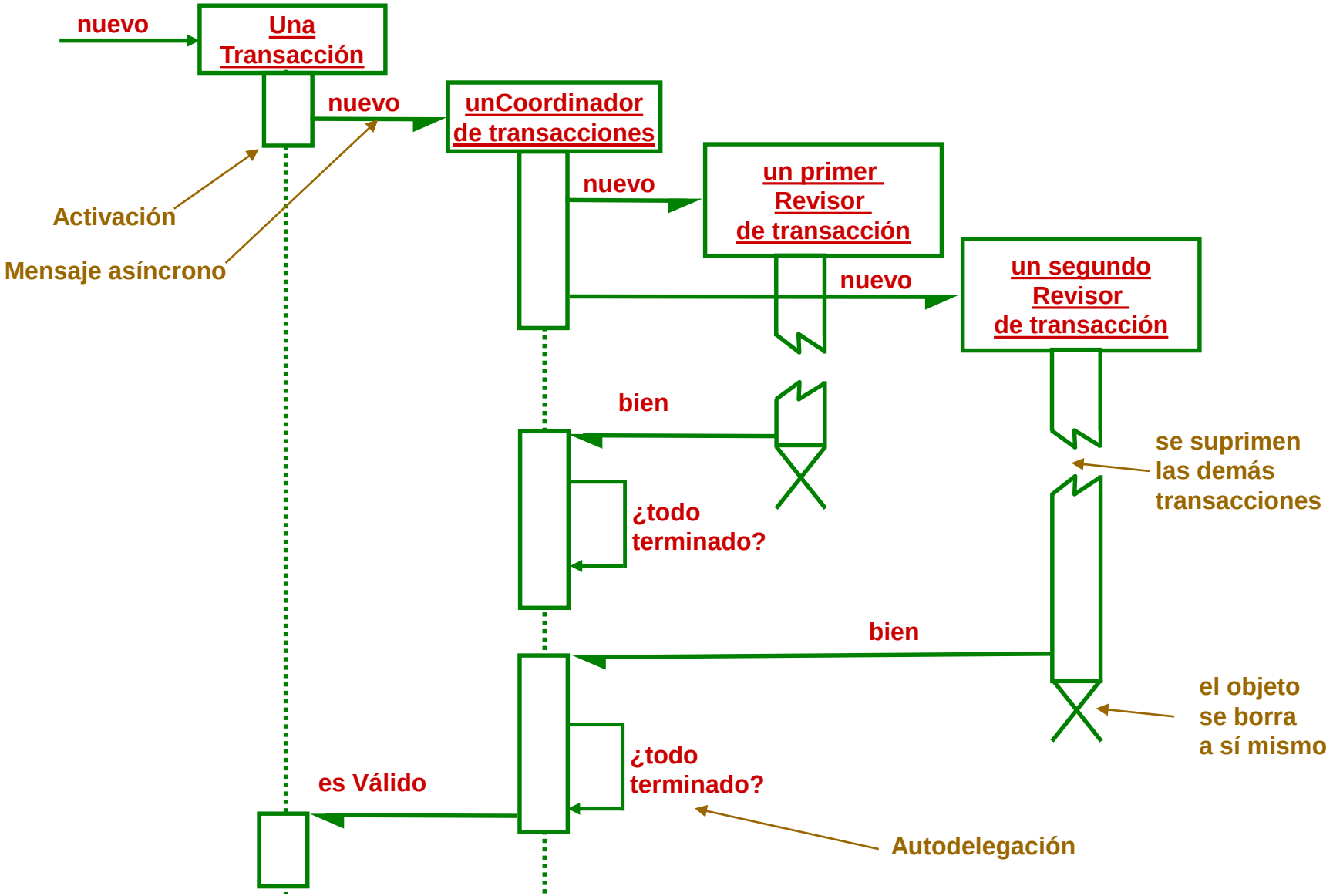
- El siguiente ejemplo muestra



- **Diagrama de secuencias. (cont.)**
  - De el objeto se desprende una línea vertical conocida como **línea de vida** del objeto y representa el tiempo de duración del objeto durante la interacción.
  - Los **mensajes** representados por líneas están en orden de arriba hacia abajo. La **autodelegación** es un mensaje que un objeto se manda a sí mismo.
  - Una **condición** indica cuándo se envía un mensaje, el cual se enviará solo si la condición es verdadera.
  - Una **iteración** muestra cuando un mensaje se envía varias veces a varios objetos receptores, como sucedería cuando se itera sobre una colección.
  - El **regreso** indica el regreso de un mensaje, no un nuevo mensaje. Los regresos SATURAN los diagramas y tienden a oscurecer el flujo, Fowler recomienda usarlo solamente cuando ayuden a aumentar la claridad.
  - El **borrado** de un objeto se muestra con una X grande. Los objetos pueden autodestruirse o pueden ser destruidos mediante otro mensaje.

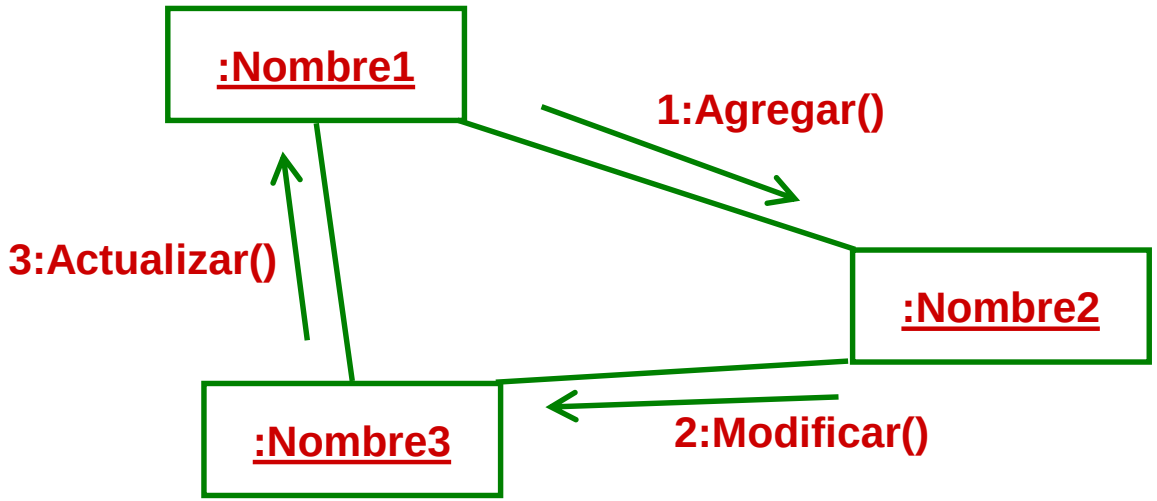
- **Diagrama de secuencias. (cont.)**
  - Otra ilustración adicional nos permitirá visualizar las activaciones y la creación de objetos.
  - Ejemplo de una transacción bancaria:
    - Cuando se crea una Transacción, ésta crea un Coordinador de transacción que coordina el trámite de la transacción. Este coordinador crea una cantidad (en el ejemplo, dos) de objetos Revisor de Transacción, cada uno de los cuales es responsable de una revisión particular. Este proceso permitirá añadir con facilidad otros proceso de revisión, porque cada registro es llamado asincrónicamente y opera en paralelo.
    - Cuando termina un revisor de transacción, se le notifica al coordinador de transacción. El coordinador comprueba si todos los revisores respondieron; de no ser así, no hace nada. Si todos han respondido indicando terminaciones exitosas, como en este ejemplo, entonces el coordinador notifica a la Transacción que todo está bien.

Diagrama de secuencias. (cont.)



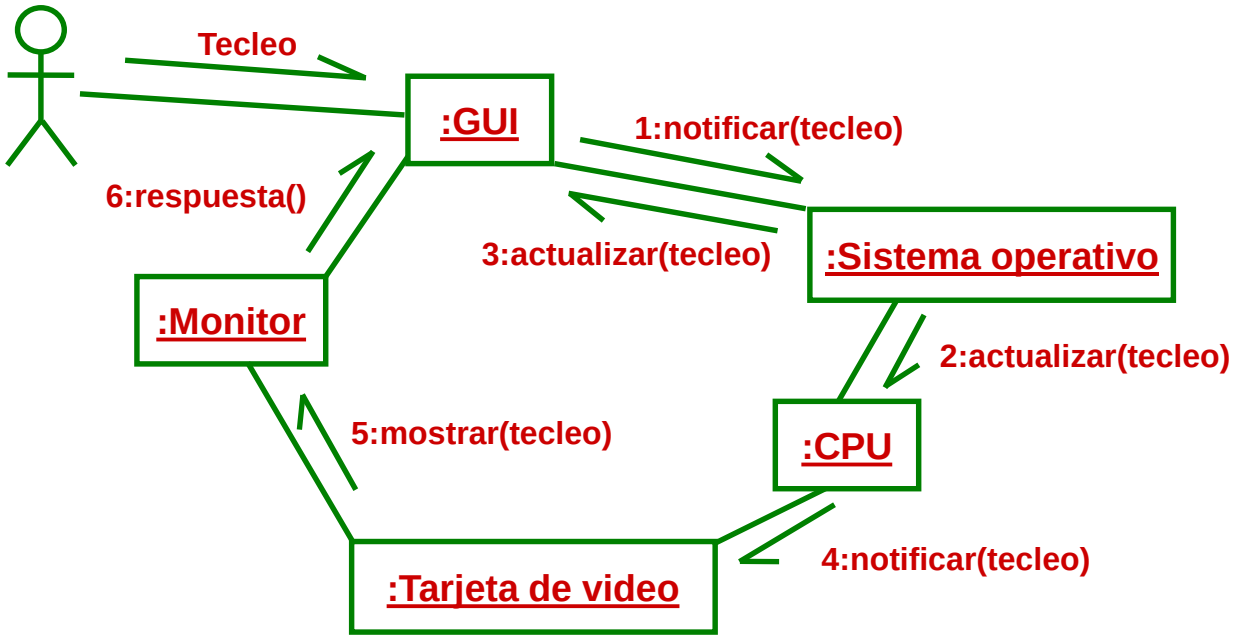
- **Diagramas de secuencias. (cont.)**
  - Cuando utilizar los diagramas de secuencias, se sugieren para:
    - Son útiles para ver la interacción entre los objetos, debido a que pone énfasis en la secuencia y es fácil apreciar el orden en el que ocurren las cosas.
    - Cuando se desee ver el comportamiento de varios objetos en un caso de uso y la secuencia de los mensajes.
  - No se sugieren para:
    - No son convenientes para representar el comportamiento condicional, debido a que son para mostrar un comportamiento simple, se sugiere usar mejor diagramas separados para cada una de las condiciones
    - No sirve para ver el comportamiento de un solo objeto a través de muchos casos de uso (usar mejor un diagrama de estados)
    - Si quiere ver el comportamiento a través de muchos casos de uso o muchos proceso mejor utilice un diagrama de actividad.

