Apuntes de Planificación, gestión y desarrollo de proyectos

Máster en Sistemas Electrónicos para Entornos Inteligentes

Antonio Bandera Rubio
Joaquín Ballesteros Gómez

V3. 2017

Sección 1

Introducción a la gestión de proyectos

Objetivos

En esta sección aprenderás:

- Por qué estudiar gestión de proyectos.
- Que es un proyecto y en que se diferencia de un proceso.
- Las cinco etapas del enfoque tradicional basado en fases.
- Qué vamos a estudiar en esta asignatura.

La historia de Antonio

A continuación vamos a narrar ***La Historia de Antonio.*** Esta historia es inventada y cualquier parecido con la realidad ¿Es pura coincidencia?

Antonio, El Empresario

Antonio era un ingeniero electrónico recién acabado. Su pasión por la electrónica le había hecho superar brillantemente todas las asignaturas de la carrera y se sentía capacitado para grandes cosas en su campo. Con ese buen currículum, Antonio comenzó la búsqueda de un trabajo. Aunque en algún momento se planteó la creación de su propia empresa, pensó que aún no tenía la experiencia necesaria y optó por buscar como asalariado.

La primera sorpresa desagradable fue la dificultad para encontrar empresas a las que ofrecerse que se dedicaran al diseño y desarrollo electrónico. Apenas encontró algunas pocas, y éstas no necesitaban un nuevo ingeniero. Comentando este problema a su amigo que trabajaba como mecánico en una fábrica de plásticos, éste le comentó la falta que le hacía a su empresa un sistema de monitorización de procesos integrado, que midiera una serie de parámetros de su fábrica y proporcionara una información integral para el control. Los sistemas de medida de que le hablaba le eran muy familiares a Antonio, como aquellos sensores que vio en Instrumentación Electrónica. Claro que esa empresa necesitaba ese proyecto concreto, no tener en plantilla un ingeniero electrónico con el que después no sabría qué hacer.

Así fue que Antonio se animó a crear su propia empresa de diseño, no sin antes cerciorarse de que había en su misma ciudad algunas empresas más que parecían interesadas en trabajos similares. Obviaremos en esta historia las 6 semanas que tuvo que invertir en preparar un plan de negocio que presentar a su banco y a amigos con algo de dinero. Era una idea prometedora, consiguió la financiación que quería y fundó su empresa. Ya tenía su primer trabajo, la empresa en que trabajaba su amigo, quien lo recomendó.

El cliente

En la primera reunión, el jefe de la empresa le explicó lo que quería y le pidió un presupuesto y un compromiso de tiempo antes de firmar nada. Antonio salió algo hundido de aquella reunión, porque su cliente sabía mucho de los procesos de fabricación de plásticos, pero él no tenía la menor idea y apenas se había enterado de lo que le habían contado.

Porque no cantara mucho su novatez, apenas se había atrevido a preguntar, sólo a tomar las notas que pudo:

- En casi todas las máquinas, el cliente quería conocer la temperatura de los plásticos y la velocidad de los rollos.
- También quería detectar de forma automática posibles grumos o malformaciones en la película de plástico circulante.
- En alguna máquina necesitaba saber la presión del aire para determinar si estaba dentro de tolerancias.
- Y toda la información quería saberla en tiempo real desde su PC en la oficina, así como poder actuar bien sobre la refrigeración, bien parando las máquinas o tal vez tocándolas directamente si no había otra opción.

Lo primero que hizo Antonio fue ponerse a buscar modelos y precios de los sensores y actuadores que necesitaba. La detección de grumos implicaba reconocimiento de imágenes, lo que podía significar un proyecto separado por sí mismo y decidió proponer al cliente dejarlo para una segunda fase. Contó cuántos sensores necesitaba de cada tipo y los metros de cable para establecer un bus hasta el PC. Estimó a ojo las horas que creía necesarias para hacer un programa de control que presentara los datos y valoró sus horas de trabajo como para ganar 2000 €/mes con un horario normal. Le añadió un 20% adicional tanto al tiempo como al coste, y le presentó el presupuesto al cliente.

Su cálculo era de 3 meses, pero el cliente lo necesitaba en mes y medio porque tenía una visita de un importante cliente suyo y quería mostrarle la excelencia de su fábrica. El precio sí le pareció bien. Su amigo le comentó después que si el jefe no le discutió el precio, tal vez se había pasado de barato.

Los socios

Antonio recurrió a otros dos amigos, Juan y Luis, ingenieros electrónicos sin trabajo por el momento, para formar parte de la empresa. Pensó que entre tres sí que podrían hacer el trabajo en el tiempo que quería el cliente, aunque no sabía hacer un cálculo más preciso de ese tiempo. Incrementó un poco el presupuesto por la reducción de tiempo y volvió al cliente. Esta vez le firmó el pedido, no sin antes quejarse un poco por el aumento de precio. El documento firmado tenía el presupuesto con la lista de materiales considerados y el plazo de finalización que exigía el cliente.

A partir de ese momento, lo primero que intentaron fue enterarse mejor de cómo funcionaba la fábrica del cliente. Los tres se fueron durante un par de días a preguntar todos los detalles de funcionamiento. Ante la abundancia de detalles que iban recopilando, tuvieron que añadir un día más que no habían previsto, para que los tres pudieran enterarse de todo lo relevante (confiaron en que la estimación del tiempo no fuera crítica en 1 día). Al cuarto día, se sentaron en el despacho que habían alquilado para analizar la información, junto con las ideas iniciales en que se basó el presupuesto. Ya aparecieron algunos problemas.

Despistes previos

En primer lugar, para los sensores de temperatura había previsto termómetros digitales con termopar tipo k que tuvieran conexión a Profibus DC-v0, un bus de campo que conocían para aplicaciones en entornos industriales. Sin embargo, los termómetros y termopares que había evaluado inicialmente no eran aptos para la máquina extrusora que presentaba temperaturas de varios cientos de grados. En estos casos hay que seleccionar otro tipo de termopares y termómetros que valen un 30% más caros. Como este problema era sólo en 3 máquinas, no lo consideraron crítico.

El segundo problema es que no habían pensado que el PC de la oficina podría requerir algún hardware adicional para conectarse con los periféricos. Una tarjeta de adaptación al Profibus eran 60 € más. Tampoco era muy grave, aunque tenían que pedirle al cliente que ampliara el disco duro pues apenas le quedaba sitio.

El tercer problema era financiero. Había que ir comprando el material, pero no había previsto que no se cobraría hasta el final, y su flamante empresa no tenía fondos para adelantar el dinero.

El cuarto problema era que Antonio había dado por hecho en su presupuesto que la instalación de cables y sensores la harían los técnicos de mantenimiento del cliente. Pero en la visita los habían conocido y, muy amablemente, les habían dejado claro que todo el proyecto era cosa de ellos, aunque podrían preguntarles las dudas que tuvieran. Este coste podría ser aún mayor, por lo que se decidieron a tratarlo con el cliente (el jefe). Éste se sorprendió por la petición.

- ¿Pretendéis llenar de cables mi fábrica? Creía que estaba claro que todo esto se haría de forma inalámbrica. No puedo permitirme levantar la fábrica para meter cables ahora y a toda prisa.

Obviamente sí se podía hacer de forma inalámbrica, pero no era eso lo previsto y presupuestado. Mirando el presupuesto que aprobaron, ciertamente que no estaba nada claro que requería toda una instalación cableada, aunque había puesto una partida para cable. La instalación inalámbrica en un entorno tan ruidoso desde el punto de vista electromagnético, podía presentar algunos problemas y requería equipos inalámbricos muy robustos. La diferencia volvía a amentar el coste (y a disminuir el margen de beneficio, si es que al final había alguno).

Durante los dos días siguientes, Antonio desplegó sus mejores dotes de negociador, pero con todo en contra ya que había un contrato firmado y lo que pedía no estaba claro en ese contrato. Finalmente consiguió que el cliente le adelantara el 20% del presupuesto, pero cualquier incremento de coste debería asumirlo Antonio. Era su primer trabajo y tuvo que aceptarlo. Pero ya había perdido una semana del tiempo disponible. Mientras tanto, Juan y Luis no sabían muy bien qué hacer.

Por fin, a desarrollar

Al comienzo de la siguiente semana ya decidieron empezar a trabajar, repartiéndose el trabajo. Cada uno tendría que diseñar e implementar un trozo. Sin embargo, ese reparto no fue inmediato. Juan y Luis querían quedarse ambos con la parte Software del PC, alegando su afición especial. Después de hora y media discutiendo, Antonio tuvo que ejercer de jefe y asignó el SW a Luis, los sensores y actuadores sobre temperatura a Juan y el resto, incluidas las compras, las llevaría él mismo. Decidir que se usaría WIFI de 54 Mb fue algo más rápido, aunque no se habló de nada más de las capas superiores del protocolo. Y los tres se pusieron a trabajar.

Antonio decidió lanzar las compras de inmediato, intuyendo que podía generar retrasos el no tenerlas. El problema era decidirse entre las distintas opciones del mercado. La respuesta le pareció clara: la más económica. Ya sabemos que los productos eran todos algo más caros que los presupuestados, así que convenía minimizar esos sobrecostes. No obstante, necesitaba que Juan le confirmara exactamente qué necesitaba, al igual que Luis. Cinco días después ya tenía hechos todos los pedidos y los proveedores le daban de plazo de entrega entre 5 y 10 días. Pero pasaron los 10 días y el router WIFI con array de 2 antenas para mejor captar las señales en ese entorno tan adverso, aún no había llegado. El proveedor, que le cobró por adelantado el 20% de su precio, le dijo que tardaría una semana más. Antonio estaba muy enfadado, pero leyendo el pedido de router, comprobó que no figuraba el plazo de entrega (se lo habían dicho de viva voz), por lo que no disponía de ningún recurso legal para reclamar. Obviamente, no volvería a comprar a ese proveedor, pero esta vez no había alternativa.

Luis había empezado a desarrollar software. Decidió que la aplicación resultaría más sencilla y elegante si se desarrollaba sobre un entorno XHX, óptimo para gestionar comunicaciones y facilitar un bonito interfaz de usuario. Cuando a las 3 semanas fue a hacer sus primeras pruebas en el PC del cliente, llevó su copia pirata de XHX para que el programa pudiera funcionar, pero el cliente le dijo que, por política de empresa, tenían absolutamente prohibido instalar ningún SW pirata. De nuevo se le cayó el mundo (los euros) encima a Luis, que se veía en la tesitura de comprarle una licencia al cliente o reescribir todo el trabajo realizado en otro lenguaje gratuito o autoejecutable. Por problemas económicos, fue la segunda opción la escogida lo que implicó un retraso de 5 días en el trabajo de Luis.

Tanto Juan como Antonio se encontraron con unos sensores y actuadores que incorporaban un microcontrolador para gestionar las comunicaciones. Habían de ser adecuadamente configurados y programados. De hecho, cada uno había hecho ya un intento de programación en el lenguaje que mejor conocía, cuando se reunieron a hablar del tema. Se dieron cuenta de que lo que tenían que hacer era casi igual en su mayor parte, por lo que decidieron que el SW que ya estaba más elaborado, hecho por Juan, lo usarían ambos adaptándolo a sus equipos. Desgraciadamente, Juan no tenía escrita ninguna descripción de su SW, y el código carecía de comentarios. Antonio perdió 3 días en familiarizarse con el lenguaje usado por Juan y otros 3 días sólo para enterarse de cómo iba ese SW y poderlo adaptar.

Donde dije diego...

Cuando Luis consiguió empezar a probar su SW en el PC del cliente, éste vio lo que presentaba:

- La interfaz es bonita, pero yo quiero que me presente los datos cada 5 minutos y los guarde junto con la hora en que se tomaron y el punto al que pertenecen.

- Sólo habíamos hablado de presentar los últimos datos y avisar en caso de alarma - dijo Luis.

Pero como el cliente siempre lleva razón (¡No habían acordado nada al respecto!), Luis se vio forzado a modificar el SW con una nueva gestión de datos no prevista. 3 días más de retraso.

En la reunión de seguimiento que tuvieron a las 3 semanas del comienzo, sólo se hablaba de retrasos y sobrecostes. Tan sólo Juan parecía que lo tenía todo controlado para empezar a instalar el Hardware. Todo, claro está, excepto que él nunca había instalado HW en un entorno real. Algún problema más empezó a vislumbrarse. Luis había probado las comunicaciones con el router WIFI y otro PC, pero no con los sensores. También había previsto la funcionalidad básica acordada con el cliente, pero resultó que si se modificaba el nº de sensores o actuadores, su posición o la máquina sobre la que operaban, había que cambiar parámetros en el SW y re-compilarlo. No había previsto que el cliente también tendría que realizar esas operaciones en el futuro. Nuevos retrasos. En esa misma reunión decidieron unos formatos específicos para la comunicación de los datos entre ellos.

Más trabajo

Con la ropa sucia, demacrados por la falta de sueño y algo hambrientos, en la fecha prevista entregaron el producto al cliente, esperando ansiosamente la recompensa económica. El cliente, razonablemente contento porque aquello parecía funcionar, les pidió para terminar los manuales de uso y de mantenimiento del sistema... Se habían olvidado de los posibles y previsibles documentos necesarios. Para que el cliente pudiera lucirse con su correspondiente cliente, Luis tuvo que quedarse otro día más para manejar el sistema y responder a posibles preguntas, mientras otro operario de la fábrica aprendía. En esta ocasión, y dado que los manuales no aparecían en el contrato, Antonio consiguió una bonificación extra del 5% del presupuesto, incluyendo la formación del operario. Y, por supuesto, el IVA.

Los problemas no terminaron exactamente ahí pues dos días después, el cliente les llamó para quejarse de que lo que describían esos manuales no coincidía con lo que aparecía en su ordenador y, además, no se entendía bien. En una última reunión de urgencia entre los tres diseñadores, averiguaron que Luis no había escrito realmente un manual (no estaba previsto), sino que había recopilado sus notas particulares a toda prisa pegándolas. Por eso no eran fáciles de entender para una tercera persona. Y esas notas las había escrito antes del último cambio solicitado por el cliente. Tenía algunas actualizaciones de esas notas posteriores, pero no sabía dónde estaban. No hubo más remedio que pedirle al cliente un par de días más para hacerle un manual en condiciones.

Epílogo

Tras pagarle a Juan y Luis lo acordado más algunas horas extra, los gastos de la oficina alquilada, las compras, el seguro de autónomo y el porcentaje de impuestos, a Antonio le quedó un sueldo mensual de 200 €. Por fortuna, era un chico espabilado y mejoró mucho su rentabilidad en siguientes proyectos. Supo aprender de los errores.

¿Porqué estudiar gestión de proyectos?

La **historia de Antonio** pone de manifiesto las múltiples formas en las que podemos realizar una mala gestión de un proyecto y los efectos que esto tiene.

Un cliente siempre va a pedir más, en el menor tiempo posible y gastando lo mínimo. Por ello, es necesario especificar detalladamente con el cliente que es lo que se va a hacer y lo que no, cuando se va a entregar y como y estimar de la forma más precisa una planificación general de las tareas a llevar a cabo.

Una vez finalizada esta fase y para aumentar nuestros beneficios y asegurar el éxito, tenemos que invertir tiempo diseñando el sistema, planificando que tareas son necesarias llevar a cabo y quien se va a encargar de ellas. Esta fase de estudio previa al desarrollo del sistema es vital para tener éxito en el proyecto.

Los conocimientos sobre la gestión de proyectos se han ido agrupando en estándares y normas recogiendo conjuntos de «buenas prácticas» que han demostrado su influencia en proyectos exitosos. Sin embargo, esto no nos garantiza el éxito...

Francia gasta 15 billones de euros en nuevos trenes que sobrepasan en ancho para entrar en 1.300 estaciones regionales (Mayo 2014)

... pero nos facilita el camino, por tanto, ¿Por qué volver a tropezar en la misma piedra?

¿Qué es un proyecto? 1/2

Estas son las definiciones que dan los estándares y normas más usados en la actualidad sobre que es un proyecto:

- *Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único* **(PMBOK v5)**
- *Un proyecto es una organización temporal que se crea con el propósito de entregar uno o más productos comerciales según un Business Case convenido* **(PRINCE2)**
- *Un proyecto es una operación restringida en tiempo y coste para realizar un conjunto de entregas definidas (el ámbito de las mismas cumple todos los objetivos del proyecto) en base a normas de calidad y requisitos* **(ICB 3.0)**
- *Es un conjunto único de procesos que consta de actividades coordinadas y controladas, con fechas de inicio y fin, que se llevan a cabo para lograr los objetivos del proyecto. El logro de los objetivos del proyecto requiere la realización de entregables que satisfagan requisitos específicos. Además un proyecto puede estar sujeto a múltiples restricciones* **(ISO 21500)**

¿Qué es un proyecto? 2/2

Dependiendo de la metodología que se siga, la nomenclatura puede variar, pero podemos resumir la definición de proyecto como:

- **Un proyecto es una secuencia de actividades únicas, complejas y conectadas que tienen un objetivo o propósito y que deben ser completadas en un tiempo específico, dentro del presupuesto, y de acuerdo con la especificación dada para producir un producto/entregable/servicio único.**

El concepto de **actividad única** se refiere a que un proyecto nunca transcurre igual dos veces y nunca volverá a ocurrir bajo las mismas circunstancias.

Las actividades son **complejas**, no son actos simples y repetitivos como cargar una batería o dar un punto de soldadura.

Las actividades están **conectadas** porque debe existir una relación lógica o técnica entre pares de las actividades. Para ser completado el proyecto, las actividades deben seguir un orden, la salida de una actividad es la entrada a otra.

Los proyectos deben tener un único **objetivo o propósito**. Los proyectos complejos o grandes pueden dividirse en varios subproyectos, cada uno de los cuales es un proyecto con su propio objetivo o propósito.

Ajustarse al **presupuesto** puede suponer pérdidas a la empresa si no se realiza una buena planificación. En un proyecto los recursos sin limitados (personal, dinero, puestos de trabajo, maquinaria, etc), estos se deben asignar y manejar para maximizar el beneficio, considerando la fecha de entrega (deadline) del proyecto, la capacidad de personal y los posibles contratiempos que podemos encontrar.

Los proyectos tienen una fecha de finalización, deben **ser completados en un tiempo específico**. Esta fecha puede ser autoimpuesta o dada por el cliente. El proyecto se considera terminado cuando se **cumple la especificación dada**.

Fáctores que afectan a la calidad

El denominado *Iron Triangle* (Imagen 1) muestra de manera gráfica como para mantener la calidad de un proyecto existe una relación entre*:*

- El ámbito del proyecto: impuesto por el objetivo del proyecto.
- El tiempo: fecha de finalización.
- Los recursos: personal, dinero, puestos de trabajo, maquinaria, etc.

Imagen 1 - Iron Triangle

La relación existente en el Iron Triangle nos marca que si requerimos más de uno (Ámbito, Tiempo o Recursos), debemos sacrificar parte de alguno de los otros. En http://office.microsoft.com/en-001/project-help/every-project-plan-is-a-triangle-HA001021180.aspx encontramos más información sobre *Iron Triangle*

Pregunta de Elección Múltiple

Si necesitamos reducir tiempo del proyecto manteniendo su calidad:

- ○ Debemos aumentar los recursos o quitar características o funcionalidades de nuestro proyecto.
- ○ Debemos reducir los recursos y quitar características o funcionalidades de nuestro proyecto.
- ○ Debemos reducir los recursos o aumentar las características o funcionalidades de nuestro proyecto.

Correct Option

Wrong

Wrong

Solution

1. Correct Option
2. Wrong
3. Wrong

Pregunta de Elección Múltiple

Si necesitamos reducir los recursos del proyecto manteniendo su calidad:

- Debemos reducir el tiempo o aumentar características o funcionalidades de nuestro proyecto.
- Debemos reducir el tiempo y quitar características o funcionalidades de nuestro proyecto.
- Debemos aumentar el tiempo o quitar características o funcionalidades de nuestro proyecto.

Wrong

Correct Option

Solution

1. Wrong
2. Wrong
3. Correct Option

Pregunta de Elección Múltiple

Si necesitamos ampliar el ámbito del proyecto manteniendo su calidad:

- Debemos aumentar el tiempo y los recursos.
- Debemos reducir el tiempo o los recursos.
- Debemos aumentar el tiempo o los recursos.

Wrong

Wrong

Correct Option

Solution

1. Wrong
2. Wrong
3. Correct Option

Proyecto vs Proceso

Un proceso es un conjunto definido de actividades o comportamientos realizados por personas o máquinas para alcanzar uno o más objetivos que no tiene por qué estar acotado en tiempo.

En las imágenes 2 y 3 observamos gráficamente la diferencia entre ambos. En la imagen 2 se muestra el resultado de una secuencia de actividades **únicas**, **complejas**, **conectadas** y completadas en un tiempo específico que han tenido un objetivo: crear un coche único. Sin embargo, en la Imagen 3 se muestran los resultados de un conjunto definido de actividades o comportamientos realizados por personas o máquinas que tienen un objetivo: fabricar coches en serie.

En http://bpmjourney.com/2013/07/que-es-un-proceso-y-en-que-se-diferencia-de-un-proyecto/ encontramos algunos ejemplos más sobre diferencias entre procesos y proyectos.

Imagen 2 - Proyecto

Imagen 3 - Proceso

Reflexión

Consideremos una empresa a la que le solicitan que instale sensores de humedad en las salas de Museo Pergamon para lanzar alarmas y avisar a la central si supera una cierta cota, ¿Estamos ante un proyecto o un proceso?

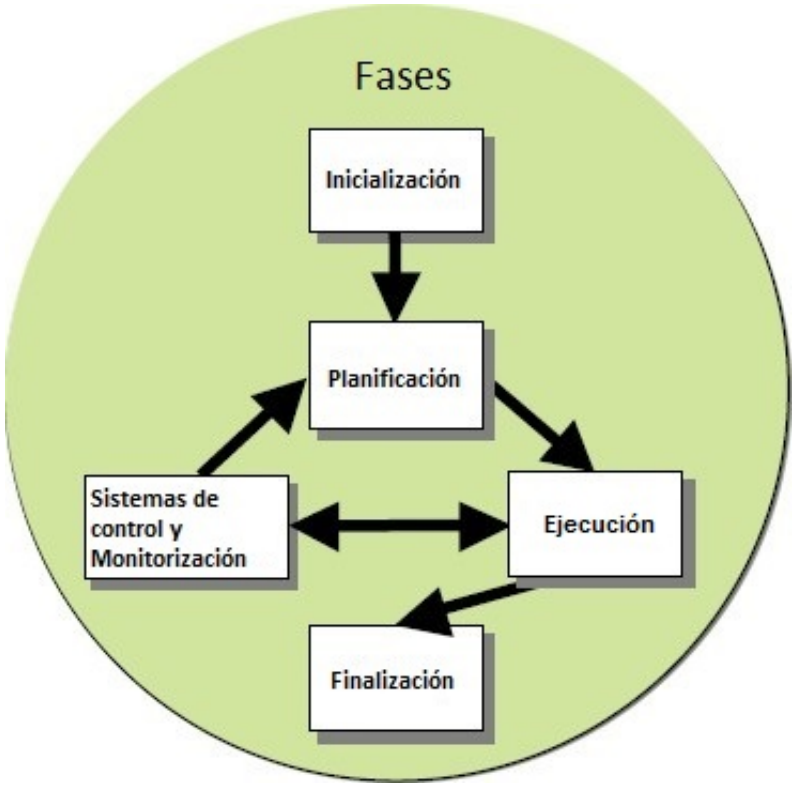
Show Feedback

Ambas soluciones son factibles, depende del tipo de empresa. Si es una empresa especializada en sensorizar músesos, tendrá ya definido el **proceso** (conjunto de actividades) que debe seguir para instalar el tipo de sensores que se le solicita. Sin embargo, podría ser una empresa dedicada a la seguidad en los músesos que acepta por primera vez este tipo de **proyecto**.

Gestión de proyectos en 5 fases

Independientemente de la metodología usada, siempre debemos tener en mente los objetivos generales del proyecto. Hoy día existe una amplia variedad de metodologías para la gestión de proyectos, la mayoría de ellas los podemos agrupar en: basadas en fases, iterativas e incrementales.

En esta asignatura vamos a seguir el enfoque tradicional, por ser el que mejor se adapta a la variedad de proyectos que podemos encontrarnos en nuestra área. Este se divide en 5 fases:



Las dos primeras fases podemos resumirlas en la respuesta a una serie de preguntas:

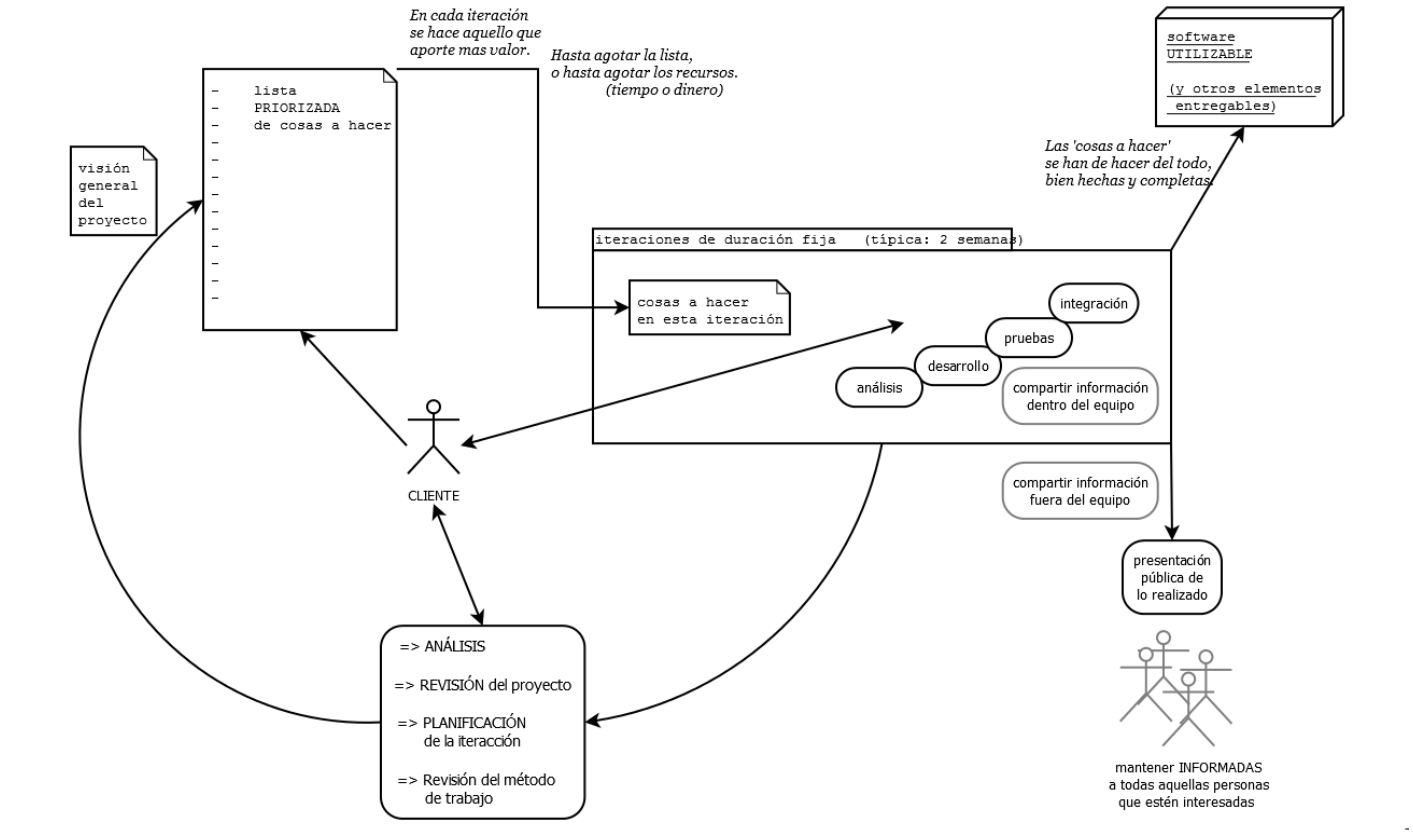
- Inicialización:
  - *¿Quien o quienes son los interesados en el proyecto?* **(Stakeholders)**
  - *¿Que quieren?* **(Objetivo del proyecto)**
  - *¿Es posible hacerlo?* **(Estudio de viabilidad)**
  - *¿Merece la pena a nuestra empresa realizar el proyecto?* **(Análisis financiero, costes vs beneficio)**
- Planificación:
  - *¿Que vamos a hacer en el proyecto?* **(Requisitos)**
  - *¿Cómo vamos a validar lo que quiere el cliente?* **(Plan de aceptación)**
  - *¿Cómo lo vamos a hacer?* **(Diseño)**
  - *¿Quien va a trabajar e el proyecto?* **(Selección de equipo)**
  - *¿Que partes va a tener nuestro proyecto?* **(Especificación de actividades)**

    - *¿Cuando se hace cada una de las partes?* **(Planificación en tiempo incluyendo entregables e hitos)**
    - *¿Cómo se manejan los imprevistos?* **(Plan de riesgos)**
    - *¿Cuanto va a costar el proyecto (una vez todo definido)?* **(Presupuesto final)**
    - *¿Tenemos vía libre para continuar?* **(Aprobación del proyecto)**
- Ejecución
    - En esta fase se desarrollan y documentan los entregables o *"deliverables"* (partes de nuestro proyecto) siguiendo la especificación y diseño establecida en la fase previa para satisfacer los requisitos impuestos por el cliente. Estos entregables deben cumplir con las normas de calidad establecidas en el proyecto.
    - En esta fase es primordial gestión del personal y la retroalimentación de los stakeholders sobre el curso del proyecto.
- Sistemas de control y monitorización
    - *¿Dónde estamos?* **(Controlamos que actividades ya han finalizado)**
    - *¿Dónde deberíamos estar?* **(Monitorización de los costes, recursos y actividades para obtener una comparación con lo acordado)**
    - *¿Cómo podemos volver sobre la planificación especificada?* **(Identificar las acciones correctivas para hacer frente a los problemas y riesgos adecuadamente)**

Gestión de proyectos alternativa: Metodologías Ágiles

Este tipo de metodologías son aplicadas generalmente sobre proyectos software (ver su manifiesto). Es una metodología iterativa e incremental en la cual los requisitos del sistema y las soluciones van evolucionando. Existe una gran variedad de metodologías ágiles, la más usada por versatilidad es Scrum. Todas esas basan su funcionamiento en generar en cada iteración (período corto de tiempo, de 1 a 4 semanas) una demo o prototipo ("sin errores"). Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto. El proyecto finaliza cuando no quedan requisitos por implementar o no hay dinero o tiempo.

