

DESEÑO DUN SISTEMA DE CONFIGURACIÓN DUN FOGAR INTELIXENTE

Víctor Xesús Barreiro Domínguez
Manuel Bendaña Gómez
David Carracedo Montero
Daniel Chenel Cea

Deseño de Software
2º Grao en Enxeñaría Informática
Universidade de Santiago de Compostela
Curso 2019/2020

Profesores:
José Varela Pet
Xosé Manuel Pardo López

Índice

1. Introducción - Descripción do problema	5
1.1. Introducción	5
1.1.1. Obxectos Intelixentes	6
1.1.2. Desenvolvemento de usuario final (EUD)	9
1.2. Sistema de configuración dun fogar Intelixente (SCFI)	10
1.2.1. Rede de sensores de propósito xeral	11
1.3. Arquitectura do SCFI	12
1.3.1. Conexións	12
1.3.2. Xestión	12
1.3.3. Control Interno	13
2. Casos de uso	15
2.1. Diagrama de casos de uso	15
2.2. Descripción detallada dos casos de uso	16
2.2.1. Caso de uso 1: Rexistrar SO	16
2.2.2. Caso de uso 2: Eliminar SO	17
2.2.3. Caso de uso 3: Seleccionar SO	17
2.2.4. Caso de uso 4: Configurar SO	18
2.2.5. Caso de uso 5: Configurar SO por formulario	19
2.2.6. Caso de uso 6: Configurar SO mediante un editor	19
2.2.7. Caso de uso 7: Configurar SO no modo avanzado	20
2.2.8. Caso de uso 8: Planificar actuación	20
2.2.9. Caso de uso 9: Visualizar información	21
3. Diagramas de Actividade	21
3.1. Unha ferramenta para describir accións	21
3.2. Diagrama de Actividade 1: Rexistrar SO	22
3.3. Diagrama de Actividade 2: Eliminar SO	22
3.4. Diagrama de Actividade 3: Seleccionar SO	25
3.5. Diagrama de Actividade 4: Configurar SO por formulario	25
3.6. Diagrama de Actividade 5: Configurar SO por editor	28
3.7. Diagrama de Actividade 6: Configurar SO no modo avanzado	28

3.8. Diagrama de Actividade 7: Planificar actuación	31
3.9. Diagrama de Actividade 8: Visualizar información	32
4. Interface de Usuario	33
4.1. Pantalla principal	33
4.2. Pantalla de información	33
4.3. Pantalla de xestión de obxectos	33
4.3.1. Pantalla Inserción	34
4.3.2. Pantalla Eliminar	34
4.3.3. Pantalla de Selección	34
4.4. Pantalla de configuración	35
5. Casos de Proba	38
5.1. Caso de Proba 1	38
5.2. Caso de Proba 2	38
5.3. Caso de Proba 3	38
5.4. Caso de Proba 4	38
5.5. Caso de Proba 5	39
5.6. Caso de Proba 6	39
5.7. Caso de Proba 7	39
5.8. Caso de Proba 8	39
5.9. Caso de Proba 9	40
5.10. Caso de Proba 10	40
5.11. Caso de Proba 11	40
5.12. Caso de Proba 12	40
5.13. Caso de Proba 13	41
5.14. Caso de Proba 14	41
5.15. Caso de Proba 15	41
5.16. Caso de Proba 16	41
5.17. Caso de Proba 17	41
5.18. Caso de Proba 18	42
5.19. Caso de Proba 19	42
5.20. Caso de Proba 20	42

6. Diagrama de Compoñentes	42
6.1. GPSN: Rede de comunicación	43
6.2. SCFI: Sistema de Control do Fogar Intelixente	44
6.3. Interfaz Gráfica	46
7. Conceptos susceptibles de ser clases	46
8. Patróns	48
8.1. Patrón <i>Strategy</i> (Estratexia)	48
8.2. Patrón <i>State</i> (Estado)	48
8.3. Patrón <i>Abstract Factory</i> (Fábrica Abstracta)	50
8.4. Patrón <i>Proxy</i> (Apoderado)	51
9. Diagrama de clases	51
10. Diagramas de Secuencia	55
10.1. Diagramas Adicionais	55
10.1.1. Recuperar SOs	55
10.1.2. Solicitar Información SO	56
10.1.3. Recuperar SO DID	57
10.2. Rexistrar SO	58
10.3. Eliminar SO	59
10.4. Seleccionar SO	60
10.4.1. Activar SO	60
10.4.2. Desactivar SO	61
10.5. Configurar SO	62
10.5.1. Configurar SO por Formulario	62
10.5.2. Configurar SO por Editor	63
10.5.3. Configurar SO no modo avanzado	63
10.6. Planificar actuación	65
10.7. Visualizar información	67
11. Conclusións	68

Abstract. Neste informe adicáremos a amosar o resultado do noso proceso de deseño dun Sistema de Configuración dun Fogar Intelixente (SCFI), que foi desenvolto a través de diferentes fases. Dividiremos o documento en varias partes, que se corresponden coas diferentes actividades que se foron proponendo ao longo do curso: começaremos por realizar unha descripción completa do problema a desenvolver, seguida da presentación dos casos de uso tanto a través dun diagrama coma da descripción individual de cada un deles. Logo pasaremos a amosar os diagramas de actividade; xusto a continuación, presentaremos a nosa proposta de deseño da interface de usuario, de cada unha das súas pantallas, e seguirase coa presentación dos diferentes casos de proba que consideramos necesarios. A partir de entón centrarémonos noutros diagramas: o de compoñentes, o de clases, que virá precedido dunha descripción de conceptos que consideramos clases e dos patróns de deseño aplicados no documento, e os de secuencia. Ao final, adicaremos unhas liñas a facer unha reflexión final sobre o traballo desenvolto neste documento.

Palabras Clave: IoT, Obxectos Intelixentes, EUD, GPSN, SCFI, UML, Casos de Uso, diagramas, Deseño.

1. Introdución - Descripción do problema

1.1. Introdución

Comezamos este proxecto presentando o problema de deseño que se aborda no mesmo, o deseño dun Sistema de Configuración dun Fogar Intelixente.

A difusión da Internet das cousas (IoT), dirixida a conectar todos os obxectos do mundo físico, está afectando a diferentes dominios como a asistencia sanitaria, a automatización industrial, o control intelixente de enerxía, a asistencia de persoas maiores, a seguridade pública, a xestión urbana, a construcción de infraestruturas, ou as casas intelixentes. Por iso, nos próximos anos espérase que se conecten mil millóns de obxectos intelixentes grazas ás tecnoloxías IoT, aínda que hai que afrontar varios retos sobre a fiabilidade dos datos, e as necesidades de interoperación e eficiencia.

A IoT permite conectar obxectos intelixentes e controlalos de xeito remoto mediante aplicacións instaladas nos dispositivos intelixentes. O principal problema das aplicacións de detección baseadas en sensores IoT é que as medidas individuais dos sensores dun único dispositivo IoT son en xeral insuficientes para tarefas de detección complexas e fiables. Para xestionar isto, introducíense sensores virtuais (VS) para operar na nube de sensores como abstracción dos dispositivos físicos.

En lugar de analizar as medicións dun único sensor, o VS pode usar medicións de sensores independentes de varios dispositivos e executar un algoritmo de estimación / predición para cumplir de xeito eficiente coa súa tarefa. Por exemplo, un VS pode usar un sistema de posicionamento global (GPS) para detectar a situación do dispositivo, ou pode estimar a situación en función das posicións doutros sensores de localización próximos cando se atopa en interiores. Os VS tamén se poden ver como sensores software que proporcionan medicións indirectas de condicións abstractas, combinando datos captados por sensores físicos heteroxéneos. Polo tanto, os VS poden realizar un amplio conxunto de operacións, como a fusión de datos, a agregación de datos, ou o manexo de modelos de predicción.

Os principais beneficios dos VS son os seguintes: (1) a reutilización de redes multiobxectivo despregadas para dar soporte a diferentes aplicacións, de xeito que poden proporcionar diferentes tipos de información dependendo das aplicacións particulares; (2) a heteroxeneidade, e (3) o enmascarado das fontes de datos, ao actuar como mediadores/...proxies entre sensores físicos e aplicacións clientes.

Neste proxecto, interésanos a aplicación da IoT aos fogares intelixentes. O termo fogar intelixente evolucionou desde a referencia exclusiva ao control centralizado e semiautomatizado dos sistemas ambientais, como o calor e as luces, ao uso de tecnoloxía para controlar calquera obxecto compatible co ambiente doméstico.

Normalmente, o obxectivo dos sistemas domésticos intelixentes é proporcionar servizos de comodidade, asistencia sanitaria, seguridade e consumo de enerxía.

Comezaremos describindo antes de nada algúns conceptos relevantes para a IoT, especialmente no ámbito dos fogares intelixentes:

A Figura 1 describe o diagrama de secuencia UML do acceso dun obxecto cliente aos datos proporcionados por un sensor virtual: no eixe horizontal están marcados os obxectos implicados, ou sexa, controlador de fonte de datos externa (Data Source), cliente (Client) e VS (Virtual Sensor); e no eixo vertical amósase a secuencia de mensaxes intercambiadas e o tempo.

A Figura 2 amosa un exemplo dun diagrama de clases para un sistema de mobilidade eléctrica, que conta con varios servizos, e cada un deles manexa datos de diferentes sensores (físicos ou virtuais). Neste caso o “Physical Sensor” (ou Data Source) e o “Virtual Sensor” poderían actuar como clases abstractas das que se poden derivar outras más específicas.

O patrón operativo proposto permite tratar por separado diferentes aspectos dun VS (por exemplo, aspectos técnicos e comerciais) que poden ser útiles tanto desde o punto de vista do desenvolvedor como do punto de vista xurídico/comercial.

1.1.1. Obxectos Intelixentes

Os obxectos intelixentes (SO) son os bloques fundamentais da IoT e combinan fontes de datos conectadas a obxectos físicos e os sensores virtuais asociados, con outros elementos software que permiten configurar o comportamento do obxecto e enviar comandos aos controladores dos seus actuadores.

Un obxecto intelixente pódese implementar empregando un amplio abano de metodoloxías. Por exemplo, a metodoloxía de implementación inspirada no estándar *Sense-Plan-Act* clásico usado normalmente na robótica. De feito, a metodoloxía *Sense-Plan-Act* permite deseñar as tres capacidades críticas que deben ter os SO para poder funcionar con eficacia: (1) adquisición de datos a partires de sensores (*Sense*); (2) execución do algoritmo para ofrecer unha resposta axeitada, de acordo con algunha estratexia predefinida (*Plan*); e (3) accións de saída (*Act*) segundo o plan.

O patrón de operacións dun obxecto intelixente comeza na fase de recoller datos de fontes de datos externas (*Sense*) por mediación dos sensores físicos. Cos datos recollidos actualízase a medición indirecta por parte do VS, no caso de que este combine os datos

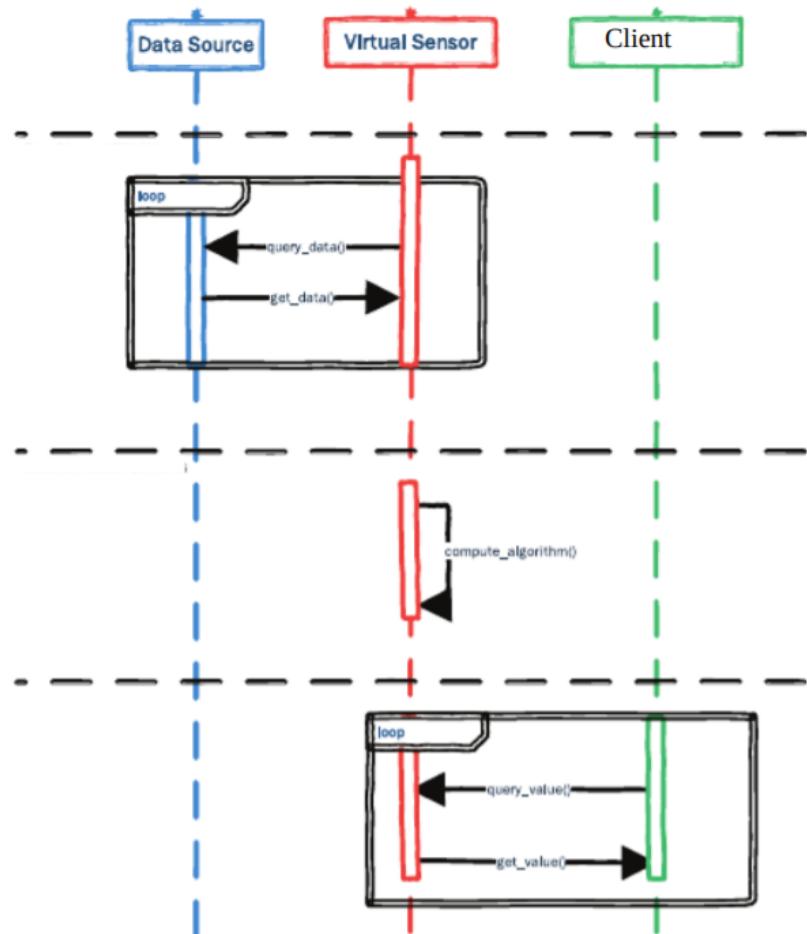


Figura 1: Diagrama de secuencia que presenta a interacción dun sensor virtual (VS)

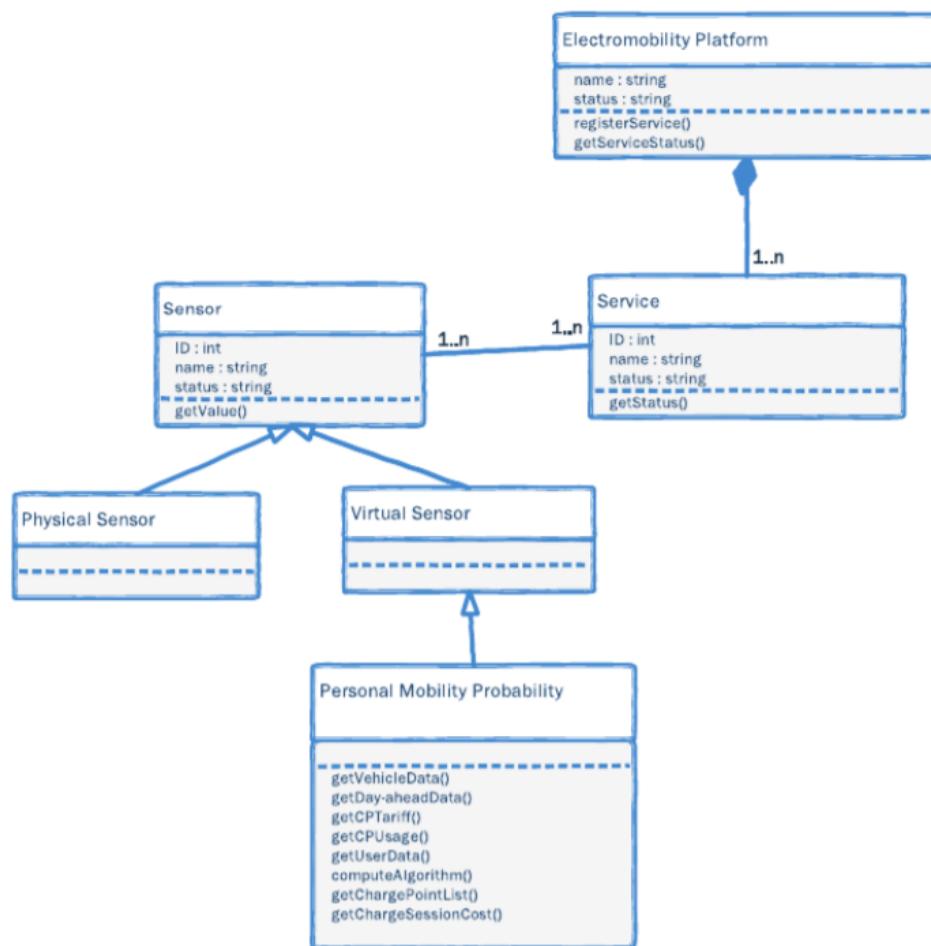


Figura 2: Diagrama de clases coas relacóns entre servizos, sensores físicos e sensores virtuais para un sistema de mobilidade eléctrico

proporcionados por varios sensores. A nova información elaborada basearase na aplicación de métodos estatísticos e / ou técnicas de intelixencia artificial, sobre os datos en bruto. No caso dun único sensor físico conectado, o resultado podería ser o mesmo que o recollido do sensor físico.