- **Diagramas de colaboraciones.**
  - Muestra los objetos,las relaciones entre ellos, los mensajes que se envían los objetos entre sí.
    - El mensaje se representa como una flecha cerca de la línea de asociación entre dos objetos. Esta flecha apunta al objeto receptor. El tipo de mensaje se mostrará en una etiqueta cerca de la flecha.
    - El mensaje le indicará al objeto receptor que ejecute una de sus operaciones.
    - Un diagrama de secuencias puede ser convertido en uno de colaboraciones y viceversa.
    - Se agregará una cifra al mensaje para indicar la secuencia propia del mensaje.



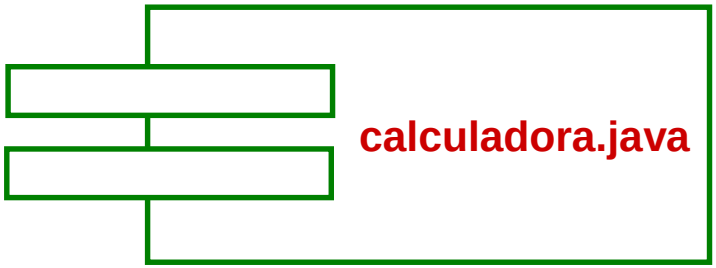
Diagramas de colaboraciones. (cont.)

- Ejemplo de un diagrama de colaboraciones:
  - El actor es el usuario quien inicia la interacción al oprimir una tecla, se inicia la siguiente secuencia:
    - La GUI notifica al sistema operativo que se oprimió la tecla
    - El sistema operativo le notifica a la CPU
    - El sistema operativo actualiza la GUI
    - La CPU notifica a la tarjeta de video
    - La tarjeta de video envía un mensaje al monitor
    - El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.



- **Diagramas de colaboraciones. (cont.)**
  - Cuando utilizar los diagramas de colaboración, se sugieren para:
    - Es la mejor forma si se quiere mostrar los objetos y mostrar como se reconectan estáticamente unos con otros.
    - Cuando se desee ver el comportamiento de varios objetos en un caso de uso.
    - Sirven para mostrar la colaboración entre los objetos, sin embargo, no sirven tan bien para la definición precisa del comportamiento
  - No se sugieren para:
    - No son convenientes para representar el comportamiento condicional, debido a que son para mostrar un comportamiento simple, se sugiere usar mejor diagramas separados para cada una de las condiciones
    - No sirve para ver el comportamiento de un solo objeto a través de muchos casos de uso (usar mejor un diagrama de estados)
    - Si quiere ver el comportamiento a través de muchos casos de uso o muchos proceso mejor utilice un diagrama de actividad.

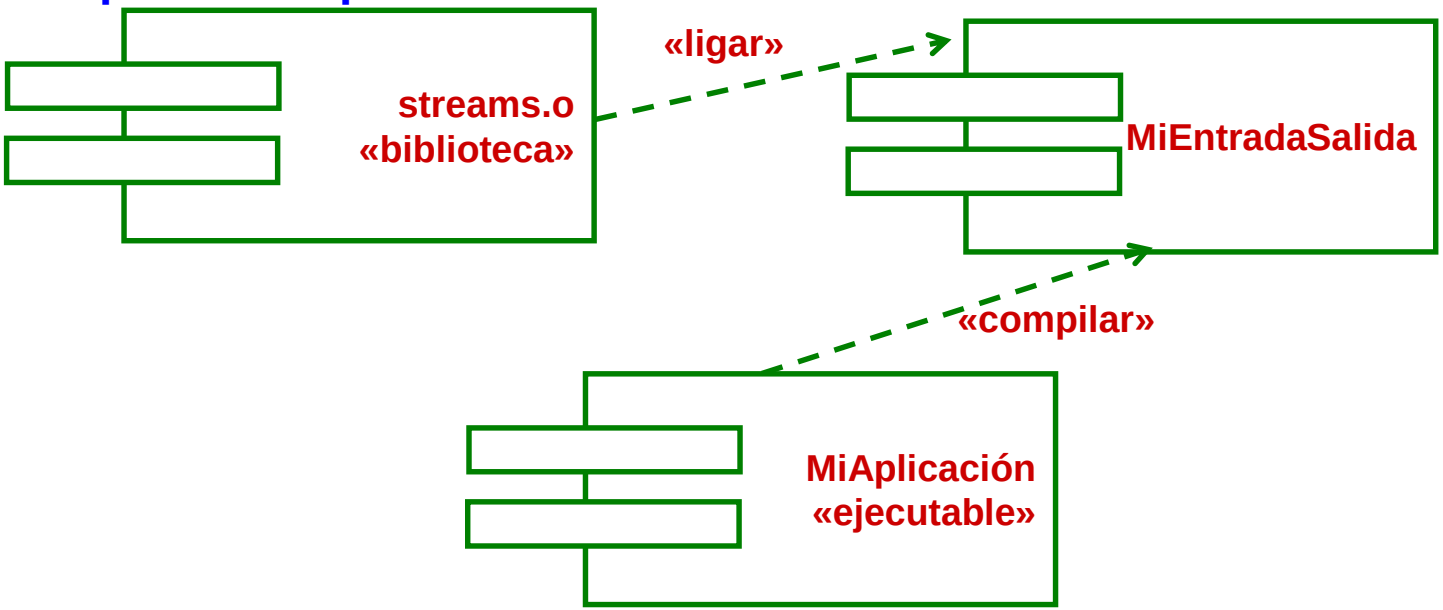
- **Diagramas de componentes**
  - Un componente es la implementación de un subsistema, la cual da las especificaciones (en términos de casos de uso) y una estructura de clases que lleva a cabo la especificación. Su representación es:



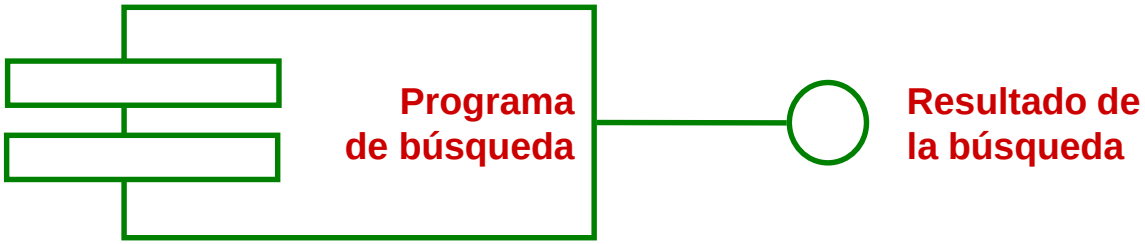
  - Hay diferentes tipos de componentes y cada uno con diferentes tipos de dependencia , Clasificándolos en relación con el proceso de compilación un componente podría ser:
    - Código fuente, el cual depende de que otros componentes estén disponibles al momento de compilación.
    - Código objeto binario, (biblioteca de clases) que depende de cualquier código objeto al cuál se ligara desde un programa ejecutable.
    - Una aplicación ejecutable, la cuál puede depender de otros programas ejecutables al tiempo de ejecución.
  - Los detalles dependen del lenguaje de programación usado, se pueden usar estereotipos como «compilar» ó «ligar», igualmente se pueden usar estereotipos para diferenciar los diferentes tipos de componentes, por ejemplo: «archivo» , «biblioteca» , «ejecutable» , «tabla», «documento»

- **Diagramas de componentes (cont.)**
  - Cada clase del modelo lógico se realiza en dos componentes: la especificación y el cuerpo.
  - La especificación contiene la interfaz de la clase mientras que el cuerpo contiene la realización de la clase.
  - En el caso de clases parametrizables se puede dar una especificación genérica.
  - Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente se refiere a los servicios ofrecidos por otro.
  - Los distintos componentes pueden agruparse en paquetes según un criterio lógico y con vistas a simplificar la implementación. Son paquetes estereotipados en «subsistemas» para incorporar la noción de biblioteca de compilación y de gestión de configuración.

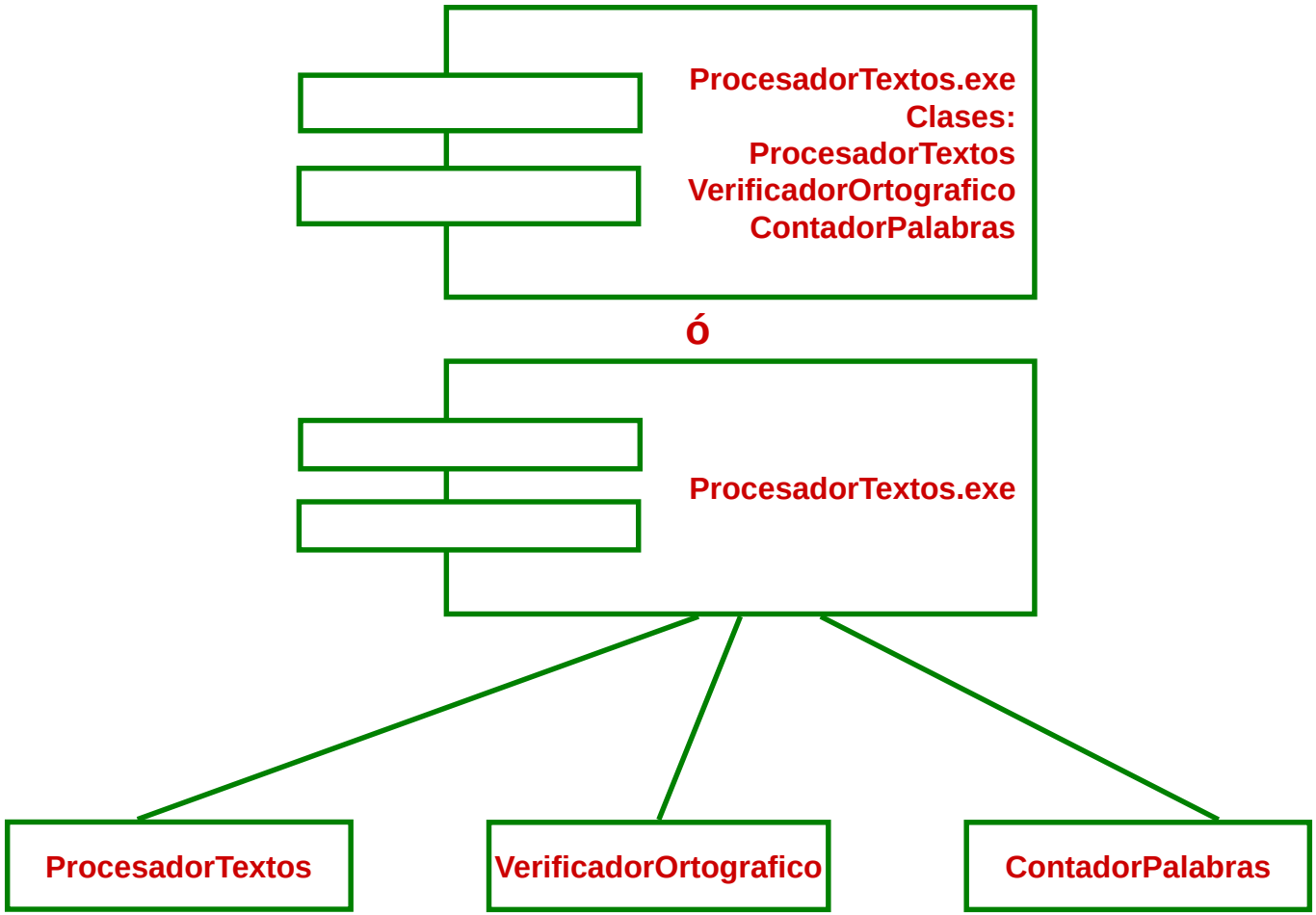
- ## Diagramas de componentes (cont.)
  - **Un diagrama de componentes mostrando las dependencias de tiempo de compilación:**