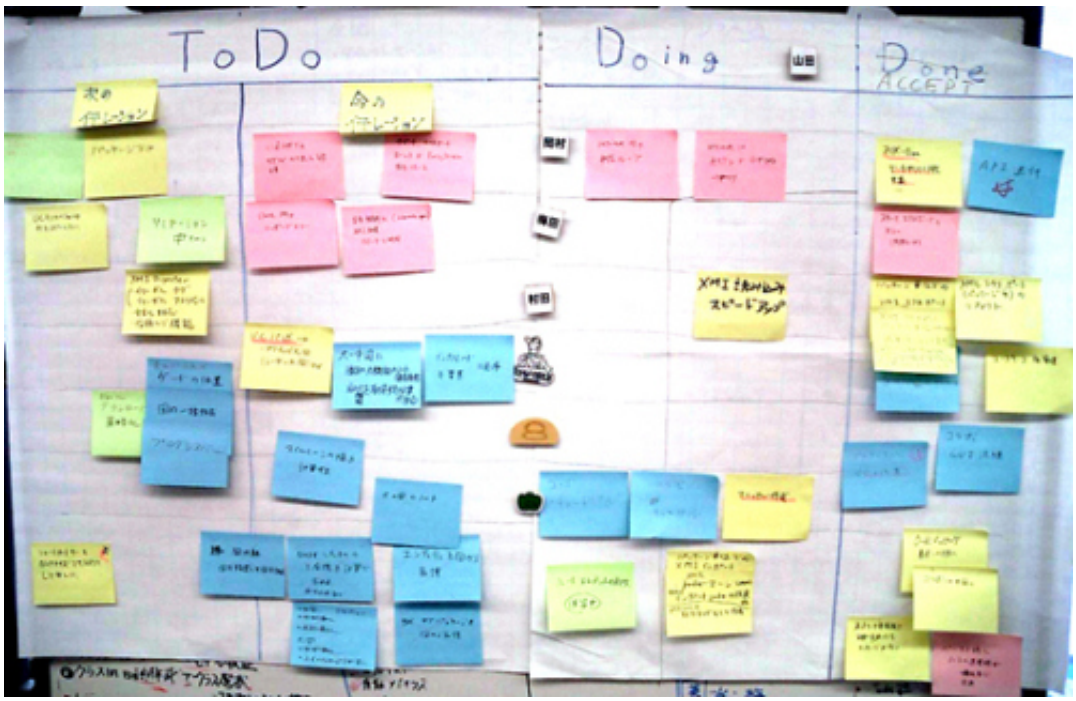
Su gestión es simple como podemos observar en la siguiente imagen:



Fuente Wikipedia (Autor Benzirpi)

Las metodologías ágiles son útiles en proyectos o subproyectos en los que los objetivos no están del todo claros o se requiere poder responder rápidamente a cambios en los requisitos. La gran desventaja es que son difíciles de predecir, desde los plazos hasta los costes. Otra desventaja que limita su uso, es la necesidad de tener al cliente trabajando en el proyecto, no siempre es posible o deseable.

La siguiente imagen refleja la filosofía de una metodología ágil:


ToDo
Doing
Done
ACCEPT

Gestión de proyectos en MSEEI

En esta asignatura nos vamos a centrar en **la fase de estudio** del enfoque tradicional aplicado a vuestro TFM haciendo uso del lenguaje de modelado SysML. El resto de fases serán documentadas y tendrán carácter informativo.

Durante la asignatura vais a documentar los requisitos de vuestro TFM incluyendo sus pruebas de verificación, se va a generar el plan de aceptación -pruebas que se van a realizar una vez esté terminado para que el cliente/tutor valide que el proyecto hace lo que deseaba - y se va a realizar el diseño a alto nivel (la arquitectura del sistema).

Una vez finalizada la asignatura, tendréis la especificación y el diseño de vuestro TFM y podréis pasar a la **fase de producción**.

Sección 2

Ingeniería de requisitos

Objetivos

En esta sección aprenderás:

- Qué es la ingeniería de requisitos y los problemas que intenta solventar.
- Qué es un requisito, cuales son las características deseables en él y una posible clasificación.
- Cómo se realiza la educción de requisitos de un proyecto.
- Cómo generar un diagrama de requisitos en SysML.

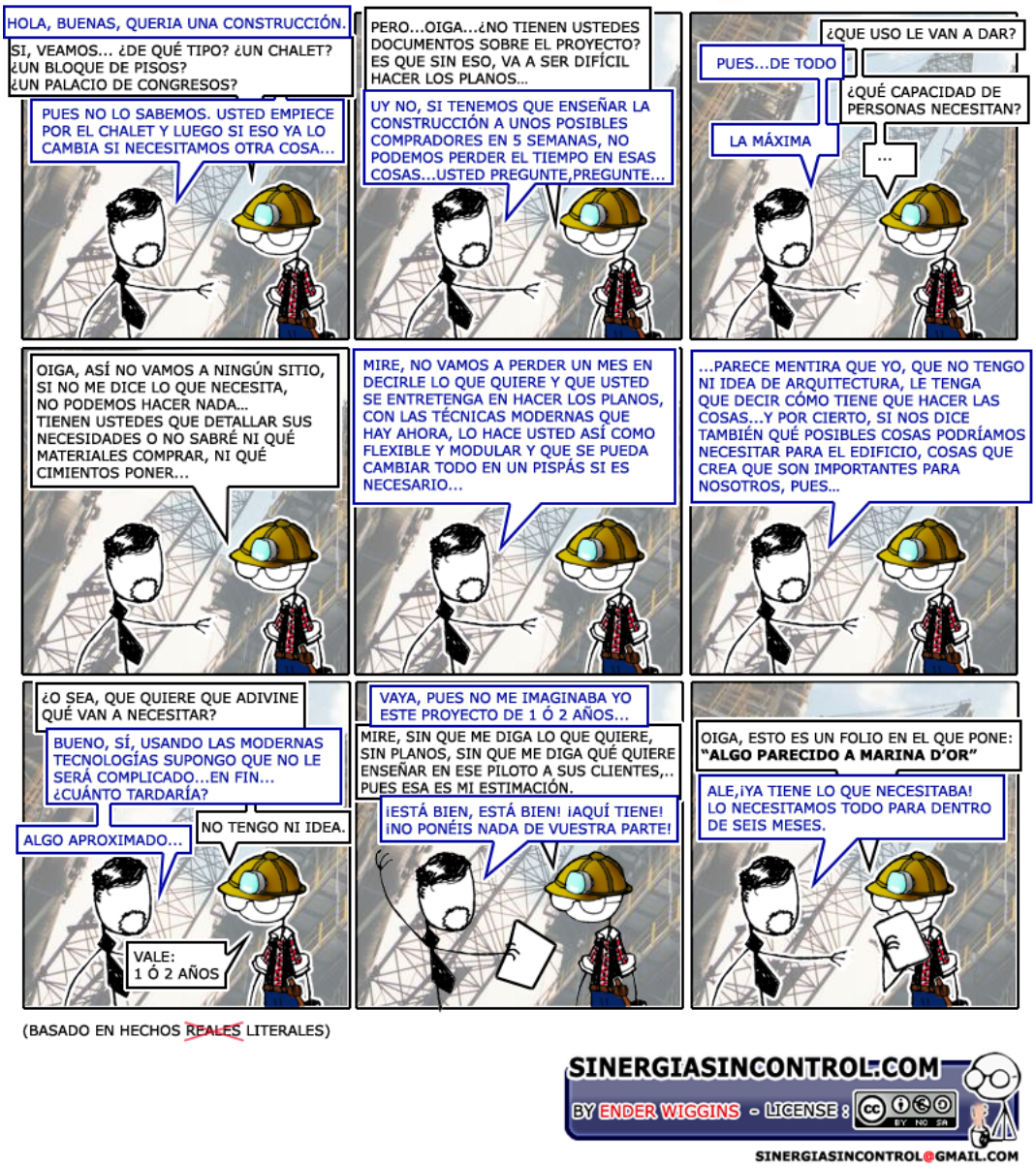
Ingeniería de requisitos

La **Ingeniería de Requisitos** se encarga de determinar las necesidades o condiciones a cumplir por un producto/entregable/servicio único.

Las actividades a realizar en la ingeniería de requisitos se resumen en:

- **Educción de los requisitos de usuario:** Se realiza con entrevistas con los *stakeholders* para saber cuáles son sus expectativas y se hace uso de la documentación de procesos de negocio para completarla.
- **Análisis de requisitos:** Determinar si los requisitos actuales cumplen las características de "un requisito bien especificado" y resolver posibles conflictos entre ellos.
- **Documentación de requisitos:** Estos pueden ser documentados en diversas formas, por lo general se define una lista resumida y se pueden incluir documentos en lenguaje natural, casos de uso, historias de usuario...
- **Verificar los requisitos:** consiste en comprobar que los requisitos han sido desarrollados acorde a lo especificado.
- **Validar los requisitos:** comprobar que los requisitos implementados cumplan para lo que inicialmente se construyó el producto.

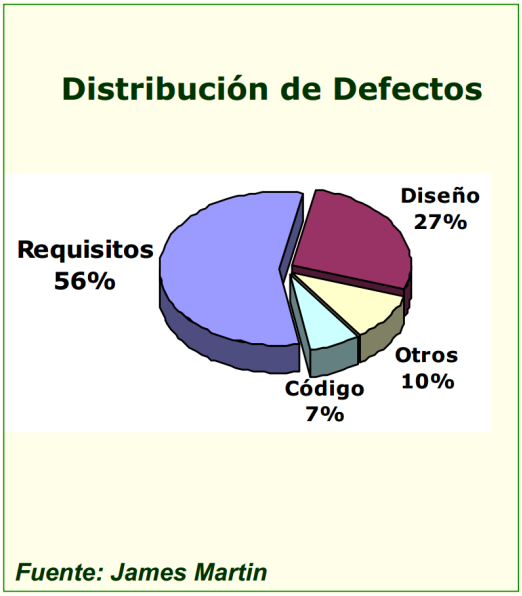
Parte de la dificultad añadida a esta etapa reside en que los *stakeholders* no necesariamente saben transmitir que quieren, pueden tener puntos de vista contradictorios y no suelen tener una visión de conjunto.

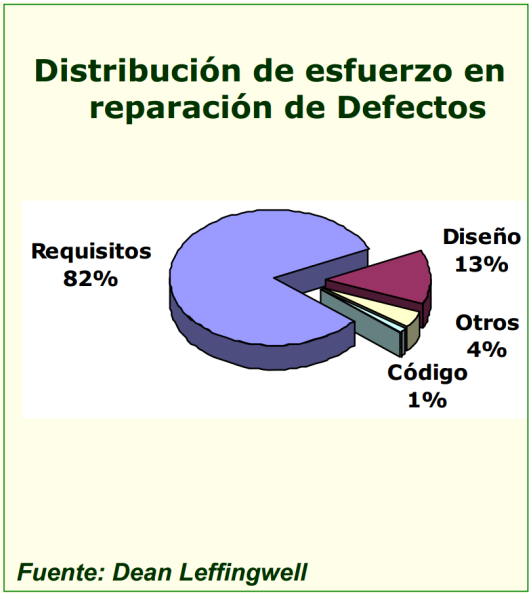


A continuación vamos a describir el concepto de requisito, características deseables en ellos y tipos que existen. En vuestro TFM partiremos de que el proyecto que vais a realizar es viable (de esto ya se ha encargado vuestro tutor), por tanto, vuestro primer trabajo comprenderá la captura, especificación y documentación de los requisitos de vuestro TFM haciendo uso del lenguaje SysML.

¿Por qué hablar de requisitos?

Los requisitos son los pilares sobre los que se sustentan las fases siguientes a la planificación. Errores en su especificación suelen provocar desecho de trabajo ya realizado, aumentando enormemente el tiempo o coste del proyecto (si queremos mantener la calidad).





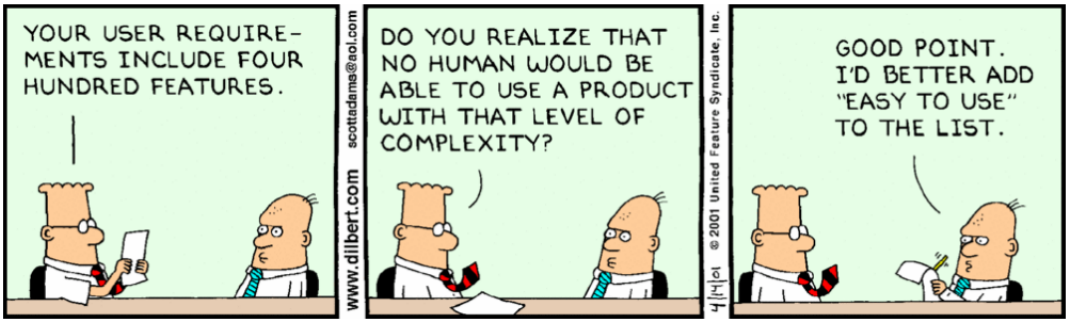
Requisitos

En IEEE Std 610.12-1990 se define un requisito como:

1. Una condición o capacidad requerida por un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe ser poseída por un sistema o componente del sistema para satisfacer un contrato, estándar, especificación, u otro documento formalmente impuesto.
3. Una representación documentada de una condición o capacidad como las descritas anteriormente.

De manera más informal, podemos definir un **requisito** como una declaración que identifica una condición o capacidad de un producto/entregable/servicio único para que tenga valor y utilidad para un *stakeholder*.

Características que debe tener un requisito bien especificado



http://dilbert.com/

Un requisito bien especificado es:

- **Completo**: Esta declarado íntegramente en su descripción y no remite a otras fuentes externas que lo explique con más detalle.
- **Consistente**: El requisito no se contradice con cualquier otro requisito y es compatible con toda la documentación externa.
- **Atómico**: No contiene conjunciones. Por ejemplo: "El sistema debe medir la temperatura, humedad y nivel de oxigeno en la caldera y controlar el acceso cuando esté encendida". Esto son claramente cuatro requisitos: "El sistema debe medir la temperatura en la caldera", "El sistema debe medir la humedad en la caldera", "El sistema debe medir el nivel de oxigeno en la caldera" y "El sistema debe controlar el acceso a la caldera cuando esté encendida"
- **Rastreable:** El requisito satisface la totalidad o parte de la necesidades de negocio impuestas por los stakeholders.
- **Actual:** El requisito no se ha quedado obsoleto por el paso del tiempo.
- **No ambiguo:** El texto debe ser claro, preciso y tener una única interpretación posible. No se debe recurrir a jerga técnica o siglas, si estas no han sido previamente definidas. No debe expresar opiniones, sólo hechos objetivos.
- **Verificable:** Se debe poder verificar con absoluta certeza, si el requisito fue satisfecho o no. Esta verificación puede lograrse mediante inspección, análisis, demostración o pruebas.

¿Que característica cumple el siguiente requisito de la Historia de Antonio?

En casi todas las máquinas, el cliente quiere conocer la temperatura de los plásticos y la velocidad de los rollos.

- ○ No ambiguo
- ○ Atómico
- ○ Rastreable

INCORRECTO: Falta mucha información por definir ¿Qué maquinas?¿Con que precisión?¿Con que frecuencia?¿Como se representa esta información?¿Dónde se muestra esta información?
INCORRECTO: No es atómico, por un lado se describe el requisito de la temperatura y por otro el de la velocidad
CORRECTO: El requisito satisface parte de la necesidades de negocio impuestas por el cliente

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta

¿Que característica cumple el siguiente requisito de la Historia de Antonio?

Toda la información quiere saberla en tiempo real desde su PC en la oficina, así como poder actuar bien sobre la refrigeración, bien parando las máquinas o tal vez tocándolas directamente si no hay otra opción.

- ○ No ambiguo
- ○ Consistente
- ○ Verificable

INCORRECTO: Faltan detalles por especificar ¿Que información se muestra en el PC? ¿Sobre que máquinas se va a poder actuar? ¿Como se va a actuar?
CORRECTO: El requisito no se contradice con cualquier otro requisito y es compatible con la inexistente documentación que tiene Antonio.
INCORRECTO: Al ser ambiguo, no es posible verificarlo ¿Cómo determinamos que la información que representamos es la necesaria para el cliente?

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto

Tipos de requisitos

La siguiente clasificación de requisitos es una de las más extendidas (BABOK Guide):

- **Requisitos de negocio**: Especificaciones a alto nivel de los objetivos o necesidades de la empresa. Describen porque el proyecto se ha iniciado, que objetivos alcanzará y que medidas se usarán para medir el éxito del mismo. Describen las necesidades de la empresa en su conjunto, no se incluyen necesidades de *stakeholder*.
- **Requisitos de *stakeholder*:** Especificaciones de medio nivel de un *stakeholder* particular o un grupo de ellos. Describen las necesidades que tienen y cómo van a interactuar con el producto/entregable/servicio único. A menudo sirve como punto intermedio entre una especificación más exhaustiva y los requisitos de negocio.
- **Requisitos de solución:** Describen las características de un producto/entregable/servicio único que cumplen con los requisitos de negocio y de *stakeholder*. Se dividen en:
  - **Requisitos funcionales**: Los requisitos funcionales establecen el comportamiento del producto/entregable/servicio único, describen los servicios (funciones) que se esperan de él.
  - **Requisitos no funcionales**: Los requisitos no funcionales son restricciones que se imponen al producto/entregable/servicio único (rendimiento, seguridad, costo, portabilidad, escalabilidad, etc).
- **Requisitos de transición:** Requisitos que especifican la transición de la situación actual de la empresa a un estado deseado futuro, pero que una vez cumplidos no serán necesarios nunca más (contratación, cambios de roles, migración de datos de un sistema a otro, etc). Se diferencian de otros tipos de requisitos por su naturaleza temporal, debido a que no pueden desarrollarse hasta que ambos productos/entregables/servicios únicos, existente y nuevo coexisten.

De los siguientes requisitos extraídos de la historia de Antonio identifica si es un requisito funcional o no funcional

El sistema medirá la temperatura de cada máquina extrusora.

- ○ Funcional
- ○ No funcional

CORRECTO: Este requisito establece una nueva funcionalidad del producto, medir la temperatura de cada máquina extrusora.
INCORRECTO: Este requisito no impone una restricción sobre el producto, describe una nueva funcionalidad del mismo, medir la temperatura de cada máquina extrusora.

Solución

1. Opción correcta
2. Incorrecto

Las comunicaciones inalámbricas respetarán la normativa vigente al respecto en España.

- ○ Funcional
- ○ No funcional

INCORRECTO: Este requisito impone una restricción sobre el producto. No es una nueva funcionalidad, es una restricción a la funcionalidad de comunicarse de manera inalámbrica.
CORRECTO: Este requisito establece una nueva restricción, toda comunicación inalámbrica debe cumplir la normativa vigente en España.

Solución

1. Incorrecto
2. Opción correcta

La precisión en la medición de la temperatura para cada una de las máquinas será siempre superior a la décima de grado.

- ○ Funcional
- ○ No funcional

INCORRECTO: Este requisito impone una restricción sobre el producto. No es una nueva funcionalidad, es una restricción a la funcionalidad de medir la temperatura.
CORRECTO: Este requisito establece una nueva restricción, nuestra precisión en la medición debe ser superior a la décima de grado.

Solución

1. Incorrecto
2. Opción correcta

El sistema permitirá actuar sobre el termostato de la refrigeración de cada maquina extrusora.

- ○ Funcional
- ○ No funcional

CORRECTO: Este requisito establece una nueva funcionalidad del producto, actuar sobre el termostato de la refrigeración de cada maquina extrusora.
INCORRECTO: Este requisito no impone una restricción sobre el producto, describe una nueva funcionalidad del mismo, actuar sobre el termostato de la refrigeración de cada maquina extrusora.

Solución

1. Opción correcta
2. Incorrecto

En el PC de la oficina se mostrarán las temperaturas de las 2 máquinas extrusoras con un retardo máximo de 0.1 s.

- ○ Funcional
- ○ No funcional

INCORRECTO: Este requisito está mal especificado. Es un requisito funcional (En el PC de la oficina se mostrarán las temperaturas de las máquinas extrusoras) y no funcional (La información sobre la temperatura de las máquinas extrusoras mostrada el PC de la oficina tendrá un retardo máximo de 0.1s) ¡Ojo al definir los requisitos!
INCORRECTO: Este requisito está mal especificado. Es un requisito funcional (En la oficina se mostrarán las temperaturas de las máquinas extrusoras) y no funcional (La información sobre la temperatura de las máquinas extrusoras mostrada el PC de la oficina tendrá un retardo máximo de 0.1s) ¡Ojo al definir los requisitos!

Solución

1. Opción correcta
2. Incorrecto

Primer entregable 1/2: Identificación de los stakeholder

Como parte del primer entregable, deberéis identificar a los *stakeholder* y usuarios potenciales de vuestro TFM usando la plantilla (*identificacion de stakeholder.doc)*. Más información aquí.

Identificar a los stakeholder, es identificar a las personas, grupos de personas u organizaciones que se ven afectadas o pueden verse afectadas por el proyecto (más información, información avanzada). En el caso de la historia de Antonio encontramos los siguientes stakeholder:

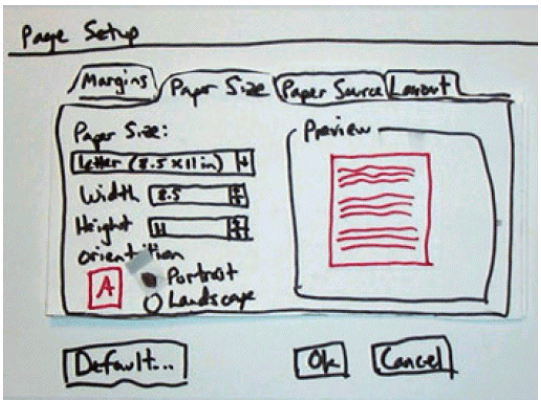
- Jefe de la Empresa de plásticos.
- Empleados de la empresa de plásticos.
- Jefe de proyecto, Antonio.
- Empleados de Antonio, Juan y Luis.

En este enlace podéis consultar el documento para la identificación de stakeholder y usuarios finales de la historia de Antonio.

Educción de los requisitos de usuario

La educción de los requisitos se ha realizado tradicionalmente con entrevistas a los *stakeholder,* para saber cuáles son sus expectativas sobre el producto/entregable/servicio único. Otros métodos que ayudan a la educción de requisitos son:

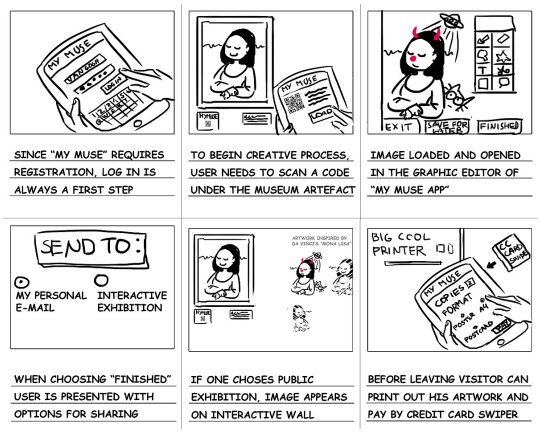
- Prototipado: Esta técnica es una de las más comunes de apoyo a las entrevistas. Consiste en realizar un prototipo (interfaz, diseño, arquitectura, etc) de como va a ser el producto/entregable/servicio único y discutir sobre el mismo.



http://www.snyderconsulting.net/

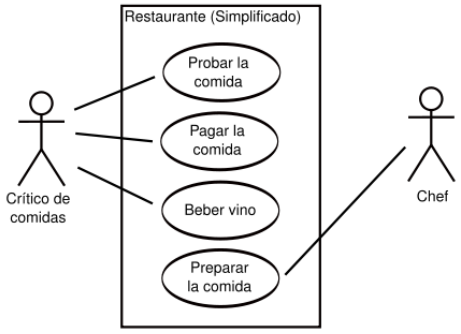
- Tormenta de ideas (*Brainstorming*): *"...La principal regla del método es aplazar el juicio, ya que en un principio toda idea es válida y ninguna debe ser rechazada... Cualquier persona del grupo puede aportar cualquier idea de cualquier índole, la cual crea conveniente para el caso tratado... Un análisis posterior explota estratégicamente la validez cualitativa de lo producido con esta técnica..."* Wikipedia.

- Guiones gráficos (*Story-boards*): La construcción de un guión gráfico significa pensar en la funcionalidad propuesta como una historia separada en una serie de pasos o acciones discretas. La componente visual hace que sea fácil de entender para obtener feedback de los *stakeholder*.