A segunda fase (*Plan*) é a fase onde se usan o estado interno do obxecto xunto ca información proporcionada polo VS. Nesta fase, o SO aplica un algoritmo coa estratexia que define o plan de acción.

A terceira e última fase é a fase de acción (*Act*), onde se pon en marcha o plan de actuación definido previamente, e que en xeral consistirá, por exemplo, no envío de comandos aos controladores dos actuadores físicos do obxecto virtual, pero tamén pode consistir en proporcionar servizos a outros subsistemas.

1.1.2. Desenvolvemento de usuario final (EUD)

Hoxe en día, a lóxica do control dos SO en fogares intelixentes resulta inaccesible á maioría dos usuarios finais. Non obstante, a entrada destes dispositivos tecnolóxicos na vida cotiá obriga a darrle aos usuarios un papel activo na súa configuración para que estes sistemas se poidan axustar dinámicamente ás súas necesidades individuais. Este fenómeno foi alcumado na literatura como Desenvolvemento de Usuario Final (EUD). EUD refírese a un conxunto de actividades, técnicas e ferramentas que permiten aos usuarios finais configurar, modificar e controlar artefactos de software e hardware.

En xeral, un SO é un subsistema (hardware/software) que pode interactuar con outros subsistemas ou humanos do seu contorno. Os SO poden empregarse para axustar automáticamente a temperatura en cada habitación, medir os niveis de monóxido de carbono ou controlar as luces a través do teléfono móvil.

A maioría dos obxectos do noso contorno non están dentro da definición anterior dun SO. De aí que existan placas de sensores e actuadores de propósito xeral que se poden acoplar a estes obxectos e así compatibilizar estes obxectos coas casas intelixentes. As placas incorporan un procesador e un transceptor inalámbrico, e diferentes tipos de sensores (luz, son, presión, aceleración, temperatura, etc.) segundo as necesidades. Como exemplo, poden engadirse sensores a unha cama para rastrexar o movemento e a posición do usuario, a outros obxectos (cadeiras, lámpadas, mesas, despertadores, etc.) para habilitar a interacción entre os obxectos e o usuario, acelerómetros no tirador da porta e no pulso do usuario para identificar a persoa, que é detectada medindo a correlación entre os dous sinais do acelerómetro, etc.

O EUD permite aos usuarios finais deseñar visualmente aplicacións domésticas intelixentes mediante unha interface táctil. Permite aos usuarios especificar dispositivos de entrada e saída e a configuración da lóxica de comportamento empregando regras "*If-Then*", que describen unha condición e a súa acción asociada. Por exemplo, unha interface formada por palabras que se poden agrupar para formar regras, combinando catro elementos de información: quen, que, onde e cando. Estes elementos describen a acción que o sistema enviará en resposta a un estímulo dado e en que momento.

Para responder ás limitacións da lóxica definida por programa, propuxéronse varios sistemas EUD que admiten a descripción do comportamento baseado en regras. As regras

definen principalmente unha condición e unha acción. As condicións son probas sobre os valores do sensor e as accións son comandos que se deben enviar aos actuadores. As normas EUD utilizan os valores de sensores e actuadores para definir condicións e accións.

1.2. Sistema de configuración dun fogar Intelixente (SCFI)

Neste proxecto imos desenvolver un sistema composto por componentes de hardware e software, que permita aos usuarios finais deseñar e configurar un sistema doméstico intelixente que responda ás súas necesidades. O sistema debe deseñarse para soportar dúas clases de usuarios: informáticos e usuarios non informáticos. Os usuarios non informáticos son os que non recibiron ningunha formación en desenvolvemento de software e hardware. Para soportar usuarios non informáticos, propoñemos a linguaxe de control simple (SCL) para o control central dos SO nun fogar intelixente.

Un fogar intelixente típico está composto por varios SO que se comunican cun sistema central denominado sistema de Control e Visualización Central (CVC). As funcións principais deste sistema pódense resumir do seguinte xeito:

1. Recibe información sensorial dos SO implementados no ambiente doméstico intelixente.
2. Controla o ambiente intelixente mediante comandos enviados a un subconxunto de SO despregados nese contorno. Estes comandos envíanse normalmente en resposta á información sensorial obtida do ambiente, un comando de usuario, ou un temporizador preconfigurado.
3. Permite ao usuario visualizar a información sensorial recollida en varios niveis de granularidade.

O SCFI interactúa con dous tipos de entidades: usuarios e SOs. Definimos a un usuario como un individuo que crea SOs usando clústers de transductores (sensores ou actuadores) de propósito xeral que aparella a obxectos do entorno formando unha rede (GPTN), configura un entorno intelixente, pode enviar comandos aos SO e/ou visualiza a información producida polos SO usando o CVC. Para configurar un entorno intelixente, o usuario especifica que debe facer o CVC en resposta aos datos recibidos dos sensores do SO. Por exemplo, considerando o seguinte SO: unha cadeira de oficina equipada cun sensor de presión e un actuador de vibración, ambos integrados na almofada do asento. O usuario pode configurar o CVC para enviar un comando ao actuador para que vibre se se detecta que a presión é alta (é dicir, unha persoa está sentada na cadeira) durante un par de horas. A vibración recordaríalle á persoa sentada que debe dar un paseo. Para configurar un ambiente intelixente, o usuario realiza o seguinte:

1. Constrúe SO usando GPTN e implantaos no contorno.
2. Usa o CVC para:
 - Rexistrar os SO dispoñibles subministrando os seus nomes e enderezos IP.
 - Configura os SO rexistrados definindo reglas para controlalos a través do CVC.
 - Visualiza información dos sensores dos SO.

Transducer	Prototype	Remarks
Wireless Node's Core (Mainboard)		Data collection, processing and communication
Pressure Sensor		I ² C IDs 1-10 are reserved for pressure sensors (FSRs)
Temperature Sensor		I ² C IDs 72-75 are reserved for Temperature Sensors (TMP102)
Accelerometer		I ² C IDs 83-84 are reserved for accelerometer (ADXL345)
Light Sensor		I ² C IDs 41, 57 are reserved for Light sensors (TSL2561)
Vibro-tactile Actuator		I ² C IDs 11-20 are reserved for vibro-tactile actuators
On/Off Actuator		I ² C IDs 21-30 are reserved for on/off actuators
Dimming Actuator		I ² C IDs 31-40 are reserved for dimming actuators
Extension		Connected to the I ² C bus when needed

Figura 3: Algunos sensores e actuadores

1.2.1. Rede de sensores de propósito xeral

A GPSN permite a creación de agrupacións de sensores en rede aparellados con obxectos, para formar SO. Cada clúster fórmanse mediante un mecanismo de plug-&-play que os conecta a unha placa principal a través de fíos mediante unha comunicación en serie, de xeito que a tarxeta principal pode recoñecer os sensores engadidos e eliminarlos dinámicamente. A rede soporta sensores de uso común como temperatura, presión, luz, aceleración, etc, coma os mostrados na Figura 3.

Asociado a cada clúster de sensores haberá un obxecto controlador que mantén a lista dos sensores conectados en cada momento e que pode devolver os datos recollidos por cada un deles.

O controlador organiza os datos dos sensores e transmíteos no formato *Language Model Sensor (SensorML)*.

Consideraremos que cada VS se conecta cunha única rede GPSN a través do seu controlador. A súa vez o SO pode conectarse con un ou con varios VS dos que recibe os datos, e tamén con un ou varios actuadores a través dos controladores respectivos.

Os SO envían datos de control no formato *Actuator Model Language* (*ActuatorML*).

1.3. Arquitectura do SCFI

O apartado máis importante desta introdución é esta descripción da arquitectura do SCFI, que a faremos dende unha perspectiva algo diferente á achegada no guión inicial. Definiremos as compoñentes deste sistema dividíndoas en tres segmentos diferentes: un de conexións (adicado ao rexistro e selección de SOs), outro de xestión de SOs (principalmente para a configuración dos obxectos intelixentes por parte do usuario) e un terceiro de control interno (unha parte que resulta máis ben oculta para o usuario):

1.3.1. Conexións

1. Rexistrador de SO:

Os SO dispoñibles son rexistrados polo usuario a través do Rexistrador de SO. Durante o rexistro, o usuario especifica o nome do SO e o enderezo IP. Toda a información recollida é almacenada na base de datos de información estática (SID).

Ademais desa tarefa de rexistro, consideramos que nesta compoñente se recolle tamén a posibilidade de eliminar un SO que non esté activo, xunto con toda a súa información almacenada na base de datos.

2. Selector de SO:

O Selector permite que o usuario seleccione un subconxunto de SO rexistrados cuxa información está almacenada no SID para estar activos no sistema. Os datos recadados dos sensores dos SOs seleccionados poden usarse para especificar as regras de actuación dos SO a través do Configurador, datos que se enviarían a través do transmisor de datos, consultando os SO seleccionados sobre a súa información dinámica. A información dinámica refírese ao número e tipo de sensores e actuadores de cada SO (xa que os transductores poden engadirse ou eliminarse dinámicamente mediante un mecanismo *plug-&-play*), e as medidas do sensor en tempo real. Tendo un SO seleccionado, poderase escolher entre activalo ou desactivalo, o que afectará ao contido presente na DID sobre o SO:

- Se se escolle activar un SO, entón solicitaráselle a información descrita previamente e almacenarase na base de datos de información dinámica (DID).
- Pola contra, se o usuario escolle desactivar o SO, o sistema reflexará na DID que o SO deixa de estar activo.

1.3.2. Xestión

1. Configurador de SO:

A definición de control dos SO baséase na lóxica difusa (fuzzy). A teoría de conjuntos difusos foi deseñada para imitar o mecanismo de razonamento humano. Por exemplo, é moito más doado para os humanos pensar

na temperatura ambiente en termos cualificativos como calor ou frío, no canto de especificar limiares nítidos que describen o estado térmico da habitación. Polo tanto, desde a perspectiva de usabilidade, empregando a lóxica difusa, faise máis intuitiva a configuración dun entorno intelixente, especialmente para usuarios con pouca ou ningunha competencia en programación informática.

Hai unha gran cantidade de linguaxes de control difuso, con todo, estes controladores son adecuados para aplicacións de enxeñaría e requiren experiencia técnica. Non obstante, o SCFI debe dar soporte a usuarios sen formación técnica formal. Por iso, introducimos o SCL, unha linguaxe baseada en regras que permite aos usuarios definir accións que os actuadores realizan en resposta aos datos do sensor ou aos comandos do usuario. É dicir, definen as estratexias a seguir polos SO. O usuario pode configurar regras de control (Plans) dun obxecto seleccionado usando un dos tres modos: **baseado en formularios**, **baseado en editor**, e **avanzado**. Os usuarios que posúen coñecementos de programación poden aprender SCL ou a crear regras de lóxica difusa moito máis rápido que outros usuarios. Así, o modo avanzado proporciona ao usuario un control de axuste fino.

- No modo baseado en formulario, o usuario define as regras de control usando SCL. Non obstante, en lugar de proporcionar ao usuario un editor de texto para escribir SCL, emprégase un editor gráfico. Este editor permítelle ao usuario crear regras SCL ao cubrir un formulario. Polo tanto, os usuarios non teñen que aprender a estrutura de SCL.
 - No modo baseado no editor, o usuario especifica SCL usando un editor de texto. O controlador do SCFI está baseado completamente en lóxica difusa. De aí que, como último paso, tanto no método baseado en formulario como no baseado no editor, as regras SCL se traduzan automáticamente en funcións difusas a través do Xerador Fuzzy.
 - No modo Avanzado, o usuario especifica regras difusas directamente sen usar SCL. Polo tanto, somentes será necesaria unha verificación da corrección da lóxica difusa (será responsabilidade do usuario que esa lóxica sexa coherente para o SO que se configura, pero non se tolerará que se introduzan regras nun formato incorrecto) En calquera caso, ao final da configuración, as regras difusas e as funcións de membresía gárdanse na Base de datos Fuzzy (FD).
2. **Visualizador:** Este compoñente envía unha solicitude de lectura aos SO que se atopan activados polo usuario a través do transmisor de datos e mostra na pantalla as súas medidas en tempo real. Tamén amosará outra información do sensor que poida resultar relevante, coma o tipo ou a identificación do mesmo.

1.3.3. Control Interno

1. **Verificador Sintaxe SCL:** Este compoñente é o responsable de verificar a sintaxe das regras SCL antes de que as traduza o Xerador Fuzzy converténdoas en regras difusas e funcións de membresía. As regras SCL son procesadas a través de dúas etapas polos módulos:
 - Analizador léxico: para converter a serie de caracteres SCL en expresións regulares (*tokens*).

- Analizador de sintaxe: para asegurar que as regras concordan coa gramática SCL, e construír unha árbore de análise con esas regras.
2. **Xerador Fuzzy:** Este compoñente é o responsable de xerar as funcións de membresía e as regras difusas baseadas nas regras SCL. Recibe dúas entradas: regras SCL organizadas nunha árbore de análise do verificador de sintaxe SCL, e os datos dinámicos do SO a configurar, que inclúen os seus sensores e actuadores, obtidos do DID. O Xerador Fuzzy produce regras difusas que coinciden coa lóxica descrita no SCL. No noso caso suporemos que as funcións de membresía xa estarán asociadas a cada un dos sensores e actuadores en particular, polo que non será necesario que sexan producidas novas funcións.
3. **Planificador:** Este compoñente, recibe datos dos SOs, borrosícaos utilizando as funcións de membresía asociadas, xera a saída difusa empregando algunha das regras almacenadas na base de datos FD e desborrosifica a saída con resultados cuantitativos. Os resultados cuantitativos xuntos cos comandos correspondentes son enviados ao actuador no formato *ActuatorML*. Ademais de actuadores tradicionais como vibradores ou interruptores eléctricos, o sistema soporta actuadores en forma de xeradores de mensaxes. Polo tanto, estes actuadores poden enviar avisos ou mensaxes informativas ao usuario en forma de correo electrónico ou SMS.
- Este compoñente carga as regras difusas da FD e solicita datos de medición de sensores en tempo real dos SO. Unha vez que recibe os datos solicitados, procésaos e xera os comandos de actuación pertinentes sempre que sexa necesario (segundo as regras difusas). O compoñente do transmisor de datos transmite estes comandos (en caso de haber algo que transmitir) aos SO asignados.
4. **Transmisor e Receptor de datos:** Estes compoñentes son os responsables de intercambiar información entre o sistema, o usuario e os SO. A información intercambiada está empaquetada en mensaxes *SensorML* e *ActuatorML*. Os comandos do usuario tamén se empaquetan en mensaxes *SensorML* onde o campo de valor conterá o ID do comando. O sistema poderá tanto solicitar información dos SO (tipo de sensores e actuadores e medicións dos sensores) coma enviar comandos ou sinais de acción aos obxectos intelixentes. Cando se reciben datos do SO, son enviados polo receptor de datos á DID.
5. **Bases de Datos:** O sistema dispón de tres bases de datos para almacenar datos relevantes sobre os obxectos intelixentes, medidas de sensores e regras do controlador. Estas bases de datos detállanse a continuación:
- Base de datos de información estática (SID): o SID almacena información sobre o rexistro dos SO, en particular, os nomes e os enderezos IP. Este último campo consideraremos que é a clave primaria, é dicir, o campo que nos permite identificar de xeito unívoco un SO. Polo tanto, non consideraremos a posibilidade de que dous obxectos intelixentes posúan a mesma IP.
 - Base de datos de información dinámica (DID): esta base de datos almacena a información dinámica recibida dos obxectos intelixentes a través do receptor de datos. Esta información inclúe o número e tipo de sensores e actuadores de cada SO, e medicións de datos en bruto en tempo real dos sensores do SO. Tamén consideramos que é necesario que se recolla se o sensor está activo ou non (posto que pudo ser activado inicialmente e logo

desactivado, e ter xa almacenada na DID información sobre el).