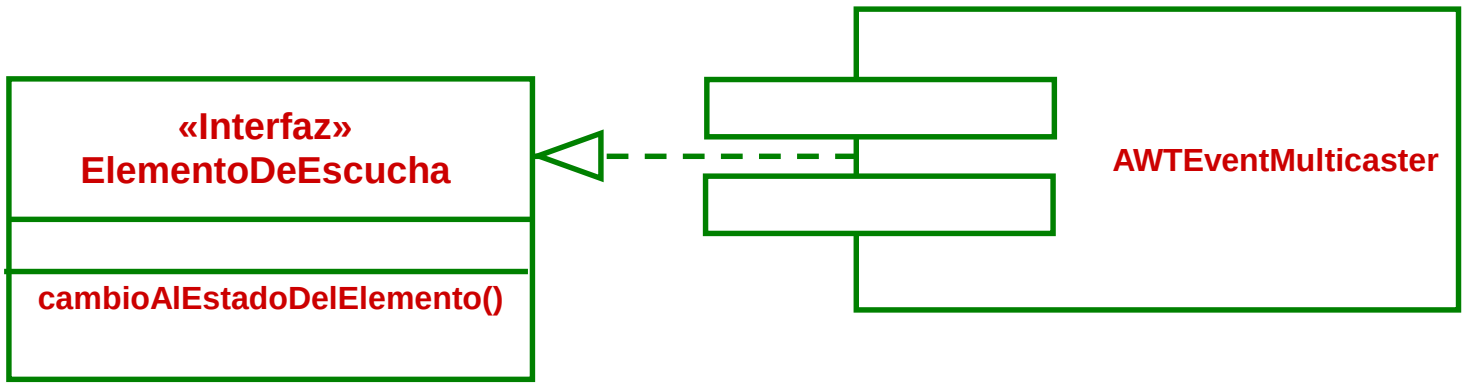
  - **La interfaz se puede representar por una línea con un círculo**



- **Diagramas de componentes (cont.)**
  - Si un componente implementa clases se puede establecer la relación entre el componente y las clases como sigue:

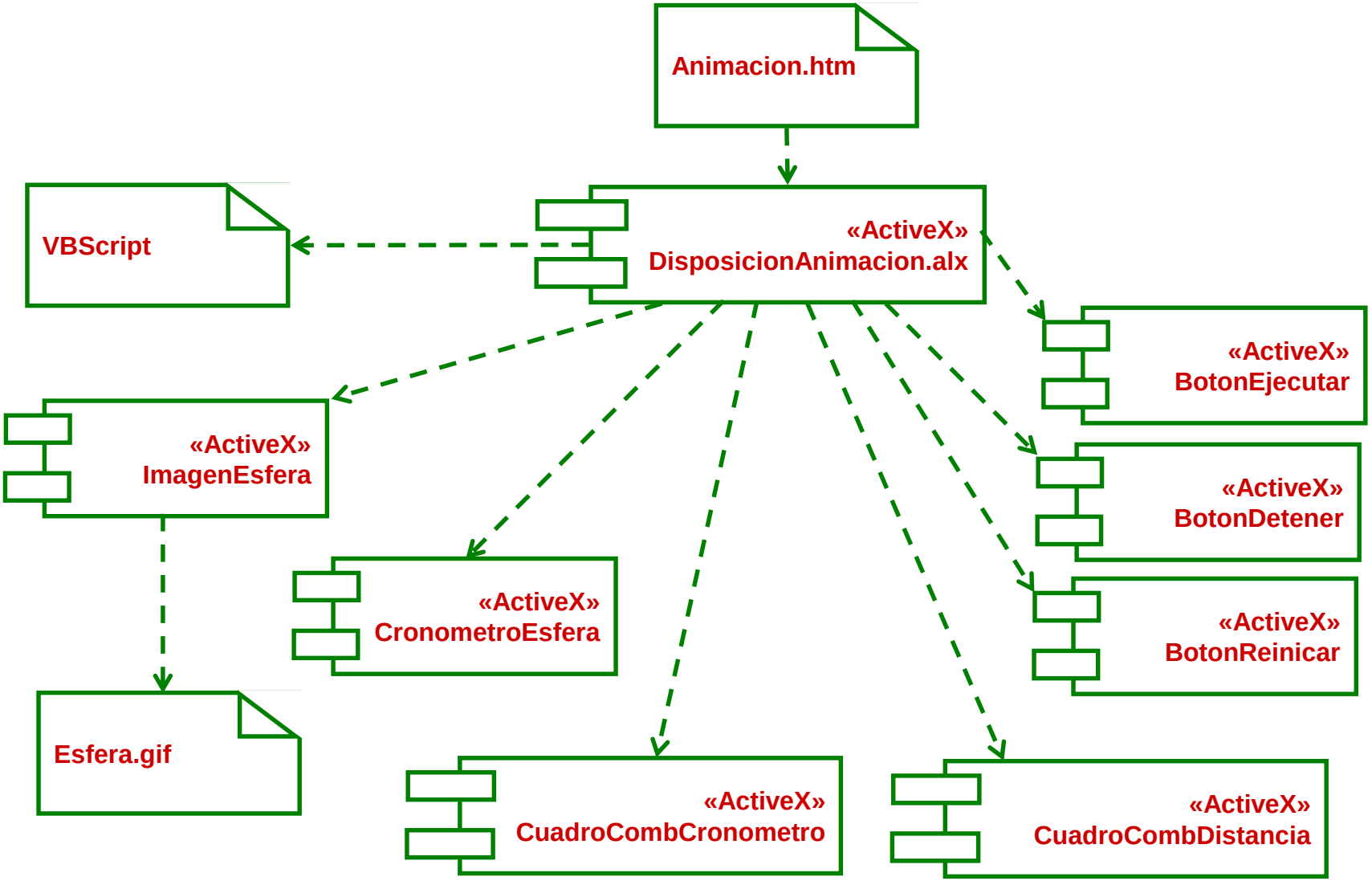


- **Diagramas de componentes (cont.)**
  - Solo se podrán ejecutar las operaciones dentro de un componente a través de su interfase. La relación entre un componente y su interfase se conoce como realización. Un componente puede acceder a los servicios de otro componente mediante el uso de su interfase, al que proporciona el servicio se dice que provee una *interfaz de exportación*, al que accede a un servicio se dice que utiliza una *interfaz de importación*.

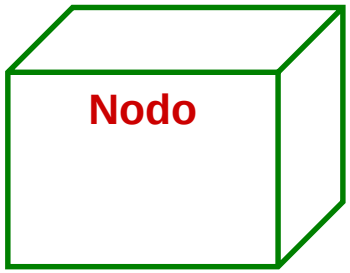


  - Se puede sustituir un componente por otro si el nuevo contiene las mismas interfases que el anterior. Se puede reutilizar un componente en otro sistema si éste puede acceder al componente reutilizado mediante sus interfases.

- **Diagramas de componentes (cont.)**
  - **Página Web con controles ActiveX.**

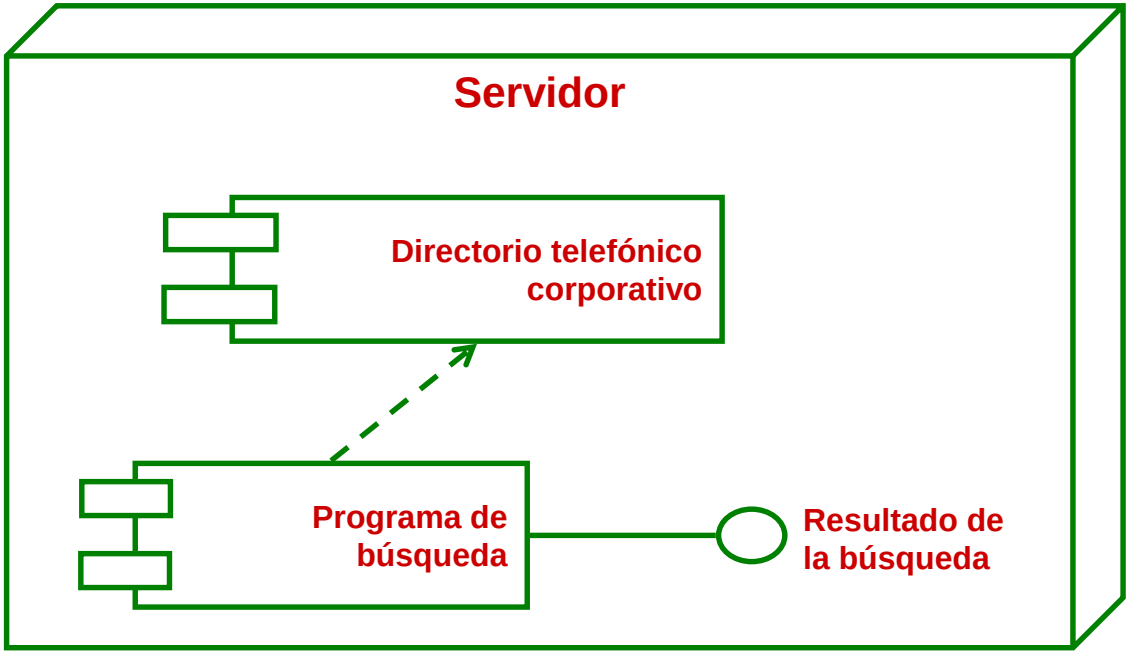


- **Diagramas de distribución / Despliegue**
  - Los diagramas de distribución muestran la disposición física de los distintos **nodos** que componen un sistema y el reparto de los componentes sobre dichos nodos.
  - Un nodo representa todo tipo de equipo de cómputo y se representa por un cubo:

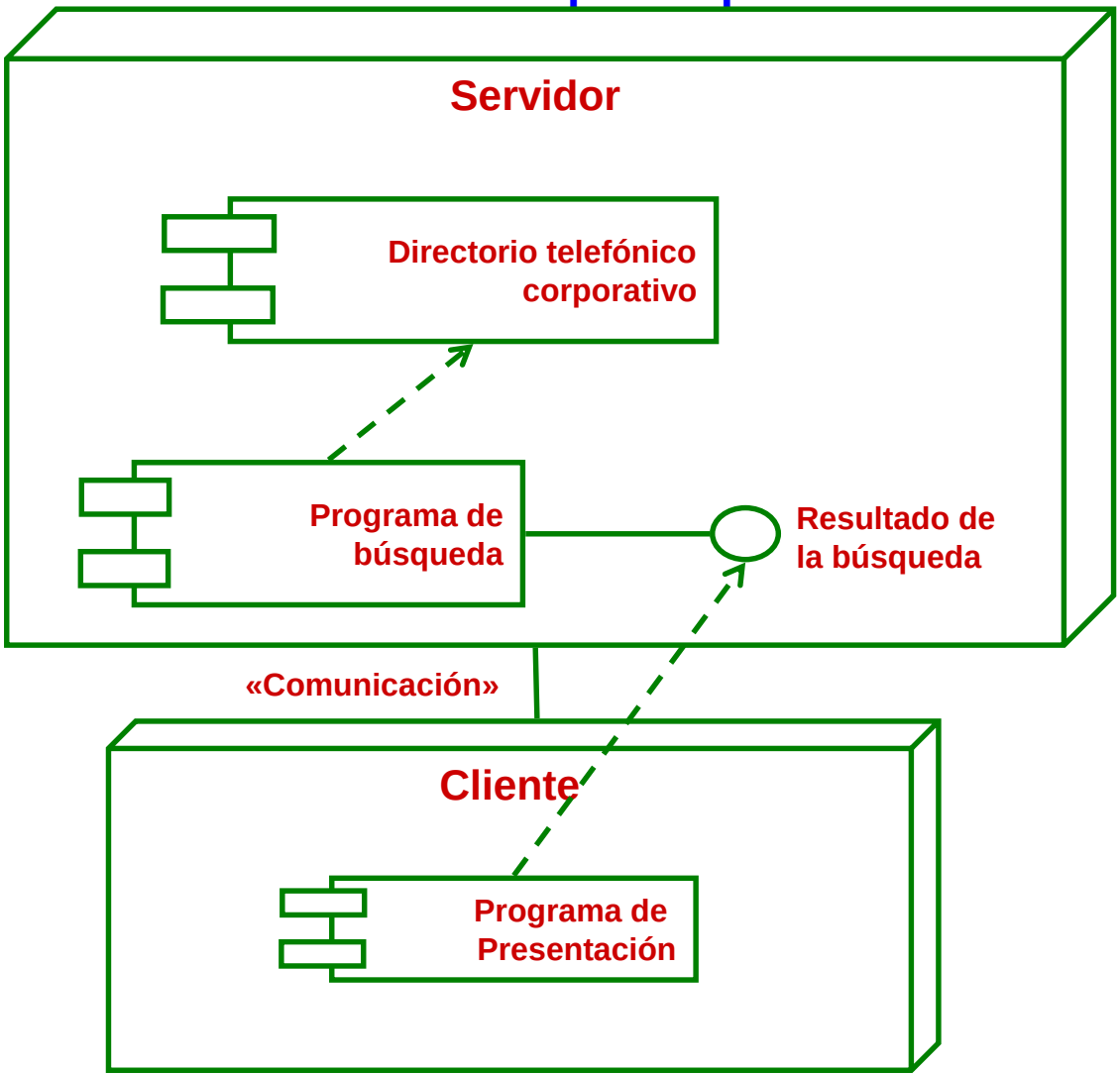


  - Los estereotipos permiten precisar la naturaleza del equipo:
    - Dispositivos
    - Procesadores
    - Memoria
    - Etc.

- **Diagramas de distribución / Despliegue**
  - Los nodos se interconectan mediante soportes bidireccionales (en principio) que puede a su vez estereotiparse.
  - Se pueden mostrar los componentes en relaciones de dependencia con un nodo:

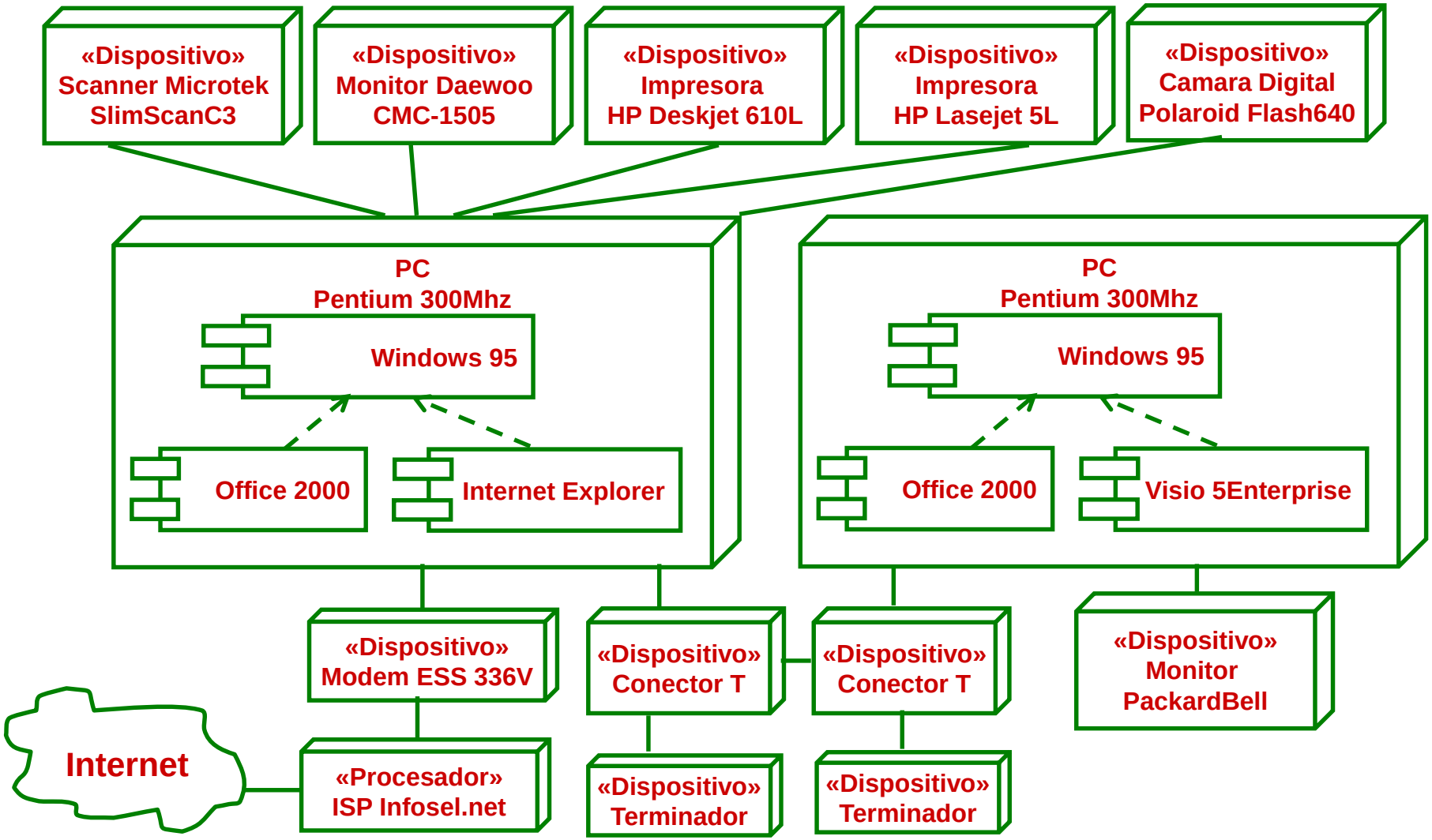


- **Diagramas de distribución /Despliegue (cont.)**
  - Las conexiones entre nodos se puede poner mediante una relación



Diagramas de distribución / Despliegue (cont.)

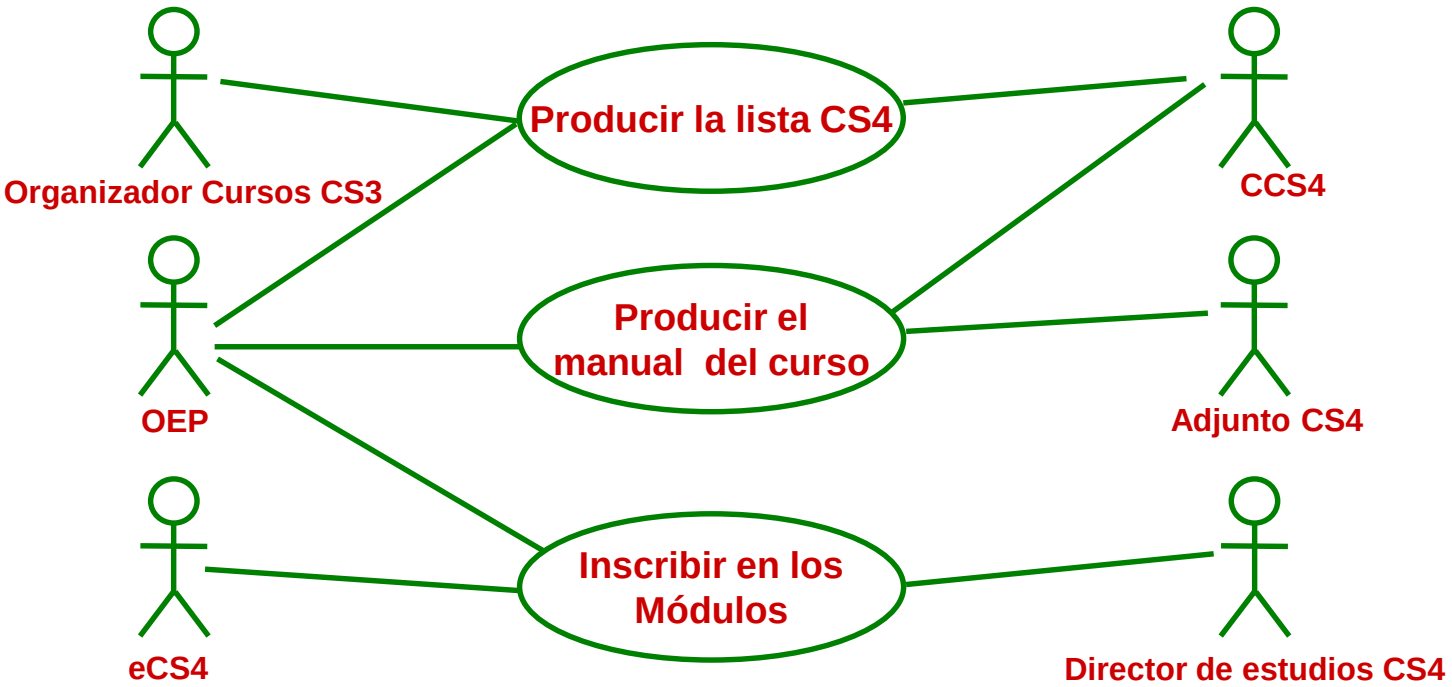
- Aplicación de los diagramas de distribución.
  - Un equipo de cómputo casero podría ser como sigue:



- # Presentación del caso.
  - ## Administración CS4
  - Un *estudiante CS4 (eCS4)*, es cualquier estudiante que esta tomando cualquier módulo de cuarto año en el departamento de ciencias computacionales, ya sea que este o no inscrito para un grado en ciencias computacionales.
  - Al final de cada año académico, el Comité Académico de el Departamento de Ciencias Computacionales determina qué módulos estarán disponibles para los *eCS4* en el siguiente año.
  - Al final de cada año el Jefe del Departamento fija actividades para los miembros del cuerpo de maestros y otros, en particular a una persona es asignada para enseñar cada uno de los módulos que se supone van a estar disponibles para el próximo año (*adjunto*)
  - Cada profesor adjunto actualiza los apuntes en el manual del curso de su módulo. El coordinador CS4 (*CCS4*) actualiza otras partes de cada manual y checa los apuntes producidos por los profesores adjuntos. Los apuntes de los módulos están escritos en el lenguaje de formato LaTeX.
  - Alguien en la Oficina de Enseñanza Profesional (*OEP*) produce la versión en papel de cada manual de curso; el *CCS4* produce la versión HTML ejecutando la aplicación *latex2html* sobre la fuente LaTeX.
  - El *CCS3* proporciona una lista de los estudiantes entrando a CS4 provenientes de CS3 al *CCS4* y al *OEP*. El *CCS4* le dice al *OEP* acerca de cualquier otro estudiante que provenga de de otros cursos que no sean CS3. El *OEP* mantiene la lista maestra de todos los *eCS4* y actualiza las listas de correo de los estudiantes tomando cursos CS4, la cual se conoce por la dirección de correo *cs4class*.
  - Cada estudiante es avisado por un miembro del cuerpo de maestros que actúa como Director de Estudios (*DdE*). Un *DdE* se asigna a un estudiante en su primer año de estudios y permanece con ésa función hasta que termina.
  - Los estudiantes se inscriben en forma provisional en los módulos llenando una forma y entregándola en la Oficina de Enseñanza Profesional . El *OEP* verifica que cada estudiante que se inscriba, esté listado como un *eCS4*, y cada *eCS4* es inscrito en un numero razonable de módulos. En caso de duda, se consulta al *DdE* del estudiante y se puede tener una plática con el estudiante.
  - El OEP produce luego las listas para los adjuntos de los estudiantes que tomarán sus módulos. Estas listas no pueden llegarles a los adjuntos antes de la semana 3. Esto es, muy tarde para que los adjuntos puedan saber cuántas copias deben preparar de su material.

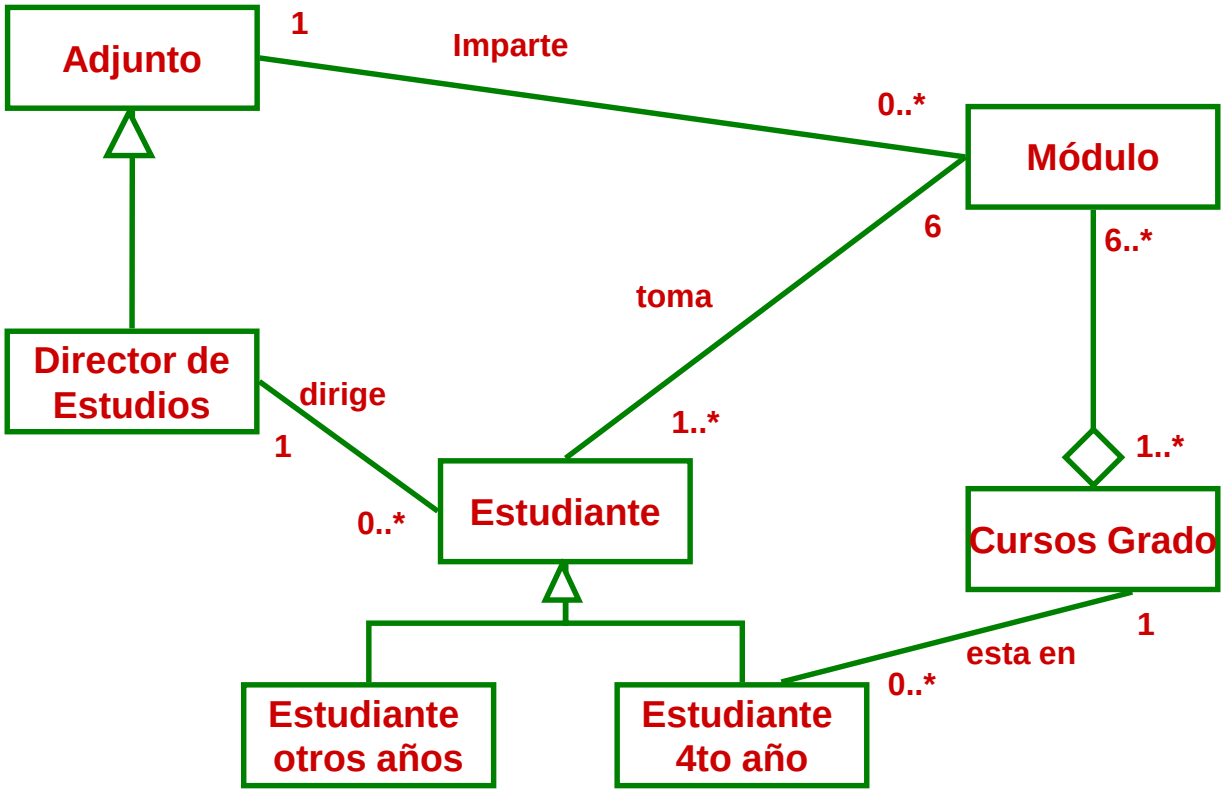
- **Administración CS4 (cont.)**
  - Cada uno de los cursos de grado tiene su propio manual de curso, los principales cursos existentes son: Ciencias Computacionales, Ciencias Computacionales e Inteligencia Artificial, Ciencias Computacionales y la Ingeniería Electrónica, etc.
  - Los detalles de la asesoría y las reglas acerca de cuáles combinaciones de módulos son aceptables, son diferentes para cada grado, igualmente hay manuales separados para cada uno de ellos.
  - Sin embargo, muchos módulos se aceptan en varios diferentes cursos de grado, y en cada caso la descripción de cada curso es igual en cada manual.
  - Cada estudiante se inscribe para un curso de grado y recibe el manual de curso apropiado.
  - El *CCS4* de producir todos los manuales de cursos (en los casos de alumnos adjuntos que lleven los cursos es posible que reciban duplicado el manual, debido a que los otros departamentos producen sus propios manuales)
  - **Se pide desarrollar un sistema que permita:**
    - Disminuir la cantidad de trabajo rutinario para todo el staff, especialmente el *CCS4*.
    - Permitir a los estudiantes inscribirse en los módulos en línea.
    - Que el *OEP* pueda obtener fácilmente información actualizada y confiable.
    - Mejorar el seguimiento de dicha información.
    - Hacer que la información tal como los manuales de cursos y las listas de los estudiantes que toman los cursos estén disponibles cuanto antes, automatizando su producción.
    - El sistema de administración CS4 deberá poder producir un informe sobre cada *eCS4* indicando si es de 4to año o aún no se gradúa, qué módulos está tomando el estudiante, cuales cursos de grado esta llevando un *eCS4*, o cuál miembro del staff es el *DdE* de un *eCS4*.
    - Deberá poder dar información sobre los módulos, quiénes los imparten, de que curso forman parte y que estudiantes los están tomando.

- **Diagrama de casos de uso (CS4)**
  - Las consultas pueden ser realizadas mediante una base de datos y usando técnicas estándar para hacer objetos persistentes. Y se dejará fuera de la respuesta, quedando pues los siguientes casos de uso:
    - Producir el manual del curso
    - Producir la lista CS4
    - Inscribir en los módulos.

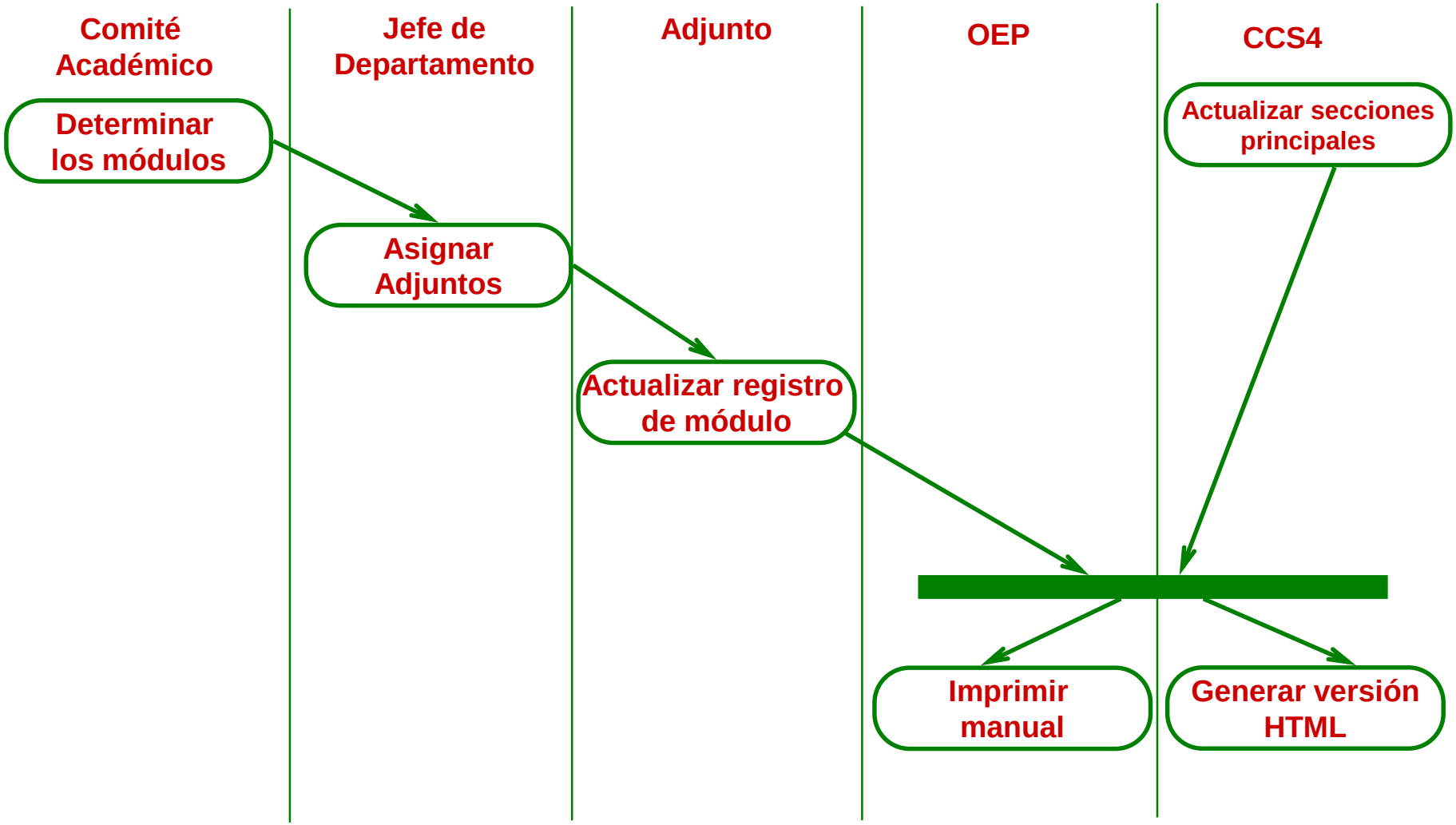


- **Diagrama de casos de uso (CS4)**
  - Descripción de caso de uso: **Producir el manual del curso**
    - Este caso de uso se puede usarse solamente cuando el comité académico ha determinado el conjunto de módulos que estarán disponibles y que el jefe de departamento ha asignado los adjuntos.
    - El organizador de CS4 actualiza las secciones principales de cada manual de curso obteniendo el texto actual del sistema, modificándolo y regresándolo modificado al sistema.
    - El adjunto de cada módulo, actualiza la descripción del módulo tomándolo del sistema, actualizándolo y regresándolo al sistema.
    - Estas actualizaciones pueden suceder en cualquier orden. El sistema conserva el registro de cuáles cambios se han hecho. Una vez que se hicieron todas las actualizaciones, el sistema envía el texto completo del manual por correo electrónico al OEP, el cual imprime y actualiza la pagina Web del mismo.
  - Desarrolle las descripciones de los casos de uso para:
    - Producir la lista CS4
    - Inscribir en los módulos.