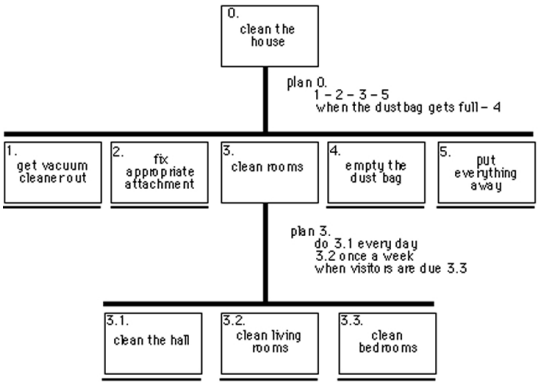
https://darjaimke.wordpress.com

- Casos de uso: Los casos de uso son básicamente guiones gráficos representados como diagramas que describen cómo funcionan los procesos discretos. Estos incluyen los actores (entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad) y describen cómo funciona la solución desde su perspectiva. Los casos de uso pueden ser más fácil de expresar para los usuarios, aunque pueden necesitar ser refinamiento posterior para obtener requisitos más específicos.



UML_diagrama_caso_de_uso [GFDL or CC-BY-SA-3.0], via Wikimedia Commons

- Hierarchical task analysis (HTA), es decir, análisis jerárquico de tareas: Es un enfoque estructurado para describir como se realiza una tarea compleja. Entre las ventajas de este método está la facilidad para comparar enfoques que solucionan una misma tarea.



Modelado jerárquico de tareas de la limpieza con una aspiradora de una casa (Fuente: HCI Book)

Educción en la historia de Antonio

En la Historia de Antonio, la educción de los requisitos consistió en una única entrevista con el jefe de la empresa en la que él obtuvo los requisitos principales del proyecto:

- En casi todas las máquinas, el cliente quería conocer la temperatura de los plásticos y la velocidad de los rollos.
- ~~También quería detectar de forma automática posibles grumos o malformaciones en la película de plástico circulante.~~
- En alguna máquina necesitaba saber la presión del aire para determinar si estaba dentro de tolerancias.
- Y toda la información quería saberla en tiempo real desde su PC en la oficina, así como poder actuar bien sobre la refrigeración, bien parando las máquinas o tal vez tocándolas directamente si no había otra opción.

¿Pudo Antonio mejorar esta educción de requisitos?

¡Si! Al contar con una interfaz que muestra toda la información, podemos hacer uso del prototipado, diseñando la interfaz a mano (con Paint, Gimp o similar). De esta forma, mostramos al stakeholder correspondiente como sería la aplicación. Esto nos ayudará a extraer más requisitos del proyecto (¡Ojo con hacerlo excesivamente realista que pueden pensar que está todo hecho!).

Supongamos que Antonio solicita una segunda reunión con el Jefe y algún operario (siempre interesa tener a varios stakeholder con diferentes roles en las reuniones) y le presenta esta interfaz para profundizar en la captura de requisitos:

Jefe - ¿Ya está todo hecho? Si que has tardado poco.

Antonio - ¡No hemos empezado aún!

Jefe - Esto tiene que estar terminado en 1 mes y medio que viene a visitar la fábrica un cliente importante.

Antonio - Lo que te muestro aquí es un diseño de la interfaz que podrías tener para ver la temperatura, la velocidad y la presión de la maquinaria. Podrías parar las máquinas si consideras que es necesario en el botón rojo. Abajo tienes las alarmas ordenadas por fecha y hora.

Operario1 - Veo peligroso que se pueda parar la máquina sin que esté fuera de rango su temperatura.

Antonio - OK. Lo limitamos a que esté fuera de rango.

Jefe - ¿Porqué salen temperaturas tan bajas?¿Sólo vas a medir 8 máquinas? ¿Dentro de unos meses vamos a instalar dos máquinas más, puedes dejar hueco para que salgan?

Antonio - Podemos dejar configurado para que podáis añadir nuevas máquinas ¿Que temperaturas suelen tener las máquinas?

Operario1 - Nos interesa poder añadir nuevas máquinas. Las extrusoras son de varios cientos de grados, entorno a 300 ºC.

Jefe - Y aquí, ¿Como miro los datos de la temperatura o la presión que han tenido las máquinas durante la semana?

Antonio - No había contemplado almacenar datos sobre las medidas anteriores. Si quieres podemos guardar en un fichero las lecturas.

Jefe - Por supuesto, si no, no puedo mejorar la eficiencia energética de mi fábrica. Quiero que los datos se muestren y se almacenen cada 5 minutos en el ordenador de mi oficina.

Antonio - Ok ¿Necesitas todas las lecturas? ¿Podemos guardar sólo el último mes o el último trimestre?

Jefe - Al menos necesito el último año.

Antonio - Vale perfecto, luego me gustaría ver el ordenador que tienes en la oficina para ver dónde tendremos que instalarte el software. Otra duda que me surge, las alarmas ¿Queréis poder configurar el rango en el que se activan desde la aplicación o las dejamos en el fichero de configuración de la aplicación?

Jefe - Eso es tema de los operarios.

Operario1 - Mejor déjanoslo en un fichero de texto plano en el que podamos introducir el rango de temperatura de cada tipo de máquina.

Antonio - ¿Qué tipos de máquinas tenéis?

Jefe - ...

Tras la reunión, Antonio visitó el ordenador en el que tenían que instalar el software, un antiguo ordenador sin apenas capacidad libre con Windows XP. Antonio tomó nota de todo, ya

tenía más requisitos y más claro que tenía que hacer:

- *En casi todas las máquinas, el cliente quería conocer la temperatura de los plásticos y la velocidad de los rollos.*
- ~~*También quería detectar de forma automática posibles grumos o malformaciones en la película de plástico circulante.*~~
- *En alguna máquina necesitaba saber la presión del aire para determinar si estaba dentro de tolerancias.*
- *Y toda la información quería saberla en tiempo real desde su PC en la oficina, así como poder actuar bien sobre la refrigeración, bien parando las máquinas o tal vez tocándolas directamente si no había otra opción.*
- Fecha de entrega 1 mes y medio.
- Posibilidad de añadir más máquinas a medir en la interfaz una vez desplegada en el cliente.
- Sólo es posible parar las máquinas que estén fuera de rango su temperatura.
- La temperatura puede ser de varios cientos de grados.
- Guardar en un fichero un histórico de las medidas tomadas durante el último año.
- Los datos se leen y almacenan cada 5 minutos.
- Configuración de las alarmas de temperatura en un fichero plano por tipo de máquina.

Ya con una idea más clara, se fue a analizar y documentar los requisitos del proyecto.

Tabla de requisitos de la Historia de Antonio

Como fase final de la educción interesa tener recogido en un documento (en nuestro caso una tabla excel) todos los requisitos que se han obtenido. De esta forma el análisis en SysML se realiza de manera más rápida. Un ejemplo es la tabla que quedaría en el caso de la historia de Antonio:

Id	Nombre	Descripción	Prioridad	Precedencia	Tipo
R1.1	Temperatura	El sistema medirá la temperatura de todas las máquinas.	F		Funcional
R1.2	Velocidad	El sistema medirá la velocidad de las máquinas con rollo.	F		Funcional
R1.3	Presión	El sistema medirá la presión de aire de las máquinas extrusoras.	F		Funcional
R2.1	MostrarTemperatura	El sistema mostrará la temperatura en el PC de la oficina.	F	R1.1	Funcional
R2.2	MostrarVelocidad	El sistema mostrará la velocidad en el PC de la oficina.	F	R1.2	Funcional
R2.3	MostrarPresión	El sistema mostrará la presión en el PC de la oficina.	F	R1.3	Funcional
R2.4	MostrarAlarmas	El sistema mostrará las alarmas en el PC de la oficina.	F	R1.1,R11	Funcional
R3	ActuarRefrige	El sistema permitirá parar las máquinas en las que se mide la temperatura.	D	R1.1	Funcional
R4	RestriccionActuarRefrige	Sólo se pararán las máquinas con temperatura fuera de rango.	D	R1.1, R3	No Funcion
R5	FechaEntrega	El proyecto debe finalizar el 18/12/2014 (1 mes y medio después de la primera reunión)	F		No Funcion
R6.1	AñadirMaquinasTemperatura	El sistema debe permitir añadir más maquinas para medir Temperatura una vez entregado	D	R1.1, R2.1	Funcional
R6.2	AñadirMaquinasVelocidad	El sistema debe permitir añadir más maquinas para medir Velocidad una vez entregado	D	R1.2, R2.2	Funcional
R6.3	AñadirMaquinasPresión	El sistema debe permitir añadir más maquinas para medir Presión una vez entregado	D	R1.3, R2.3	Funcional
R7	ValorTempatura	El sistema debe permitir medir temperaturas de hasta 400 ºC	F	R1.1, R2	No Funcion
R8	GuardarHistorico	El sistema almacenará todas las lecturas y las alarmas en un histórico en el PC de la oficina del cliente	D	R1.*	Funcional
R9	MaximoHistorico	El sistema almacenará el histórico durante 1 año.	D	R8	No Funcion
R10.1	ActualizaciónDatos	Los datos se leen cada 5 minutos.	F	R1.* R2.*	No Funcion
R10.2	AlmacenajeDatos	Los datos se almacenan cada 5 minutos.	F	R1.* R2.*, R8	No Funcion
R11	RangoAlarma	Los valores válidos de temperatura se almacenarán en un fichero de texto plano según la máquina.	D		Funcional
R12	Ganancias	Las ganancias deben ser de 3000 euros (2000 por mes)	D		No Funcion

¿Porqué estos datos?

Establecer la prioridad y la precedencia servirán como guía a la hora de la planificación temporal. En el caso de Antonio, GuardarHistórico y RestriccionActuarRefrige son los requisitos menos prioritarios, estos pueden ser finalizados en las últimas etapas sin que afecte a las funcionalidades más importantes del sistema.

Conocer si es o no un requisito funcional es un aporte interesante. En el caso de los requisitos funcionales, podemos usar casos de uso para mejorar su comprensión. Por otra parte, en caso de los requisitos no funcionales, podemos tener el caso de ir generando documentos de requisitos no funcionales para nuestra empresa (seguridad, calidad, etc) y reusarlos en futuros proyectos.

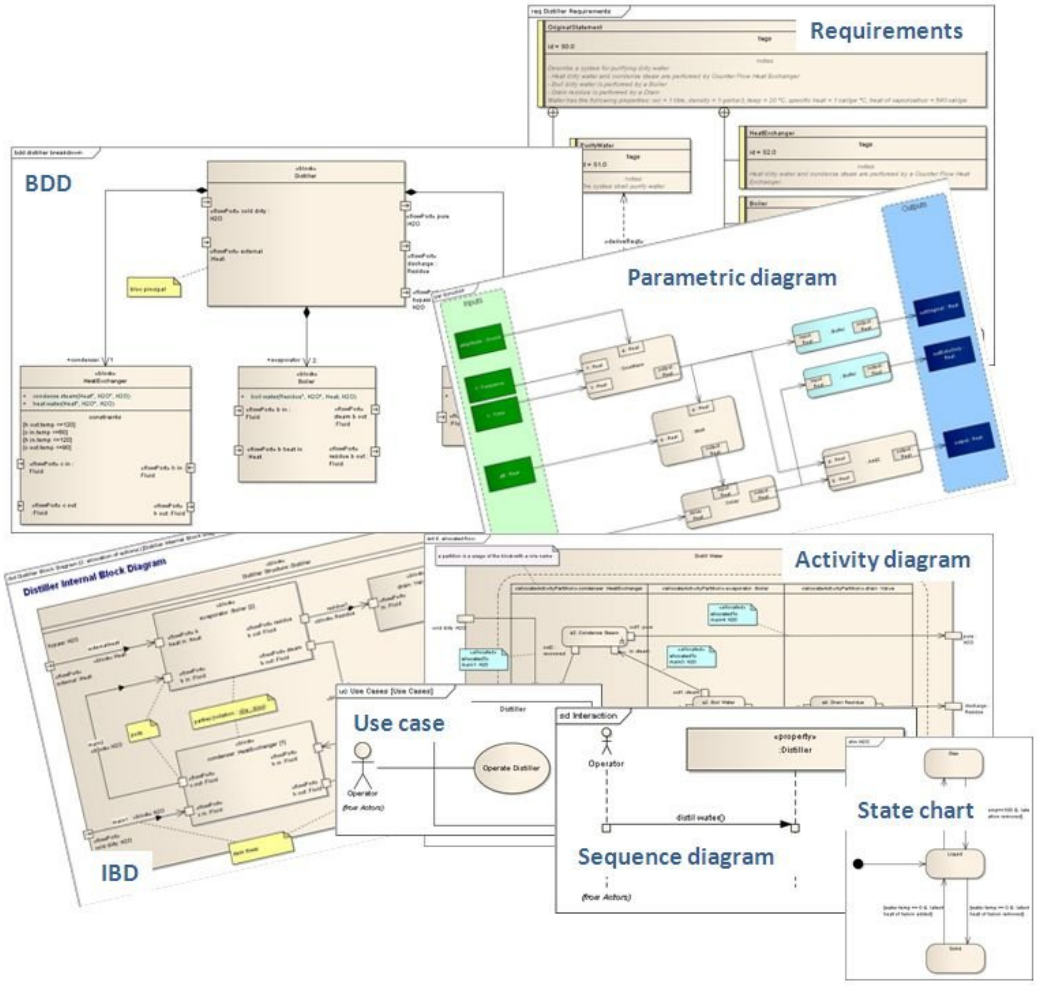
Primer entregable 2/2: Tabla de requisitos

Como parte del primer entregable deberéis realizar la educción de los requisitos de vuestro TFM y rellenar la plantilla que os hemos facilitado en el campus virtual (Requisitos.xlsx). Es deseable que los requisitos que redactéis cumplan las características de un requisito bien especificado descritas anteriormente.

Más información aquí.

SysML

Systems Modeling Language (SysML) es un lenguaje gráfico de propósito general que permite el análisis, especificación, diseño, verificación y validación de sistemas de ingeniería complejos. Estos sistemas pueden incluir hardware, software, información, procesos, el personal y las instalaciones. Este lenguaje es un subconjunto ampliado de UML 2.0 y desde finales de 2007 es un estándar de la OMG.



wikipedia

¿Por qué SysML?

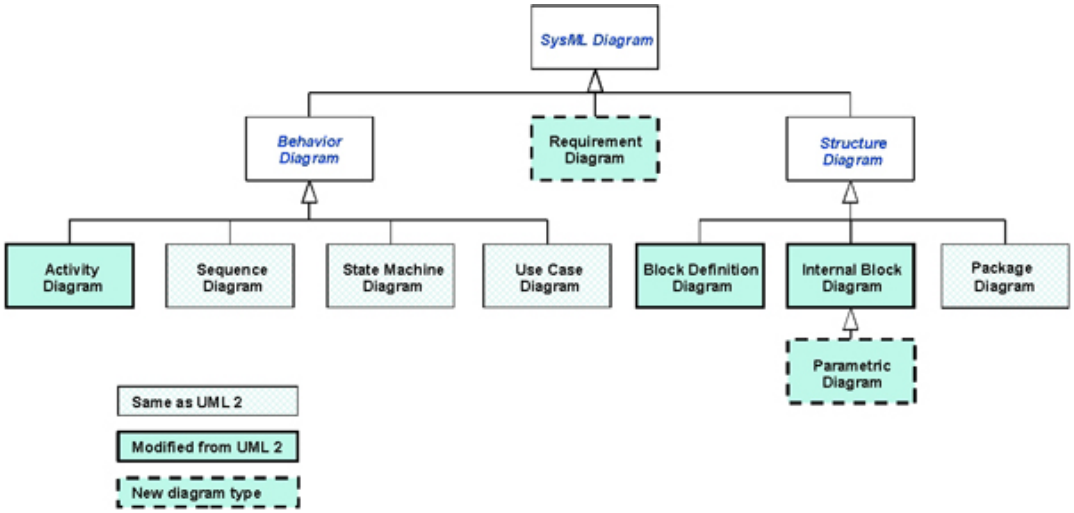
SysML mejora la precisión y aumenta la eficiencia en las comunicaciones entre las partes integrantes de un proyecto. Como lenguaje basado en modelos, además aporta otras ventajas:

- Facilitan la comunicación entre los stakeholder.
- Capturan y gestionan la propiedad intelectual de las empresas.
- Proporciona una estructura escalable para la resolución de problemas.
- Suministra abstracciones que permiten manejar sistemas complejos y de gran tamaño.
- Permite explorar múltiples soluciones en paralelo con un riesgo mínimo.
- Permite detectar errores u omisiones (falta de requisitos) en las fases tempranas del proyecto.

Diagramas

La Real academia Española define un diagrama como: *Dibujo en el que se muestran las relaciones entre las diferentes partes de un conjunto o sistema.*

SysML contiene nueve tipos de diagramas, siete de los cuales comparte en común con UML 2.0.

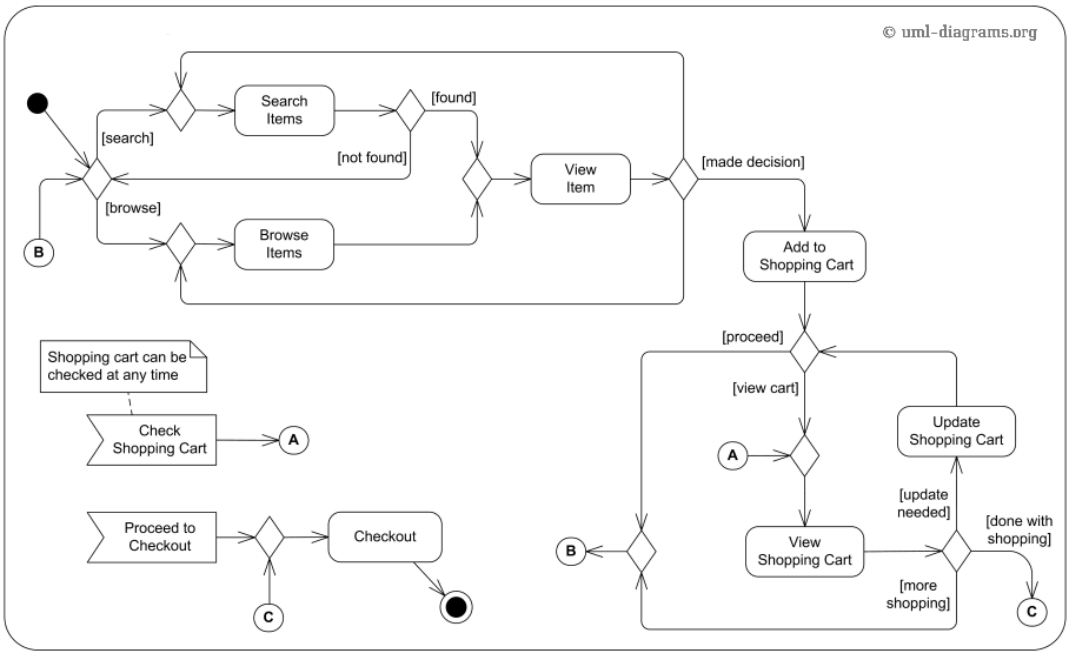


http://www.omgsysml.org/

- **Diagrama de actividad:** Representa el comportamiento en cuanto a la ordenación de las acciones basadas en la disponibilidad de entradas, salidas, y el control, y cómo las acciones transforman las entradas hacia las salidas (modificación de diagrama de actividades UML).
- **Diagrama de secuencia:** Representa el comportamiento en términos de una secuencia de mensajes intercambiados entre las partes (igual que el diagrama de secuencia UML).
- **Diagrama de máquina de estados:** Representa el comportamiento de una entidad en términos de sus transiciones entre estados provocados por eventos.
- **Diagrama de casos de uso:** Representa la funcionalidad (requisitos funcionales) en términos de cómo se utiliza un sistema por entidades externas (los actores) para lograr un conjunto de objetivos.
- **Diagrama de requisitos**: Representa los requisitos del sistema, las relaciones entre ellos, los elementos de diseño y los casos de prueba, facilitando la trazabilidad de los requisitos.
- **Diagrama de definición de bloques:** Representa elementos estructurales llamados bloques, su composición y clasificación (modificación del diagrama de clases UML).
- **Diagrama de bloques interno:** Representa la interconexión y las interfaces entre las partes de un bloque (modificación del diagrama de estructura compuesta de UML).
- **Diagrama paramétrico:** Representa restricciones en los valores que se utilizan para apoyar el análisis de ingeniería.
- **Diagrama de paquetes:** Representa la organización de un modelo en términos de paquetes que contienen elementos del modelo.

A continuación, para entender la utilidad de SysML, responde a las siguientes cuestiones sobre diagramas SysML sin conocer como se definen, sólo sabiendo para que sirve.

Ejemplo de diagrama: Diagrama de actividad



¿Cual de las siguientes opciones representa mejor el comportamiento del sistema descrito en la siguiente imagen?

Sugerencia

- Búsqueda en una plataforma de artículos (item).
- Búsqueda de artículos, gestión del carro de la compra y pasar por caja.
- Gestión de artículos en un carrito de compra.

INCORRECTO: A parte de buscar, el sistema descrito también permite gestionar artículos en un carrito de compra y pasar por caja.

CORRECTO

INCORRECTO: A parte de la gestión, el sistema descrito permite buscar artículos y pasar por caja.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto

¿Permite el sistema descrito añadir más de un artículo diferente al carrito de compra?

- Sí.
- No.

CORRECTO: La salida B permite añadir artículo al carrito de compra comenzando de nuevo las actividades correspondientes.

INCORRECTO: La salida B permite añadir otro artículo al carrito de compra comenzando de nuevo las actividades correspondientes.

Solución

1. Opción correcta
2. Incorrecto

¿Es necesario añadir un nuevo artículo al carro de compra para pasar por caja (*checkout*)?

- Sí.
- No.

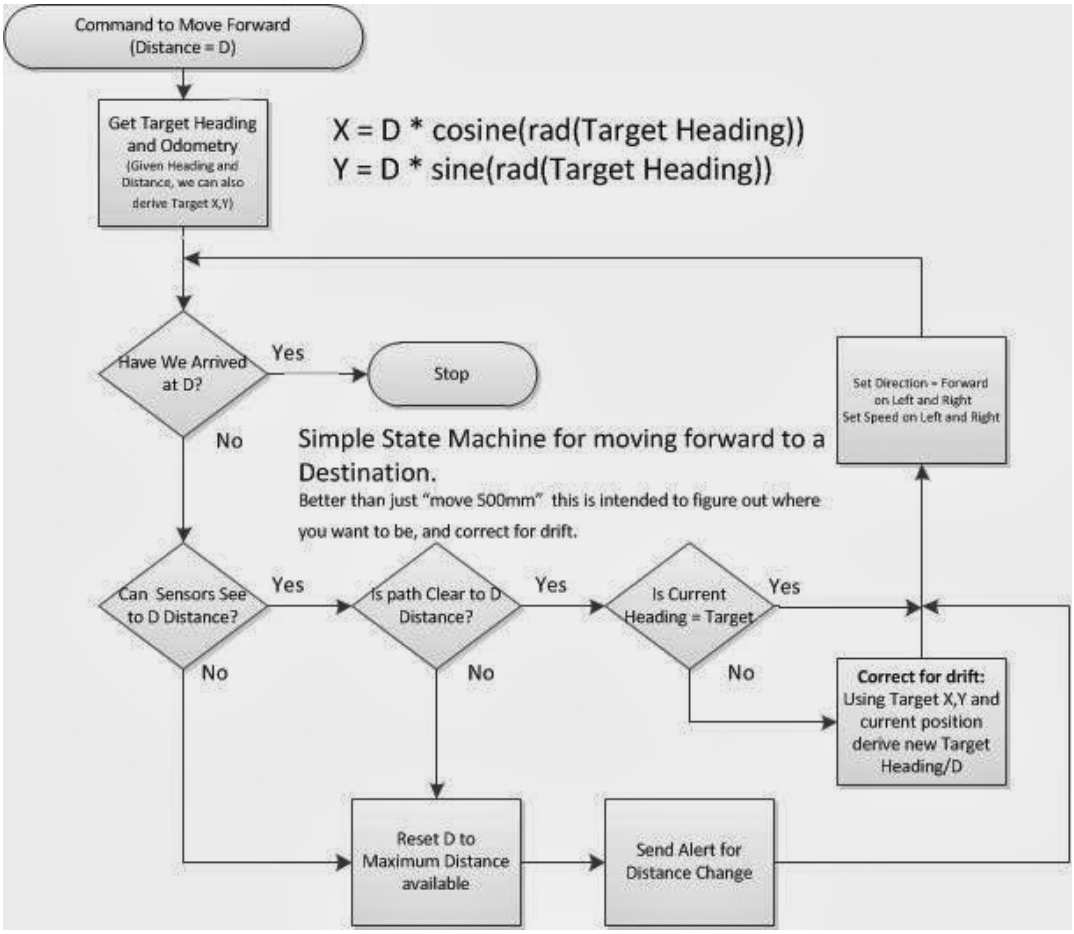
INCORRECTO: El evento *proceed to checkout* permite en cualquier momento que se genere pasar por caja, sin necesidad de añadir un nuevo artículo al carro de compra.

CORRECTO: El evento *proceed to checkout* permite en cualquier momento que se genere pasar por caja, sin necesidad de añadir un nuevo artículo al carro de compra.

Solución

1. Incorrecto
2. Opción correcta

Ejemplo de diagrama: Diagrama de máquina de estados



http://letsmakerobots.com/

¿Permite el navegador definido evitar obstáculos móviles?

- Sí.
- No.

INCORRECTO: Antes de mandar a mover los motores, el navegador se asegura que el camino está libre (*Path is clear*) y la orientación es correcta (*Current heading = Target*), tras esto, no se vuelven a comprobar los obstáculos.

CORRECTO: Antes de mandar a mover los motores, el navegador se asegura que el camino está libre (*Path is clear*), tras esto, no se vuelven a comprobar los obstáculos.

Solución

1. Incorrecto
2. Opción correcta

Si un camino tiene un obstáculo a distancia d < D ¿El navegador queda bloqueado?

- Sí.
- No.

INCORRECTO: El navegador modifica el valor (D = d) cuando encuentra que no puede navegar y envía una alerta avisando del cambio.

CORRECTO: El navegador modifica el valor (D = d) y envía una alerta avisando que ha cambiado la distancia D

Solución

1. Opción correcta
2. Incorrecto

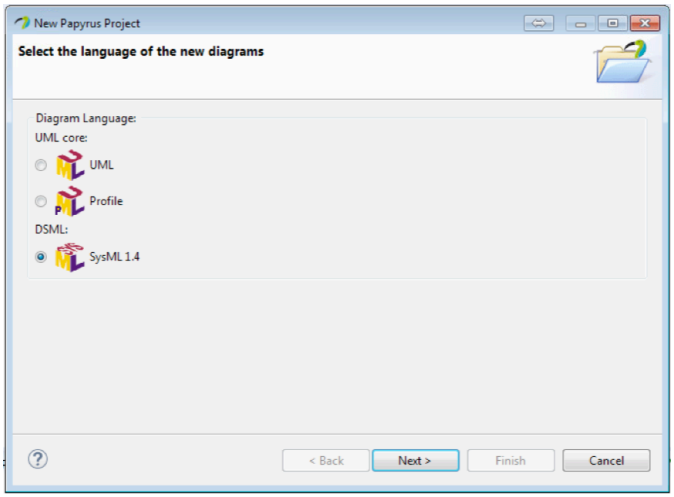
Análisis de Requisitos en SysML

A continuación vamos a describir el proceso de analizar los requisitos en el lenguaje SysML usando el plugin Papyrus para Eclipse. Analizar implica buscar incompatibilidades entre ellos, describirlos de forma que cumplan las características que debe tener un requisito bien especificado, añadir nuevos o borrar duplicados, etc. Este proceso manual se realiza más eficientemente cuando los representamos en un diagrama de requisitos y establecemos las relaciones entre ellos.