- Base de datos difusa (FD): esta base de datos é a responsable de almacenar as funcións e regras de membresía xeradas como un ficheiro difuso durante a fase de configuración: pode que se xerase a mesma regra para dous sensores, polo que se recollerán asociacións entre regras xeradas e SOs configurados que as posúan. O controlador recupera esta información para poder difundir os datos brutos dos sensores.

Como xa comentamos na SID, entendemos que calquera SO poderá ser identificado a través da súa IP, que consideraremos a clave primaria.

2. Casos de uso

2.1. Diagrama de casos de uso

Trala especificación inicial do problema, procedemos a amosar o diagrama de casos de uso que deseñamos para o sistema. Na Figura 4 amósase o mencionado diagrama, no que se pensamos que se recollen todos os casos de uso que involucran directamente ao usuario e algunha cuestión adicional que cremos conveniente representar.

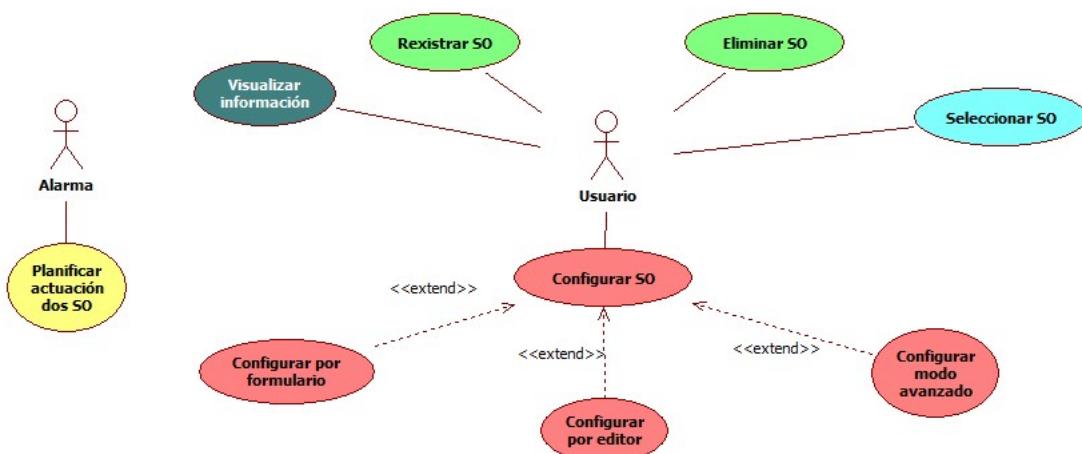


Figura 4: Diagrama de casos de uso do SCFI

Consideramos que é necesario realizar unha pequena xustificación das decisións tomadas, para que así o lector poida interpretar mellor o resultado que se recolle neste documento.

Para comezar, decidimos incluir dous actores, que son o 'Usuario' que accede ao sistema (o máis importante de todos, posto que o que estamos a deseñar vai encamiñado á utilización do sistema por parte del), e unha 'Alarma', un actor de tipo temporal que representa a frecuencia coa que se realizará a planificación dun SO, e que nos parece igualmente interesante que se vexa reflexado neste diagrama dado que, inda que non sexa algo co que o usuario vai interactuar directamente, pensamos que é un comportamento derivado doutras accións deste o suficientemente importante como para recollelo aquí dalgún xeito (non faremos o mesmo con outras partes do sistema).

Vamos a comentar o contido deste diagrama (en canto a casos de uso) en tres partes:

- A parte superior reflexa as transaccións que pode realizar o usuario para o que englobaremos como a xestión dos SOs: rexistrar, eliminar, seleccionar SOs e visualizar a información dos SOs seleccionados.
- A parte inferior (cor vermella) reflexa a parte de configuración do SO, que modelamos mediante un caso de uso “principal” que representa a parte común da configuración, e varios casos que extenden ao principal, un por cada posible modo de configuración (formulario, por editor e avanzado). Esta idea poderíase modelar tamén mediante unha xerarquía, de feito, parécenos que son dúas representacións equivalentes, pero tras varias discusións chegamos a que resulta mellor e más intuitivo entender que se trata de extensións da configuración (dependendo do modo).
- A parte esquerda representa unha parte do sistema que non está vinculada directamente co usuario, pero que evidentemente é provocada pola configuración realizada: o planificador. Ao caso de uso da planificación dos SO asóciasele ese actor alarma que representamos previamente, xustificando tamén así a súa aparición neste diagrama.

Distinguimos así dúas partes importantes do sistema: unha parte de xestión de SOs e outra adicada á configuración e planificación. Estas partes son as que percibimos que, para a descripción dos casos de uso, deben estar presentes. No sistema evidentemente hai máis cuestións, como as bases de datos ou o envío de mensaxes aos SOs (que irán aparecendo mencionadas en cada un dos casos de uso), pero que non nos ten sentido modelar como casos de uso ao non ter ningún tipo de interacción con ningún actor.

2.2. Descripción detallada dos casos de uso

A continuación levaremos a cabo unha descripción en detalle de cada un dos casos de uso, para aclarar o seu funcionamento. En cada caso que describamos, iremos especificando o seu propósito, os actores que interveñen e as precondicións e poscondicións dos mesmos, así como os posibles cursos dos eventos (o normal e os alternativos) e as relacións que teñan con outros casos de uso seguindo o diagrama presentado previamente.

Para distinguir o comportamento do actor do sistema, empregaremos cores diferentes para describir os pasos realizados por cada unha das partes: en verde os pasos dos actores e en azul os pasos realizados polo sistema.

2.2.1. Caso de uso 1: Rexistrar SO

- **Propósito:** Dar de alta un obxecto intelixente no sistema.
- **Actores:** Usuario
- **Precondicións:** -
- **Poscondicións:** o sistema rexistrará un novo SO cos datos introducidos polo usuario (nome e IP) e almacenará a información na base de datos estática (SID), ou devolverá un erro se corresponde.
- **Curso Normal dos Eventos:**
 1. O usuario escolle a opción de Rexistrar un novo SO.
 2. O usuario introduce o nome do obxecto intelixente e a IP.
 3. O sistema validará os datos introducidos. É dicir, comprobará se cubriron os campos e se son correctos.

4. O sistema accede á base de datos estática (SID) para almacenar os datos do novo SO.
 5. O sistema informa de que o proceso se executou correctamente.
- **Curso alternativo 1**
 3. Non se introduciu nada nos campos.
 4. O sistema informa ao usuario do erro producido.
 - **Curso alternativo 2**
 3. A IP introducida ou o nome teñen un formato incorrecto (supérase o límite dos campos ou úsase un carácter inválido).
 4. O sistema avisa ao usuario do erro producido.
 - **Curso alternativo 3**
 4. A IP introducida xa está rexistrada.
 5. O sistema avisa ao usuario do erro producido.
- **Relacións con outros casos de uso:** non existe ningunha relación con outros casos de uso que sinalar.

2.2.2. Caso de uso 2: Eliminar SO

- **Propósito:** Retirar un obxecto intelixente do sistema.
- **Actores:** Usuario
- **Precondicións:** Hai algún SO rexistrado e inactivo no sistema.
- **Poscondicións:** O sistema actualizará a Base de Datos Estática (SID), a Dinámica (DID) e a Difusa (FD) eliminando toda a información sobre o obxecto intelixente que se decidiu eliminar.
- **Curso Normal dos Eventos:**
 1. O usuario escolle a opción de eliminar un SO.
 2. O sistema accede á SID para tomar a información de todos os SOs rexistrados.
 3. O sistema accede á DID para comprobar se cada SO está activo ou non.
 4. Recollidos os SOs, o sistema filtra por aqueles que non están activos e amósallos ao usuario.
 5. O usuario selecciona un SO dos amosados.
 6. O sistema accede á FD para borrar as asociacións entre o SO seleccionado e as regras difusas, e borra aquelas regras que queden sen SOs vinculados.
 7. O sistema accede á DID para borrar a información dinámica que poida estar rexistrada dese SO.
 8. O sistema accede á SID para borrar a información estática rexistrada dese SO.
 9. O sistema informa de que o proceso rematou correctamente.
- **Relacións con otros casos de uso:** non existe ningunha relación con outros casos de uso que sinalar.

2.2.3. Caso de uso 3: Seleccionar SO

- **Propósito:** Permite escoller un obxecto intelixente dos rexistrados no sistema para o perfil de configuración actual, e activalo/desactivalo (dependendo do caso).

- **Actores:** Usuario
- **Precondicións:** É necesario que esté activa a conexión coa rede de sensores (GPSN) para posibles comunicacóns cos SOs, e hai SOs rexistrados no sistema.
- **Poscondicións:** O sistema escollerá un obxecto dos rexistrados para activalo ou desactivalo e, no primeiro caso, solicitaralle a súa información dinámica, que almacenará na DID. No segundo caso, almacenará unha indicación de que se desactiva.
- **Curso Normal dos Eventos:**
 1. O usuario escolle a opción de seleccionar un SO.
 2. O sistema accede á SID para recuperar a información de todos os SOs rexistrados.
 3. O sistema comproba na DID se cada SO está activo ou non.
 4. O sistema amosa os SOs rexistrados ao usuario.
 5. O usuario selecciona un obxecto intelixente rexistrado no sistema para activalo.
 6. Como o SO pasa a estar activo, o sistema envía unha mensaxe ActuatorML ao mesmo para solicitarlle información.
 7. O sistema accede á DID para almacenar os datos recibidos.
 8. Sistema informa ao usuario de que o proceso se executou correctamente.
- **Curso alternativo 1**
 5. O usuario escolleu un SO para desactivalo.
 6. O sistema garda na DID que o SO seleccionado pasa a estar inactivo.
 7. O sistema informa de que o proceso se realizou correctamente.
- **Relacións con outros casos de uso:** non existe ningunha relación con outros casos de uso que sinalar.

2.2.4. Caso de uso 4: Configurar SO

- **Propósito:** Xerar regras de control para un SO activo a partir das preferencias do usuario. Tras xerarse as regras correspondentes, estas almacenaranse na base de datos.
- **Actores:** Usuario
- **Precondicións:** Hai ao menos un SO rexistrado e configurado como activo.
- **Poscondicións:** Produciranse unha serie de regras difusas sobre algún SO activo que se almacenarán na FD, ou devolverase algún erro se é preciso.
- **Curso Normal dos Eventos:**
 1. O usuario escolle a opción de configurar un SO.
 2. O usuario escolle un dos modos (editor, formulario ou avanzado) como <MC>.
 3. O sistema recupera a información dos SO activos que polo tanto poden ser configurados da DID e da SID.
 4. O sistema amosa os SOs seleccionados para estar activos ao usuario.
 5. O usuario escolle un dos SOs seleccionados para ser configurado.
 6. Punto de extensión: execútase o caso de uso asociado ao modo de configuración <MC> seleccionado polo usuario.
 7. O sistema accede á base de datos FD para almacenar as regras xeradas a través dunha das configuracións.

8. O sistema informa que se completou a configuración correctamente.
- **Relacións con outros casos de uso:** este caso de uso está relacionado con outros tres mediante extensións:
 1. **Configurar SO por formulario:** escollendo <MC> por formulario.
 2. **Configurar SO por editor:** escollendo <MC> por editor.
 3. **Configurar SO no modo avanzado:** escollendo <MC> avanzado.

2.2.5. Caso de uso 5: Configurar SO por formulario

- **Propósito:** Xerar regras de control para un SO mediante a introducción por parte do usuario de información nos campos dun formulario.
- **Actores:** Usuario
- **Precondicións:** Comezou a executarse o caso de uso 'Configurar SO', e seleccionsouse unha configuración por formulario.
- **Poscondicións:** O configurador do sistema xerará a partir dos datos introducidos unha serie de regras difusas, ou devolverá un erro se se produciu algún problema.
- **Curso Normal dos Eventos:**
 1. O sistema accede á DID para recuperar a información do SO que foi seleccionado para configurar.
 2. Grazas aos datos recuperados, o sistema elabora e amosa o formulario a cubrir ao usuario.
 3. O usuario cubre os diversos campos do formulario presentado.
 4. O sistema valida que a información dos campos sexa correcta.
 5. O sistema xera regras SCL segundo os valores introducidos.
 6. O sistema xera unha árbore de análise ao verificar as regras SCL xeradas.
 7. O sistema xera as regras difusas a partir da árbore de análise.
- **Curso alternativo 1**
 4. Hai campos obligatorios baleiros.
 5. O sistema informa dos problemas atopados.
- **Curso alternativo 2**
 4. Insertouse información sobre a configuración que resulta incorrecta para xerar as regras SCL dese SO.
 5. O sistema avisa dos problemas producidos.
- **Relacións con otros casos de uso:** este caso de uso é unha extensión de **Configurar SO**.

2.2.6. Caso de uso 6: Configurar SO mediante un editor

- **Propósito:** Xerar regras de control para os SO mediante a introducción por parte do usuario de sentenzas na linguaxe SCL.
- **Actores:** Usuario
- **Precondicións:** Comezou a executarse o caso de uso 'Configurar SO', e seleccionsouse unha configuración por editor.
- **Poscondicións:** O configurador do sistema xerará coas regras introducidas polo usuario unha serie de regras difusas, en caso de que as primeiras sexan correctas. Se non, devolverá un erro.

- **Curso Normal dos Eventos:**
 1. O usuario escribe regras en SCL para configurar o SO seleccionado para configurar.
 2. O sistema valida que se introducira algunha regra.
 3. O sistema accede á DID para recuperar a información dinámica do SO.
 4. O sistema xera unha árbore de análise ao verificar as regras SCL introducidas polo usuario.
 5. O sistema xera as regras difusas a partir da árbore de análise.
- **Curso alternativo 1**
 2. Non se introduce ningunha regra.
 3. O sistema informa ao usuario da situación.
- **Curso alternativo 2**
 4. Algunha das regras que se introduciu presenta erros.
 5. O sistema informa dos erros producidos ao usuario.
- **Relacións con outros casos de uso:** este é un caso de uso que extende a **Configurar SO**.

2.2.7. Caso de uso 7: Configurar SO no modo avanzado

- **Propósito:** Introdución das regras difusas directamente por parte do usuario para configurar un obxecto intelixente.
- **Actores:** Usuario
- **Precondicións:** Comezou a executarse o caso de uso 'Configurar SO', e selecciónouse unha configuración avanzada.
- **Poscondicións:** O sistema recolle as regras difusas do usuario e válidaas, ou devolve un erro se houbo problemas.
- **Curso Normal dos Eventos:**
 1. O usuario introduce regras difusas para aplicarle ao SO seleccionado.
 2. O sistema comproba que o usuario introducira algunha regra.
 3. O sistema comproba que a lóxica introducida polo usuario sexa válida.
- **Curso alternativo 1**
 2. Non se introduce ningunha regra.
 3. O sistema informa ao usuario da situación.
- **Curso alternativo 2**
 3. A lóxica introducida polo usuario non é válida.
 4. O sistema informa dos problemas atopados ao usuario.
- **Relacións con otros casos de uso:** este caso de uso é unha extensión de **Configurar SO**

2.2.8. Caso de uso 8: Planificar actuación

- **Propósito:** Xerar comandos de actuación axeitados para os SO en función das súas medicións e a lóxica difusa creada polo usuario, cando así o indique a alarma.
- **Actores:** Alarma
- **Precondicións:** Debe estar activa a conexión coa rede de sensores GPSN para as comunicacións co SO, e determinase planificar un SO que se encontra activo e configurado no SCFI.

