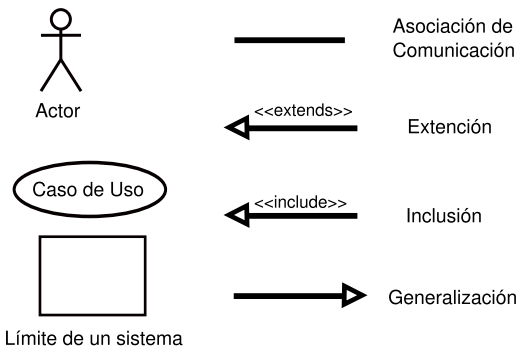
Casos de uso
umh2818-TADS



Notación de caso de uso

Un **caso de uso** es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Los más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos, donde se originaron, si bien puede utilizarse con resultados igualmente satisfactorios con otros paradigmas de programación.

1 Un poco de historia en la programación

En 1986, Ivar Jacobson, importante contribuyente al desarrollo de los modelos de UML y proceso unificado, creó el concepto de caso de uso.[1] Se han realizado muchas mejoras al concepto que se estableció entonces, pero probablemente la más influyente y significativa, en términos de definición del término caso de uso, fue la de Alistair Cockburn en el libro *Escribir casos de uso efectivos* publicado en el año 2000.

Durante los años 1990 los casos de uso se convirtieron en una de las prácticas más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos, donde se originaron, si bien puede utilizarse con resultados igualmente satisfactorios con otros paradigmas de programación.

2 Definiciones básicas

2.1 Actores

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas, como el tiempo.

En el caso de los seres humanos se pueden ver a los actores como definiciones de rol por lo que un mismo individuo puede corresponder a uno o más Actores. Suele suceder sin embargo, que es el sistema quien va a tener interés en el tiempo. Es frecuente encontrar que nuestros sistemas deben efectuar operaciones automáticas en determinados momentos; y siendo esto un requisito funcional obvio, resulta de interés desarrollar alguna forma de capturar dicho requisito en el modelo de caso de uso final.

3 Tipos de relaciones

- Comunica (<<*communicates*>>): Relación (asociación) entre un actor y un caso de uso que denota la participación del actor en dicho caso de uso.
- Usa (<<*uses*>>) (o <<*include*>> en la nueva versión de UML): Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro.
- Extiende (<<*extends*>>): Relación de dependencia entre dos casos de uso que denota que un caso de

uso es una especialización de otro. Por ejemplo, podría tenerse un caso de uso que extienda la forma de pedir azúcar, para que permita escoger el tipo de azúcar (normal, dietético o moreno) y además la cantidad en las unidades adecuadas (cucharadas o bolsas).

Se utiliza una relación de tipo <<extends>> entre casos de uso cuando nos encontramos con un caso de uso similar a otro pero que hace algo más que éste (variante). Por contra, utilizaremos una relación tipo <<uses>> cuando nos encontramos con una parte de comportamiento similar en dos casos de uso y no queremos repetir la descripción de dicho comportamiento común.

En una relación <<extends>>, un actor que lleve a cabo el caso de uso base puede realizar o no sus extensiones. Mientras, en una relación <<include>> el actor que realiza el caso de uso base también realiza el caso de uso incluido.

En general utilizaremos <<extends>> cuando se presenta una variación del comportamiento normal, y <<include>> cuando se repite un comportamiento en dos casos de uso y queremos evitar dicha repetición.

Por último en un diagrama de casos de uso, además de las relaciones entre casos de uso y actor (asociaciones) y las dependencias entre casos de uso (<<include>> y <<extends>>), pueden existir relaciones de herencia ya sea entre casos de uso o entre actores.

Llamamos modelo de casos de uso a la combinación de casos de uso y sus correspondientes diagramas. Los modelos de casos de uso se suelen acompañar por un glosario que describe la terminología utilizada. El glosario y el modelo de casos de uso son importantes puntos de partida para el desarrollo de los diagramas de clases.

Por último se debe tener en cuenta, que aunque cada caso de uso puede llevar a diferentes realizaciones, es importante reflejar en cada representación el motivo que nos ha llevado a descartarla, si es el caso.

Pasos para la Definición de un Caso de Uso:

- ID
- NOMBRE
- REFERENCIAS CRUZADAS
- CREADO POR
- ULTIMA ACTUALIZACIÓN POR
- FECHA DE CREACIÓN
- FECHA DE ULTIMA ACTUALIZACIÓN
- ACTORES
- DESCRIPCIÓN
- TRIGGER
- PRE-CONDICIÓN
- POST-CONDICIÓN
- FLUJO NORMAL
- FLUJOS ALTERNATIVOS
- INCLUDES
- FRECUENCIA DE USO
- REGLAS DE NEGOCIO
- REQUERIMIENTOS ESPECIALES
- NOTAS Y ASUNTO

4 Normas de aplicación

Los casos de uso evitan típicamente el lenguaje técnico, prefiriendo la lengua del usuario final o del experto del campo del saber al que se va a aplicar. Los casos del uso son a menudo elaborados en colaboración por los analistas de requerimientos y los clientes.

Cada caso de uso se centra en describir cómo alcanzar una única meta o tarea. Desde una perspectiva tradicional de la ingeniería de software, un caso de uso describe una característica del sistema. Para la mayoría de proyectos de software, esto significa que quizás a veces es necesario especificar decenas o centenares de casos de uso para definir completamente el nuevo sistema. El grado de la formalidad de un proyecto particular del software y de la etapa del proyecto influenciará el nivel del detalle requerido en cada caso de uso.