A continuación describiremos los pasos que debéis seguir para generar el diagrama de requisitos a partir de la tabla de requisitos que habéis generado.

Generación de un proyecto en Papyrus

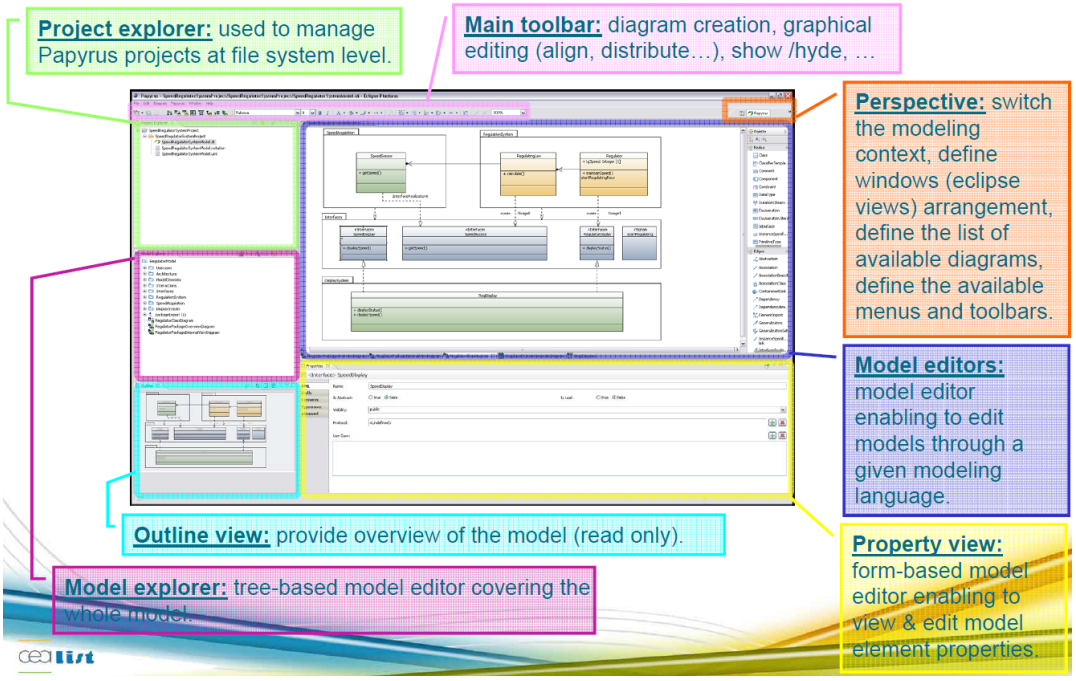
Abrimos eclipse y generamos un nuevo proyecto, *File* > *New* > *Papyrus Project*. Tras esto seleccionamos la opción SysML:



Seleccionas siguiente y eliges un nombre para tu proyecto (por ejemplo el nombre de vuestro TFM) y lo genéras (Pulsa Finish):

A partir de aquí, ya tenemos generado el proyecto sobre el que incluiremos todos nuestros diagramas.

El interfaz con el que vamos a trabajar es el siguiente:

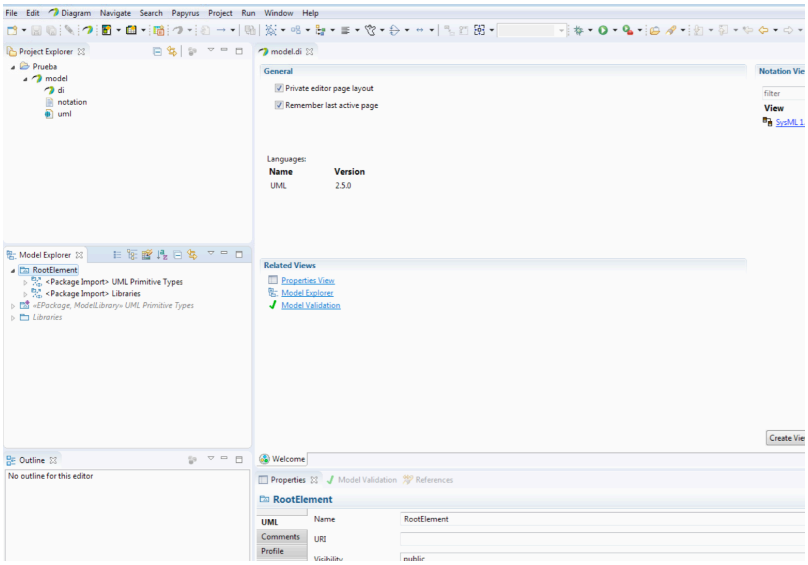


http://www-list.cea.fr/en

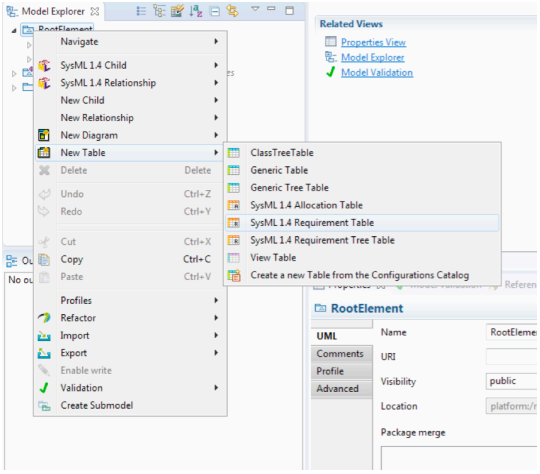
Ya habéis leído lista de requisitos de la historia de Antonio (Requisitos Historia Antonio), con ella, vamos a proceder a generar una tabla de requisitos de SysML y un diagrama de requisitos. Este diagrama va a representar los requisitos del sistema y las relaciones entre ellos, facilitando la trazabilidad y revelando posibles errores que hayamos cometido en la educción de los mismos.

Importar requisitos a la tabla de requisitos de Papyrus

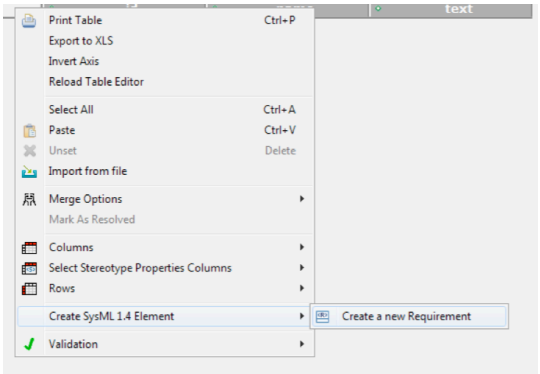
Para crear la tabla de requisitos en *Papyrus,* marcamos en la ventana model explorer, la carpeta RootElement



Pulsamos sobre el botón derecho en esta carpeta y navegamos en el menú *New Table* > *New SysML1.4 Requeriment Table*



Podemos probar a crear un nuevo elemento, pulsando con el botón de la derecha sobre la tabla y seleccionando *Create SysML1.4 element* > *Create a new Requeriment*

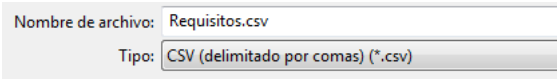


Resultado:

Para facilitarnos el trabajo, Papyrus permite añadir los requisitos desde un fichero csv a la tabla de requisitos en formato SysML. Para ello debemos seguir dos pasos:

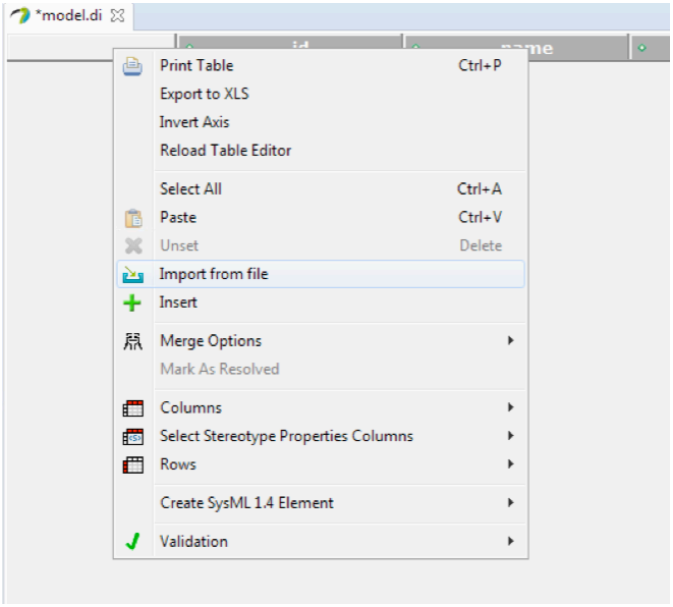
1. Debemos tener sólo 3 columnas: identificador, nombre y descripción.
2. Debemos cargar ficheros CSV.

Por tanto, debes abrir la tabla excel de requisitos de la historia de Antonio, borrar las columnas que no son necesarias y darle a guardar como csv:

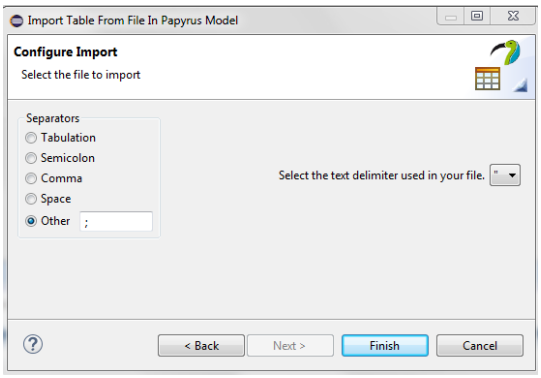


¡Ojo! Antes de importar los requisitos a Papyrus, abre el fichero CSV con un editor de texto plano (bloc de notas o similar) y revisa el separador que se está usando entre campos (en el ejemplo propuesto es ;).

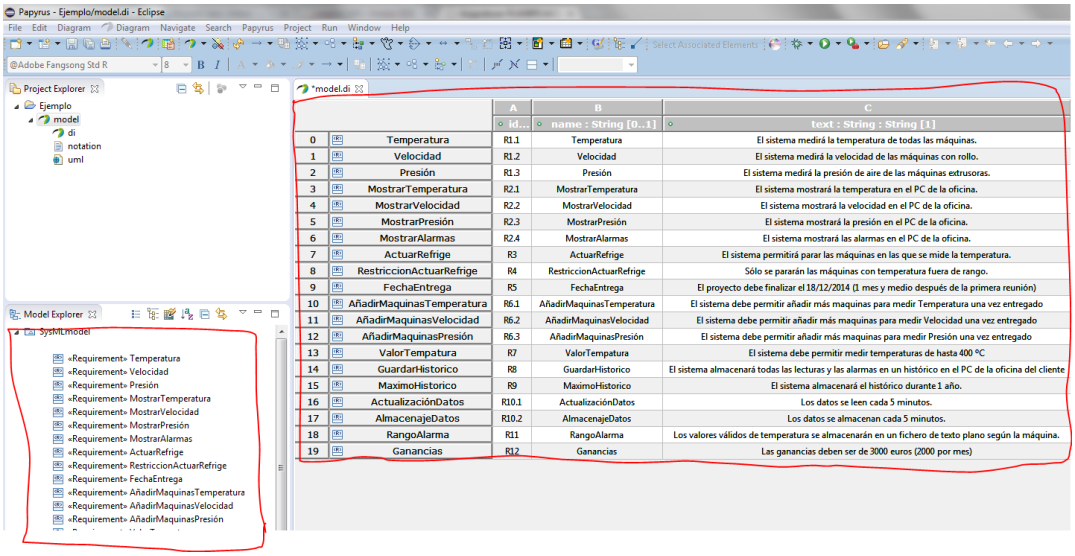
Volviendo a la tabla de requisitos que se ha generado, selecciona el área que está a la izquierda de la primera columna, pulsa el botón de la derecha y selecciona *Import from file*:



Busca y selecciona el fichero csv que has generado anteriormente. Selecciona el separador de tu fichero csv (en nuestro caso *Other ;*) y finaliza la operación:

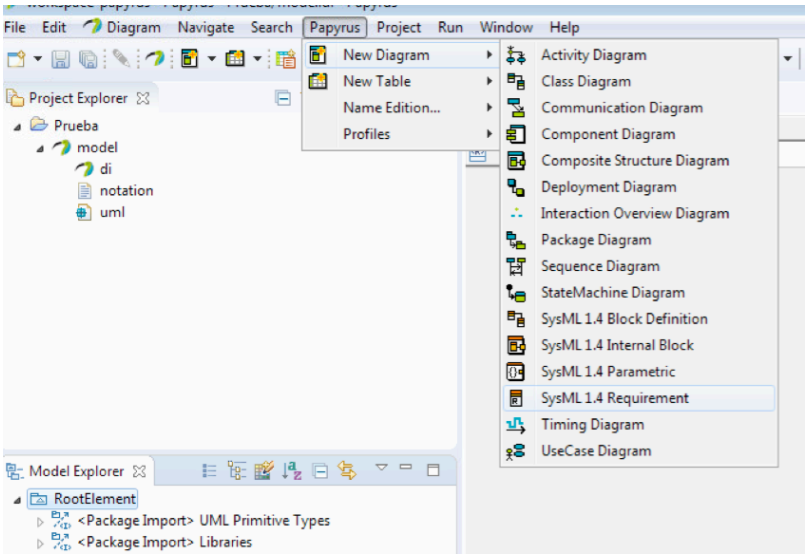


Ya tenemos los requisitos cargados en nuestro proyecto y en nuestra tabla, y están listos para el siguiente paso, realizar el diagrama. En algunos casos la cabecera del fichero CSV de requisitos se incluye, debes borrar este falso requisito:



Creación del diagrama de Requisitos

Pulsamos en la ventana del model explorer sobre nuestro el paquete *Requisitos* y a continuación vamos al menú *Papyrus > New Diagram > Create a new SysML 1.4 Requirement* elegimos un nombre y aceptamos.



Generación del diagrama de Requisitos

Tras crearlo, podemos ir arrastrando uno a uno los requisitos a la ventana del editor de modelos para formar el diagrama completo.

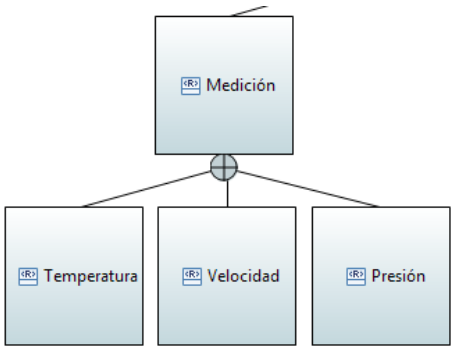
Existen las siguientes relaciones que se pueden establecer entre requisitos en un diagrama de requisitos de SysML:

- *Containment* (Decompose): Esta relación se establece para descomponer un requisito en otro u otros más concretos.
- *Derive* (Derive): Esta relación se establece cuando un requisito deriva de otro. El requisito derivado no es una especificación más concreta del que deriva, esa relación es Containment.
- *Copy* (Copy): Esta relación se establece entre requisitos con el mismo texto y diferente id. Esta relación suele ser útil para reutilizar requisitos no funcionales.

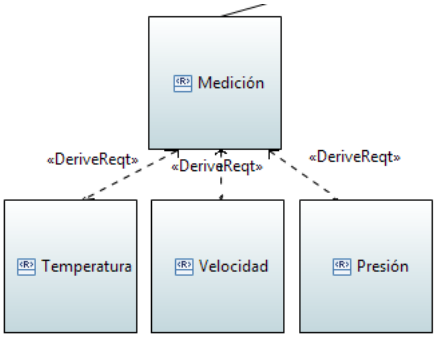
¿Cómo modelarías las siguientes relaciones de la historia de Antonio?

Si generamos un requisito medir con el cual se relacionan los requisitos de medir temperatura, presión y velocidad, ¿Cómo lo modelarías?

- *Containment*



- *Derive*



CORRECTO: El requisito medir, se descompone en los requisitos medir temperatura, presión y velocidad. Estos requisitos aumentan el nivel de detalle del requisito medir.

INCORRECTO: Los requisitos temperatura, velocidad y presión no se derivan de medir, sin especificaciones más concretas de él.

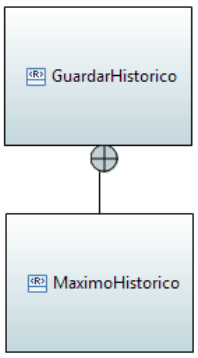
Solución

1. Opción correcta
2. Incorrecto

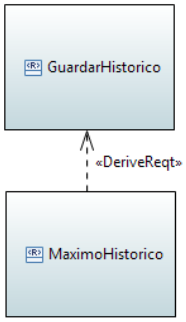
¿Cómo modelarías las siguientes relaciones de la historia de Antonio?

Cómo se relacionan los requisitos GuardarHistórico (*El sistema almacenará todas las lecturas y las alarmas en un histórico en el PC de la oficina del cliente*) y MáximoHistórico (*El sistema almacenará el histórico durante 1 año*)

- *Containment*



Derive



INCORRECTO: El requisito MáximoHistórico no está contenido en el requisito GuardarHistórico. Se deriva de él.

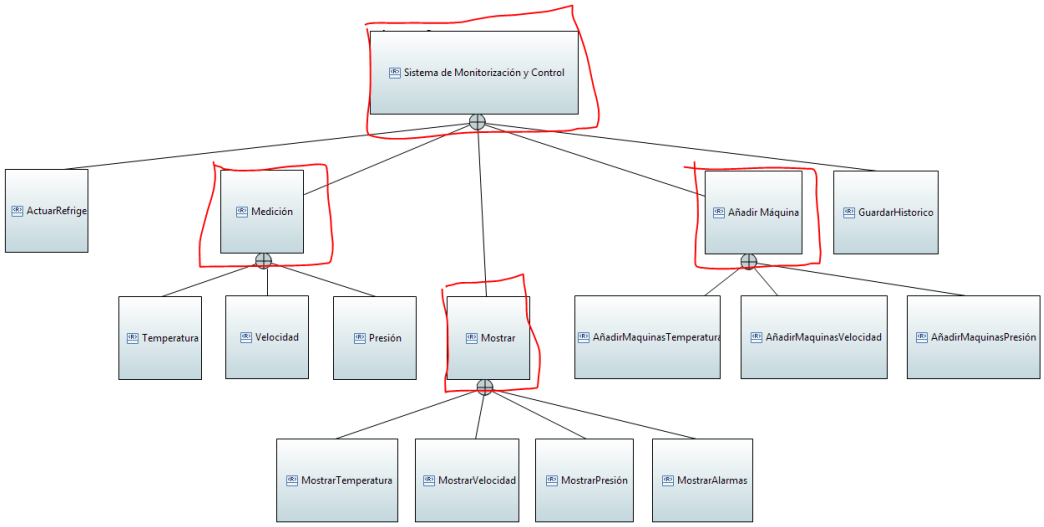
CORRECTO: El requisito MáximoHistórico se deriva del requisito GuardarHistórico, pero no forma parte de él.

Solución

1. Incorrecto
2. Opción correcta

Diagrama de requisitos: Historia de Antonio

Es una buena costumbre al definir el diagrama de requisitos hacerlo de manera descendente usando la relación *containment*. Por este motivo vamos a generar un requisito que sea el nombre de nuestro TFM, del cual se desglosarán el resto de requisitos. En el caso de la historia de Antonio, vemos que los primeros 3 requisitos también podemos agruparlos en uno más general que es medición. Esta realización de generalización también puede ser aplicada a los requisitos de MostrarXX y AñadirX. En la siguiente imagen se muestra el resultado, remarcando en rojo los nuevos requisitos generados:



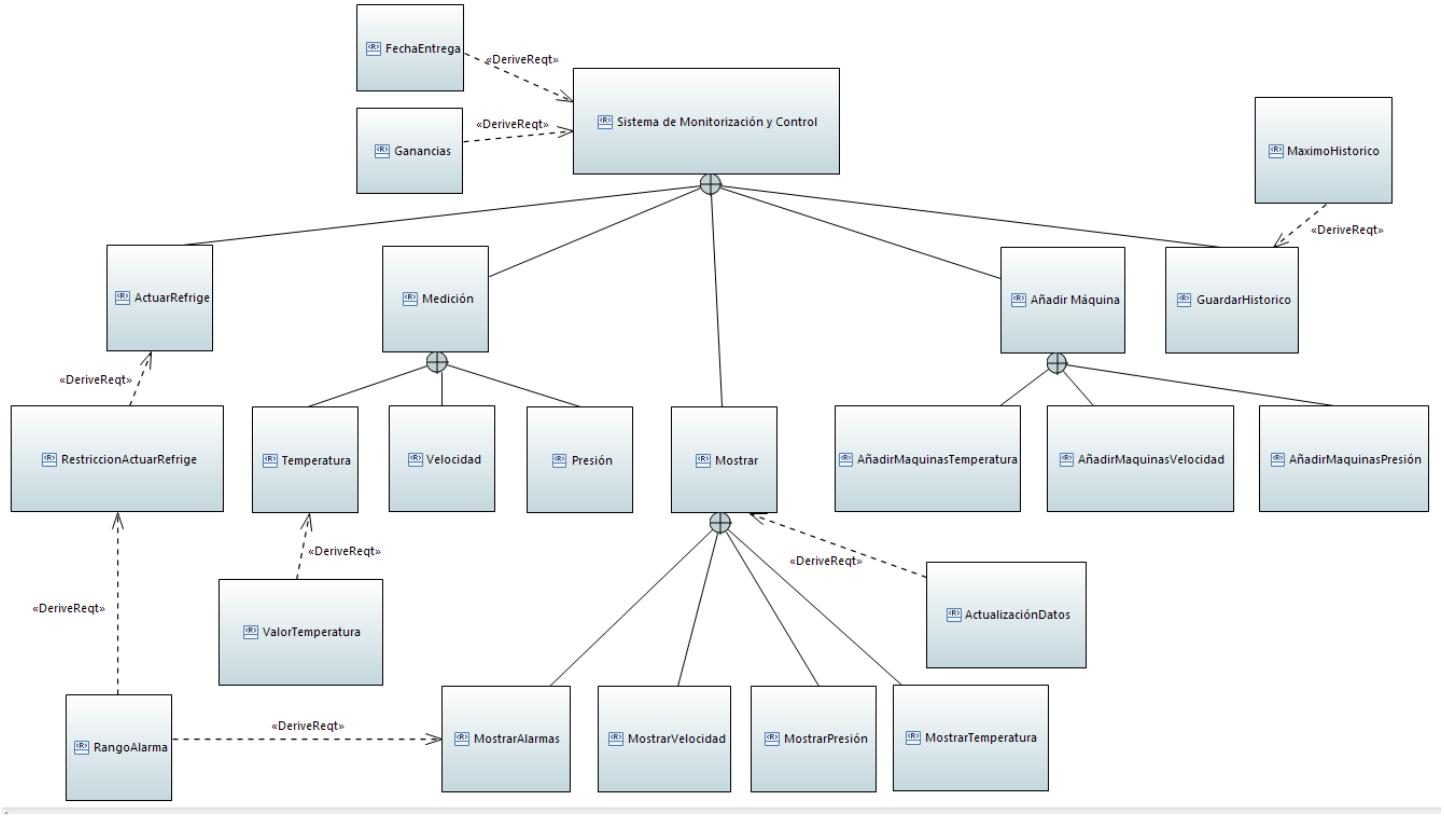
Existen una serie de requisitos de negocio que son requisitos no funcionales que se derivan del requisito general de nuestro proyecto:

- FechaEntrega (*El proyecto debe finalizar el 18/12/2014 (1 mes y medio después de la primera reunión)*).
- Ganancias (*Las ganancias deben ser de 3000 euros (2000 por mes)*).

El resto de requisitos no funcionales que quedan son restricciones sobre un requisito funcional y se modelan como una relación de *Derive:*

- Deriva de ActuarRefrige -> RestriccionActuarRefrige (*Sólo se pararán las máquinas con temperatura fuera de rango*).
- Deriva de Temperatura-> ValorTemperatura (*El sistema debe permitir medir temperaturas de hasta 400 ºC*).
- Deriva de GuardarHistorico-> MaximoHistorico (*El sistema almacenará el histórico durante 1 año*).
- Deriva de Mostrar-> ActualizaciónDatos (*Los datos se leen cada 5 minutos*).
- Deriva de GuardarHistorico-> AlmacenajeDatos (*Los datos se almacenan cada 5 minutos*).

Por último, el requisito funcional RangoAlarma (*Los valores válidos de temperatura se almacenarán en un fichero de texto plano según la máquina*) se deriva del requisito funcional MostrarAlarmas y RestriccionActuarRefrige, quedando el diagrama de requisito de la historia de Antonio:


FechaEntrega
«DeriveReqt»
Ganancias
«DeriveReqt»
Sistema de Monitorización y Control
MaximoHistorico
«DeriveReqt»
ActuarRefrige
Medición
Añadir Máquina
GuardarHistorico
«DeriveReqt»
RestriccionActuarRefrige
Temperatura
Velocidad
Presión
Mostrar
AñadirMaquinasTemperatura
AñadirMaquinasVelocidad
AñadirMaquinasPresión
«DeriveReqt»
«DeriveReqt»
ValorTemperatura
«DeriveReqt»
ActualizaciónDatos
RangoAlarma
«DeriveReqt»
MostrarAlarmas
MostrarVelocidad
MostrarPresión
MostrarTemperatura

Primer entregable 3/3: Diagrama de requisitos

Como última parte del primer entregable deberéis realizar el diagrama de requisitos de vuestro TFM en Eclipse como os hemos mostrado en la historia de Antonio y adjuntarlo al entregable como una imagen.

Más información aquí.

Sección 2

Ingeniería de requisitos

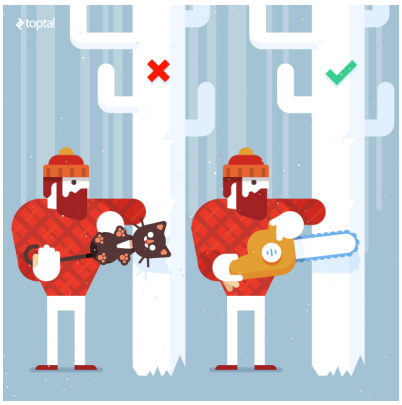
)

Objetivos

En esta sección aprenderás:

- Qué es la validación y porqué es útil
- Cómo generar un plan de validación.

Validación del sistema



La validación es el proceso por el cual el cliente o el stakeholder acordado por el cliente comprueba que lo que se ha generado es lo que se deseaba. Esta validación se lleva a cabo con diferentes tipos de pruebas de aceptación y genera un informe de aceptación en el que se detalla que todos los entregables están acorde a lo esperado.

Las pruebas de aceptación siempre se realizan sobre el sistema completo previo a su finalización. Ojo, no es el objetivo de estas pruebas el centrarse en identificar problemas estéticos de la interfaz, como errores ortográficos, tampoco es el detectar errores de diseño, como derivaciones a tierra. Este tipo de pruebas de integración o sistema ya son realizadas en la etapa de verificación por los desarrolladores.

Pasar de manera satisfactoria estas pruebas de validación implica que el cliente certifica que el sistema o producto generado cumple con lo deseado. Estas pruebas pueden requerir varias iteraciones y se pueden postergar en el tiempo.

Los tipos de pruebas que se usan para validar son 2:

1. Pruebas alfa: realizadas por el cliente - o los stakeholder acordados por el cliente o empresa - con el equipo de desarrollo o parte de él como observador en un entorno controlado.
2. Pruebas beta: realizadas por cliente - o los stakeholders acordados por el cliente o empresa - en el entorno de trabajo real y sin observadores.