- **Poscondicións:** Xeraranse comandos de actuación que se enviarán aos SO para que configuren os seus actuadores (se é preciso).
- **Curso Normal dos Eventos:**
 1. A alarma avisa ao planificador para que se leve a cabo a planificación de SOs.
 2. Escollido un SO a configurar, o sistema envía unha mensaxe *ActuatorML* a este para solicitar datos de medicións.
 3. O sistema accede á FD para recuperar as regras difusas asociadas ao SO.
 4. O sistema almacena a información recibida na DID.
 5. En función das regras difusas e os datos do SO, xéranse comandos de actuación para el.
 6. Envíase unha mensaxe *ActuatorML* ao SO cos comandos de actuación.
- **Curso alternativo 1**
 5. En función das regras difusas e os datos do SO, non se xerou ningún tipo de comando de actuación que haxa que enviar.
 6. O sistema non realiza ningún envío, e remata a execución do caso de uso.
- **Relacións con outros casos de uso:** non existe ningunha relación con outros casos de uso que sinalar.

2.2.9. Caso de uso 9: Visualizar información

- **Propósito:** Amosar información dos SOs que foron seleccionados polo usuario.
- **Actores:** Usuario
- **Precondicións:** Debe estar a conexión coa rede GPSN activa para a comunicación cos SOs, e debe haber SOs rexistrados xa no sistema.
- **Poscondicións:** Amosarse información dinámica acerca dos SOs activos nese momento ao usuario.
- **Curso Normal dos Eventos:**
 1. O usuario escolle a opción de visualizar información dos SOs.
 2. O sistema accede á SID e á DID para recuperar información dos SOs activos.
 3. O sistema envía mensaxes *ActuatorML* a todos os SOs que estén activos para solicitar información e datos de medicións.
 4. O sistema recolle a información recibida dos SOs e almacéna na DID.
 5. O sistema organiza a información que foi recibindo e amósalla ao usuario.
- **Relacións con otros casos de uso:** non existe ningunha relación con outros casos de uso que sinalar.

3. Diagramas de Actividade

3.1. Unha ferramenta para describir accións

Presentado xa todo o relativo aos casos de uso, neste apartado centrarémonos nos diagramas de actividade que deseñamos a partir deles.

A descripción feita previamente dos casos de uso co texto plano pode resultar máis incómoda para o usuario. Alternativamente, podemos modelala mediante un diagrama de actividade, que nos permite reflexar o fluxo procedural de accións que son parte dunha

actividade maior, o que nos permite recoller esa descripción dos casos de uso dun xeito más visual.

Nas seguintes páxinas, adicarémonos a amosar os diferentes diagramas que deseñamos. A idea que tivemos foi a de intentar ter un por cada funcionalidade contemplada nos casos de uso, para evitar formar diagramas que fosen excesivamente complexos de entender ou de interpretar. Nos seguintes puntos, iremos incluindo cada un dos diagramas xunto con algúns razoamento acerca das decisións que se foron tomando en cada caso para así telas xustificadas.

3.2. Diagrama de Actividade 1: Rexistrar SO

Modelamos o rexistro dun SO a través da introdución dos seus datos por parte do usuario. Na Figura 5 pódese observar o diagrama elaborado para este primeiro caso.

Este diagrama resulta sinxelo, comezando coa introdución dos datos dun SO por parte do usuario, que logo o sistema tentará verificar. Tras isto, temos unha decisión, posto que dependendo de se os datos son correctos ou non permitimos camiños diferentes:

- Se os datos son correctos, continúase co fluxo principal, que levará a intentar a inserción na base de datos. Pero dita inserción pode non levarse a cabo se a IP xa estaba rexistrada. Nese caso, informarase do erro e rematarase.
- Se son incorrectos, consideramos que o axeitado é dar opción a que o usuario poida volver a introducir os datos e regresar á verificación dos mesmos por parte do sistema. Estimamos máis axeitado modelar este tipo de bucles neste diagrama (cousa que non consideramos na descripción dos casos de uso para evitar maior complexidade).

Matizar algo que pode chamar a atención: ¿por que no caso de que os campos introducidos sexan incorrectos deixamos introducir de novo os datos e no caso de que a IP esté xa rexistrada non? Isto facémolo porque, se ben no primeiro caso se puido producir unha equivocación, no segundo si se meteu unha IP válida, polo que nos ten máis senso que se remate directamente a actividade (parece unha situación na que o usuario se puido confundir e meter un SO que realmente xa ten rexistrado: a IP é a clave primaria que identifica ao SO de maneira única na Base de Datos, e en caso de ser unha equivocación de verdade cambiaríase a IP a introducir, polo que o interpretamos como unha nova inserción, non coma un segundo intento da mesma).

3.3. Diagrama de Actividade 2: Eliminar SO

Eliminar un Obxecto Intelixente será unha acción que tamén poderá facer o usuario a partir de ter obxectos rexistrados. Na Figura 6 pódese consultar o diagrama de actividade que representa esa situación.

Neste diagrama, con respecto ao que tratamos no punto anterior, temos que sinalar algunas diferenzas e novas utilidades que foron empregadas.

Para comezar, mencionar a introdución dos estados de sincronización, que empregamos para modelar que hai algunas tarefas que cremos que se poden realizar simultáneamente ou ben en calquera orde. Ditas tarefas son o borrado dos datos do SO da DID, da SID e da FD (das asociacións SO-regra difusa e das regras que queden sen asociacións con SOs), posto que pode haber información dos mesmos en calquera das bases de datos.

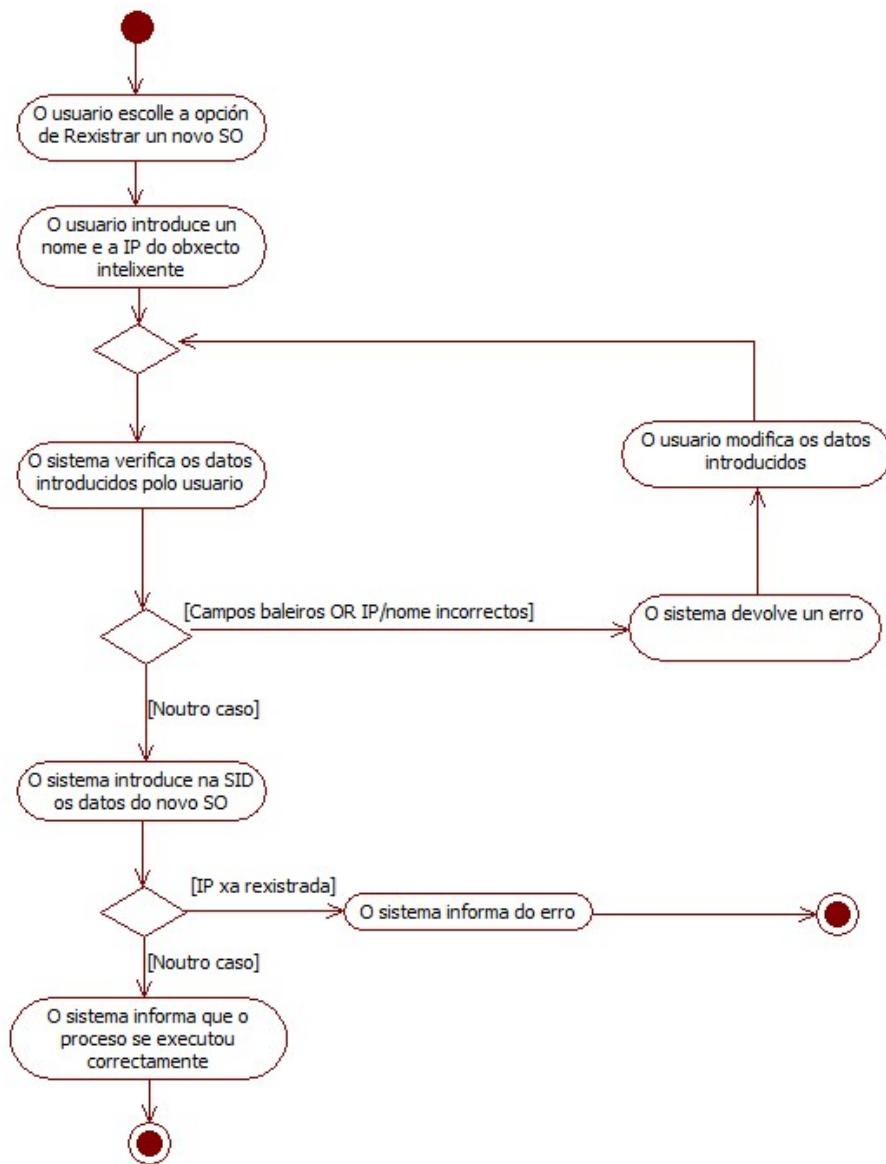


Figura 5: Diagrama de actividade que modela o rexistro dun novo SO por parte do usuario

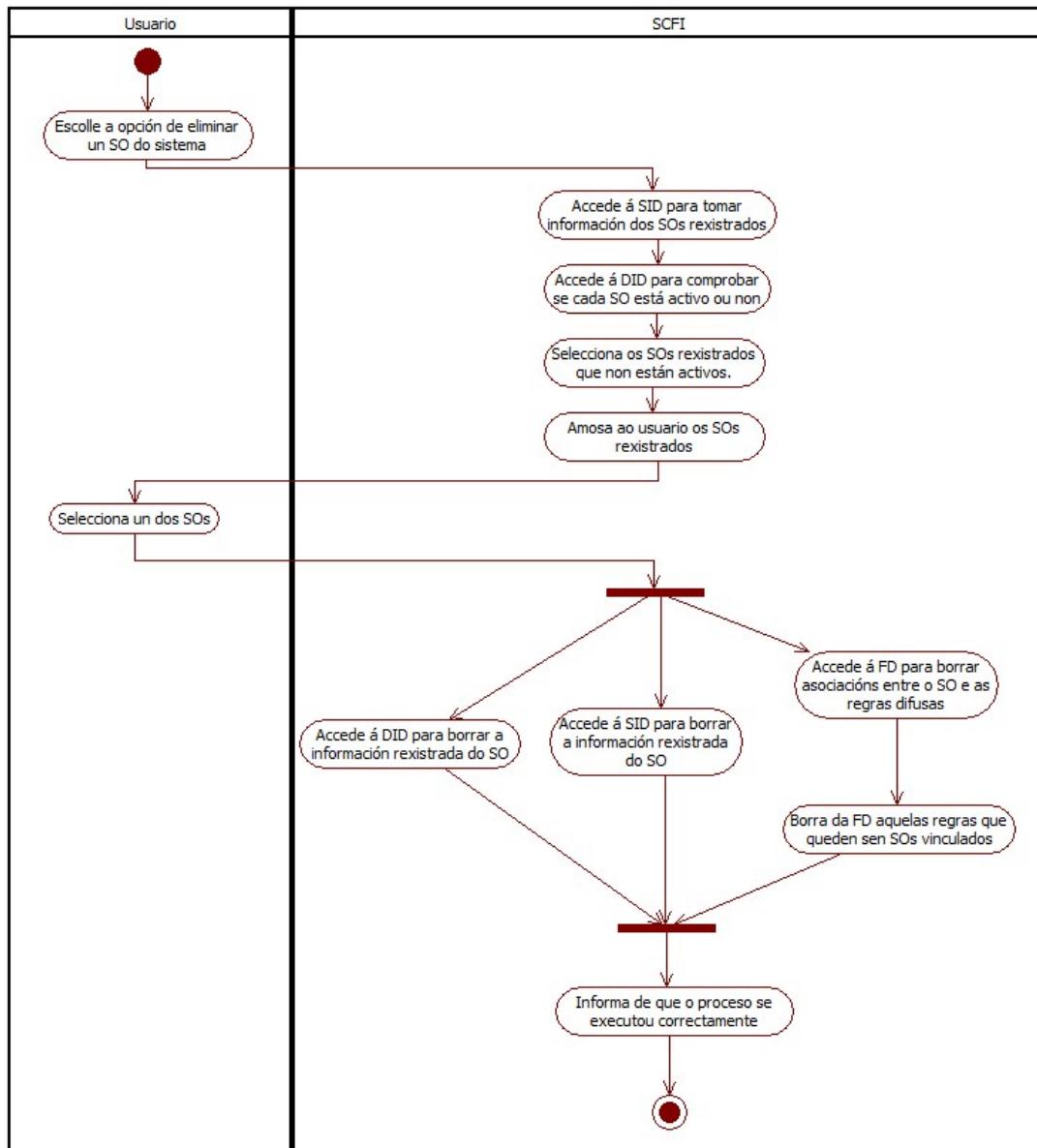


Figura 6: Diagrama de actividad que modela a eliminación de SOs

En todos estes diagramas, ao igual que nos casos de uso, unha precondición recorrente é a de que haxa algúun SO rexistrado, posto que se non engadiríamos en moitos casos máis complexidade ao diagrama dun xeito innecesario: cremos que se acaba tratando de algo secundario e que o usuario debería verificar antes de executar este tipo de casos de uso.

Tamén resaltar que introducimos neste caso outra ferramenta interesante dos diagramas de actividade: as chamadas *swimming lanes* ou canles en galego. Usamos dúas, unha para o usuario e outra para o SCFI, o que nos permite:

1. Escribir únicamente na descripción do estado o que fai ese elemento, sen ter que comezar cada unha das frases que o describen por comodíns como 'O sistema...' ou 'O usuario...' .
2. Ter más claro dunha maneira visual o que fai cada parte na actividade representada.

As canles empregarémolas nalgúns diagramas coma estes, se apreciamos que cada parte ten algo de participación ao longo de toda a actividade. Aquí o usuario intervén un par de veces, por iso se incluiu, pero no da Figura 5 non, dado que só participaba ao comezo e non volvía a intervir.

3.4. Diagrama de Actividade 3: Seleccionar SO

Pasamos a outra parte do SCFI como é a selección dos SO para seren activados ou desactivados. Na Figura 7 amósase o diagrama de actividade correspondente a ese caso de uso.