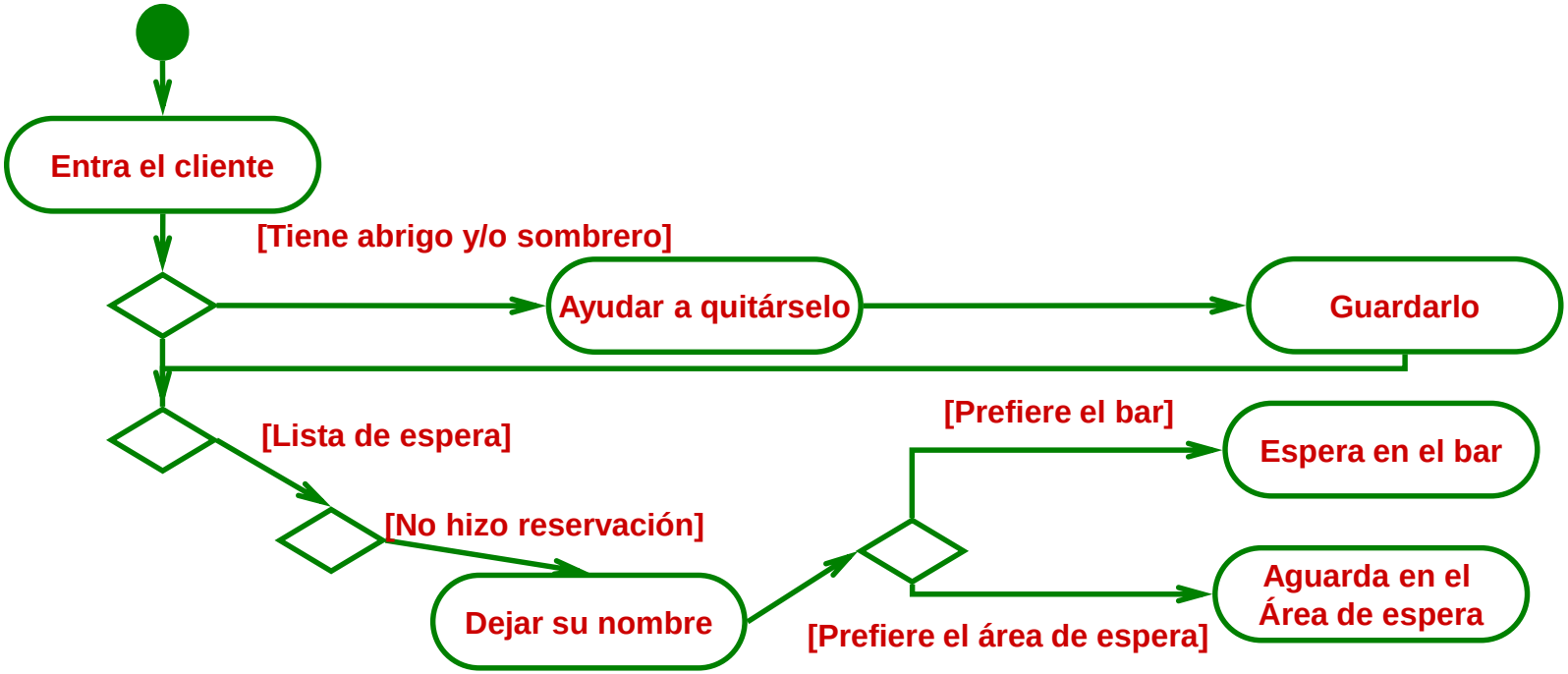
- **Diagrama de clases (CS4)**
  - **Un diagrama de clases a nivel conceptual, sin incluir actividades y operaciones:**



- ## Diagrama de actividad (CS4)
  - **El siguiente diagrama muestra el flujo de trabajo requerido para determinar qué cursos hay, quien los imparte, generar los manuales de los cursos:**



- **Presentación del caso.**
  - Las empresa Restaurantes,S.A. ha hecho una encuesta sobre el mundo de los restaurantes y ha llegado a sorprendentes conclusiones: a la gente le gusta comer fuera, pero no disfruta algunos momentos de ésa experiencia. Y deciden construir el restaurante del futuro, que deje a la gente darse el gusto y que mejore los resultados para el cliente.
  - **Entrevista de Análisis**
  - ¿Qué sucede cuando un cliente entra al restaurante? R: Si el cliente tiene un abrigo,le ayudamos a quitárselo, lo almacenamos en el guardarropa y le damos un boleto para solicitarlo posteriormente. Lo mismo hacemos con el sombrero.
  - ¿Y si hay línea de espera, se forma? R: Se le pregunta si hizo reservación, en cuyo caso lo pasamos lo más pronto posible. Si no hay reservación deja su nombre y puede ir al bar a tomar algo antes de comer o esperar sentado en área de espera.

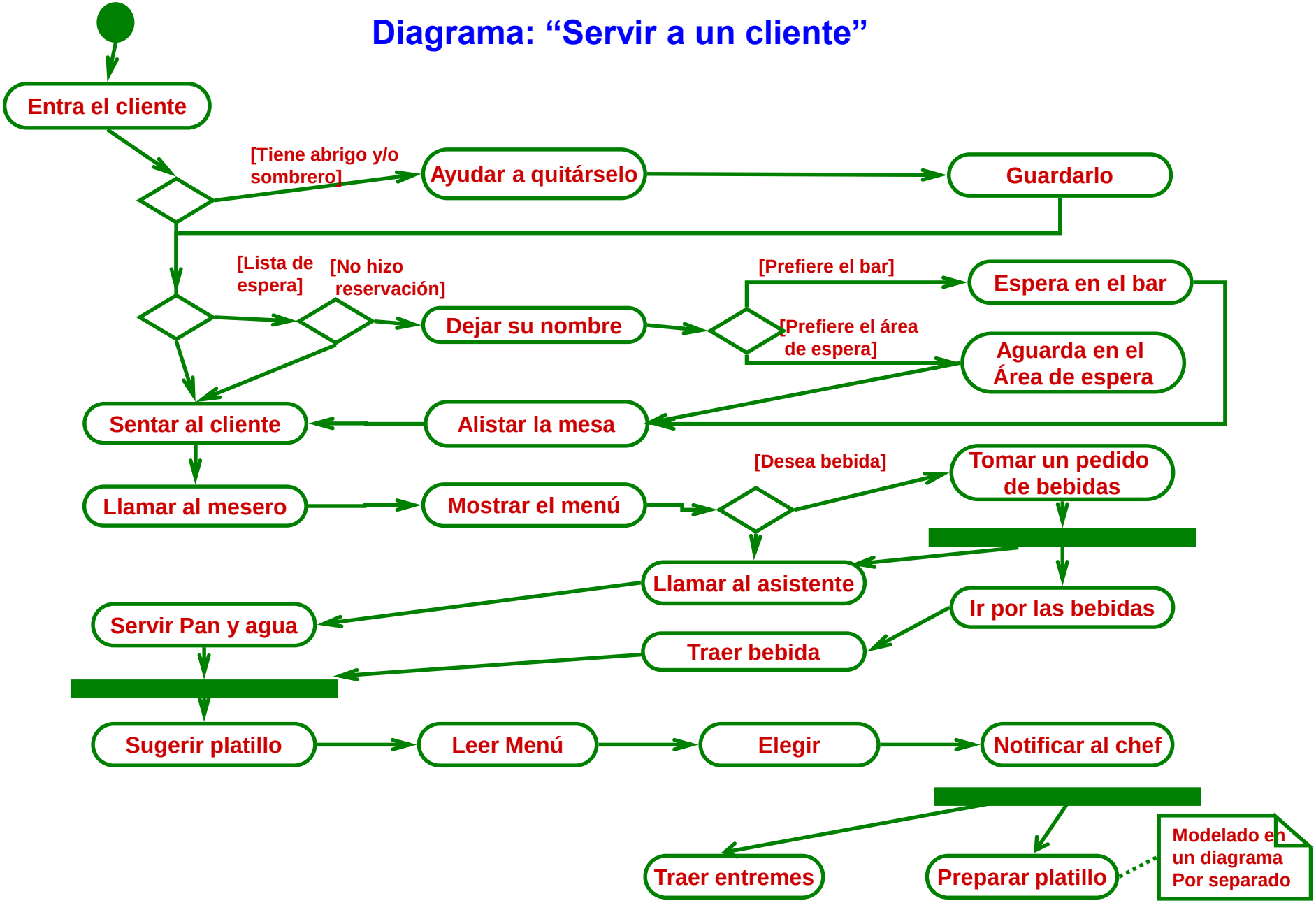


Presentación del caso. (cont.)

Entrevista de Análisis (cont.)