Todas las pruebas son recogidas en un documento de plan de aceptación que suele redactar el jefe de proyectos y que es aprobado por el cliente previo al comienzo del mismo. El lenguaje para estas pruebas es cercano al cliente, ya que será el encargado de realizarlas. El plan de aceptación se adjunta al contrato, el cual debería incluir al menos:

- Una descripción textual de los requisitos funcionales y no funcionales.
- Una fecha de entrega.
- Una descripción de los entregables y la forma de realizar dicha entrega.
- Un presupuesto junto con su plan de contingencia.

En nuestro caso, vamos a resumir y sólo vamos a generar el informe de aceptación junto con las pruebas necesarias.

Validación en la historia de Antonio

Tras definir como iba a verificar los requisitos, Antonio debería haber pasado a la validación, el tema más peliagudo. Supongamos que esto ocurrió y concertó una cita más con su cliente....

En esta cita Antonio le explicó que iban a redactar junto con el contrato, las pruebas que él como jefe de su empresa iba a realizar para confirmar que todo funcionaba según él esperaba.

Tras más de 20 minutos intentando explicar porqué era importante la validación, el cliente le dejó claro a Antonio que tenía que funcionar todo bien y que él no se iba a poner a hacer pruebas. Antonio le insistió y le comentó que una vez el sistema estuviera instalado y las pruebas desde su despacho realizadas, su trabajo habría concluido. Y que tras esto, las horas extras por no estar la aplicación totalmente a su gusto las facturaría a parte.

El cliente entro en razón y Antonio lo guió en el proceso de redacción de las pruebas. Lo primero que le recordó es los tres entregables que iba a hacerle: programa, manual de usuario y la red montada en su fábrica. Antonio le preguntó y el cliente confirmó que sería el que revisaría que estos tres entregables eran correctos.

Nombre del entregable	Pruebas asociadas para su validación	Responsable
Programa software	Prg1, Prg2, Prg3, Prg4 y Prg5	Jefe de la empresa
Instalación de red	Prg1 y Prg5	Jefe de la empresa
Manual de usuario	Mnl1	Jefe de la empresa

Tras esto, Antonio le recordó que era lo que iba a hacer el software y por tanto que es lo que tenían que probar:

- Lectura de las máquinas de temperatura, presión y velocidad.
- Comprobación de rangos de alarmas y actuación sobre la refrigeración.
- Almacenaje de datos en los históricos.
- Incorporar nuevas máquinas.

El cliente estuvo de acuerdo en que esto es lo que quería, como ya le había comentado. Antonio redactó las pruebas para validar este software y le comentó al cliente que cuando tuviera completo el sistema, el realizaría las pruebas sólo y rellenaría los resultados. El cliente confirmó que las pruebas le parecían bien. Antonio le insistió que en caso de validar correctamente todas las pruebas, él daría por terminado su trabajo, cosa que al cliente no le hizo mucha gracia, pero que tras algunas discusiones más acabo aceptando.

A continuación se muestra un ejemplo de prueba de validación: la lectura de sensores. Observa como la descripción y los pasos son descripciones textuales sin entrar en detalle y considerando que el sistema ya está terminado y desplegado listo para usarse. Se deja un campo de resultado obtenido para que el cliente rellene que ha ocurrido al terminar la prueba y en caso de ser fallida, cuanto afecta a sus expectativas (inadmisible, es algo a mejorar, etc).

Lectura sensores	Prg1
Descripción: Se procederá a realizar lecturas durante una hora de todas las máquinas de fábrica con sensores externos que confirmen sus valores.	
Prerrequisitos Maquinarías encendidas y funcionando; mediciones de sensores externos funcionando.	
Pasos: 1.- El usuario enciende la aplicación. 2.- El usuario puede leer desde la interfaz de la aplicación las lecturas de los sensores.	
Resultado esperado: Los valores de los sensores externos coinciden con los de la aplicación	

Resultado obtenido:

¿Qué ocurre si alguna de estas pruebas no convencen al cliente? Si está justificado y no son nuevas funcionalidades, hay que hacer los cambios en lo que se requiera. Si quedaba de forma ambigua lo que había que realizar (requisito ambiguo), el cliente siempre lleva la razón.

En caso de coincidir con lo esperado, la prueba es satisfactoria y el cliente realiza una firma sobre esta fila para realizar la confirmación. Una vez aceptadas todas las pruebas el trabajo a concluído y se debe proceder al cierre del proyecto.

En el caso que nos atañe estas firmas no son necesarias ya que para presentar el TFM tenéis que tener el visto bueno del tutor y por tanto, todas las pruebas de validación han debido ser pasadas de manera satisfactoria.

Entregable 3 1/1: Genera el plan de aceptación

Como parte de este tercer entregable tendrás que realizar el plan de aceptación. Este tiene dos partes:

- En la primera parte desglosarás los entregables de tu proyecto: informes, manuales de usuario, placas, software, etc. Para cada entregable asignarás el stakeholder encargado de validarlo.
- En la segunda parte generarás pruebas para validar los entregables. Algunas líneas para definir estas pruebas:
  - Las pruebas las realizará el cliente o stakeholder designado por él, por tanto debes redactarlos evitando jerga técnica.
  - Los casos de uso pueden ser usados como pruebas para validar.
  - Las pruebas están enfocadas a que el cliente certifique que los entregables son correctos.
  - Las pruebas NO SON para ver que las funcionalidades no tienen errores, de esto ya se encarga la verificación.
  - Una vez el cliente certifica que es correcto, lo notifica en la fila ***Resultado obtenido*** y firma sobre la misma.

Usa como guía el ejemplo de la Historia de Antonio del campus virtual para rellenar los nuevos apartados del documento de especificación de tu TFM.

Inco

Sección 3

Ingeniería de requisitos: Documentación

)

Objetivos

En esta sección aprenderás:

- A definir casos de uso para refinar la descripción de los requisitos funcionales del TFM.
- A generar un diagrama de casos de uso.
- A definir el conjunto de pruebas a realizar para verificar los requisitos.
- A generar la documentación necesaria para describir los requisitos y casos de uso del TFM.

Casos de uso

Para añadir más detalle a la especificación de un proyecto, es necesario ampliar la descripción de los requisitos funcionales describiendo parte de su comportamiento. Un **caso de uso,** es una descripción de los pasos o actividades que deberán realizarse entre el sistema visto como caja negra[1] y un **actor o actores** para lograr satisfacer un o unos requisitos funcionales (o sólo una parte de un requisito funcional). Los personajes, entidades o sistemas externos que participarán en un caso de uso es lo que se denomina **actor**. Estos actores son siempre externos al propio sistema que estamos desarrollando.

Los casos de uso pueden ser usados tanto para afinar la educción de requisitos (por ejemplo encontrando nuevos o redefiniendo los existentes) como **para conocer el comportamiento que el cliente desea que tenga el sistema (validación).**

La descripción textual de un caso de uso incluye:

- Contexto de uso: Cuando se va a usar por el actor este caso de uso.
- Actor Principal: Quién o que requiere invocar al caso de uso.
- Participantes y Objetivos: Quién o que interacciona con el sistema en el caso de uso.
- Pre condiciones: Que condiciones son necesarias que se den para que el caso de uso pueda comenzar.
- Garantías mínimas: Condiciones que se dan cuando el caso de uso finaliza.
- Escenario de éxito principal: Descripción de los pasos o actividades que deberán realizarse para llevar a cabo el caso de uso.
- Extensiones: Descripción de los pasos o actividades que se realizan en un escenario alternativo en el caso de uso (no se tienen porqué cumplir las garantías mínimas).

Los casos de uso suelen ir acompañados de un **diagrama de casos de uso** que es una representación visual de todos los casos de uso y los actores del proyecto.

1 El concepto caja negra hace referencia a tratar el sistema generado como una caja en la que se desconoce el contenido pero si se conocen las interfaces de entrada y sal

Casos de uso en la Historia de Antonio

Lo primero que vamos a hacer es recordar que requisitos funcionales tenemos. De esta lista de requisitos buscamos extraer cuales de ellos podríamos ampliar con casos de uso. No tiene porqué ser un requisito un caso de uso. Puede ser un requisito (o una parte de él) o unos requisitos funcionales los que satisfacen un sólo caso de uso.

Recordemos que el proceso de análisis se añadieron algunos requisitos más:

Id		Nombre	Descripción	Prioridad	Precedencia	Tipo
R1		Medición	El sistema medirá la temperatura, velocidad y presión de todas las máquinas.	F		Funcio
	R1.1	Temperatura	El sistema medirá la temperatura de todas las máquinas.	F		Funcio
	R1.2	Velocidad	El sistema medirá la velocidad de las máquinas con rollo.	F		Funcio
	R1.3	Presión	El sistema medirá la presión de aire de las máquinas extrusoras.	F		Funcio
R2		Mostrar	El sistema mostrará la temperatura, velocidad y presión en el PC de la oficina.	F	R1	Funcio
	R2.1	MostrarTemperatura	El sistema mostrará la temperatura en el PC de la oficina.	F	R1.1	Funcio
	R2.2	MostrarVelocidad	El sistema mostrará la velocidad en el PC de la oficina.	F	R1.2	Funcio
	R2.3	MostrarPresión	El sistema mostrará la presión en el PC de la oficina.	F	R1.3	Funcio
	R2.4	MostrarAlarmas	El sistema mostrará las alarmas en el PC de la oficina.	F	R1.1,R11	Funcio
R3		ActuarRefrige	El sistema permitirá parar las máquinas en las que se mide la temperatura.	D	R1.1	Funcio
R6		Añadir	El sistema debe permitir añadir más maquina para medir Temperatura, Velocidad y Presión	F	R1	Funcio
	R6.1	AñadirMaquinasTemperatura	El sistema debe permitir añadir más maquinas para medir Temperatura una vez entregado	D	R1.1, R2.1	Funcio
	R6.2	AñadirMaquinasVelocidad	El sistema debe permitir añadir más maquinas para medir Velocidad una vez entregado	D	R1.2, R2.2	Funcio
	R6.3	AñadirMaquinasPresión	El sistema debe permitir añadir más maquinas para medir Presión una vez entregado	D	R1.3, R2.3	Funcio
R8		GuardarHistorico	El sistema almacenará todas las lecturas y las alarmas en un histórico en el PC de la oficina del cliente	D	R1.*	Funcio
R11		RangoAlarma	Los valores válidos de temperatura se almacenarán en un fichero de texto plano según la máquina.	D		Funcio

Observamos como los requisitos R1.1, R1.2 y R1.3 (el requisito R1 es una generalización de ellos) tienen el mismo comportamiento y pueden ser especificados bajo el mismo caso de uso. La diferencia de qué están midiendo no afecta al comportamiento, ya que todos leen de un sensor. En caso de que alguno de estos requisitos leyera de un sistema externo, o solicitara datos a un servidor, si afectaría a su comportamiento y sería separado en otro caso de uso.

El mismo caso lo tenemos en los requisitos 2.1,2.2,2.3 (R2 generaliza) y 6.1,6.2,6.3 (R6 generaliza), por tanto, los casos de uso que nos quedarían en la Historia de Antonio serían (estamos definiendo comportamiento, el nombre debe ir acorde al contenido):

- C1 Medir los valores de una máquina.
- C2 Mostrar los valores de una máquina.
- C3 Actuar sobre la refrigeración de una máquina.
- C4 Añadir una nueva máquina al sistema existente.
- C5 Guardar Histórico.
- C6 Almacenar valores válidos de temperatura.

Observamos como algunos de los casos de uso se activan a una frecuencia dada (C1,C2,C5). El sistema en sí no puede desencadenar un caso de uso, por tanto, ¿Como modelamos estos casos de uso? Hacemos uso de un actor que modela el reloj que invoca a una frecuencia dada estos casos de uso, el actor Tiempo. Tenemos también el actor máquina, el cual representa a todas las máquinas de la fábrica. El actor Usuario representa a todos los usuarios que van a hacer uso del sistema de Monitorización y Control.

A continuación vamos a describir cada uno de los casos de uso.

C1 Medir los valores de una máquina

Contexto de uso: A una frecuencia dada, se miden los datos de un sensor situado en la máquina de la cual se le ha solicitado información.
Actor Principal: Tiempo.
Participantes y Objetivos:

Participante	Objetivo
Tiempo	Activar la lectura de los datos de una máquina dada.
Máquina	Proveer los datos de su sensor.

Pre condiciones: Máquina conectada.
Garantías mínimas: Se almacena el valor de la máquina (válido o no).
Escenario de éxito principal:

1.- El Tiempo solicita al Sistema de Monitorización y Control la lectura de una máquina.

2.- El Sistema de Monitorización y Control se comunica con la máquina.

3.- La máquina devuelve el valor de su sensor.

4.- El Sistema de Monitorización y Control actualiza su valor interno del sensor.

Extensiones:

3.a- No obtenemos respuesta.

4.a.- El Sistema de Monitorización y Control almacena un estado de fallo.

3.b- Obtenemos un valor erróneo (fuera de rango).

4.b.- El Sistema de Monitorización y Control almacena un estado de fallo.

Al definir este caso de uso nos damos cuenta que no hemos tenido en cuenta que el sistema nos informe de fallos en las lecturas. Esta funcionalidad hay que añadirla como requisito, se nos ha pasado. El requisito RangoAlarma (*Los valores válidos de temperatura se almacenarán en un fichero de texto plano según la máquina*) debe ser reescrito para que también se almacenen los valores válidos de los otros sensores.

Como vimos en el bloque anterior, los casos de uso son también un proceso de educción de requisitos (obtienes requisitos funcionales). Aunque el proceso de generar los requisitos es iterativo e incremental, es primordial encontrar todos los requisitos antes de pasar a la fase de diseño. Por tanto, los nuevos requisitos quedan:

Id		Nombre	Descripción	Prioridad	Precedencia	Tipo
R1		Medición	El sistema medirá la temperatura, velocidad y presión de todas las máquinas.	F		Funcio
	R1.1	Temperatura	El sistema medirá la temperatura de todas las máquinas.	F		Funcio
	R1.2	Velocidad	El sistema medirá la velocidad de las máquinas con rollo.	F		Funcio
	R1.3	Presión	El sistema medirá la presión de aire de las máquinas extrusoras.	F		Funcio
R2		Mostrar	El sistema mostrará la temperatura, velocidad y presión en el PC de la oficina.	F	R1	Funcio
	R2.1	MostrarTemperatura	El sistema mostrará la temperatura en el PC de la oficina.	F	R1.1	Funcio
	R2.2	MostrarVelocidad	El sistema mostrará la velocidad en el PC de la oficina.	F	R1.2	Funcio
	R2.3	MostrarPresión	El sistema mostrará la presión en el PC de la oficina.	F	R1.3	Funcio
	R2.4	MostrarAlarmas	El sistema mostrará las alarmas en el PC de la oficina.	F	R1.1,R11	Funcio
R3		ActuarRefrige	El sistema permitirá parar las máquinas en las que se mide la temperatura.	D	R1.1	Funcio
R6		Añadir	El sistema debe permitir añadir más maquina para medir Temperatura, Velocidad y Presión	F	R1	Funcio
	R6.1	AñadirMaquinasTemperatura	El sistema debe permitir añadir más maquinas para medir Temperatura una vez entregado	D	R1.1, R2.1	Funcio
	R6.2	AñadirMaquinasVelocidad	El sistema debe permitir añadir más maquinas para medir Velocidad una vez entregado	D	R1.2, R2.2	Funcio
	R6.3	AñadirMaquinasPresión	El sistema debe permitir añadir más maquinas para medir Presión una vez entregado	D	R1.3, R2.3	Funcio
R8		GuardarHistorico	El sistema almacenará todas las lecturas y las alarmas en un histórico en el PC de la oficina del cliente	D	R1.*	Funcio
R11		RangoAlarma	Los valores válidos de cada sensor se almacenarán en un fichero de texto plano según la máquina.	D		Funcio
R12		InformeFallos	El sistema notificará con una alarma cuando un sensor de valores no válidos.	D	R1	Funcio

El nuevo requisito funcional no requiere de un caso de uso para describirlo ya que está incluido en este que estamos describiendo.

C2 Mostrar los valores de una máquina

Contexto de uso: A una frecuencia dada se muestran los datos almacenados de una máquina de la cual se le ha solicitado información.
Actor Principal: Tiempo.
Participantes y Objetivos:

Participante	Objetivo
Tiempo	Activar el procedimiento para mostrar los datos de una máquina dada.

Pre condiciones: Datos de la máquina almacenados.
Garantías mínimas: El Sistema de Monitorización y Control muestra el valor de la máquina.
Escenario de éxito principal:

1.- El Tiempo solicita al Sistema de Monitorización y Control que muestre los datos de una máquina dada.

2.- El Sistema de Monitorización y Control muestra el valor almacenado para esa máquina.

Extensiones:

2a .- Si el valor almacenado es un fallo, informa con una alarma.

C3 Actuar sobre la refrigeración de una máquina

Contexto de uso: El usuario desea actuar sobre la refrigeración de una máquina.
Actor Principal: Usuario.
Participantes y Objetivos:

Participante	Objetivo
Usuario	Actuar sobre la refrigeración de una máquina.
Máquina	Control de la refrigeración.

Pre condiciones: Máquina conectada, refrigeración funcionando.
Garantías mínimas: El Sistema de Monitorización y Control avisa si se puede o no actuar.
Escenario de éxito principal:

1.- El usuario escoge una máquina sobre la que actuar en la refrigeración.

2.- El Sistema de Monitorización y Control se comunica con la máquina.

3.- La máquina devuelve el valor de su sensor de temperatura.

4.- El Sistema de Monitorización y Control comunica a la máquina que active su refrigeración e informa con una alarma.

Extensiones:

4a.- El Sistema de Monitorización y Control **no actúa** sobre la refrigeración porque está dentro de rango y lo notifica al usuario con una alarma.

C4 Añadir una nueva máquina al sistema existente

Contexto de uso: El usuario desea añadir una nueva máquina el sistema.
Actor Principal: Usuario.
Participantes y Objetivos:

Participante	Objetivo
Máquina	Proveer los datos de sus sensores.
Usuario	Añadir una nueva máquina.

Pre condiciones: Máquina conectada y sensor integrado.
Garantías mínimas: El Sistema de Monitorización y Control añade una nueva máquina al sistema.
Escenario de éxito principal:

1.- El usuario añade una nueva máquina, estableciendo el tipo, los valores fuera de rango para la máquina y las conexiones al sensor.

2.- Sistema de Monitorización y Control se comunica con la máquina.

3.- La máquina devuelve el valor de su sensor.

4.- El Sistema de Monitorización y Control notifica con una alarma al usuario que la máquina ha sido correctamente añadida.

Extensiones:

3a.- La máquina no devuelve nada.

4a.- El Sistema de Monitorización y Control notifica al usuario con una alarma que no puede conectar con la nueva máquina.

C5 Guardar Histórico

Contexto de uso: El usuario desea tener almacenadas todas las lecturas de los sensores a cierta frecuencia en un histórico.
Actor Principal: Tiempo.
Participantes y Objetivos:

Participante	Objetivo
Tiempo	Activar el almacenaje de los datos de todos los sensores.

Pre condiciones: Sistema operativo con espacio en disco.
Garantías mínimas: Datos almacenados.
Escenario de éxito principal:

1.- El Tiempo activa la lectura de cada una de las máquinas.

2.- El Sistema de Monitorización y Control se comunica con cada una de las máquinas.

3.- Cada máquina devuelve el valor de su sensor.

4.- El Sistema de Monitorización y Control almacena en un fichero los datos de los sensores de cada máquina.

Extensiones:

4a.- El Sistema de Monitorización y Control detecta un fallos en la lecturas, informa de ellos con una alarma y almacena los datos de los sensores de cada máquina.

C6 Almacenar valores válidos de un sensor

Contexto de uso: El usuario desea establecer el rango de valores válidos de una máquina.
Actor Principal: Usuario.
Participantes y Objetivos:

Participante	Objetivo
Usuario	Almacenar cuando los valores están fuera de rango.

Pre condiciones: Sistema operativo con espacio en disco.
Garantías mínimas: Datos almacenados.
Escenario de éxito principal:

1.- El usuario da los rangos válidos para una máquina.

2.- El Sistema de Monitorización y Control los almacena en un fichero.

Extensiones:

No hay.

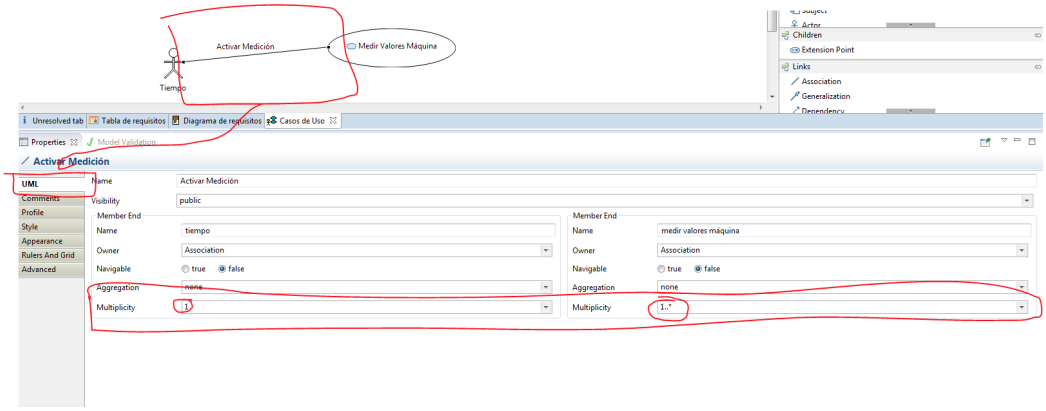
Segundo entregable 1/2: Casos de uso, descripción textual

En esta semana se van a documentar los requisitos y casos de uso de vuestro TFM en el documento Documento de Especificación del TFM. Rellena la descripción textual de los casos de uso y los actores como hemos visto anteriormente para la Historia de Antonio. El documento está comentado con las dudas más comunes que te pueden surgir.

Diagrama de casos de uso

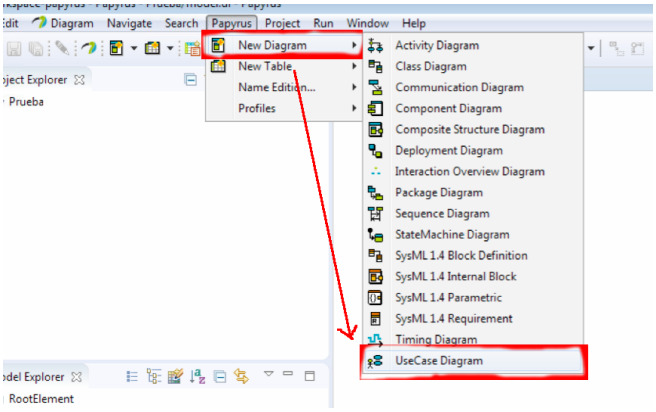
Un diagrama de casos de uso es una representación de las interacciones entre los actores y el sistema a través de los diferentes casos de uso. Dentro del diagrama de casos de uso distinguimos dos elementos y relaciones entre ellos:

- Actores( ): Personajes, entidades o sistemas externos que participarán en un caso de uso.
- Caso de uso (Medir Valores Máquina): Caso de uso definido previamente de manera textual.
- Relaciones entre casos de uso:
  - Inclusion( ): La relación de inclusión permite que un caso de uso, denominado caso de uso base, incluya la funcionalidad de otro caso de uso, llamado el caso de uso incluido, como parte de su funcionalidad cuando se realiza. El caso de uso incluido se realiza siempre que se produce el caso de uso base. Un comportamiento que realiza el caso de uso base hace referencia a menudo el comportamiento del caso de uso incluido.
  - Extension( ): Un caso de uso se puede extender un caso de uso base utilizando la relación de extensión. El caso de uso extendido es un fragmento de una funcionalidad que no se considera parte de la funcionalidad del caso de uso base. A menudo este describe un comportamiento excepcional en la interacción, como el control de errores.
  - Classification(Generalization): Los casos de uso se pueden clasificar utilizando la relación de generalización estándar SysML. Los escenarios para el caso de uso general (tanto el de éxito principal como el extendido), son también escenarios del caso de uso especializado. También significa que los actores asociados con un caso de uso general no participan en ningún escenario únicamente descritos por un caso de uso especializado. Clasificación de los casos de uso se muestra mediante el símbolo generalización estándar SysML.
- Relaciones entre actores:
  - Classification(Generalization): Los actores se pueden clasificar utilizando la relación de generalización estándar SysML. Un actor especializado representa un subconjunto del actor general que tiene ciertas características que la diferencia del resto de actores incluidos en el actor general.
- Relación entre actores y casos de uso:
  - Association(Association): Cada actor se puede relacionar con un caso de uso y establecer la multiplicidad. Puede ser que ese actor invoque al caso de uso 1 sóla vez (1), al menos 1 vez (1..*), como máximo 1 vez (0..1) o puede que lo invoque muchas veces o ninguna (0..*). La multiplicidad de establece en la pestaña UML de la asociación:



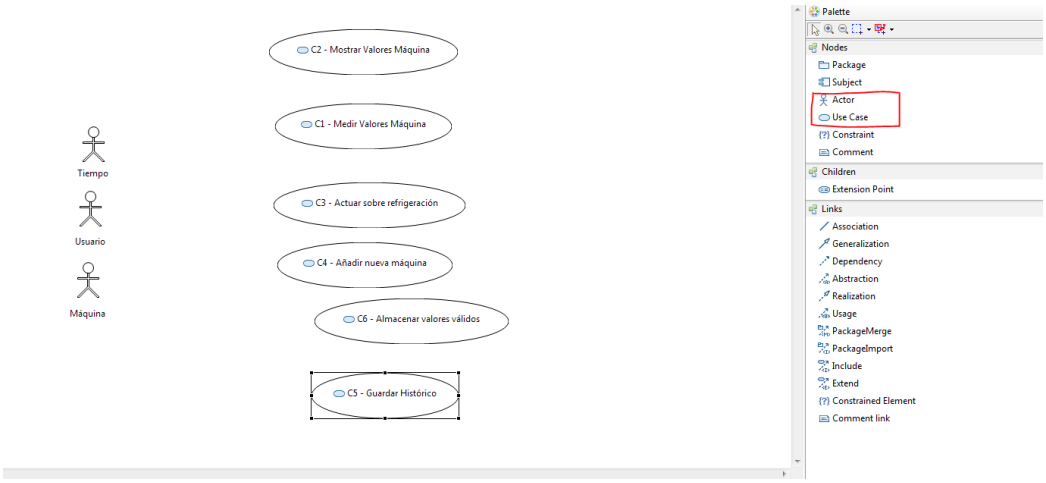
Creación diagrama casos de uso

Pulsamos en la ventana del model explorer sobre nuestra carpeta del modelo y a continuación vamos al menú *Papyrus* > *New Diagram* > *UseCase Diagram* elegimos un nombre y aceptamos.