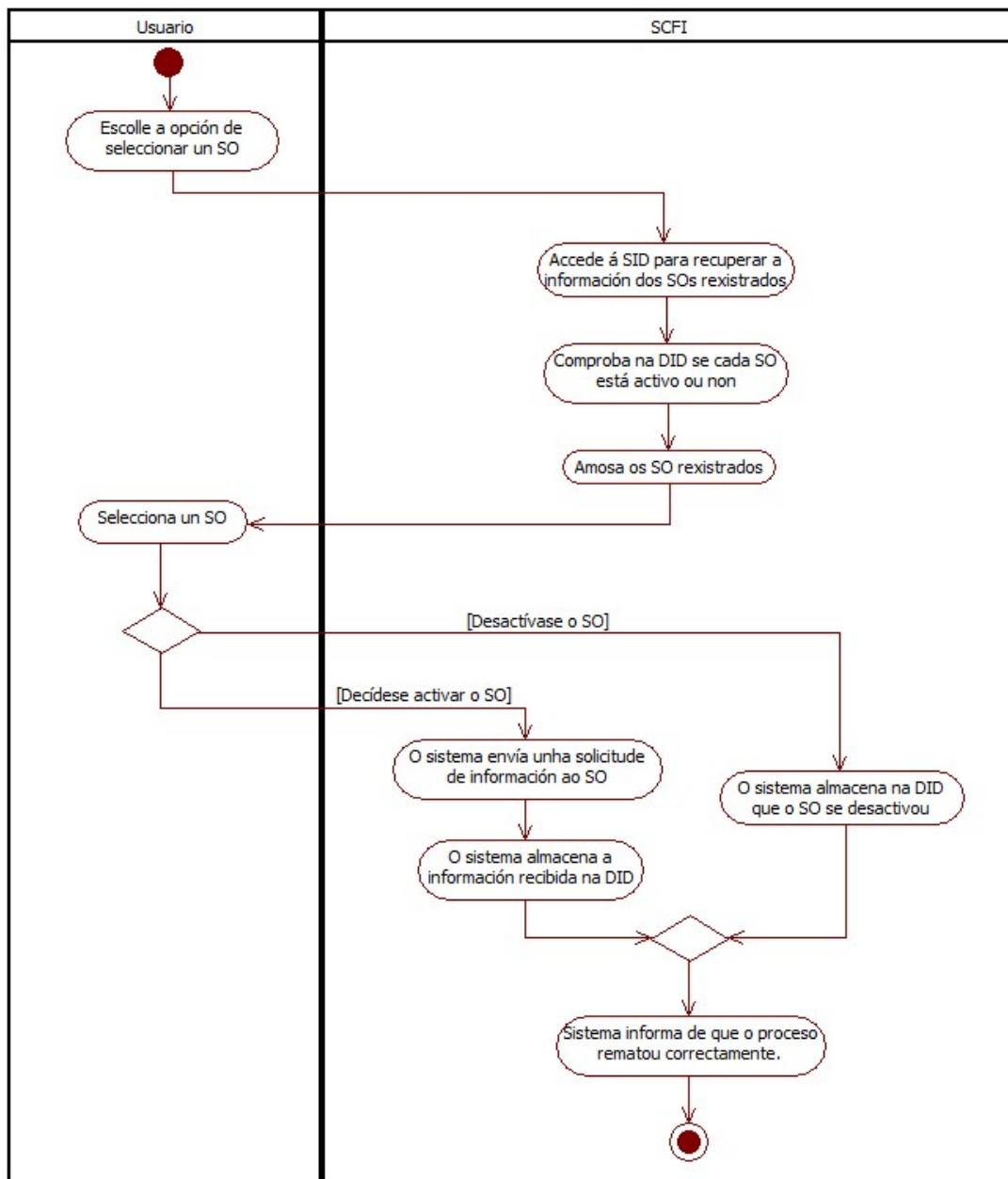
Neste diagrama volvemos a recorrer ás canles, ao haber unha clara interacción entre usuario e sistema en diferentes pasos. Ademais, botamos man dun punto de decisión para representar que, unha vez escollido o SO, este pode activarse ou desactivarse, dependendo do estado no que se atope actualmente (supoñemos que o usuario somentes poderá activar os SOs desactivados e desactivar aqueles que estén activados).

Independentemente dessa decisión, o punto final será a confirmación da execución do proceso por parte do sistema. Temos que anotar que o punto onde converxen as dúas situacións o modelamos co mesmo rombo ca o punto de decisión (algo que xa empregamos do mesmo xeito en diagramas anteriores, pero que non mencionamos para intentar equilibrar os comentarios sobre cada un dos diagramas), dado que non dispoñemos doutro símbolo máis acertado para representalo. De feito, nalgúns das referencias que consultamos nas que tamén se amosaban diagramas de actividade elaborados no StarUML, facíase o mesmo.

3.5. Diagrama de Actividade 4: Configurar SO por formulario

Se ben inicialmente realizamos unha primeira aproximación á representación da configuración do SO nun único diagrama, a súa grande complexidade e a intención de introducir canles para representar unha interacción usuario-sistema leváronnos a considerar un diagrama para cada modo de configuración.

Deste xeito, neste diagrámase modélase a configuración dun SO mediante formulario, e como se pode observar na Figura 8, o seu tamaño é bastante considerable por existir varias tarefas que realizar, polo que preferimos separar os tres tipos de configuración en tres diagramas diferentes.

**Figura 7: Diagrama de actividad que representa o proceso de selección dun SO**

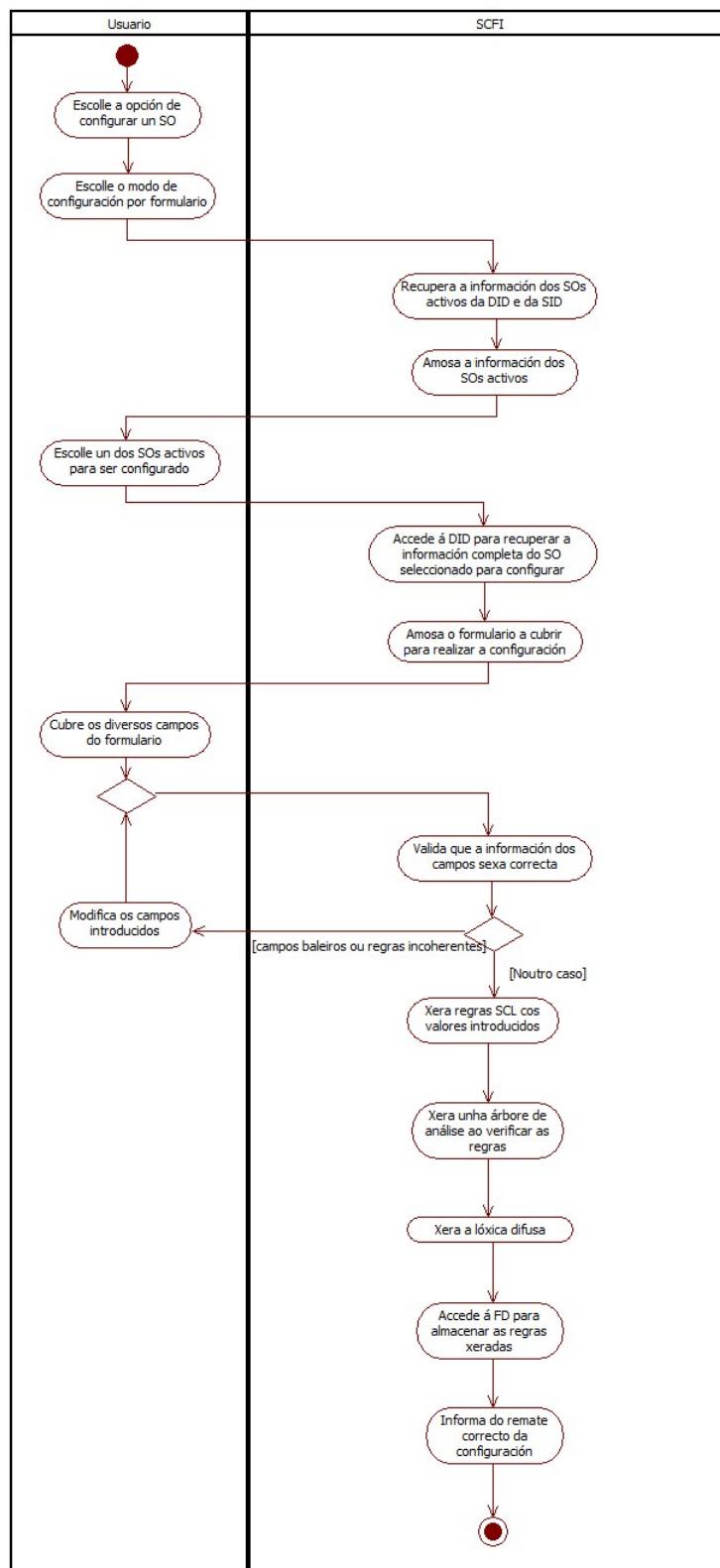


Figura 8: Diagrama de actividad que representa o proceso de configuración dun SO mediante formulario

Destacar deste diagrama o punto de decisión que se ten na validación dos campos do formulario, onde de novo se permite volver atrás para que o usuario poida modificalos e intentar de novo a configuración.

Polo demais, simplemente sinalar un detalle: cando se comeza a configuración por formulario coméntase que se accede á SID e á DID, pero cando se escolle un SO accédese á DID de novo. Isto cremos que é necesario precisalo: no primeiro acceso á DID só se identifica se cada SO está activo ou non, mentres que no segundo, xa escollido o SO, recupérase toda a información dinámica para poder proceder á configuración do obxecto seleccionado. Isto facémolo así para evitar recuperar demasiada información innecesaria no caso de facer un só acceso á DID (habería que traer toda a información de todos os OSs activos).

3.6. Diagrama de Actividade 5: Configurar SO por editor

Modelamos neste novo diagrama a configuración dun obxecto intelixente por medio dun editor no que se poden introducir directamente regras en linguaxe SCL. Pódese consultar este na Figura 9.

Pódese ver unha clara similitude en varios pasos co diagrama da Figura 8, posto que os casos de uso que representan os diferentes tipos de configuración son extensión dun mesmo caso coñecido como 'Configurar SO' .

De todos os xeitos, hai diferencias que motivan a separación en diagramas diferentes:

- Agora non hai a consulta previa na DID para coñecer información do SO, porque as regras son introducidas directamente polo usuario (non é necesario coñecer información dinámica do SO para amosar un formulario). Esa consulta faise agora ao comezar a configuración.
- Entendemos que as regras agora poden ser incorrectas, por iso temos un punto de decisión despois da parte de verificación das regras SCL que no caso de configuración por formulario non tiñamos, posto que as regras son xeradas polo sistema e xa se verifican os parámetros antes de pasalas ao verificador. Tamén permitiremos a volta atrás para corrixilas, como se pode observar na Figura 9.

3.7. Diagrama de Actividade 6: Configurar SO no modo avanzado

Pechamos os diagramas de actividade que modelan os modos de configuración dun SO con este, no que se amosa o modo avanzado. Pódese analizar en maior detalle na Figura 10.

Neste diagrama volvemos a observar algunas similitudes cos anteriores, pero tamén máis diferenzas, pois cómpre lembrar chegados a este punto que non é preciso que participe o verificador SCL nin que se xeren regras difusas co Xerador Fuzzy. A maiores, resaltar o seguinte:

- Introducimos un par de puntos de decisión que representan as comprobacións que se poden realizar: que o usuario introduza algunha regra e que estas sexan correctas. Resaltar isto último, dado que a pesar de que non haxa xeración de regras seguimos reflexando unha comprobación que nos valide as regras introducidas.
- Como se mencionou previamente, ao non haber verificación SCL nin xeración de lóxica difusa o seguinte paso xa é almacenar as regras na base de datos, simplificándose o número de pasos neste diagrama.

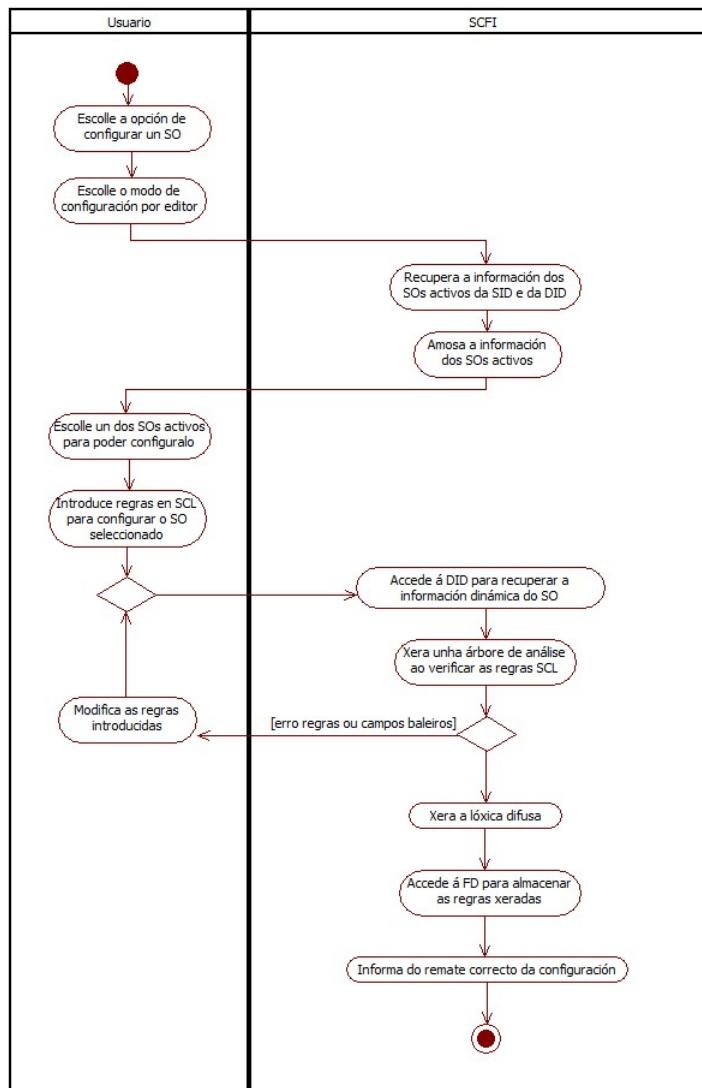


Figura 9: Diagrama de actividad que representa o proceso de configuración dun SO por editor.

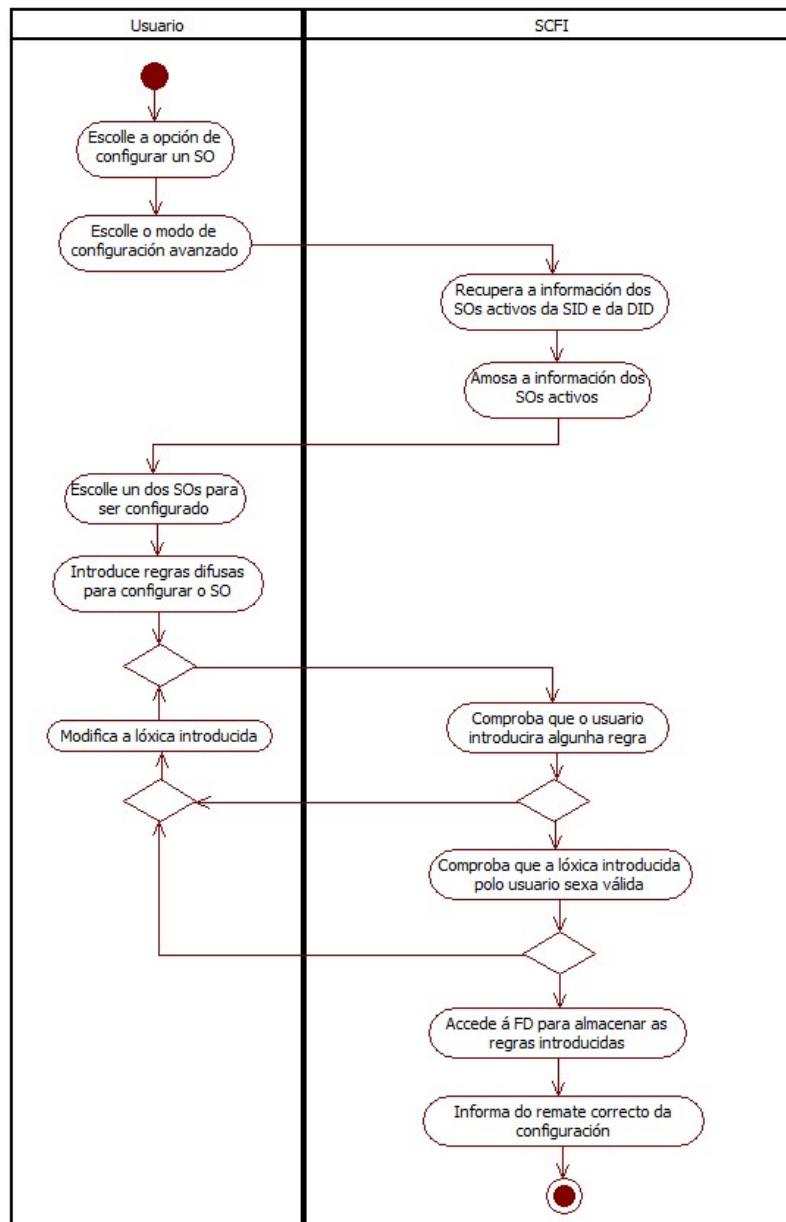


Figura 10: Diagrama de actividad que representa o proceso de configuración dun SO no modo avanzado

3.8. Diagrama de Actividade 7: Planificar actuación

Este diagrama de actividad representa as accións a realizar dentro do caso de uso 'planificar actuación', que tamén incluimos dentro do desenvolvemento deste sistema, inda que non haxa interacción directa co usuario. Na Figura 11 amósase o mencionado diagrama, que pasaremos a comentar a continuación.