Los casos de uso pretenden ser herramientas simples para describir el comportamiento del software o de los sistemas. Un caso de uso contiene una descripción textual de todas las maneras que los actores previstos podrían trabajar con el software o el sistema. Los casos de uso no describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican cómo se implementará. Simplemente muestran los pasos que el actor sigue para realizar una operación.

Un caso de uso debe:

- Describir una tarea del negocio que sirva a una meta de negocio.
- Tener un nivel apropiado del detalle.
- Ser bastante sencillo como que un desarrollador lo elabore en un único lanzamiento.

Situaciones que pueden darse:

- Un actor se comunica con un caso de uso (si se trata de un actor primario la comunicación la iniciará el actor, en cambio si es secundario, el sistema será el que inicie la comunicación).

- Un caso de uso extiende otro caso de uso.
- Un caso de uso utiliza otro caso de uso.

5 Ventajas

La técnica de caso de uso tiene éxito en sistemas interactivos, ya que expresa la intención que tiene el actor (su usuario) al hacer uso del sistema.

Como técnica de extracción de requerimiento permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema, evitando que la gente especializada en informática dirija la funcionalidad del nuevo sistema basándose solamente en criterios tecnológicos.

A su vez, durante la extracción (*elicitation* en inglés), el analista se concentra en las tareas centrales del usuario describiendo por lo tanto los casos de uso que mayor valor aportan al negocio. Esto facilita luego la priorización del requerimiento.

Aunque comúnmente se asocian a la fase de Test de una aplicación, esta idea es errónea, y su uso se extiende mayormente a las primeras fases de un desarrollo.

6 Limitaciones

Los casos de uso pueden ser útiles para establecer requisitos de comportamiento, pero no establecen completamente los requisitos funcionales ni permiten determinar los requisitos no funcionales. Los casos de uso deben complementarse con información adicional como reglas de negocio, requisitos no funcionales, diccionario de datos que complementen los requerimientos del sistema. Sin embargo la ingeniería del funcionamiento especifica que cada caso crítico del uso debe tener un requisito no funcional centrado en el funcionamiento asociado.

7 Véase también

- Requisito funcional
- Diagrama de casos de uso
- Puntos de casos de uso
- Caso de abuso

8 Enlaces externos

- Precise Use Cases
- Use-Case Modeling

9 Herramientas de administración de requerimientos

- Open source requirement management tool

10 Referencias

[1] Jacobson, I., P. Jonsson, M. Christerson and G. Overgaard, Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.

11 Origen del texto y las imágenes, colaboradores y licencias

11.1 Texto

- **Caso de uso** *Fuente:* https://es.wikipedia.org/wiki/Caso_de_uso?oldid=89170535 *Colaboradores:* Benjavalero, Emijrp, Magister Mathematicae, Superzerocool, Yrbot, GermanX, Kelovy, The Photographer, Gothmog, Banfield, Pviojo, Cheveri, Lasneyx, Cad, BOTpolicia, CEM-bot, Alejandrosilvestri, Pablotortorella, -jem-, Jfrank~eswiki, Antur, Thijs!bot, Esoya, Alvaro qc, Tyrannosaurusreflex, Nightwish, Góngora, Migp~eswiki, Mansoncc, Iulius1973, ZrzlKing, Pólux, Manuel Trujillo Berges, VolkovBot, Snakeyes, Matdrodes, DJ Nietzsche, Tatvs, Muro Bot, Bucho, SieBot, Manwë, Pascow, Belb, Tirithel, Jarisleif, Carlosdemoron, Nicop, Farisori, Veon, Qwertymith, Luis Arenal Kuster, Leonpolanco, Poco a poco, Hernaldo, Ravave, AVBOT, LucienBOT, Vgrunberg, Diegusjaimes, Arjuno3, Andreasmperu, Jkbw, Rubinbot, Dreitmen, Botarel, Vihuarar, Hprmedina, Encvirtual, HUBOT, Nachetex, PatruBOT, EmausBot, Jorge123321, Grillitus, Khiari, MerlIwBot, KLBot2, Thehelpfulbot, Serchkat, Pccoronado, Invadibot, Justincheng12345-bot, Balles2601, Jarould, Matiia, Crystallizedcarbon, David Condrey, Gargola 123, Esanhueza y Anónimos: 156

11.2 Imágenes

- **Archivo:Commons-emblem-question_book_orange.svg** *Fuente:* https://upload.wikimedia.org/wikipedia/commons/1/1f/Commons-emblem-question_book_orange.svg *Licencia:* CC BY-SA 3.0 *Colaboradores:* + *Artista original:* GNOME icon artists, Jorge 2701
- **Archivo:Notacion_Caso_de_Uso.svg** *Fuente:* https://upload.wikimedia.org/wikipedia/commons/9/9e/Notacion_Caso_de_Uso.svg *Licencia:* CC BY-SA 3.0 *Colaboradores:*
- Notacion_Caso_de_Uso.png *Artista original:* Notacion_Caso_de_Uso.png: Wilfredo R. Rodriguez H.
- **Archivo:Use_case_restaurant_model.svg** *Fuente:* https://upload.wikimedia.org/wikipedia/commons/1/1d/Use_case_restaurant_model.svg *Licencia:* CC BY-SA 3.0 *Colaboradores:* Own work, redrawn of w:File:Restaurant Model.png *Artista original:* Kishorekumar 62 (redrawn by Marcel Douwe Dekker)

11.3 Licencia del contenido

- Creative Commons Attribution-Share Alike 3.0