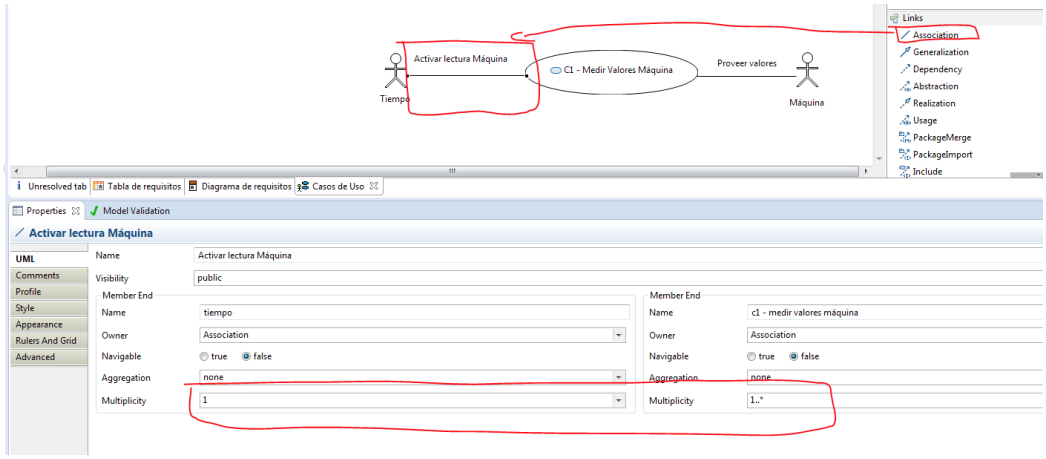


Generación del Diagrama de Casos de Uso: Historia de Antonio

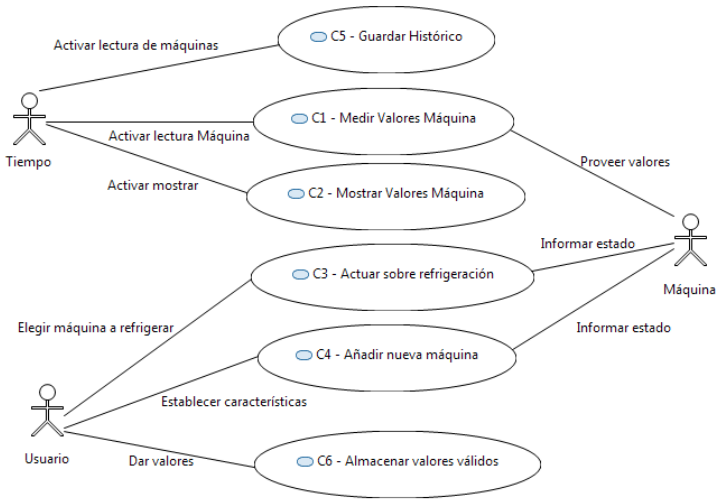
Tras crearlo, lo abrimos y generamos los actores y casos de uso que hemos descrito previamente de manera textual:



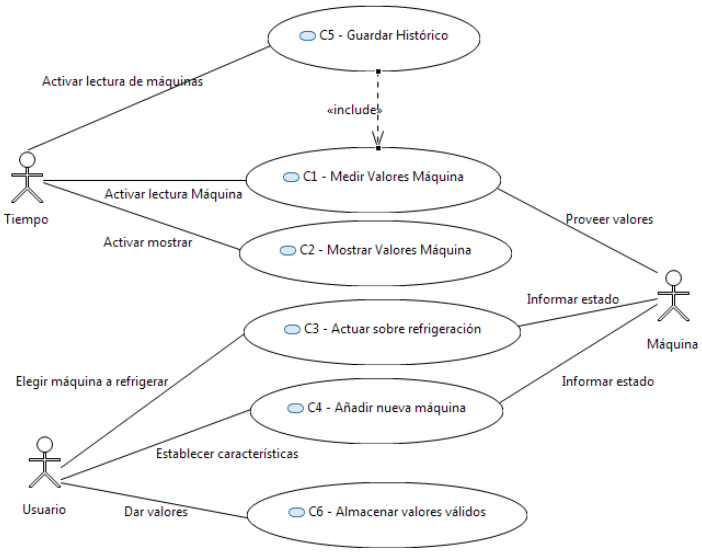
En el primer caso de uso intervienen los actores Tiempo y Máquina. Usamos el *Link Association* para describir que el Actor participa en el caso de uso. Podemos rellenar el campo *Name* con un valor descriptivo sobre su intervención (*Activar lectura máquina*). Es importante definir la Multiplicidad de la relación. En este caso el Tiempo puede invocar muchas veces el caso de uso (1..*).



Una vez realizamos todos los enlaces entre actores y casos de uso, el diagrama queda:



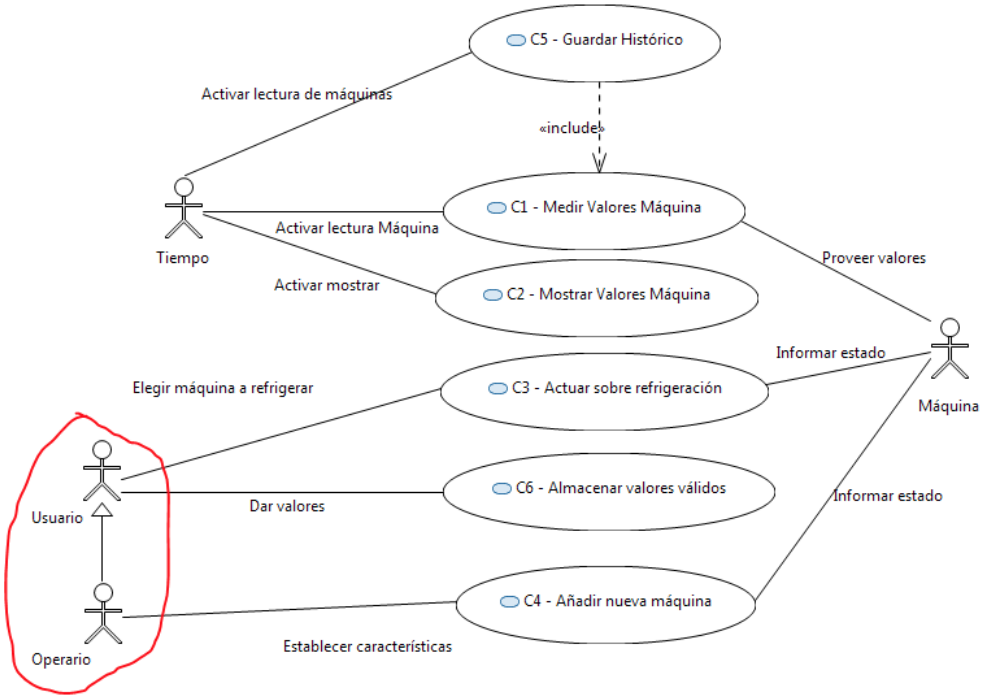
Una vez definidas las relaciones entre actores y casos de uso cabe preguntarse si existen relaciones entre casos de uso. Podemos observar como tenemos sólo una relación en este diagrama de casos de uso: el caso de uso C5 Guardar Histórico incluye al caso de uso C1 Medir los valores de una máquina, este mide todos los valores de las máquinas antes de almacenarlos.



No existen relaciones entre los actores, con lo cual damos por terminado nuestro diagrama de casos de uso de la historia de Antonio.

Modificaciones a la Historia de Antonio para el diagrama de casos de uso

Supongamos ahora que existe un actor más que es el operario. Este es un usuario especial que es el único que puede añadir nuevas máquinas ¿Cómo lo modelamos? Con la relación *Classification* entre actores:



De esta forma hemos definido que el Operario es el único usuario que puede añadir una nueva máquina. Hay que destacar que el operario es un usuario, por tanto puede actuar sobre la refrigeración o almacenar los valores válidos de los sensores.

Segundo entregable 2/3: Diagrama de casos de Uso

Tras rellenar la descripción textual de los casos de uso y los actores de tu TFM debes añadir el diagrama de casos de uso al documento Documento de Especificación del TFM. El documento está comentado con las dudas más comunes que te pueden surgir.

Verificación de requisitos basado en pruebas

La verificación tiene por objeto comprobar que un producto/entregable/servicio único cumple con el conjunto de especificaciones dadas para él a priori. Es decir, responde a la pregunta ¿El producto/entregable/servicio único es el que queríamos construir?

La verificación se puede aplicar tantas veces como sea necesaria en la fase de desarrollo, ampliación o producción. A menudo el proceso de verificación es interno, sin embargo, es posible contratar a empresas externas para realizarlo.

En la fase de desarrollo, los procedimientos de verificación implican la realización de pruebas para modelar o simular un subconjunto, o la totalidad de un producto/entregable/servicio único. En la fase posterior al desarrollo, los procedimientos de verificación implican la repetición regular de pruebas diseñadas específicamente para asegurar que el producto/entregable/servicio único siguen cumpliendo las especificaciones iniciales a medida que avanza el tiempo.

La verificación por tanto consiste la definición del conjunto de pruebas y en la revisión o análisis de los resultados obtenidos en estas pruebas para comprobar si son los esperados, y si no, evaluar el efecto sobre el producto, servicio o sistema para valorar un posible cambio. Hay que hacer hincapié en la forma de la verificación que vamos a realizar. Aunque existen áreas formales dedicadas a esto, como model checking o normas que guían a una correcta forma, como la ISO1012-2012, la aproximación que vamos a seguir en esta asignatura es textual debido a la diversidad de proyectos y falta de tiempo.

Por tanto, se van a realizar descripciones textuales de las pruebas a realizar. Para ello, previamente se va a definir el entorno en el que se van a ejecutar dichas pruebas. Tras esto, para cada prueba definiremos:

- El **requisito** que se va a verificar.
- La **descripción** de la prueba a realizar.
- La **salida** esperada para esa prueba.
- Los **errores** que se podrían obtener.

Verificación de requisitos de la Historia de Antonio

Como hemos comentado anteriormente, lo primero es definir el entorno en el que se van a realizar las pruebas. En la historia de Antonio se contemplaron dos entornos, uno con todo el sistema montado en casa y otro con el sistema ya desplegado:

1. Entorno del sistema beta: *Se cuenta con el sistema de comunicación inalámbrico y el sistema software/hardware completo desarrollado. Aún no está desplegado y se cuenta con lo necesario para simular los sensores.* Es decir, tenemos el sistema completo a falta de montar en la fábrica del cliente.
2. Entorno del sistema alfa: *Se cuenta con el sistema desplegado en la fábrica del cliente con al menos una máquina de cada tipo conectada.* Es decir, tenemos el sistema completo montado en la fábrica del cliente.

Una vez definidos los entornos, fué necesario definir que pruebas se iban a realizar en él. Para ello, se fue recorriendo la lista de requisitos definiendo pruebas que los verificarán. Hubo casos en los que una misma prueba sirvió para verificar más de un requisito.

En el documento de *Documento de Especificación del TFM: Historia de Antonio* se desglosan todas las pruebas que se van a realizar. A continuación se muestra un ejemplo de como se va a verificar en el entorno beta la lectura (R1.1, R2.1, R10.1) y almacenamiento (R10.2) de los sensores de temperatura acorde a las restricciones especificadas (R7).

Requisito/s	R1.1, R2.1, R7, R10.1, R10.2
Prueba	Se simularán las lecturas de los sensores de temperatura en paralelo (5 máquinas) con cambios de valor cada minuto en los rangos acotados por el requisito R7. Los valores serán diferentes para cada máquina. La simulación se realizará en el extremo de la red inalámbrica de la máquina.
Salida	a. En la interfaz del PC de la oficina se mostrarán los valores simulados, con cambios cada 5 minutos b. Los datos se almacenan cada 5 minutos.
Errores	a. Los valores leídos/almacenados difieren de los simulados. b. Los valores leídos/almacenados son correctos, pero en máquinas distintas. c. Los valores se actualizan/almacenan más rápido o más lento de lo especificado en R10.1 (5 minutos)

Observamos como el campo Requisito/s especifica que requisitos van a ser **total o parcialmente verificados**. Esto es importante ya que puede ser necesario más de una prueba para verificar totalmente un requisito. Por otro lado, es posible verificar con una sola prueba más de un requisito.

Los campos prueba y salida describen de forma textual la prueba que se va a realizar y las salidas que se esperan obtener.

El campo errores desglosa los posibles errores que se pueden obtener y es un complemento al campo salida, ya que describe salidas que son incorrectas.

Observa como estas primeras pruebas que se realizan en la especificación no tienen en cuenta cuestiones muy profundas del diseño. Esta sección de pruebas es ampliada durante las siguientes fases del proyecto.

Segundo entregable 3/3: Completa el documento de requisitos y casos de uso

El documento que se va a usar para la documentación de los requisitos de la semana anterior y los casos de uso de esta semana es una modificación del documento de visión del Proceso Racional Unificado (RUP *Rational Unified Process*), Documento de Especificación del TFM. Este es un proceso de desarrollo del software creado por la empresa Rational Software (actualmente parte de IBM).

El documento modificado se encuentra en el campus virtual en el módulo M22. Este fichero está **totalmente comentado**, las secciones que lo componen son:

1. **Introducción**: Se describe el proyecto a grandes rasgos, los objetivos y alcance del mismo.
2. **Directivas del proyecto**: Descripción del problema y el producto/servicio que se va a generar.
3. **Descripción de participantes y usuarios**: Se describen los stakeholder y usuarios finales de la aplicación (mismo contenido que *Identificación de stakeholder.doc*)
4. **Requisitos**: En esta sección se describen los requisitos funcionales y no funcionales del sistema. Amplia la descripción de tus requisitos generados y pega la tabla y el diagrama dónde corresponde.
5. **Verificación de requisitos:** En esta sección se describen las pruebas que se van a realizar para la verificación de requisitos.
6. **Casos de uso:** En esta sección se describen los actores y casos de uso. Se añade un diagrama general de casos de uso para tener una visión de conjunto.
7. **Cuestiones Abiertas**: Declaración de los factores inciertos que podrían contribuir a diferencias significativas en el producto/servicio.

Este documento correctamente rellenado es la entrega de esta semana. Con este documento finalizamos la fase de especificación de tu TFM y comenzamos el diseño.

Inicio

Sección 4

Arquitectura del sistema

Objetivos

En esta sección aprenderás:

- A generar diagramas de bloques en SysML.
- A definir la arquitectura lógica de un proyecto.
- A definir la arquitectura física de un proyecto.

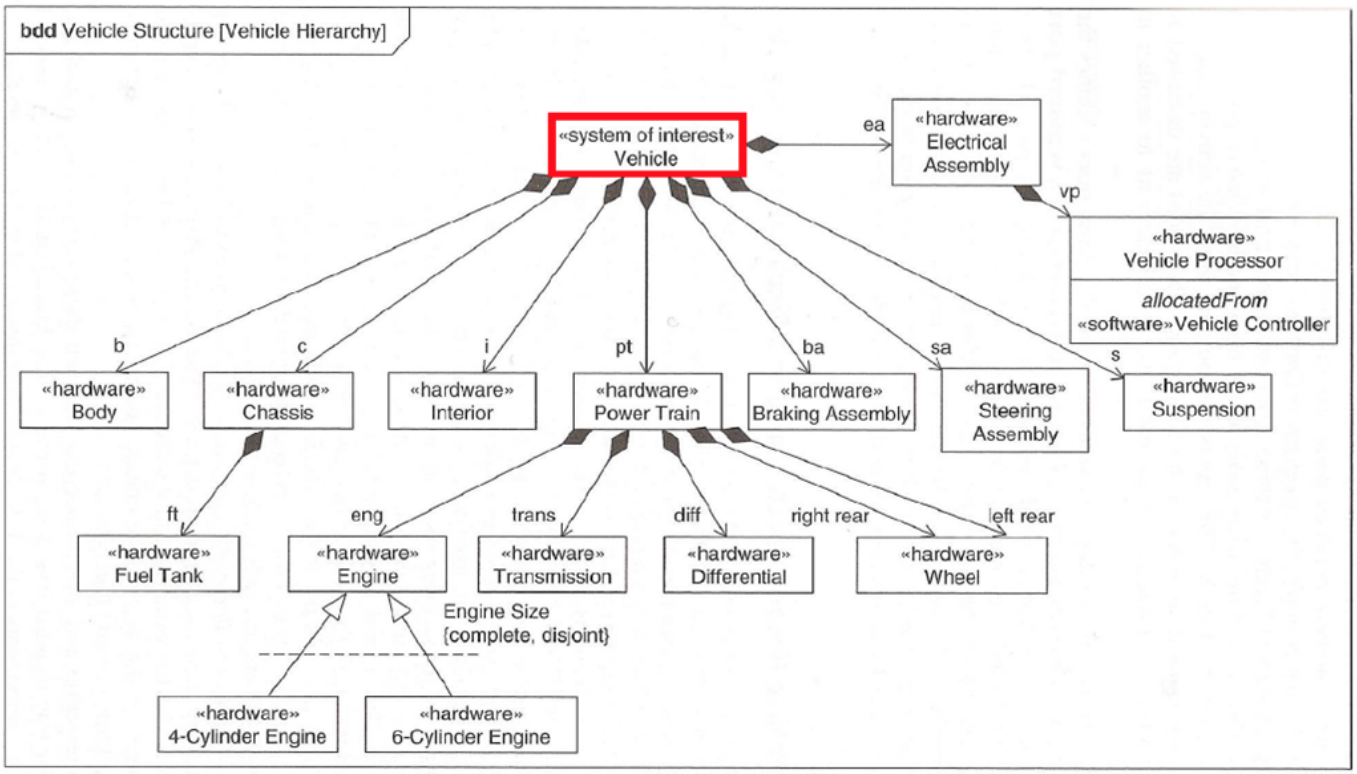
Arquitectura

La arquitectura de un sistema define como el sistema o producto va a ser construido, describe cuales son los bloques principales y como encajan entre sí, siempre partiendo de una perspectiva a alto nivel, evitando en lo posible el bajo nivel.

Nos vamos a centrar sólo en dos tipos de arquitectura: lógica y física.

La arquitectura lógica define los componentes software del proyecto y las relaciones que existen entre ellos.

La arquitectura física de un proyecto describe los elementos hardware del mismo y como estos se relacionan.



http://tekniskit.blogspot.com.es/

Diagrama de Bloques

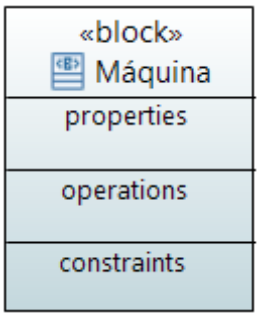
Este diagrama es el que vamos a usar para definir tanto la arquitectura lógica como física de nuestro sistema. El **Diagrama de definición de bloques** representa elementos estructurales llamados bloques, su composición y clasificación. El diagrama de bloques es una modificación de SysML al diagrama de Clases de UML.

¿Qué entendemos por bloque?

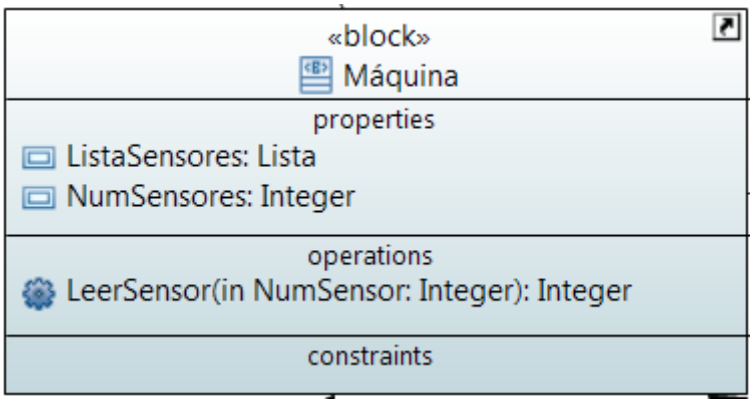
Un bloque es la unidad modular fundamental. Este puede definir un tipo de una entidad lógica o conceptual, una entidad física, un componente de hardware, software o datos, una persona (actor), una instalación, una entidad que fluye a través del sistema (por ejemplo, agua), una entidad en el medio natural (por ejemplo, la atmósfera o el océano), etc.

Un bloque podemos verlo también como un tipo, es decir, una descripción de un conjunto de instancias similares u objetos, cada uno de los cuales presenta las mismas características (aunque no los mismos valores). Un ejemplo de un bloque es una máquina de la historia de Antonio, esta tendrá una serie de características como pueden ser el rendimiento, consumo, peso, antigüedad, etc. Cada instancia del bloque máquina (cada máquina en particular) incluirá estas características y será identificada por el valor de algunas de sus propiedades.

En Papyrus un bloque tiene el siguiente aspecto:



Una vez creado el bloque, podemos definir las propiedades, operaciones y restricciones sobre el bloque:

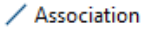


Parte de la arquitectura lógica de la Historia de Antonio

Relaciones entre bloques

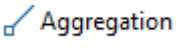
Como ya hemos visto en otros diagramas, tenemos las unidades básicas (en este caso los bloques) y las relaciones entre ellas. En el caso de los diagramas de bloques tenemos tres tipos de relaciones: por un lado las que están a nivel de instancia de bloque, en el caso de Antonio una máquina en concreto se relaciona con un tipo de sensor en concreto; por otro lado las que están a nivel de bloque como tipo, en el caso de la historia de Antonio, existen diferentes tipos de sensores, temperatura, presión o velocidad; y por último está la relación general de dependencia.

Las relaciones a nivel de instancia son:

- **Association**(

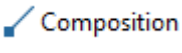
 ):

  Una asociación es la relación más general que existe a nivel de instancia. Esta puede vincular cualquier número de bloques. Una asociación puede tener vinculado un nombre, y en los extremos, puede tener asociado roles, indicadores de propiedad, multiplicidad, visibilidad y otras propiedades.

- **Aggregation**(

 ):

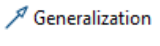
  La agregación es una variante de la asociación que podemos denominar "tiene una...". La agregación tiene la restricción de que sólo vincula a dos bloques (el contenedor y el contenido) y no puede tener más (puedes realizar desde el mismo bloque más, pero necesitas una agregación por cada nuevo bloque).

- **Composition**(

 ):

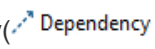
  La composición es una variante de la agregación más específica. La composición tiene una fuerte dependencia con el ciclo de vida de las instancias con el bloque contenedor. Podemos verlo como si el contenido no existe en el diagrama sin el contenedor es una composición, si no, es una agregación.

Las relación que se puede establecer a nivel de tipo es:

- **Generalization**(

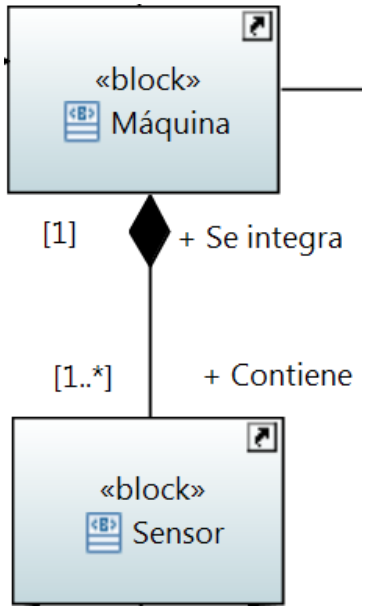
 ):
  La relación de generalización se corresponde con "es un". Esta relación indica que uno de los dos bloques relacionados (el específico) se considera que es una forma especializada del general. En la práctica, esto significa que cualquier instancia (elemento concreto) del bloque específico es también una instancia del bloque general (el sensor de temperatura es un sensor).

La relación general es:

- **Dependency**(

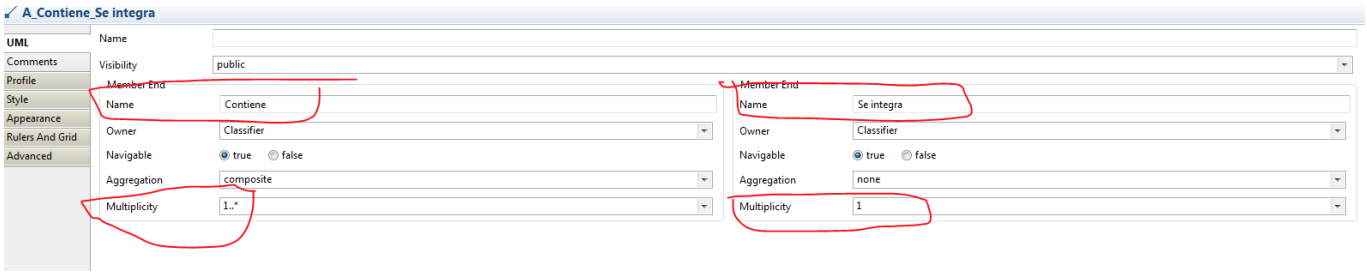
 ):
  La dependencia es una forma más débil de relación que indica que un bloque depende de otro elemento (otro bloque, un actor, un caso de uso, etc). En el caso de depender un bloque de otro el significado es diferente de una asociación, donde un atributo del bloque es dependiente es una instancia del bloque asociado.

Definiendo en Papyrus



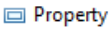
Una relación puede tener vinculado un nombre (*Nombre Relación*), y en los extremos, puede tener asociado roles (*Contiene, Se integra*), indicadores de propiedad, multiplicidad ([1..*],[1]), visibilidad y otras propiedades.

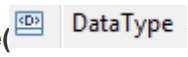
Lo podemos definir en la pestaña UML de la relación:

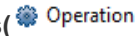


Esta relación describe que una máquina contiene 1 o más sensores, y un sensor se integra en 1 y sólo 1 una máquina. Recordemos que esta relación es a nivel de instancia (una máquina concreta se relaciona con un sensor concreto).

Definiendo propiedades, tipos y operaciones dentro de un bloque

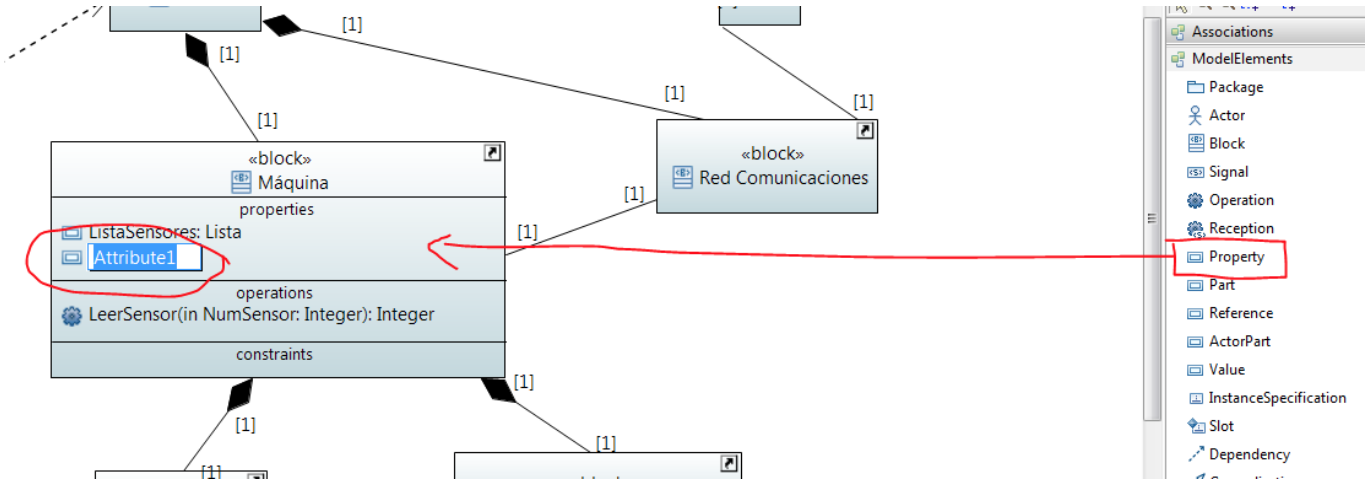
Properties ( **):** Son los miembros de nuestro bloque, variables comunes a todas las instancias de él. En el ejemplo de la Historia de Antonio, en el bloque máquina tendríamos por ejemplo una lista de sensores (algunas máquinas podían tener a parte del sensor de temperatura uno de presión o velocidad).

DataType( **):** Tipos de elementos o datos que se pueden definir para ser usados en las propiedades o operaciones de un bloque.