Neste caso o proceso de planificación é activado por unha alarma, e a partir de entón realiza a mesma tarefa para cada un dos SOs que estén activos (neste diagrama suporremos que xa os temos recuperados para evitar engadir máis pasos). Decidimos por iso amosar un bucle (a diferencia do feito no caso de uso, coma noutros comentados antes).

Dentro desa tarefa, introdúcese un estado de sincronización no punto no que se accede á base de datos (para obter as regras difusas) e se envían mensaxes aos SOs (para obter a súa información dinámica que será ao mesmo tempo gardada na DID), posto que son dúas tarefas de recuperación de información que entendemos que se poden realizar simultáneamente ou en calquera orde.

Polo demais, pode haber que mandar comandos de actuación aos SOs ou non, o que reflexamos no diagrama mediante un punto de decisión.

Anotar para rematar que non introducimos neste diagrama canles porque realmente a alarma só intervén no primeiro paso, e a partir de ese punto únicamente participa o sistema de diferentes xeitos, reforzando así a xustificación feita xa en diagramas anteriores sobre a introdución delas.

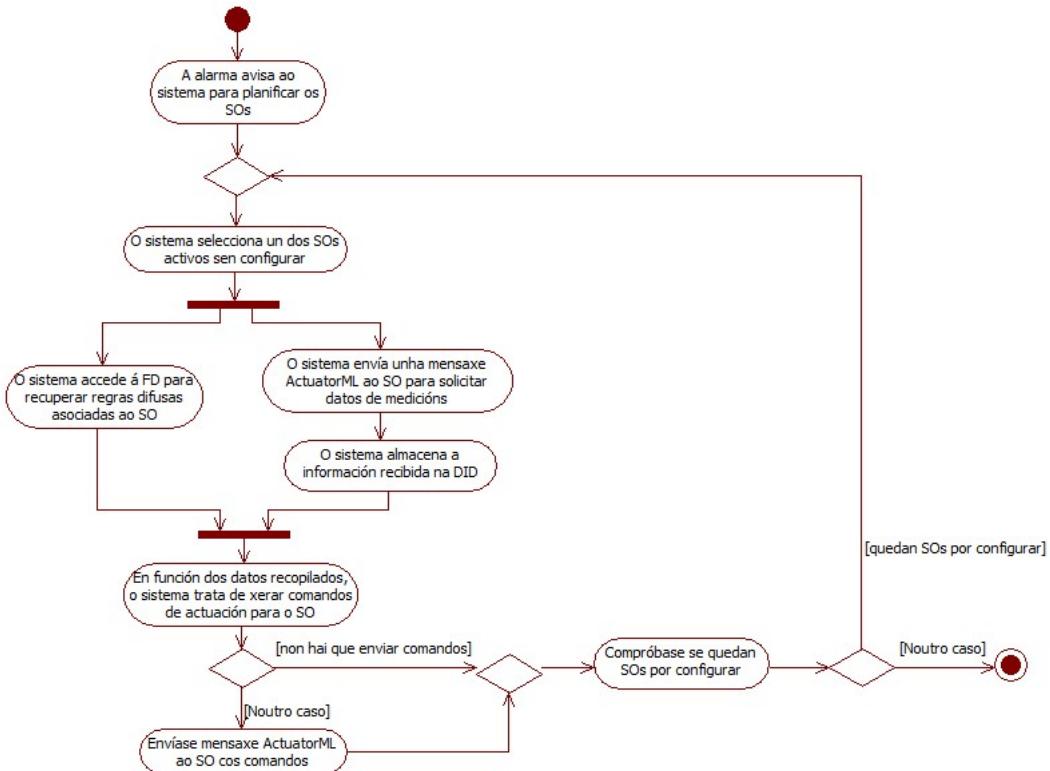


Figura 11: Diagrama de actividad que representa o proceso de planificación dun SO

3.9. Diagrama de Actividade 8: Visualizar información

Quedaría con isto unha situación sen modelar, que é a visualización da información recopilada dos SOs. O diagrama que representa este último caso de uso pódese observar na Figura 12.

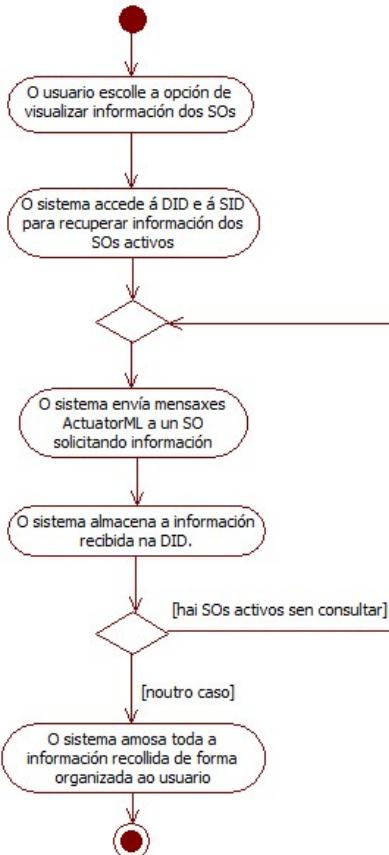


Figura 12: Diagrama de actividad que representa o proceso de visualización da información

Este diagrama non é mais ca unha secuencia de pasos que se seguen dende que o usuario escolle a opción de visualizar información dos SOs. O único resaltante é que se introduciu un bucle no que se representa que se vai solicitando a información dos SOs activos (que son dos que se poderá amosar esta) un a un, e se vai almacenando na DID. A partir de aí, consideramos que non hai nada máis interesante que se poida engadir de forma xustificada e tampouco nos ten moito sentido a introdución de canles, posto que temos de novo unha participación moi escasa por parte do usuario, somentes ao comezo da actividade.

Deste xeito, temos para rematar cos diagramas de actividades deseñados un dos que máis simplicidade presenta, se nos referimos a número de pasos, elementos empregados na súa construcción e á apariencia.

4. Interface de Usuario

A continuación describiremos o deseño da interface da aplicación para a xestión do fogar intelixente xunto coas funcionalidades que consideramos. Comentar que realizamos este deseño pensando en que esta aplicación se poida desenvolver nun entorno móvil, dado o avance das tecnoloxías e a comodidade que pode supoñer o dispor desa aplicación no teléfono.

4.1. Pantalla principal

Esta pantalla corresponde á vista inicial cando se inicia a aplicación. Nela mostrárase información xenérica como a hora e á temperatura ademais dunha mensaxe de benvida. Na zona inferior da pantalla aparecerá un menú que nos permitirá acceder o resto de funcionalidades da aplicación (Figura 13a)

4.2. Pantalla de información

Esta pantalla mostará os dispositivos disponibles, así como unha pequena información acerca do seu estado, por exemplo se unhas luces están encendidas ou se un sistema de audio está reproducindo algúns sons. O usuario terá a posibilidade de premer enriba dos iconos dos dispositivos que aparezan en pantalla e, ao facelo aparecerá outra pantalla con toda a información disponible sobre o obxecto en cuestión. Esta información adaptarase en función do tipo de dispositivo (Figuras 13b e 13c).



Figura 13: Pantallas de información

4.3. Pantalla de xestión de obxectos

Esta pantalla da interface permitirá ao usuario dar de alta obxectos intelixentes, eliminarlos do sistema ou seleccionar cales se queren que estén activos. A interfaz amosará un pequeno menú no que se poderá elixir a acción a realizar. Cada un destes botóns abrirá outra pantalla, as cales se describen ao longo dos puntos seguintes (Figura 14a).

4.3.1. Pantalla Inserción

Nesta pantalla haberá dous campos que permitan ao usuario introducir o nome e a dirección IP do novo SO que se está dando de alta. Unha vez cubertos, mediante o botón 'Gardar' poderá rexistrar o obxecto intelixente (Figura 14b).

Cando se prema no mencionado botón, recibirase a cambio unha mensaxe que informe do resultado, é dicir, que confirme se se puido levar a cabo a inserción ou se produciu algúin problema.

4.3.2. Pantalla Eliminar

Nesta pantalla haberá un ítem despregable no que se mostrarán todos os obxectos rexistrados no sistema que están inactivos. O usuario pode seleccionar un deles para ser eliminado, o que poderá levar a cabo a través dun botón que desencadeará a acción de eliminar (Figura 15a).

Trala eliminación, como ocorre coa inserción, amosarase unha confirmación que indique que este proceso foi correcto.

4.3.3. Pantalla de Selección

Nesta pantalla haberá un selector para activar e desactivar os obxectos intelixentes. Aparecerán unha serie de botóns con iconas que corresponderán ao respectivo obxecto intelixente. Pulsando cada botón, o usuario pode activar ou desactivar os SO, dependendo do seu estado actual (Figura 15b).

Deste xeito, só se poderán desactivar os SOs que no momento estén activados, e só se poderán activar aqueles que no momento se encontren desactivados.



Figura 14: Pantallas de Xestión dun SO (I)



Figura 15: Pantallas de Xestión dun SO (II)

4.4. Pantalla de configuración

Esta pantalla permitirá a configuración dos SOs por parte do usuario, que se poderá realizar en tres pasos diferentes.

Primeiro mostrarse unha pantalla na que se lle permitirá ao usuario seleccionar o modo de configuración: mediante formulario, editor ou modo avanzado; a continuación aparecerá un item despregable onde o usuario seleccionará un obxecto de entre aqueles que estean activos no sistema (Figura 16a); finalmente, aparecerá unha pantalla en función do modo de configuración.

As pantallas que aparecerán segundo o modo que o usuario seleccionase previamente son as seguintes:

1. **Formulario** Nesta pantalla mostraráse unha lista coas condicións do obxecto e outra lista coas accións a realizar que se poden ir introducindo. Ademais haberá tres botóns: un para engadir novas regras, outro para engadir novas actuacións e outro botón para gardar os cambios (Figura 17b).

- Ao pulsar no botón de engadido dunha regra, amosareñse unha serie de items despregables nos que o usuario poderá seleccionar distintos parámetros e opcións para configurar a regra desexada. Despois darase a opción de facer efectivo o engadido da regra mediante un botón situado na parte inferior (Figura 16c).
- Se o usuario preme o botón de engadir actuación, aparecerá unha pantalla con diversos items nos que se lle permitirá seleccionar distintas opcións que permitan configurar a acción que se desea que se leve a cabo. Despois poderá gardarse dita actuación mediante un botón (Figura 17a).

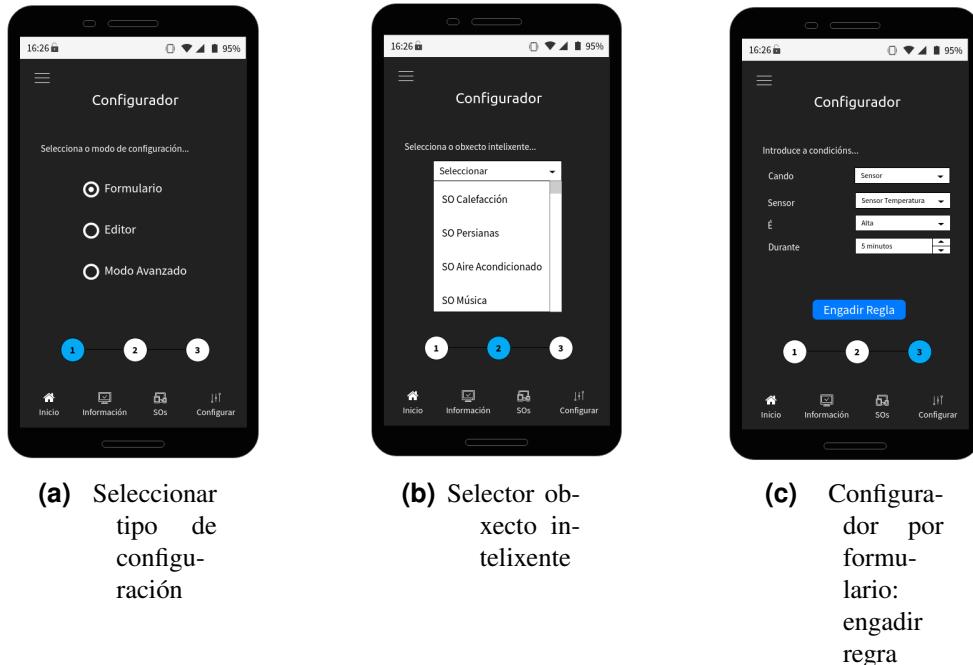


Figura 16: Pantallas de configuración (I)

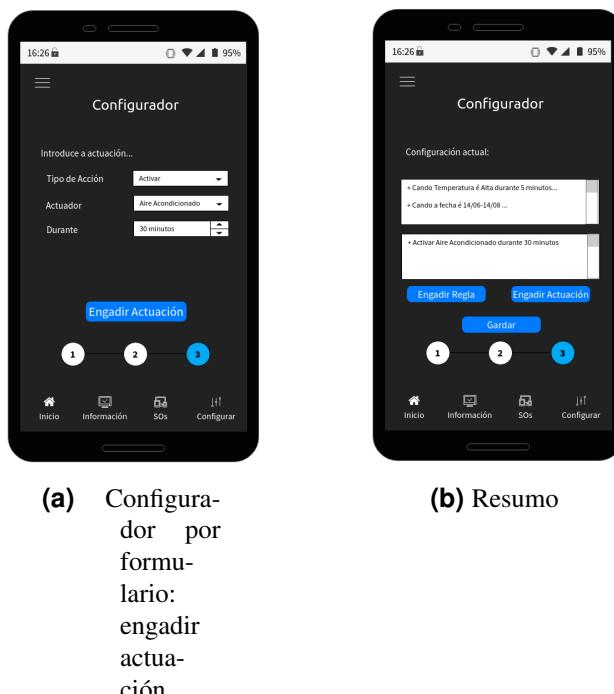


Figura 17: Pantallas de configuración (II)

2. **Editor** O usuario poderá introducir as regras SCL que desexe mediante un campo de texto no que se poderá escribir. Tamén haberá un botón que permitirá gardar as sentenzas que escribe no sistema (Figura 18a).
3. **Avanzado** O usuario poderá introducir as regras difusas que desexe escribindo no

campo de texto dispoñible. Ademáis haberá un botón que permita gardar toda a lóxica introducida no sistema (Figura 18b).

Unha vez que o usuario remate de cubrir os campos para a configuración, independentemente do modo escollido, imprimirase unha mensaxe indicando se dita configuración puido finalmente levarse a cabo ou non.

Destacar, a maiores de todo o comentado previamente, que na parte inferior da sección de configuración haberá un ítem interactivo a modo de guía que lle permita ao usuario saber en que fase da configuración se atopa.