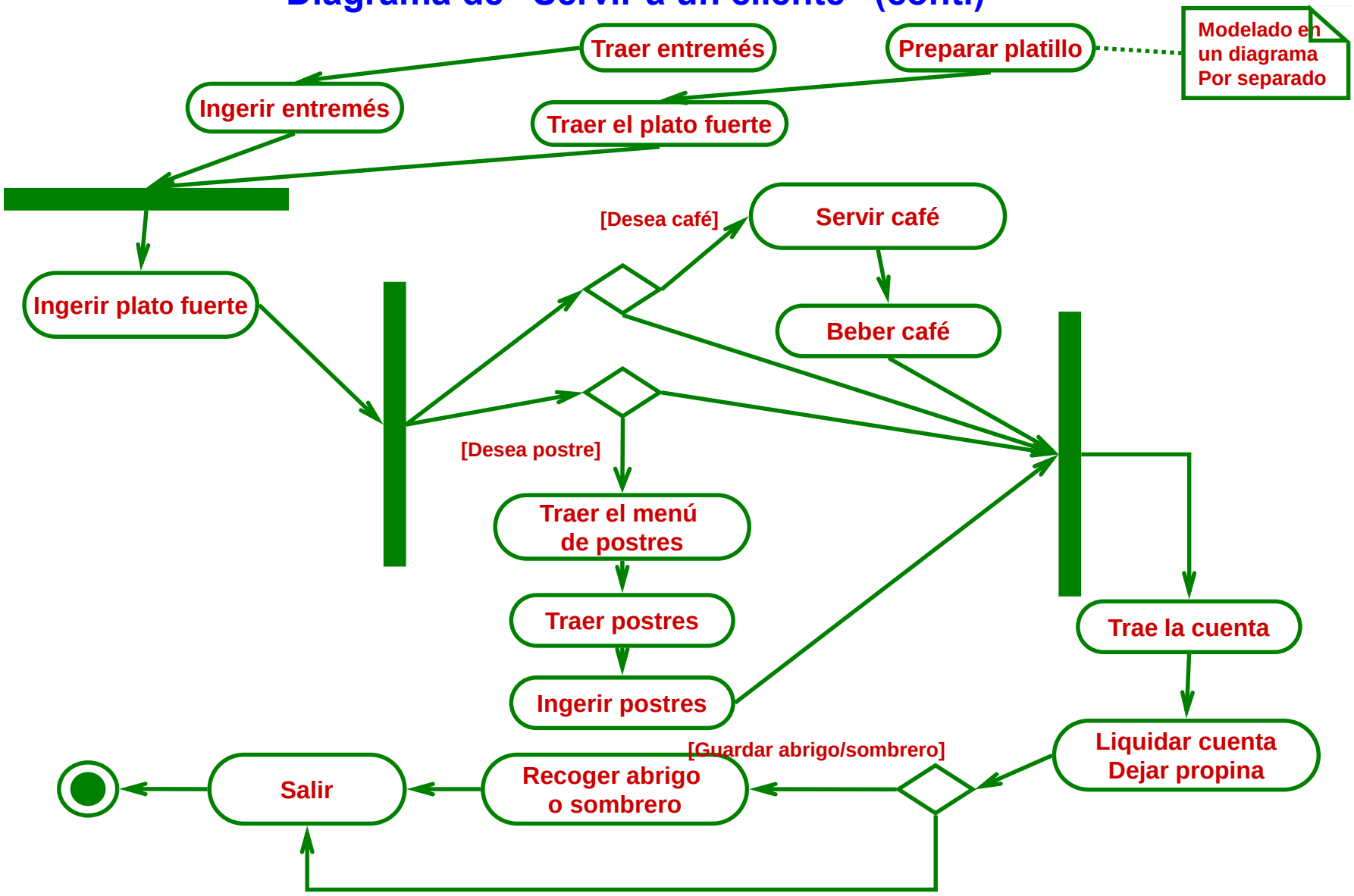
- ¿Cuando le llega el turno al cliente, es hora de sentarlo? R: La mesa deberá estar lista; para ello, deberá ser limpiada. Un mozo de piso debe cambiar el mantel usado por el cliente anterior y acomodar la mesa. Cuando esta lista el capitán de meseros lleva al cliente a su mesa y luego llama a un mesero, si el cliente estaba bebiendo en el bar, se podrá llevar su bebida. Los meseros tienen asignadas sus áreas de servicio y saben cuando está lista una mesa.
- ¿Luego que ocurre? R: El mesero llega a la mesa, entrega un menú a cada comensal y les pregunta si desean alguna bebida mientras deciden. Luego llama a un "asistente" quién coloca una charola con pan y mantequilla. Si alguien ha pedido una bebida el mesero la traerá.
- ¿Cómo deciden los clientes qué van a consumir? R: El mesero propone siempre a los clientes la sugerencia del día y les da de 5 a 10 minutos para que decidan. Cuando el cliente llama la atención del mesero, éste regresa a tomarle su orden. Y luego lo notifica al chef. Mediante la transcripción de la elección en un formulario, llamado comanda.
- ¿Qué contiene la comanda? R: La mesa, la elección y la hora. Debido a que generalmente la cocina está muy ocupada y el chef tiene que dar prioridad a las comandas en el orden que hayan llegado.
- ¿Por qué es tan importante la prioridad? R: La mayoría de las comidas incluyen un entremés antes del plato fuerte. A la mayoría de la gente le gusta comer el plato fuerte caliente, el chef prepara el entremés y el mesero se lo lleva al cliente. El reto está en llevar el plato fuerte, caliente a todos los comensales en la mesa al mismo tiempo.

Diagrama: "Servir a un cliente"



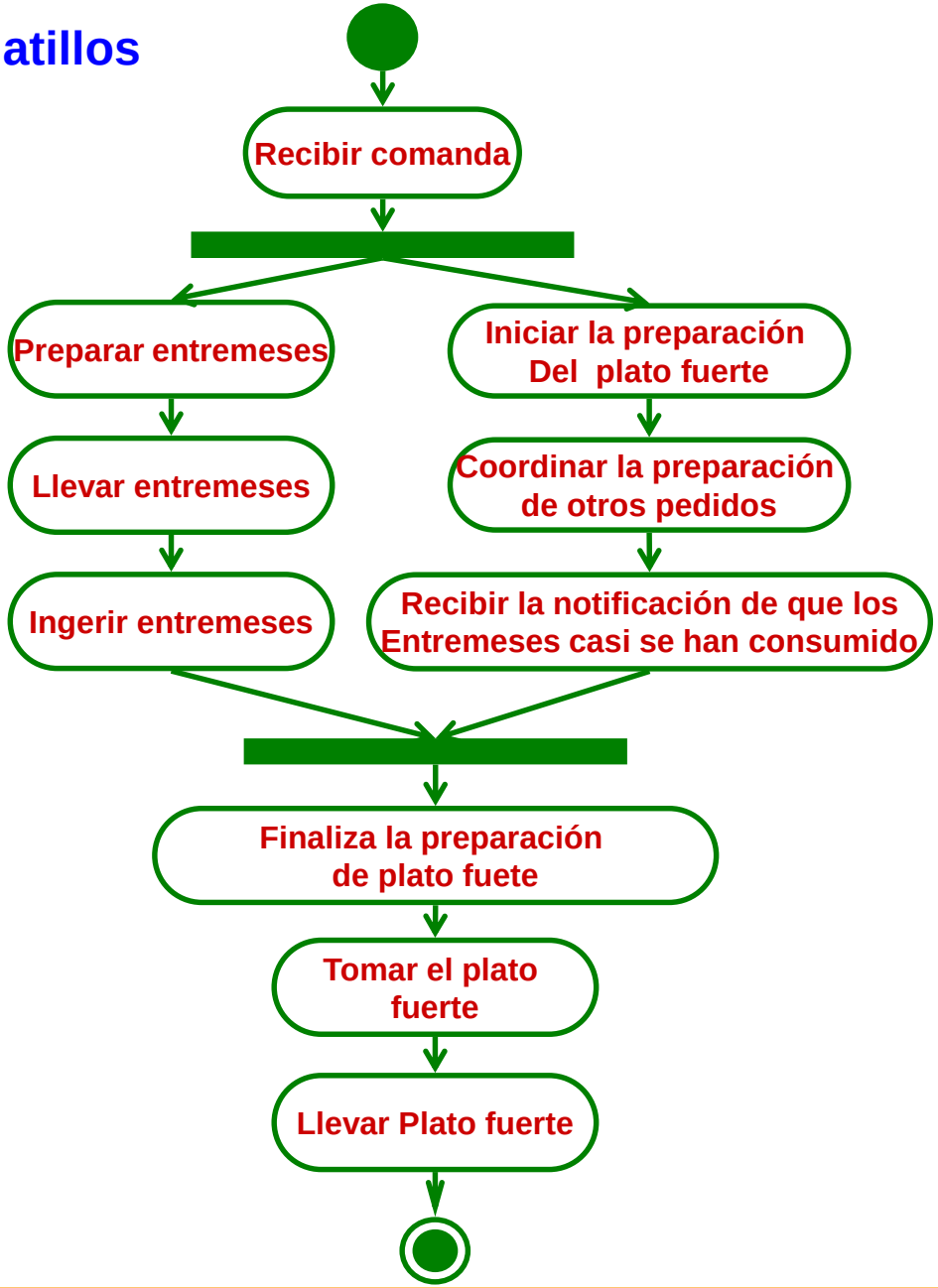
- **Entrevista de Análisis (cont.)**
  - ¿Cómo le sirven al cliente? R: El chef preparará el plato fuerte y el mesero lo recogerá cuando se dé cuenta de que los comensales han terminado con el entremés; después el mesero lleva el plato fuerte a la mesa. Los comensales ingerirán sus alimentos, y el mesero regresará al menos una vez para verificar si todo está bien.
  - ¿Qué sucede cuando los clientes terminan de comer? R: El mesero regresa y pregunta si desean postre, en cuyo caso el mesero dará el menú de postres y tomará las órdenes. En caso de no desearlo pregunta si desean café, en caso de aceptarlo, traerá el café, tazas y les servirá. Si no desean nada, traerá la cuenta. Luego regresará y obtendrá el pago en efectivo o en tarjeta de crédito. Luego traerá el cambio o los vouchers, los clientes dejarán una propina, recogerán sus abrigos y se irán.
  - ¿Qué debe hacer el mesero cuando el cliente salga? R: Llamar al mozo de piso para limpiar la mesa, la preparará y estará lista para los siguientes comensales.

Diagrama de "Servir a un cliente" (cont.)