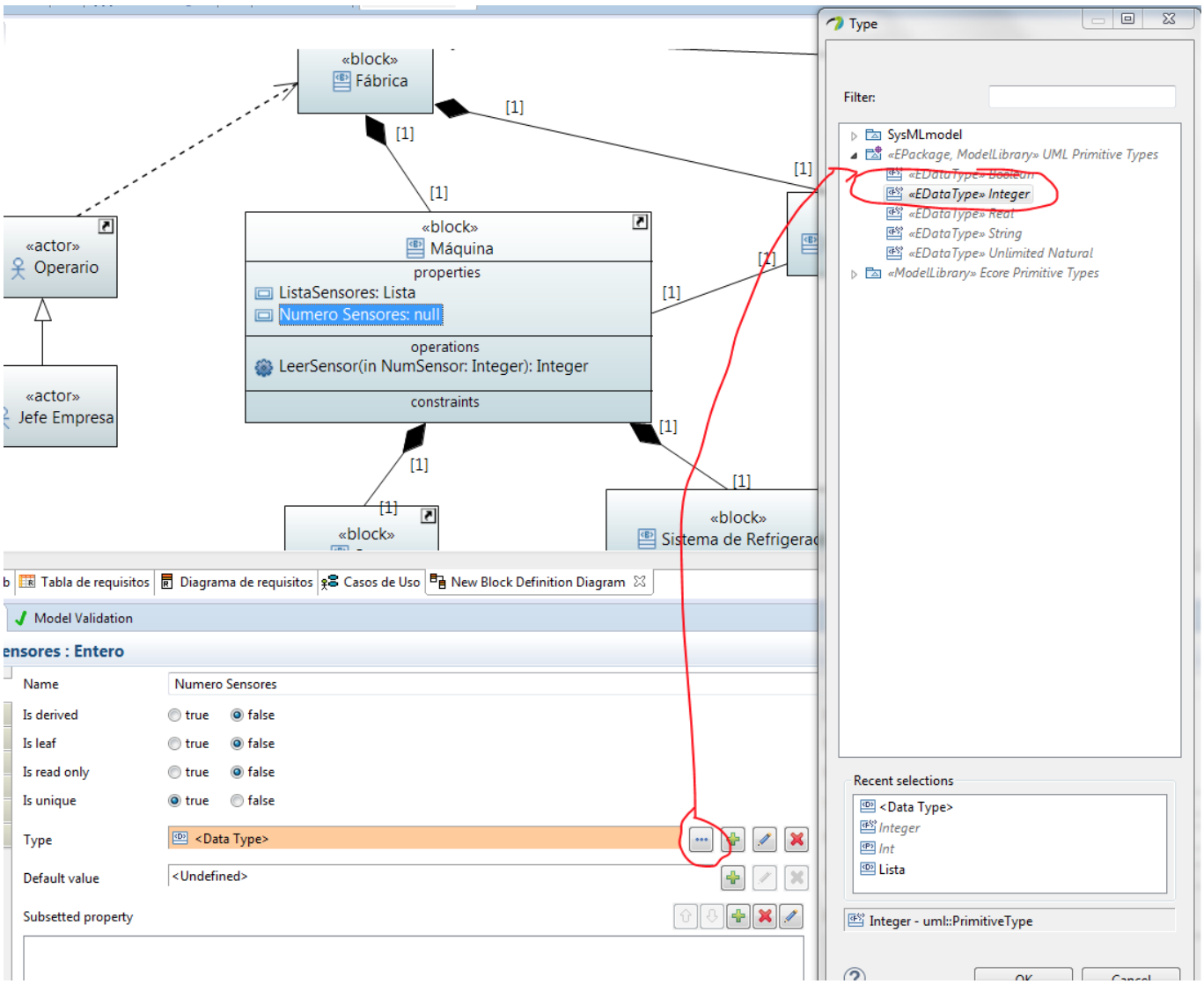
Operations( **):** Operaciones que realiza el bloque, se pueden especificar entradas y salidas en estas operaciones.

Propiedades

Para definir una propiedad, seleccionamos el elemento *Property* y hacemos click sobre la zona *properties* del bloque:

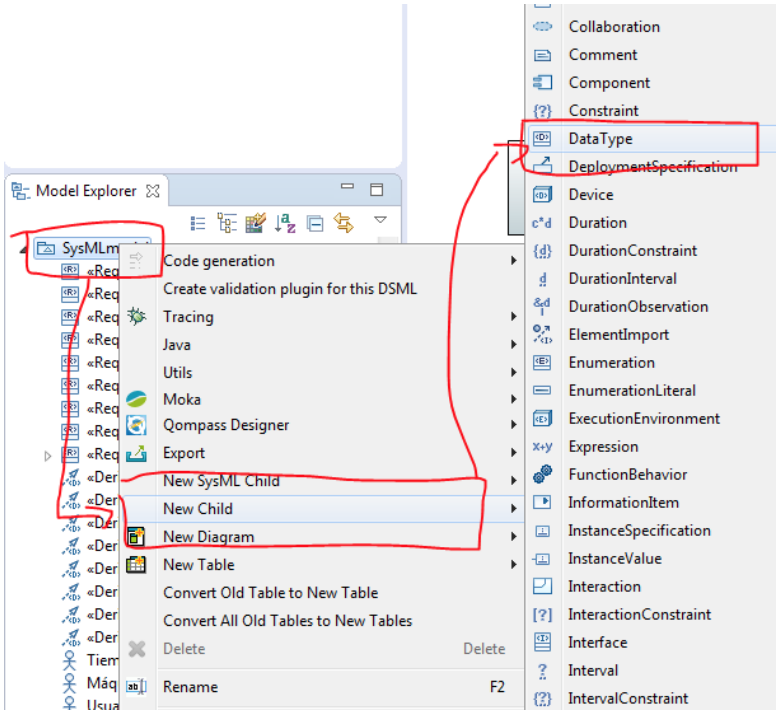


Una vez creado, podemos asignarle un nombre y algunas propiedades. Dentro de las propiedades más interesantes está el tipo (*Type*), valor por defecto y multiplicidad. En la imagen siguiente os mostramos como se puede seleccionar el tipo dentro de los existentes:

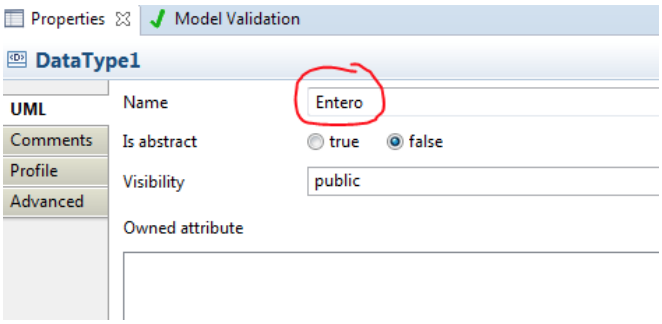


Tipos de datos

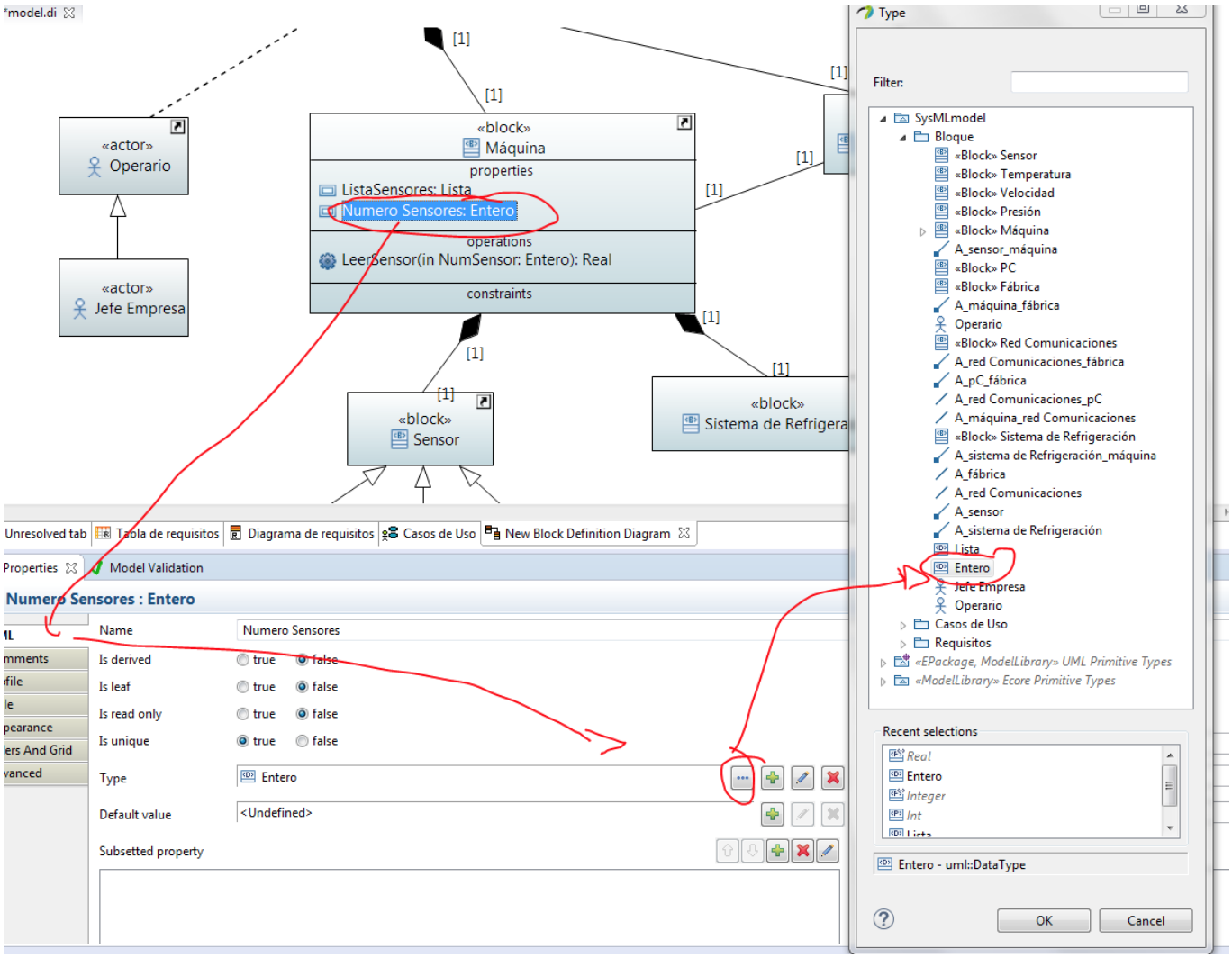
En caso de querer definir nuestros propios tipos para usarlos en las propiedades o las operaciones, podemos agregar un nuevo elemento a nuestro modelo que sea de tipo *DataType*:



De este nuevo tipo de datos podemos especificar su nombre:

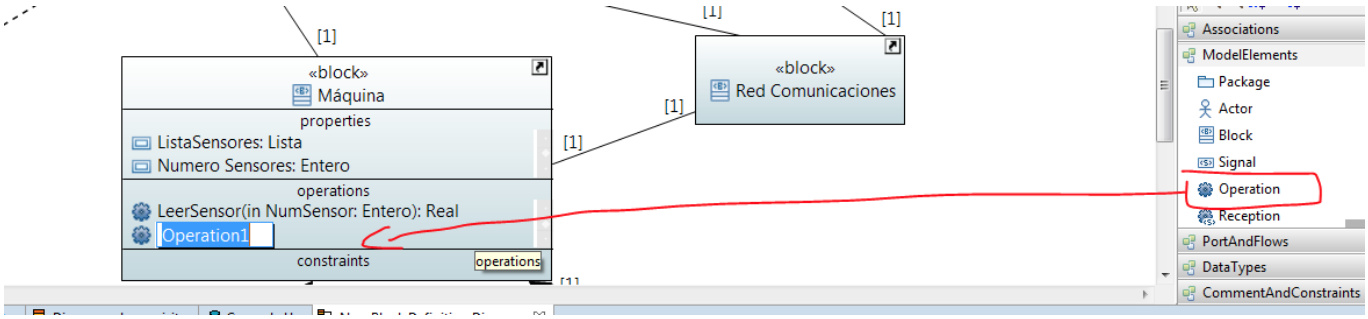


Una vez definido podemos asignar el nuevo tipo a la propiedad que deseemos, quedando los tipos del bloque personalizados:

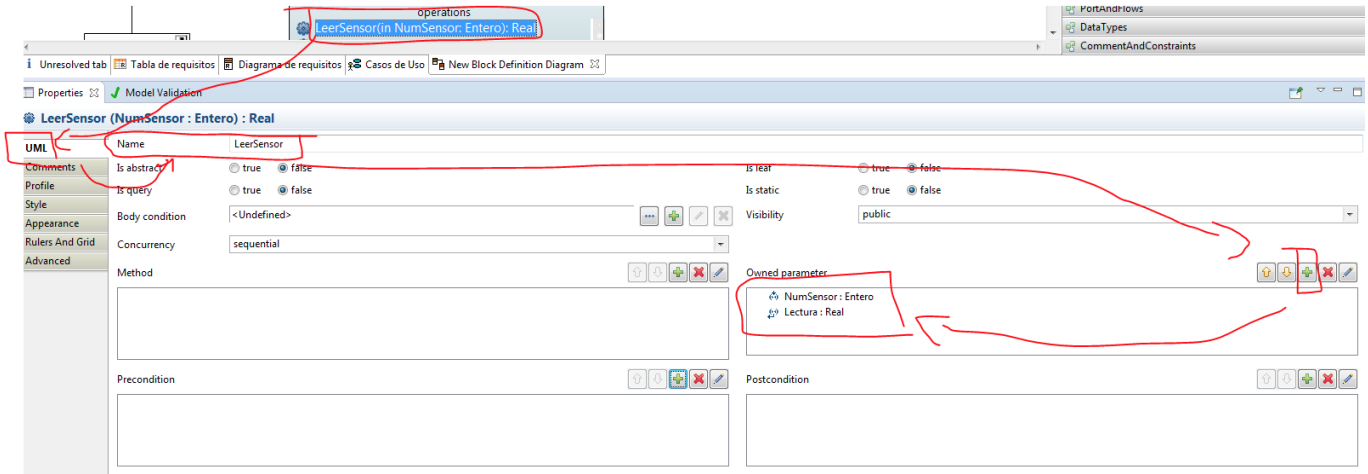

*model.di
«actor»
Operario
«actor»
Jefe Empresa
«block»
Máquina
properties
ListaSensores: Lista
Numero Sensores: Entero
operations
LeerSensor(in NumSensor: Entero): Real
constraints
«block»
Sensor
«block»
Sistema de Refrigera
[1]
Unresolved tab
Tabla de requisitos
Diagrama de requisitos
Casos de Uso
New Block Definition Diagram
Properties
Model Validation
Numero Sensores : Entero
Name
Numero Sensores
Is derived
Is leaf
Is read only
Is unique
true
false
Type
Entero
Default value
<Undefined>
Subsetted property
Type
Filter:
SysMLmodel
Bloque
«Block» Sensor
«Block» Temperatura
«Block» Velocidad
«Block» Presión
«Block» Máquina
A_sensor_máquina
«Block» PC
«Block» Fábrica
A_máquina_fábrica
Operario
«Block» Red Comunicaciones
A_red Comunicaciones_fábrica
A_pC_fábrica
A_red Comunicaciones_pC
A_máquina_red Comunicaciones
«Block» Sistema de Refrigeración
A_sistema de Refrigeración_máquina
A_fábrica
A_red Comunicaciones
A_sensor
A_sistema de Refrigeración
Lista
Entero
Jefe Empresa
Operario
Casos de Uso
Requisitos
«EPackage, ModelLibrary» UML Primitive Types
«ModelLibrary» Ecore Primitive Types
Recent selections
Real
Entero
Integer
Int
Entero - uml::DataType
OK
Cancel

Operaciones

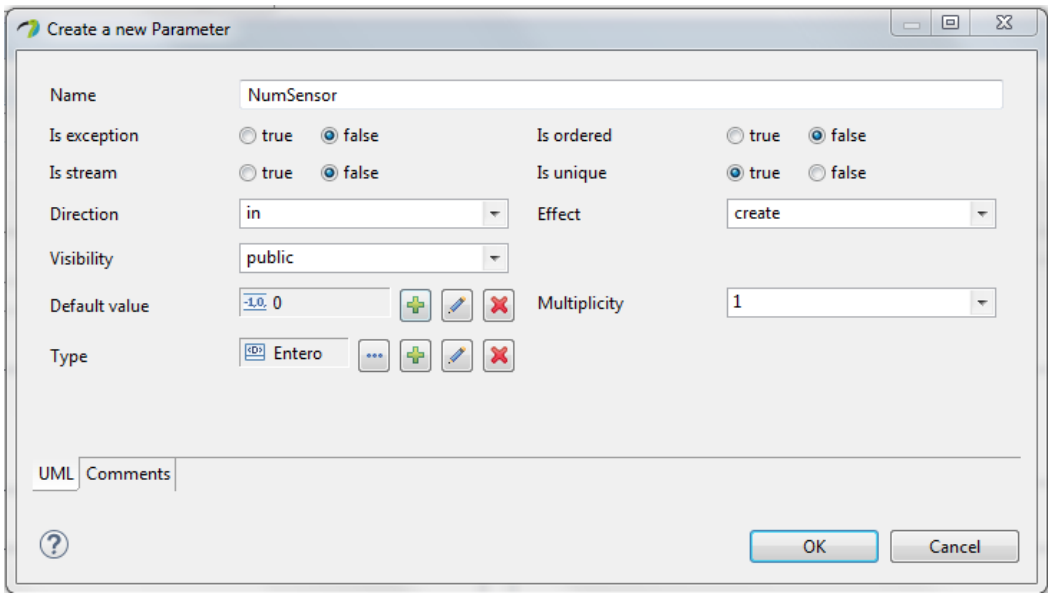
Para definir una operación, seleccionamos el elemento *Operation* y hacemos click sobre la zona *operations* del bloque:



Tenemos un amplio abanico de posibilidades al definir una operación, vamos sólo a comentar como definir las variables de entrada y salida de la operación. Para ello vamos al área *Owned Parameter* de la pestaña *UML* de la operación:



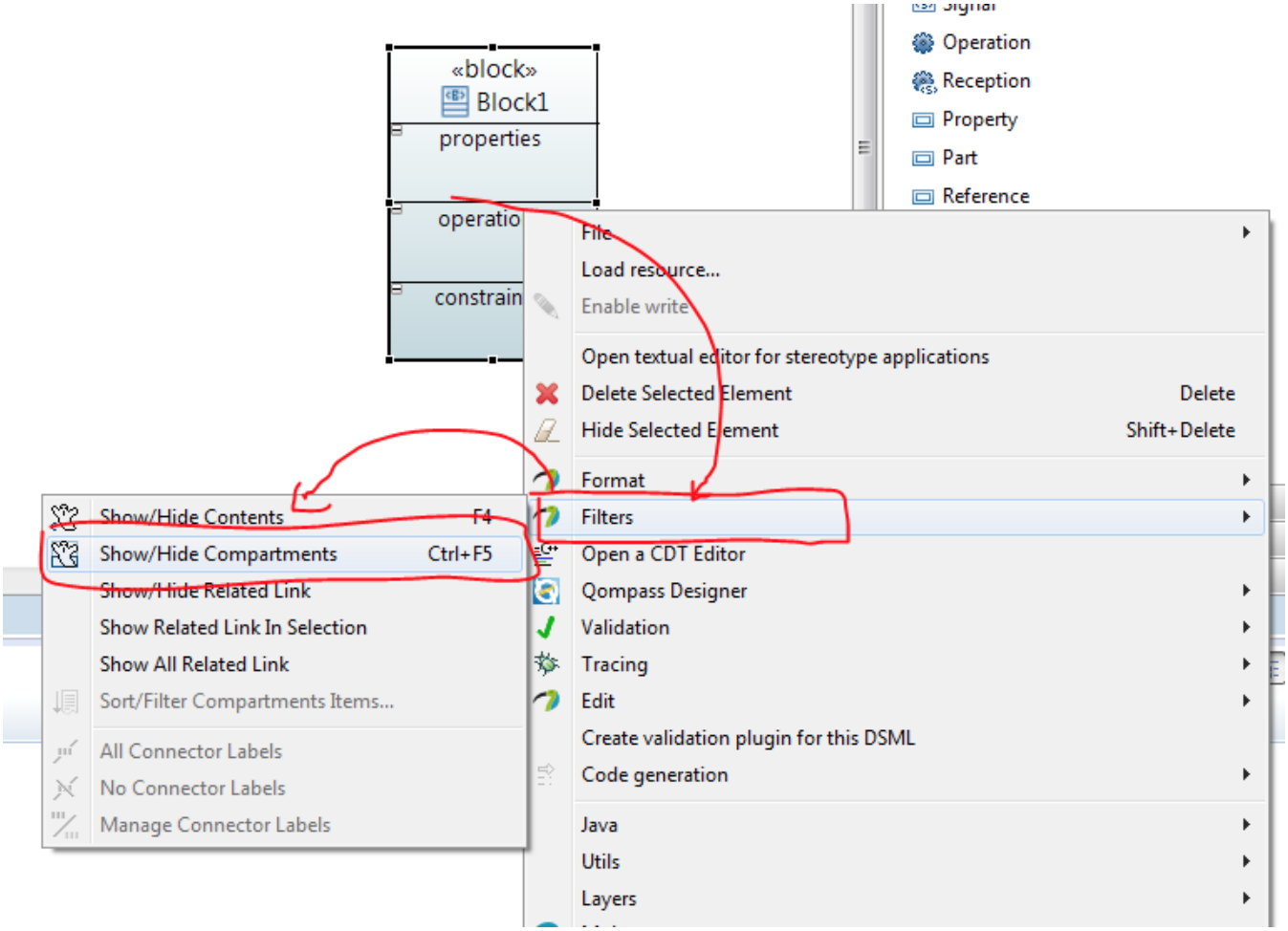
Una vez pulsamos el +, nos encontramos con el siguiente formulario para definir una variable de nuestra operación:



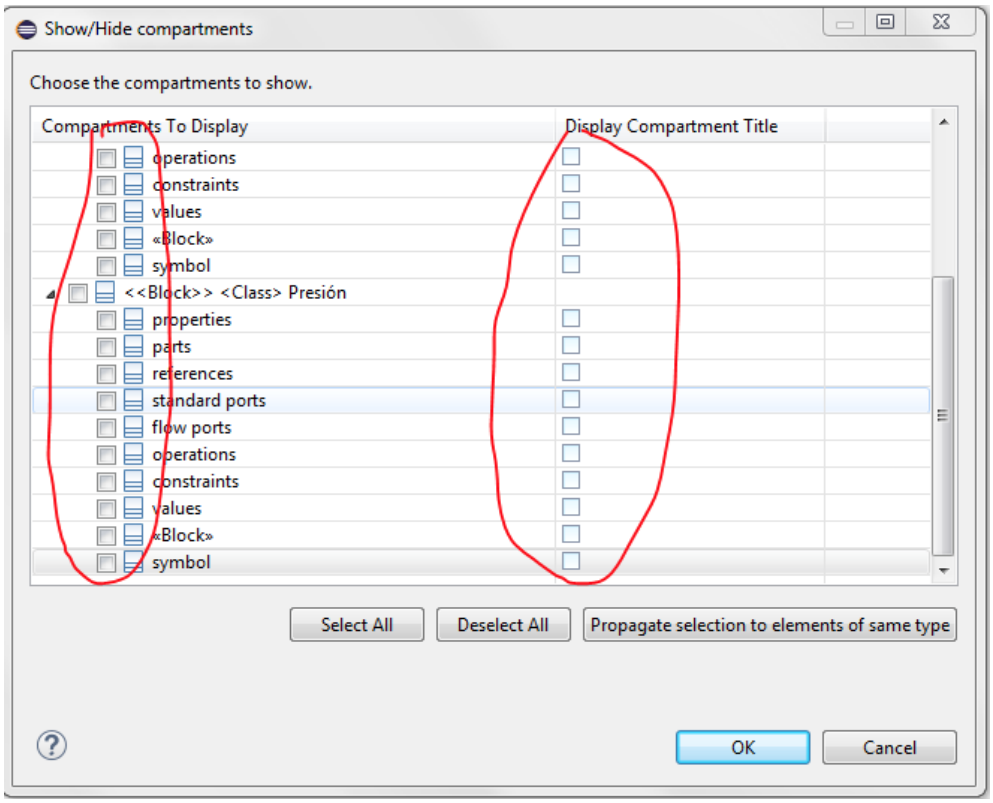
Primero le damos nombre al parámetro (*NumSensor*), seleccionamos si es de entrada (le pasamos este valor a la operación), salida (la operación nos devuelve el valor) o entrada/salida. Tras esto podemos definir el tipo y el valor por defecto como en el caso de las propiedades. Con estos campos ya tenemos definida nuestra variable de entrada NumSensor que es un entero y como valor por defecto toma el 0 LeerSensor(in NumSensor: Entero):

Bloques sin nada definido

Si no se va a definir ninguno de estos elementos se pueden ocultar pulsado sobre el bloque con el *botón de la derecha > Filters > Show/Hide Compartment*s:



Una vez en este menú, deseleccionamos de ambas columnas todo el contenido:



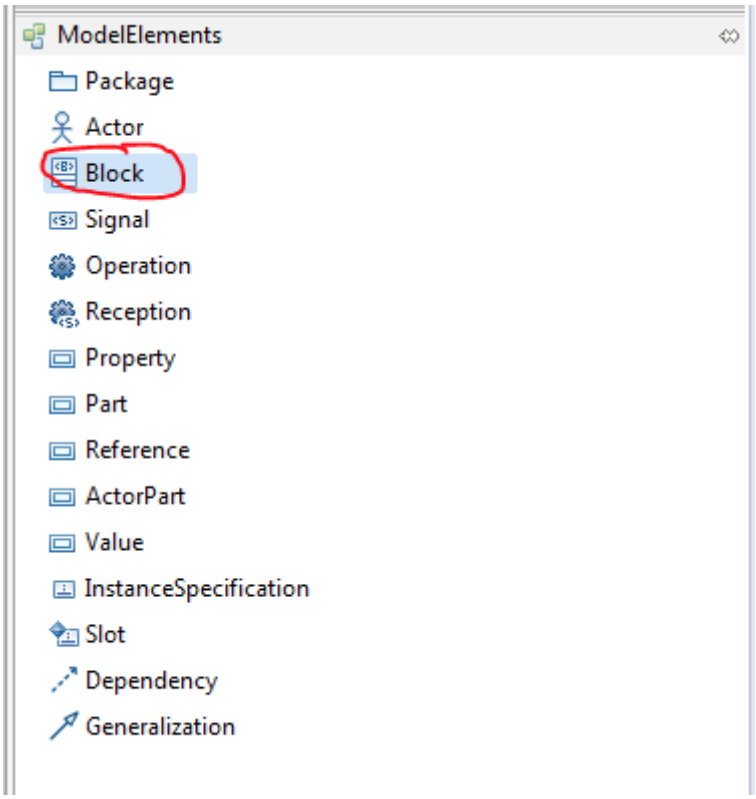
Quedándonos sólo con el nombre del bloque:

Crear un Diagrama de bloques

Al igual que en los otros diagramas, pulsamos en la ventana del model explorer sobre nuestro nuestro modelo y a continuación vamos al menú *Papyrus > New Diagram > Create a new SysML Block Definition* elegimos un nombre y aceptamos.