Figura 18: Pantallas de configuración (III)

Con todo isto, temos xa descrito a través das capturas e da explicación como desenaríamos a parte da interface gráfica da aplicación que permitiría a xestión do noso fogar intelixente, baixo o noso parecer, da forma máis axeitada.

5. Casos de Proba

Co obxectivo de comprobar que a implementación realizada é correcta, resulta conveniente ter definidos algúns casos de proba, é dicir, situacións que se utilizarán para cerciorarnos de que a funcionalidade que fomos describindo ata o de agora está presente no sistema. Deste xeito, presentamos a continuación os casos de proba que nos pareceron axeitados para cada un dos casos de uso que describimos previamente.

5.1. Caso de Proba 1

- **Caso de Uso:** Rexistrar SO.
- **Funcionalidade a probar:** O sistema introduce os datos do SO na SID ao ser válidos.
- **Estado Inicial:** Escollemos a opción para rexistar novos SOs e estamos en condicións de introducir os datos dun novo SO.
- **Entrada:** Insertamos un nome correcto e unha dirección IP válida.
- **Saída Esperada:** O sistema comproba que se introduciu algo correcto no campo do nome e da IP, é dicir, non están baleiros e teñen un formato válido, polo que se gardarán os datos e se informará de que todo rematou correctamente.

5.2. Caso de Proba 2

- **Caso de Uso:** Rexistrar SO.
- **Funcionalidade a probar:** O sistema non introduce os datos do SO na SID porque se introduciu a IP cun formato incorrecto.
- **Estado Inicial:** Escollemos a opción para rexistar novos SOs e estamos en condicións de introducir os datos dun novo SO.
- **Entrada:** Insertamos incorrectamente a IP, é dicir, escribímola cun formato incorrecto (incumprindo o formato textual dunha dirección IP).
- **Saída Esperada:** O sistema atopa de que hai errores no campo da IP e informa do erro que se produciu, rematando así a operación.

5.3. Caso de Proba 3

- **Caso de Uso:** Rexistrar SO.
- **Funcionalidade a probar:** O sistema non introduce os datos do SO na SID porque se introduciu o nome nun formato incorrecto.
- **Estado Inicial:** Escollemos a opción para rexistar novos SOs e estamos en condicións de introducir os datos dun novo SO.
- **Entrada:** Insertamos incorrectamente o nome, é dicir, escribímolo cun formato incorrecto (demasiada extensión ou caracteres non permitidos).
- **Saída Esperada:** O sistema atopa de que hai errores no campo do nome e informa do erro que se produciu, rematando así a operación.

5.4. Caso de Proba 4

- **Caso de Uso:** Rexistrar SO.
- **Funcionalidade a probar:** O sistema non introduce os datos do SO na SID porque non se cubriron os campos asociados ao rexistro (nome e IP).
- **Estado Inicial:** Escollemos a opción para rexistar novos SOs e estamos en condicións de introducir os datos dun novo SO.
- **Entrada:** Non escribimos nada en calquera dos dous campos: no nome ou na IP.
- **Saída Esperada:** O sistema atopa de que falta información para poder abordar a inserción e informa do erro, rematando así esta operación.

5.5. Caso de Proba 5

- **Caso de Uso:** Rexistrar SO.
- **Funcionalidade a probar:** O sistema non introduce os datos do SO na SID porque xa existe un SO rexistrado coa mesma IP.
- **Estado Inicial:** Escollemos a opción para rexistrar novos SOs e estamos en condicións de introducir os datos dun novo SO.
- **Entrada:** Escribimos un nome calquera e unha IP que tamén teña un SO xa rexistrado na base de datos.
- **Saída Esperada:** O sistema atópase con problemas ao tentar insertar na base de datos dado que a IP xa pertence a outro SO, polo que informará do problema, rematando así a operación.

5.6. Caso de Proba 6

- **Caso de Uso:** Eliminar SO.
- **Funcionalidade a probar:** O sistema elimina a información do SO de todas as bases de datos.
- **Estado Inicial:** Escollemos a opción do menú para eliminar un SO e estaremos en condicións de seleccionar un deles para eliminarlo.
- **Entrada:** Escollemos un dos SOs amosados para eliminar.
- **Saída Esperada:** O sistema accede á FD, á DID e á SID, elimina aquela información asociada ao SO seleccionado para eliminar, e indica que rematou a operación correctamente.

5.7. Caso de Proba 7

- **Caso de Uso:** Seleccionar SO.
- **Funcionalidade a probar:** O usuario decide activar un dos SOs rexistrados, polo que o sistema envía a dito SO unha petición de información, cuxo resultado se gardará na DID.
- **Estado Inicial:** Escollemos a opción do menú para seleccionar un SO, e estaremos en condicións de escoller un deles.
- **Entrada:** Escollemos un SO desactivado e activámolo.
- **Saída Esperada:** O sistema comproba que se ten que activar o SO, enviando unha solicitude de información ao mesmo que resulta correcta e almacena o resultado recibido na DID, avisando de que se puido levar a cabo a activación correctamente.

5.8. Caso de Proba 8

- **Caso de Uso:** Seleccionar SO
- **Funcionalidade a probar:** O usuario decide desactivar un dos SOs rexistrados, polo que o sistema almacenará que o SO deja de estar activo.
- **Estado Inicial:** O usuario escolle a opción de seleccionar dende o menú da aplicación e ten a posibilidade de escoller un dos SO.
- **Entrada:** Escollemos un SO activado e desactivámolo.
- **Saída Esperada:** O sistema comproba que se ten que desactivar o SO, almacenando na DID (na base de datos de información dinámica) que dito SO pasa a estar inactivo. Para rematar, informa de que todo rematou correctamente.

5.9. Caso de Proba 9

- **Caso de Uso:** Configurar SO por formulario
- **Funcionalidade a probar:** O sistema procesa correctamente os campos introducidos polo usuario ao ser correctos e almacena a lóxica difusa xerada.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por formulario.
- **Entrada:** Dentro do modo por formulario, o usuario introduce os campos de forma correcta.
- **Saída Esperada:** O sistema comprobará que a información é válida e xerará a lóxica difusa que posteriormente se almacenará na FD, rematando así a operación.

5.10. Caso de Proba 10

- **Caso de Uso:** Configurar SO por formulario
- **Funcionalidade a probar:** O sistema non xera lóxica algúna, posto que non se introduciu algún dos campos necesarios.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por formulario.
- **Entrada:** Non se introduce algún dos campos do formulario disponible.
- **Saída Esperada:** O sistema comprobará que hai campos baleiros e informa dos problemas, rematando a operación.

5.11. Caso de Proba 11

- **Caso de Uso:** Configurar SO por formulario
- **Funcionalidade a probar:** O sistema non xera lóxica algúna, xa que se insertou información que leva a regras que son incorrectas.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por formulario.
- **Entrada:** Introduceuse algún dato para configurar o SO que provocaría errores na xeración das regras SCL (por exemplo, un rango de temperaturas incorrecto).
- **Saída Esperada:** O sistema comprobará que hai datos incorrectos, informa dos problemas e remata a operación.

5.12. Caso de Proba 12

- **Caso de Uso:** Configurar SO mediante un editor
- **Funcionalidade a probar:** O sistema procesa correctamente as regras introducidas ao ser correctas e almacena a lóxica difusa xerada.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por editor.
- **Entrada:** Insertamos o código SCL de maneira correcta.
- **Saída Esperada:** O sistema comprobará que a información e a sintaxe sexan correctas, xerará a lóxica difusa, e almacenará dita lóxica na base de datos, informando do éxito da operación.

5.13. Caso de Proba 13

- **Caso de Uso:** Configurar SO mediante un editor
- **Funcionalidade a probar:** O sistema non xera lóxica difusa nin a almacena posto que non se introduciu ningunha regra.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por editor.
- **Entrada:** Non se inserta nada, é dicir, ningunha regra.
- **Saída Esperada:** O sistema comprobará que non se introduciron regras SCL e informa do erro, rematando así a operación.

5.14. Caso de Proba 14

- **Caso de Uso:** Configurar SO mediante un editor
- **Funcionalidade a probar:** O sistema non xera lóxica difusa nin a almacena posto que se introduciron regras non válidas.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo por editor.
- **Entrada:** Introducíense regras SCL que presentan errores (formato ou sintaxe) e polo tanto non son válidas.
- **Saída Esperada:** O sistema comprobará que se introduciron regras SCL con errores e informa do problema, rematando así a operación.

5.15. Caso de Proba 15

- **Caso de Uso:** Configurar SO no modo avanzado
- **Funcionalidade a probar:** O sistema garda a lóxica difusa introducida porque é correcta.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo avanzado.
- **Entrada:** Introducimos lóxica difusa correctamente.
- **Saída Esperada:** O sistema comprobará que se introduciron as regras e que a lóxica é correcta, almacenándoa na base de datos.

5.16. Caso de Proba 16

- **Caso de Uso:** Configurar SO no modo avanzado
- **Funcionalidade a probar:** O sistema non garda nada porque non se introduciu ningunha regra.
- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo avanzado.
- **Entrada:** O usuario non introduce regras para o SO.
- **Saída Esperada:** O sistema comproba que non se introduciron regras e informa ao usuario da situación.

5.17. Caso de Proba 17

- **Caso de Uso:** Configurar SO no modo avanzado
- **Funcionalidade a probar:** O sistema non garda a lóxica introducida porque presenta errores.

- **Estado Inicial:** Escollemos a opción de configurar un SO, e seleccionamos o modo avanzado.
- **Entrada:** O usuario introduce regras incorrectas: con errores sintácticos.
- **Saída Esperada:** O sistema comproba que a lóxica introducida non é valida e informa dos problemas atopados ao usuario.

5.18. Caso de Proba 18

- **Caso de Uso:** Planificar actuación
- **Funcionalidade a probar:** O sistema prepara aos SOs, os comandos de actuación correspondentes segundo as súas medicións e lóxica difusa, por activación dunha alarma.
- **Estado Inicial:** A alarma actívase e avisa ao sistema para comenzar a configuración.
- **Entrada:** O usuario configurou previamente algún dos SOs e xerouse a raíz diso algunha regra que provocará que se xeren comandos de actuación para os obxectos intelixentes.
- **Saída Esperada:** O sistema recolle toda a lóxica difusa e información dos SOs para xerar os comandos de actuación que a continuación enviará a eses obxectos.

5.19. Caso de Proba 19

- **Caso de Uso:** Planificar actuación
- **Funcionalidade a probar:** Tras comprobar as medicións do SO e a lóxica difusa asociada ao mesmo, o sistema non ten que xerar comandos de actuación nin, polo tanto, envialos.
- **Estado Inicial:** A alarma actívase e avisa ao sistema para comenzar a configuración.
- **Entrada:** O usuario non realizou ningunha configuración previa sobre algún SO que implique o envío de comandos de actuación trala planificación.
- **Saída Esperada:** O sistema recolle toda a lóxica difusa e información dos SOs pero non ten que enviar nada, rematando así a execución.

5.20. Caso de Proba 20

- **Caso de Uso:** Visualizar información
- **Funcionalidade a probar:** O sistema amosa información xeral de todos os SOs activos correctamente.
- **Estado Inicial:** Escollemos a opción de visualizar información, e estaremos en condicións de consultar información dos SOs.
- **Entrada:** O usuario non ten que introducir nada a maiores.
- **Saída Esperada:** O sistema devolve correctamente un resumo da información dos SOs activos.

6. Diagrama de Compoñentes

No seguinte punto centrarános en tratar o diagrama de compoñentes que deseñamos no contexto do noso SCFI. Na elaboración de dito diagrama tentouse tamén o emprego de diferentes cores e o resultado pódese consultar na Figura 19.

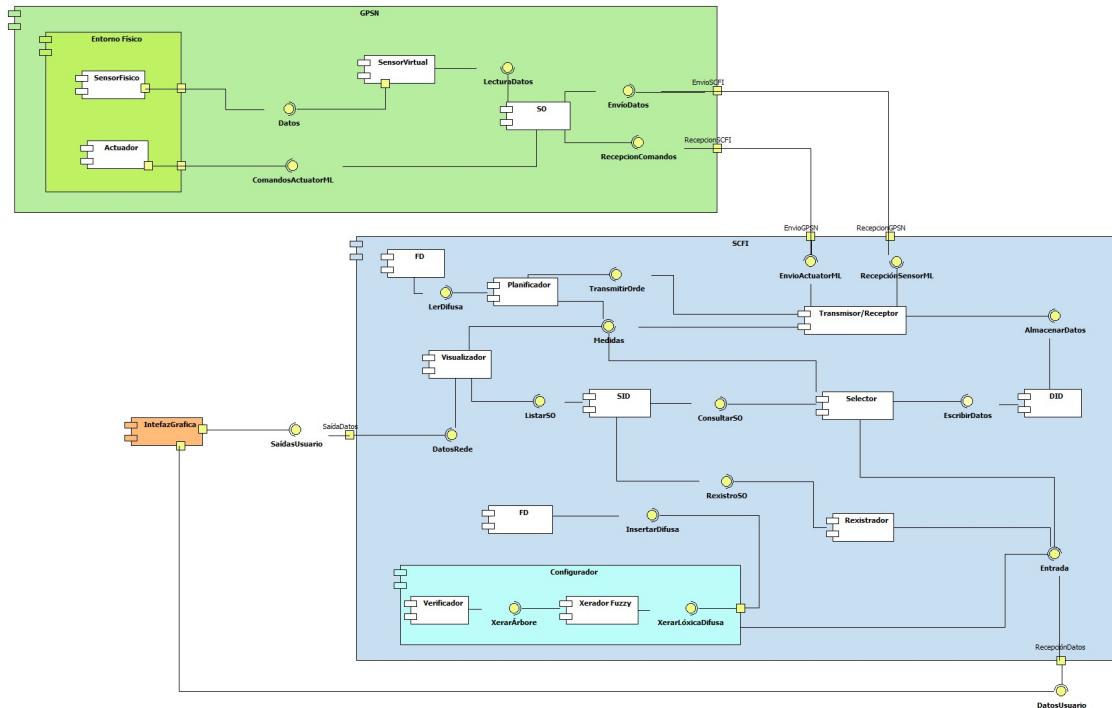


Figura 19: Diagrama de compoñentes

O diagrama de compoñentes é unha ferramenta que permite representar de forma estática o sistema de información. Este diagrama proporciona unha vista de alto nivel dos compoñentes do sistema, que poden ser dende software, unha base de datos, unha interfaz de usuario ou ata unha compoñente hardware como un circuito ou un dispositivo.

No noso proxecto fixemos un diagrama de compoñentes que representa unha visión completa do sistema (**maioritariamente** a nivel do software, inda que se introduciu algúns compoñentes hardware), incluíndo tanto o propio sistema de control do fogar intelixente, o SCFI, como unha visión da rede de comunicación dos obxectos intelixentes, a GPSN.

Decidimos facer unha análise destes dous subsistemas xa que con iso podemos ter unha visión máis ampla de como o sistema principal que estamos a deseñar interactúa co exterior mediante a comunicación cos obxectos intelixentes, e estes, á súa vez, cos seus sensores físicos e os seus actuadores. Ademais, engadimos unha compoñente situada fóra dos subsistemas presentados (dado que non é parte nin da rede nin do sistema de control propiamente dito), que representará a interface gráfica coa que o usuario interactúa.

A continuación, exporemos en detalle os dous subsistemas recollidos no diagrama e esa compoñente adicional, e faremos un razoamento das decisións tomadas en canto ao motivo de desenvolverlas dese xeito.

6.1. GPSN: Rede de comunicación

Neste subsistema atópanse, principalmente, os obxectos intelixentes e os seus sensores virtuais. Será o responsable de recoller os datos do medio físico e de enviar as ordes aos actuadores. Deste xeito será o que faga de ponte co SCFI transmitindo os datos entre o

exterior e o sistema principal.