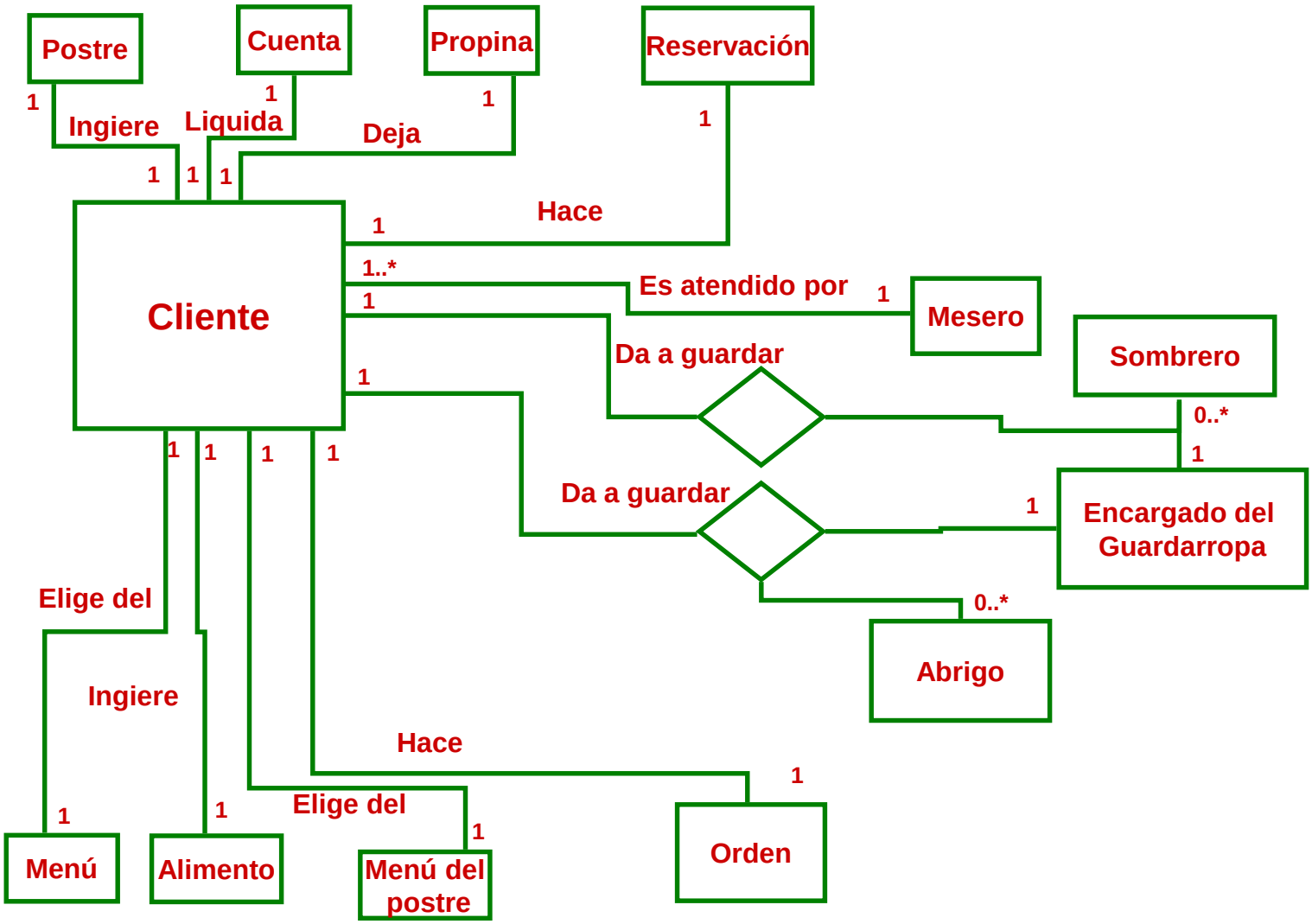


- **Preparación de platillos**
  - ¿Cómo se coordina el chef para tener los platillos a tiempo? R: La gente en una mesa casi siempre termina sus entremeses, en momentos distintos. Entre el mesero y el chef se coordinan para traerles a todos los platos fuertes al mismo tiempo. El chef recibe la comanda y empieza a preparar los entremeses y el plato fuerte, cuando esta terminado el entremés, el mesero va a la cocina, los toma y los lleva a la mesa.
  - ¿Cómo se entera el mesero que ya están listos los entremeses? R: El mesero va a la cocina de vez en cuando. El chef luego de dar el entremés al mesero, espera que este le avise cuando la mayoría de los comensales ya casi ha terminado con sus entremeses para poderle dar el toque final a cada plato fuerte. El mesero va a la cocina, y le avisa al chef que ya casi están listos para el plato fuerte, el chef termina su preparación. El mesero los toma y los lleva a la mesa

- **Preparación de platillos**



- **Clases y asociaciones**
  - El cliente se asocia con una gran cantidad de clases, como muestran las asociaciones a continuación:



- **Detalle de las clases**
  - **Cliente, sus atributos serían:**

Cliente
nombre horallegada orden horaservicio
ingerir() beber() ordenar() pagar()

- **Detalle de las clases**
  - Empleado es una clase abstracta que puede contener la información de los empleados y como clases secundarias, se pueden tener Mesero, Chef, Gerente, Asistente.

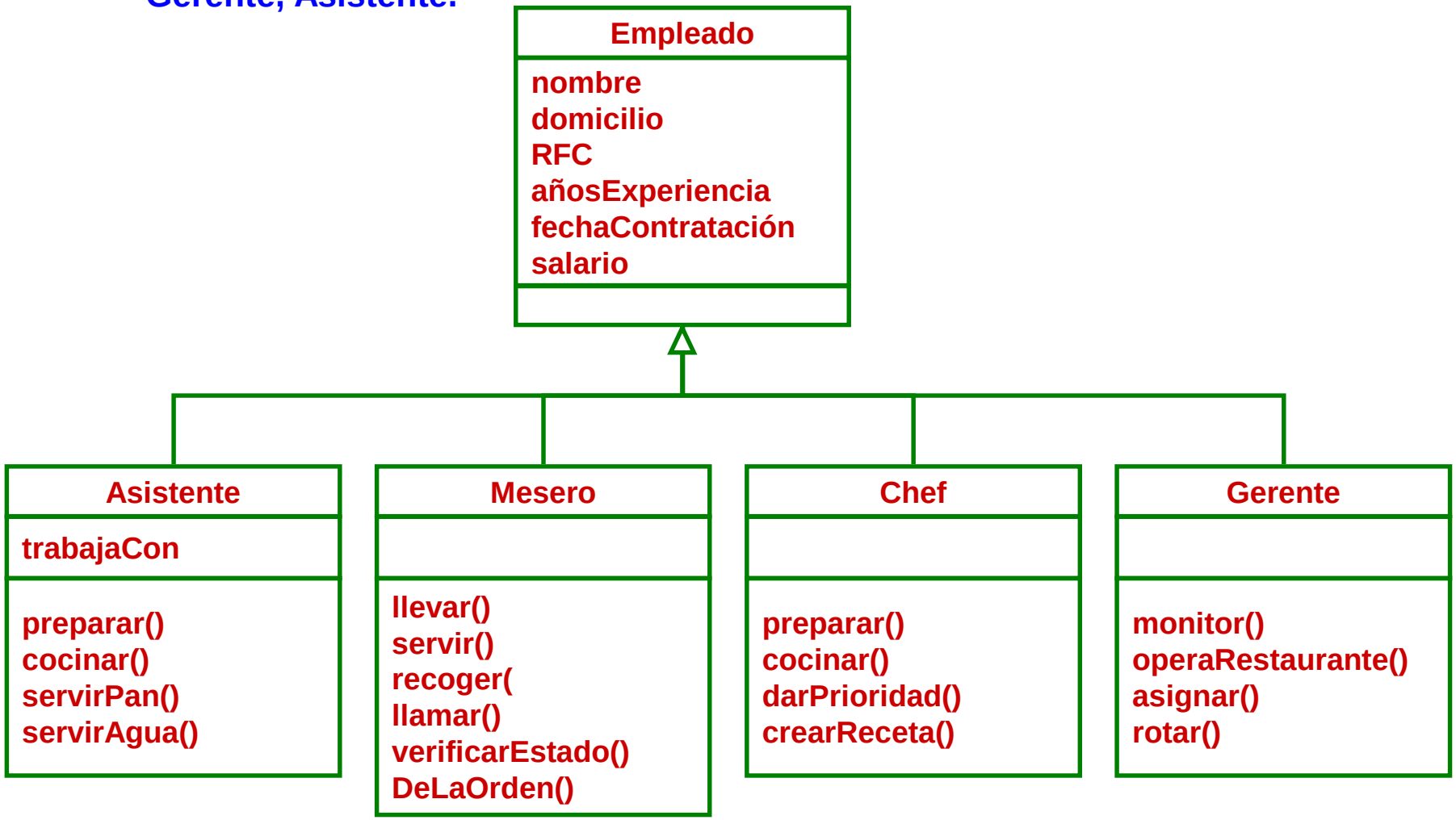
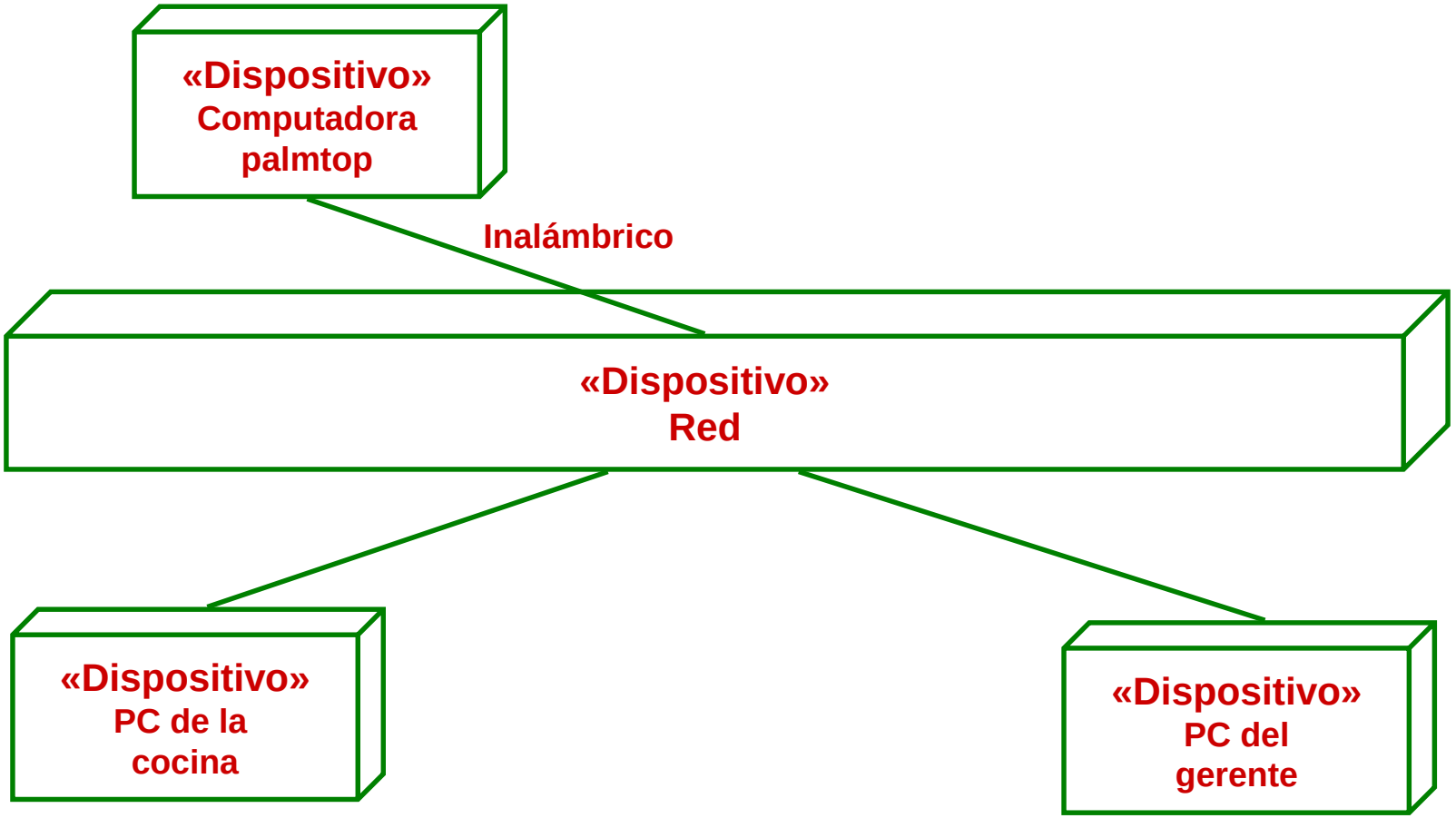


Diagrama de distribución, Restaurante

- Los meseros contarán con una computadora palmtop y la utilizarán para comunicarse con la cocina y con los mozos de piso. Estos últimos también tendrán palmtops para comunicarse. La cocina tendrá una terminal de escritorio y una o varias pantallas. El gerente tendrá una en su oficina.



- **Casos de uso**
  - El paquete mesero