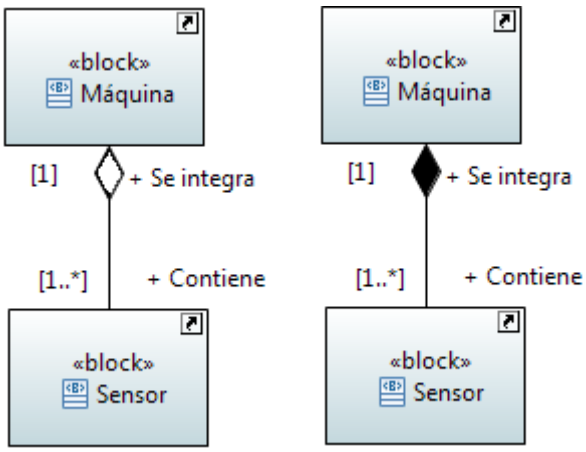
Por comodidad puedes crear un paquete para ir añadiendo todo el contenido del diagrama dentro (*botón de la derecha en model explorer > New Child > Package*).

Una vez creado y abierto el diagrama, puedes crear un bloque pulsado en la pestaña *ModelElements* sobre el ítem *block* y luego sobre el diagrama. También puedes generarlo directamente desde el *model explorer* pulsando en el *botón de la derecha sobre el paquete > New SysML child > Block*):



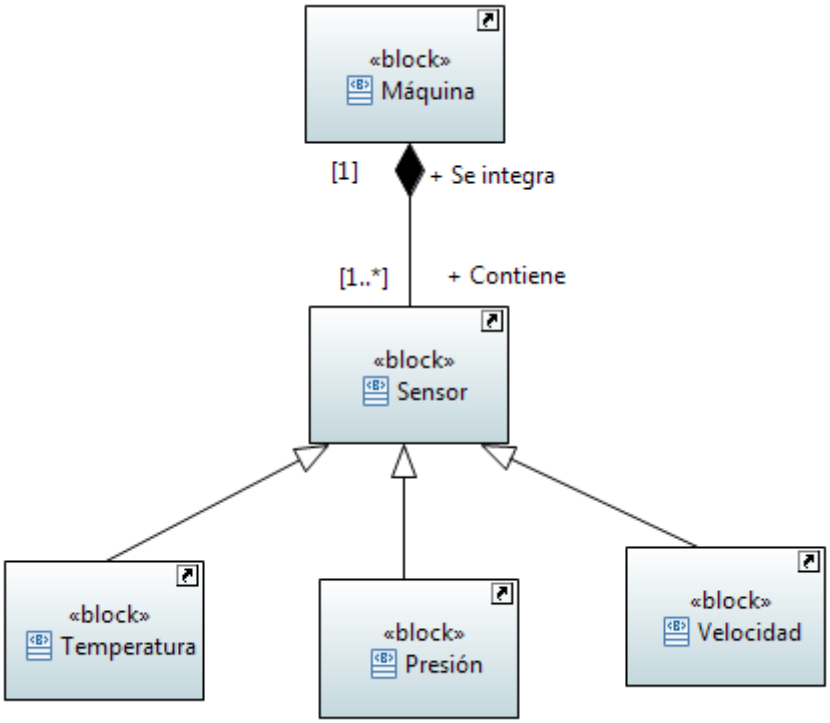
Historia de Antonio: Arquitectura Física

La arquitectura física de un proyecto describe los elementos hardware del mismo y como estos se relacionan. En el caso de la Historia de Antonio podemos identificar claramente Máquinas y Sensores (temperatura, presión y velocidad) ¿Cómo podemos relacionarlos? Una máquina en concreto tiene una serie de sensores incorporados, por tanto en principio podemos modelarlo como una **composición** o como una **agregación**:

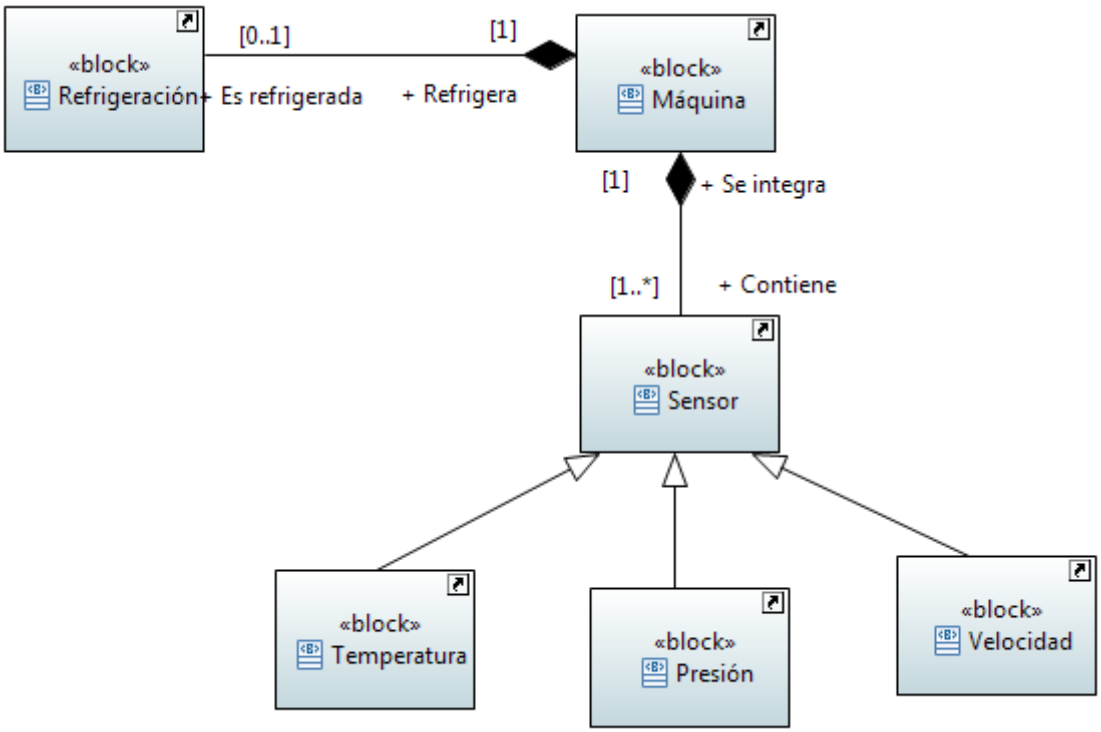


Para saber cual usar hay que preguntarse si la "existencia" del sensor está ligada a la de la máquina. Aunque no tenemos el diagrama terminado, sabemos que los sensores en esta fábrica sólo tienen utilidad asociados a una máquina, por tanto lo modelamos como una **composición**.

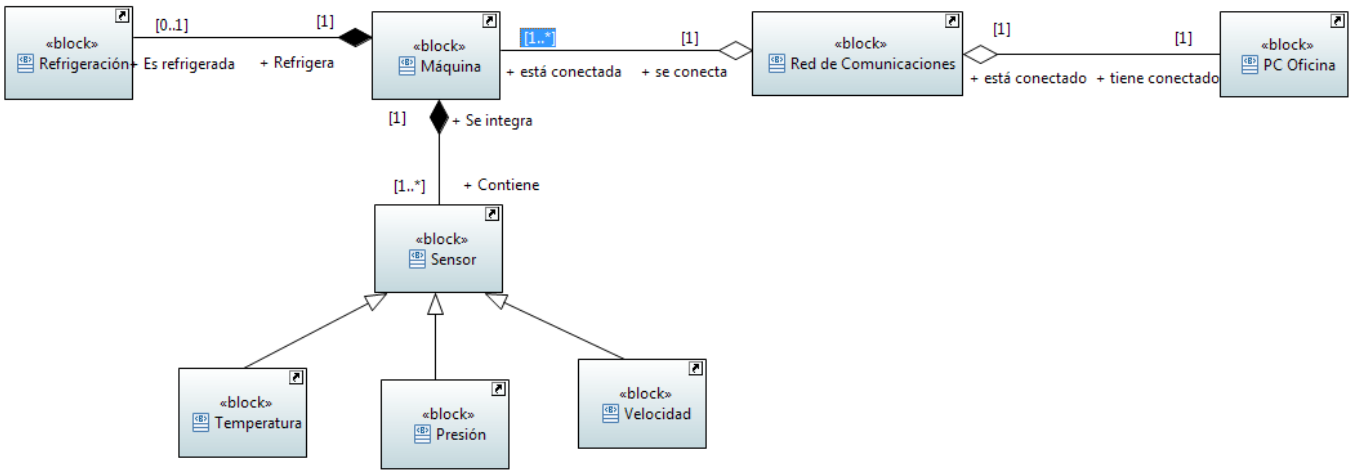
Sabemos que tenemos diferentes tipos de sensores (temperatura, velocidad y presión), esta relación es entre tipos (entre sensores en general, no como caso particular), por tanto las relaciones que pueden existir se limitan a dependencia y generalización. Como un sensor de temperatura es un sensor, lo modelamos como una generalización:



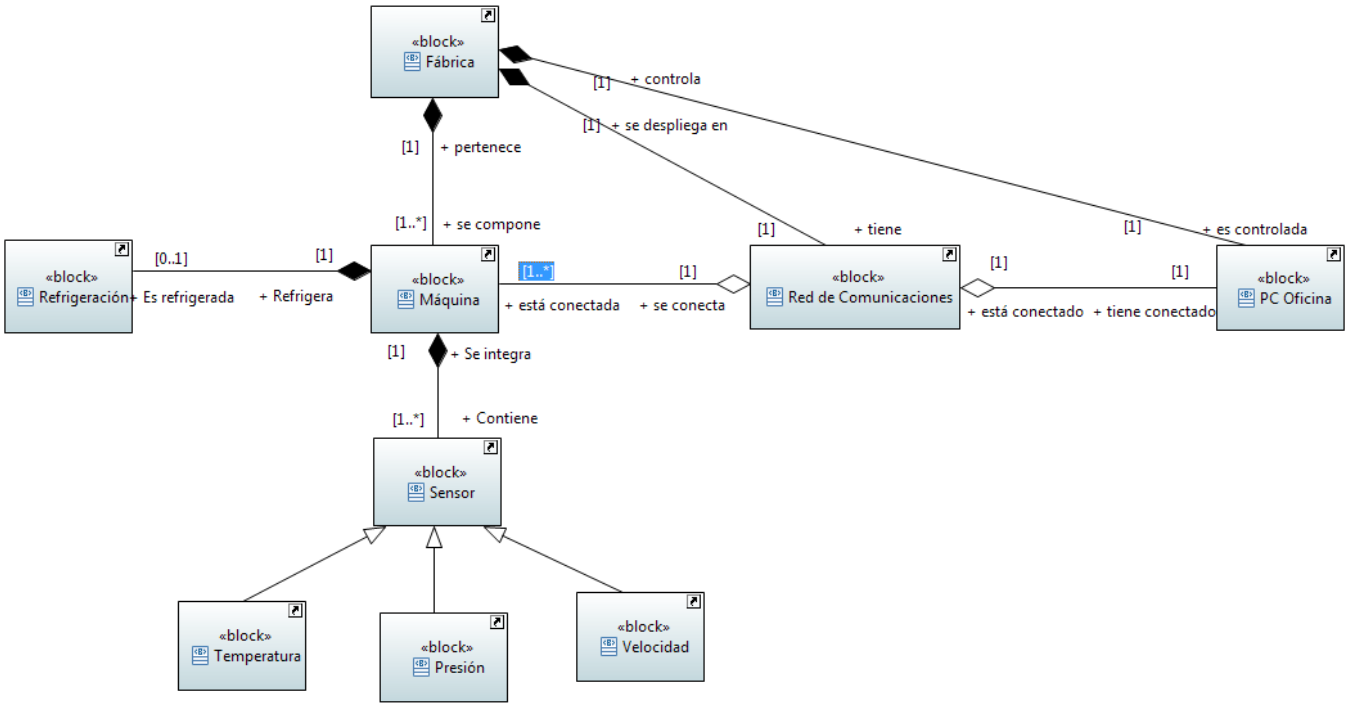
Una máquina también puede tener un sistema de refrigeración (el *puede tener* se modela con la multiplicidad), al igual que en el caso de los sensores lo modelamos como una composición (mismas razones):



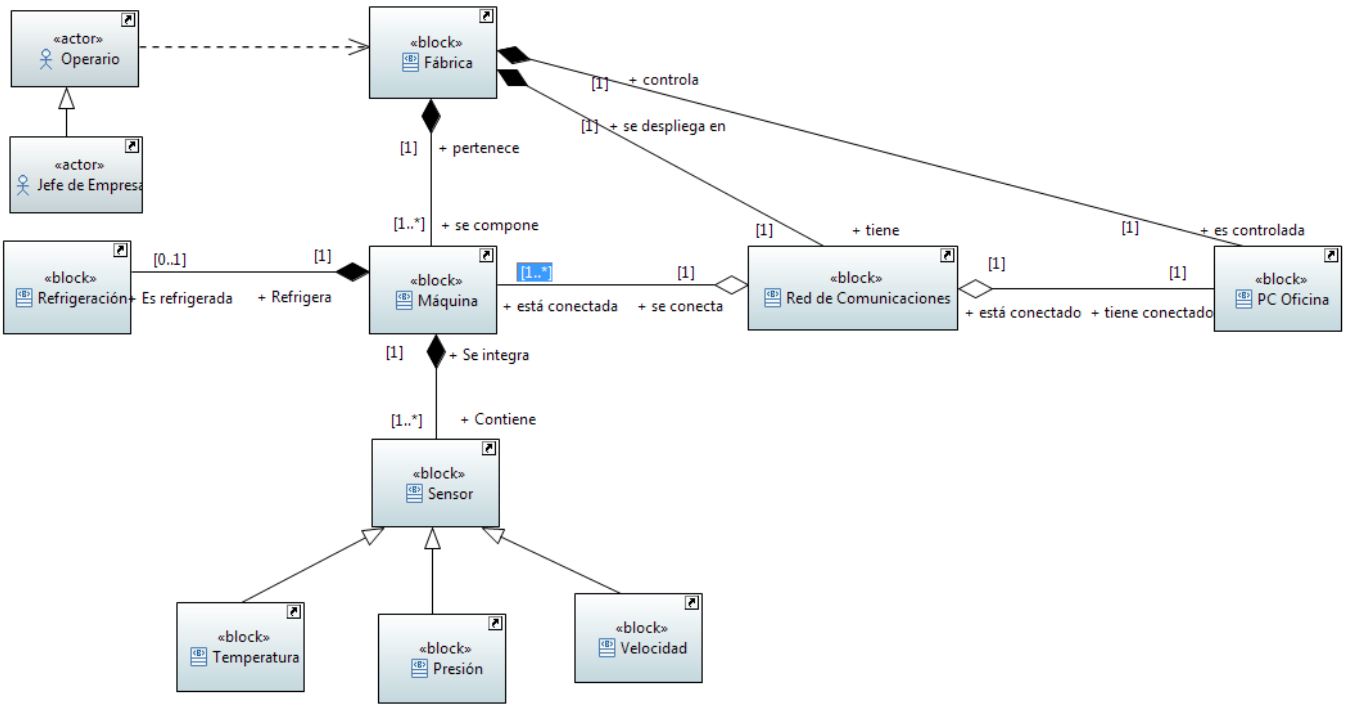
La red interna tiene todas las máquinas conectada a ella para poder leer de los sensores. Las maquinas no dependen de la existencia de la red, por tanto lo modelamos como una agregación. En esta red también está el PC de la oficina conectado:



Todo esto que hemos definido es lo que compone una fábrica de plásticos:



En esta arquitectura podemos hablar de los actores que sean entidades físicas. En nuestro sistema tenemos el operario (como es un actor y no es un bloque sólo podemos relacionarlo usando la dependencia), También tenemos un caso especial de actor que es el jefe de empresa (que es un operario, por tanto relación de generalización):

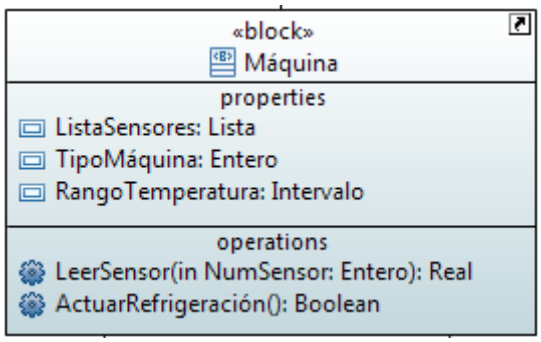


El diagrama de bloques mostrado anteriormente describe la arquitectura física del sistema, esta incluye los actores físicos y todos los elementos hardware del sistema de monitorización y control.

Historia de Antonio: Arquitectura Lógica

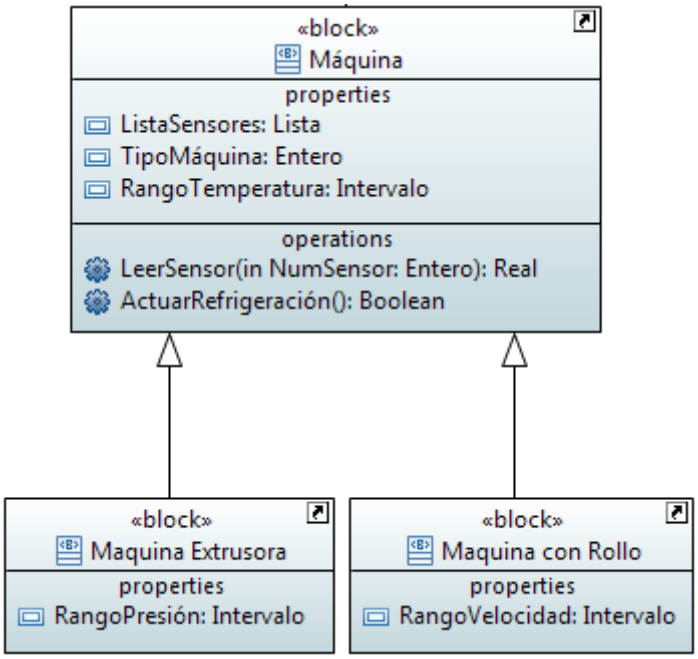
La arquitectura lógica de un proyecto describe los elementos software del mismo y como estos se relacionan.

En el caso de la Historia de Antonio podemos identificar claramente un componente software Máquina, que es el que va a proporcionar al sistema todos los valores de los sensores que tenga (en implementación se verá como se conecta a los sensores y obtiene información, ahora estamos a alto nivel). Este componente también nos va a permitir actuar sobre la refrigeración. Cómo todas las máquinas tienen estas características podemos modelarlo como un solo bloque:

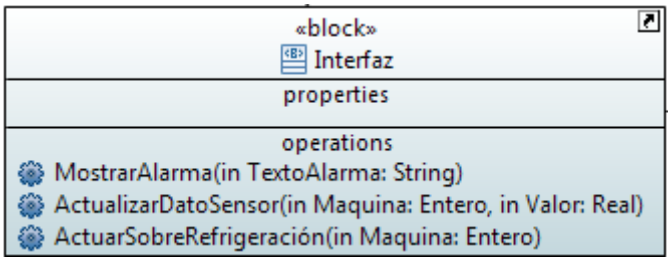


Observamos como hemos decidido en este momento, diseñando la arquitectura, que sea el componente Máquina el que se encarga de almacenar y controlar el rango de temperatura.

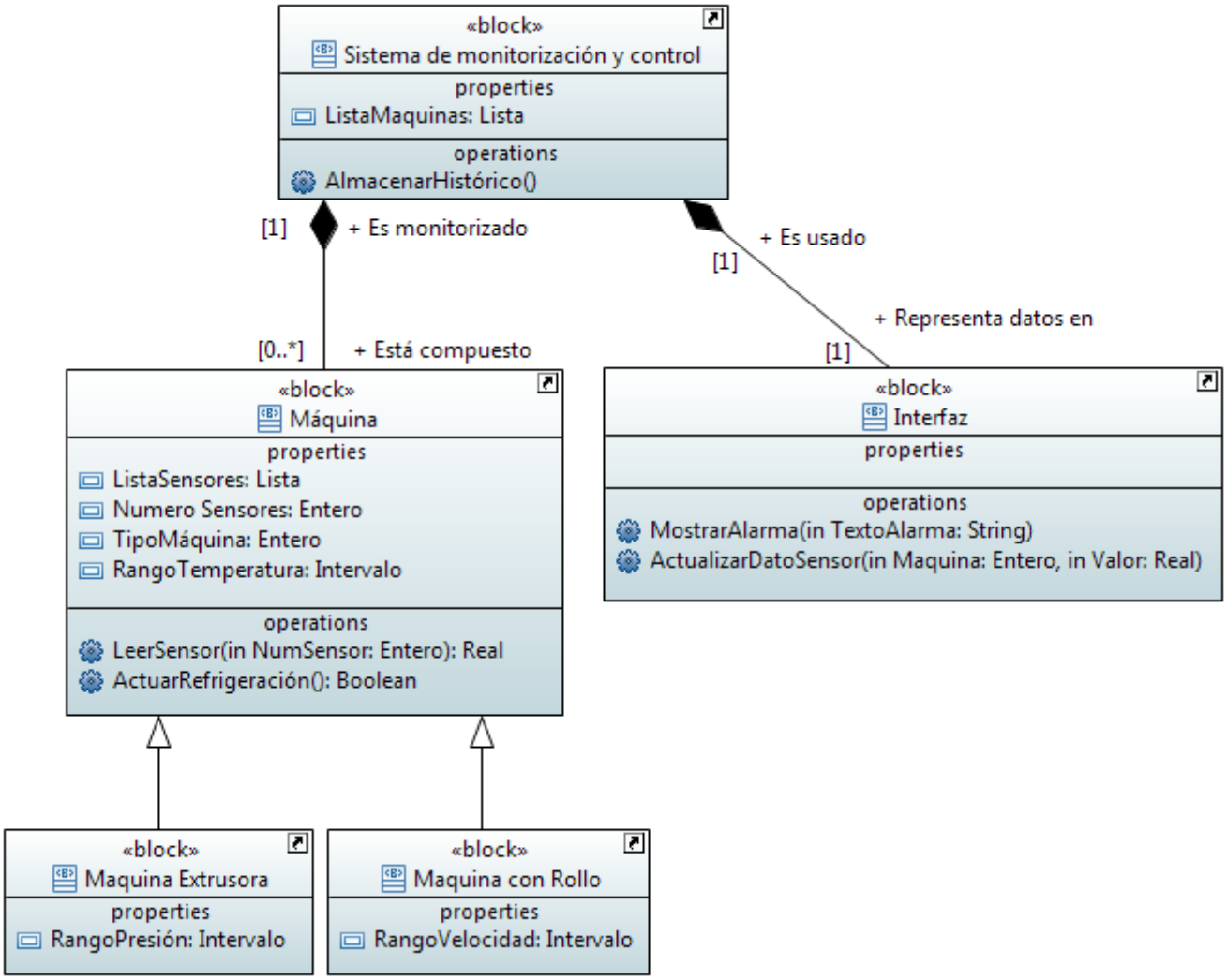
Tenemos dos tipos de máquinas más específicas, las máquinas de rollo y extrusoras, las cuales tienen rangos específicos para sus sensores (más propiedades a parte de las que el componente Máquina tiene). Claramente esta relación entre componentes es una generalización (La máquina extrusora y la máquina de rollo son máquinas):



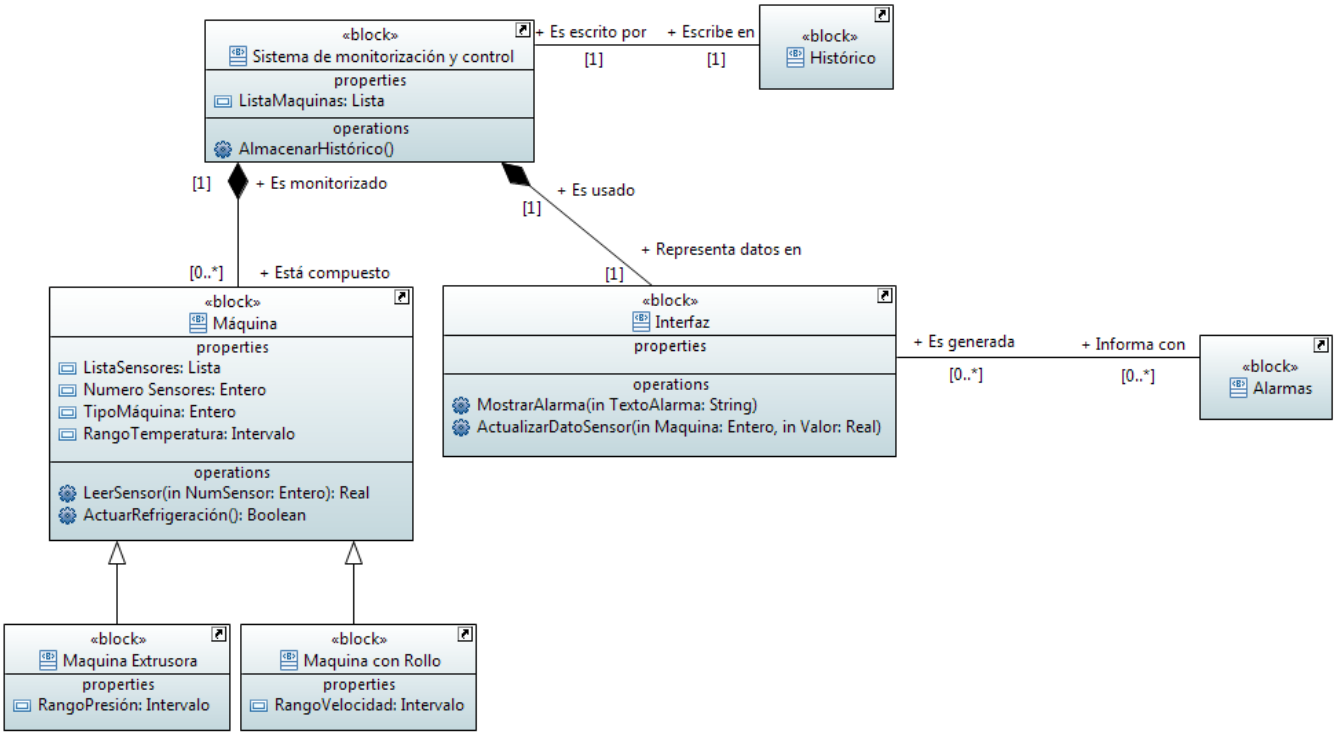
También existe un componente (lo llamaremos *Interfaz*) que es el encargado de mostrar las alarmas y los valores por pantalla:



El sistema de monitorización y control está compuesto por un conjunto de máquinas, de las que lee valores y sobre las que puede actuar en su refrigeración; también está compuesto por una interfaz gráfica que informa al usuario y le permite indicar que máquina debe refrigerarse. Esta relación se puede definir como una **agregación** o como una **composición**. Aunque el diagrama aún no ha sido completado, sabemos que ambos componentes no existirán sin el sistema de monitorización y control, por tanto la relación la definimos como una composición:



Existen dos funcionalidades más que no han sido descritas aún: guardar historial y generar alarmas. Ambas funcionalidades no componen el sistema de monitorización y control (no podemos usar la agregación o la composición) pero si se relacionan con él o con algunos de sus componentes. En concreto, generar alarmas se relaciona con el componente interfaz (El Interfaz informa con Alarmas o Las alarmas son generadas por el interfaz) y guardar historial se relaciona con el Sistema de monitorización y control (El histórico es escrito por el sistema de monitorización y control o el sistema de monitorización y control escribe en el histórico):



Por último, al igual que en el caso anterior, podemos representar como los actores interactúan con el interfaz (el actor Tiempo es una creación ficticia para los casos de uso, internamente dónde sea necesario se usará un timer para llamar periódicamente a las funciones que lo requieran, no es parte de la arquitectura):