A continuación describimos as diferentes compoñentes consideradas dentro do subistema GPSN:

- **Sensor Virtual:** Encargado de recibir a información dos sensores físicos. Este sensor comunícase co SO enviando os datos das medicións a través da interface para a lectura de datos, a cal lle permite proporcionar a funcionalidade de transmitir os datos obtidos dos sensores físicos (a través doutra interface), facendo que o SO poida obter finalmente eses datos.
- **SO:** Encárgase da comunicación cos sensores (virtuais) e actuadores do sistema. Máis detalladamente, recibe as medicións do sensor virtual, envía comandos de actuación aos actuadores e comunícase bidireccionalmente co SCFI, ao cal lle envía os datos recollidos dos sensores e do que recibe esos comandos de actuación que os actuadores poderán recibir posteriormente.
- **Entorno Físico:** Este subsistema da GPSN fai referencia á parte externa, é dicir, á parte física que interactúa co noso sistema. Consideramos conveniente engadila xa que esta parte comunícase cos SOs e cos sensores virtuais correspondentes, e estas compoñentes resultan tamén de relevancia para o SCFI. Dentro do entorno físico cómpre destacar dúas compoñentes:
 - **Sensor Físico:** Encárgase de recuperar os datos do mundo real e enviarlos ao sensor virtual para que este os procese.
 - **Actuador:** Encárgase de executar os comandos que lle son enviados polos obxectos intelixentes.

Desta forma reflexamos o que consideramos como unha compoñente necesaria para o sistema, xa que dalgunha forma é a base sobre a que se cementa a funcionalidade do SCFI.

6.2. SCFI: Sistema de Control do Fogar Intelixente

Este subsistema supón a parte máis importante do diagrama, posto que dentro deste teremos todos os compoñentes necesarios para controlar os obxectos intelixentes así como a interacción co usuario, ben sexa proporcionando interfaces para a entrada de datos ou interfaces para a saída de información por pantalla para que o usuario interactúe ou consulte, respectivamente.

A continuación describimos as distintas compoñentes deste subistema:

- **Rexistrador:** Esta compoñente encárgase de rexistrar os datos dos SO que introduce o usuario (que serán o nome e a IP do SO que se quere rexistar). A SID ofrece a funcionalidade de introducir datos na propia base, e reflexamos isto mediante a interface de rexistro dun SO.
- **Selector:** O selector encárgase de activar ou desactivar os SO. Comunícase ca SID para listar os SO disponibles mediante o uso da interface que permite consultar os obxectos gardados na mesma. Tamén se comunica co Transmisor-Receptor, para que este lle indique ao SO seleccionado que se necesita obter a información. Recollemos que esta información recibida, xunto coa confirmación de que o sensor se atopa activo, se almacena na DID mediante a interface de almacenamento de datos ofrecida pola mencionada base de datos.

- **Transmisor/Receptor:** Encárgase da comunicación cos SOs, podendo realizar transmisión (con comandos *ActuatorML*) e recepción (con comandos *SensorML*) de datos. É empregado polo Visualizador, polo Selector e polo Planificador para recibir datos dos SOs, e polo Planificador tamén para enviar datos directamente a eses obxectos. No sentido deste último uso, ofrece a posibilidade de transmitir ordes mediante mensaxes *ActuatorML*. Toda esa funcionalidade que se describiu vén representada a través das súas interfaces.
- **Planificador:** Encárgase de enviar os plans de actuación aos SO. Para iso comunícase co Transmisor/Receptor, ao cal lle envía ditas instruccións. Por outro lado tamén precisa de ler datos da Base de Datos Difusa (FD), para coñecer as funcións de membresía e a lóxica difusa que permiten controlar os SOs, e faino coa interface que esta ofrece para ler.
- **Visualizador:** Encárgase de amosar os datos dos SOs aos usuarios. Para iso precisa ler os datos recollidos na SID e tamén obter medidas dos obxectos intelixentes. Esta comunicación faise a través das interfaces xa mencionadas anteriormente, ademais, o visualizador interactúa coa SID para poder amosar a información básica dos SOs. Tamén resulta fundamental destacar que esta compoñente ofrece a interface de saída de información para o usuario.
- **Bases de datos:** No caso das bases de datos pensamos que é máis adecuado ter unha compoñente para cada base en lugar dun subsistema que as agrupe a todas elas, posto que isto nos permite visualizar a interacción coas demais compoñentes dunha forma más clara. Deste xeito diferenciamos:
 - **SID:** Esta compoñente do sistema almacena a información estática (IP, nome) dos SO. O Rexistrador envíalle a información de rexistro, a cal é almacenada nesta base de datos. Ofrece a funcionalidade de ler e escribir os datos estáticos dos obxectos intelixentes.
 - **DID:** Esta compoñente almacena a información dinámica dos SOs. Ofrece a funcionalidade a través das interfaces de ler e escribir na propia compoñente.
 - **FD:** Esta compoñente almacena as funcións de membresía e regras difusas, e ofrece coas súas interfaces, tanto ao configurador como ao planificador, a funcionalidade de escribir e ler nela. Anotar neste punto que tivemos que duplicar esta compoñente do sistema para poder evitar cruces de liñas que fagan ilexible o diagrama por completo e provoquen confusión, de aí que apareza dúas veces.
- Poderíase tamén considerar a opción de eliminar datos na base de datos, pero dada a complexidade do diagrama e para evitar introducir demasiadas relacóns e sobrecargalo, pensamos que é mellor deixala aparte nesta descripción de compoñentes e centrarnos nos aspectos fundamentais do sistema que ademais viñan descritos inicialmente (a posibilidade de eliminar os datos dos SOs engadímola nós a maiores).
- **Configurador:** Cremos que é conveniente representar esta compoñente como un subsistema do propio SCFI, xa que dentro dela se levan a cabo outras accións que teñen algo en común: a configuración dun SO. A función principal desta compoñente será recibir os datos introducidos polo usuario para a configuración dun SO e producir a lóxica difusa derivada.

Dentro desta compoñente teremos outras dúas, que serán o verificador de sintaxe de SCL e o Xerador Fuzzy. A continuación procederemos a describelas:

- **Verificador de sintaxe:** Esta compoñente recibe o código SCL introducido polo usuario. Dito código procésase e posteriormente verícase se a sintaxe do mesmo é correcta. Se é así, xérase unha árbore de expresión, que será ofrecida ao xerador Fuzzy, o que representamos mediante a interface ‘Xerar Árbore’.
- **Xerador Fuzzy:** O xerador fuzzy encárgase de xerar as funcións de membresía e as regras difusas que serán gardadas na Base de datos Fuzzy. Primeiro obtén a funcionalidade do verificador, o cal xera a árbore que esta compoñente convertirá en lóxica difusa. Despois, ofrece a funcionalidade de xerar a lóxica difusa, que será almacenada na base de datos correspondente grazas a outra interfaz ofertada pola FD: ‘*InsertarDifusa*’.

6.3. Interfaz Gráfica

Esta última compoñente que consideramos ten a misión de comunicarse co usuario, ben sexa para recoller os datos necesarios que precisa o sistema do SCFI, como pode ser aqueles para rexistrar un novo SO ou para configuralo, ben para amosar os datos xerados polo sistema ao usuario. Este subsistema xestiona as comunicacións de entrada/saída coa interacción humana, por iso, representámolo fóra do que é o propio SCFI que deixamos para a parte máis ‘interna’. De todos os xeitos, existe gran relación entre o mencionado subsistema e a interface.

7. Conceptos susceptibles de ser clases

A parte restante do noso proceso de deseño desenvolveuse en varias fases, fases que temos que tentar relacionar dalgún xeito e amosalas dunha maneira coherente neste documento, porque algunas das cuestións poderían presentarse en diferente orde sen problema algúin. Comezamos distinguindo algúns dos conceptos más importantes que poderían ser identificados coma clases no noso deseño; logo fomos elaborando os primeiros diagramas de secuencia, ao mesmo tempo que montamos o diagrama de clases coas relacións que se derivaron. O seguinte paso foi a introdución de novos patróns de deseño (aparte dos aplicados dende o comezo: DAO, Fachada e MVC), o que provocou cambios en todos os diagramas e tamén chegou a afectar a algún caso de uso e diagrama de actividade, que reestructuramos convenientemente para tentar manter a coherencia de todo o modelo.

Dado que se poderían amosar todos estes elementos de diferentes xeitos, decidimos indicar aquí a decisión que tomamos para construír os demais apartados do documento: presentaremos primeiro os conceptos que consideramos susceptibles de seren clases neste apartado (xa que así o lector poderá entender mellor os seguintes apartados que se presentarán), logo describiremos os patróns que foron aplicados e o diagrama de clases correspondente e, finalmente, presentaremos os diagramas de secuencia para cada un dos casos de uso. Esta é a orde que nos pareceu máis coherente para amosar o contido e que o lector o poida seguir da mellor maneira posíbel.

Comentar que as clases que se van a amosar a continuación non se definiron nunha primeira iteración, senón que moitas delas foron xurdindo a medida que avanzamos no proceso de deseño. Tamén, tivemos en conta as ideas presentadas na clase e que se recollen no campus virtual sobre algúns patróns de deseño. Concretamente:

- Tentamos realizar unha separación entre a parte de interacción co usuario, as respostas aos seus comandos e o control interno do sistema seguindo o patrón MVC.
- Decidimos representar as clases que modelan as bases de datos como clases DAO (*Data Access Object*).
- Finalmente, tamén tentamos na medida do posible seguir o patrón Fachada, incluíndo clases que sirvan de referencia, conectando cun conxunto doutras clases e realizan diferentes xestións.

Deste xeito, imos amosar unha pequena descripción das clases que consideramos que, de entrada, deberíamos ter no SCFI, para o que distinguiremos algúns grupos:

- Por unha banda, diferenciamos claramente a parte da interfaz gráfica. Teremos, ademáis dunha FachadaGUI, outras tres clases que se corresponden, ademais, cos tres grandes grupos de pantallas amosados no Apartado 4:
 - VentaXestion: para a parte de rexistro, eliminación e selección de SOs.
 - VentaConfiguracion: para a parte de configuración de SOs en calquera das tres variantes (formulario, editor, avanzado).
 - VentaVisualizacion: para a parte de visualización da información.
- Por outro lado, temos claro tamén que hai que distinguir a base de datos do demais. Isto levámolo a cabo introducindo unha clase FachadaBD que se comunicará cos DAOs:
 - DAOSID: para acceso á SID (Información estática).
 - DAODID: para acceso á DID (Información dinámica).
 - DAOFD: para acceso á FD (Lóxica difusa).
- Temos outra fachada, chamada FachadaXestion, que se irá comunicando con outras compoñentes xa descritas e que se atopará máis enfocada na parte de xestión de SOs (rexistro, eliminación, selección...), de aí o seu nome.
- A última fachada das que consideramos incialmente é a FachadaConfiguracion e, como o seu propio nome indica, estará enfocada na parte de configuración e planificación interna. Consideramos conveniente facer esta distinción para non sobrecargar unha única clase con toda a parte de control interno. Ademais, esta fachada estará vinculada con outras dúas clases para repartir máis as responsabilidades:
 - Configurador: é o que se adicará á parte de configuración. Nesa clase atoparanse métodos propios da compoñente homónima: verificar a sintaxe SCL e xerar as regras difusas. Para isto introducimos outras dúas clases que nos permitan esta funcionalidade:
 - O VerificadorSCL será o que se adique á parte da configuración relativa ao SCL: xerar regras SCL e analizalas.
 - O XeradorFuzzy será o encargado de verificar e xerar as regras difusas a partir da árbore de análise obtida no verificador.
 - Planificador: estará exclusivamente enfocado na tarefa de planificación; é dicir, a partir das regras difusas e dos datos do SO, tratará de proporcionar os comandos de actuación que terán que ser enviados aos SOs.
- Finalmente, decidimos distinguir unha clase pola importancia da compoñente que representa, o SO, que será empregada dende outras para o almacenamento de datos como o nome ou a IP, e para a comunicación cos SOs. Ademáis, decidimos representar tamén os sensores e actuadores asociados a cada un destes SO mediante as clases homónimas (Sensor e Actuador).

Tamén destacar que incluimos unha clase para representar as regras difusas (*RegraDifusa*), nas que se almacenará, para cada unha delas, as condicións e a respectiva forma de actuación (as regras difusas son do tipo se [evento] entón [acción]).

Isto non é todo, senón que somentes conseguimos ter unha primeira aproximación ao que son as clases que deseñamos. Fáltannos os patróns de deseño que foron aplicados a maiores do MVC, da Fachada e do DAO, tema no que nos centraremos no seguinte apartado.

8. Patróns

A continuación, explicaremos como introducimos os patróns no noso deseño do SCFI a partir do conxunto de clases descrito previamente. Procuramos aplicar patróns dos tres tipos vistos, para dar algo de variedade: de creación, estruturais e de comportamento. Ademais, tentouse sempre buscar unha utilidade real e que nos permita un deseño mellor do noso sistema ou, en todo caso, que se proporcione unha utilidade que permita que a realización de modificacións futuras non resulten demasiado complexas de efectuar.

Optamos por empregar catro patróns, cada un deles nun contexto concreto que desenvolveremos e xustificaremos a continuación: o *Abstract Factory* (de creación), o *Proxy* (estrutural), o *Strategy* (de comportamento) e o *State* (de comportamento).

8.1. Patrón *Strategy* (Estratexia)

Este é un patrón de comportamento, que nos permite definir unha familia de algoritmos, encapsulalos, e facelos intercambiables de maneira sinxela.

No noso proxecto, decidimos aplicar este patrón sobre a parte do configurador. Se ben temos os métodos para a configuración na clase *Configurador*, a parte de SCL é xestionada por un lado e a da lóxica difusa por outro. Este patrón estratexia permitenos definir un algoritmo para, por exemplo, analizar a sintaxe do SCL, e poder ter diferentes implementadores concretos de este algoritmo (é dicir, diferentes clases que implementen cada unha das operacións de xeito diferente), de forma que se poidan intercambiar se é necesario nalgún momento.

Desta forma, temos dúas interfaces, que son *VerificadorSCL* e *XeradorFuzzy*, e para cada unha destas, temos un implementador concreto que nos proporciona a funcionalidade (estes serán o *VerificadorConcreto* e *XeradorConcreto*), sendo sinxelo poder engadir outros implementadores que empreguen variacións dos algoritmos para cada unha das tarefas.

Podemos dicir, en conclusión, que este patrón o aplicamos dúas veces: unha na clase que implementa os métodos da verificación de linguaxe SCL e xeración de regras, e outra na clase do xerador *Fuzzy*.

8.2. Patrón *State* (Estado)

O patrón de estado é o segundo patrón de comportamento que empregaremos, e este permítenos alterar o comportamento dun obxecto concreto cando o seu estado cambia.

Poderíamos valorar a aplicación deste patrón nalgunha das nosas clases, onde poida existir un comportamento variable, e onde poida tamén haber a posibilidade de ter novos

estados nun futuro. A clase que seleccionamos foi a que representa o obxecto intelixente, pois así tamén separamos o comportamento variable do mesmo, e repartindo máis as responsabilidades desa clase.

Antes de comezar coa aplicación do patrón, resulta fundamental coñecer os estados polos cales pode pasar un SO, para o que nos foi útil elaborar en primeiro lugar un **diagrama de estados**, que precisamente nos permite reflectir o comportamento dunha parte do sistema, polo que nos parece interesante empregalo no contexto dos SOs.

Deste xeito, tamén incluiremos aquí o mencionado diagrama de estados, que podemos consultar na figura 20. A continuación, faremos un pequeno comentario sobre o mesmo.

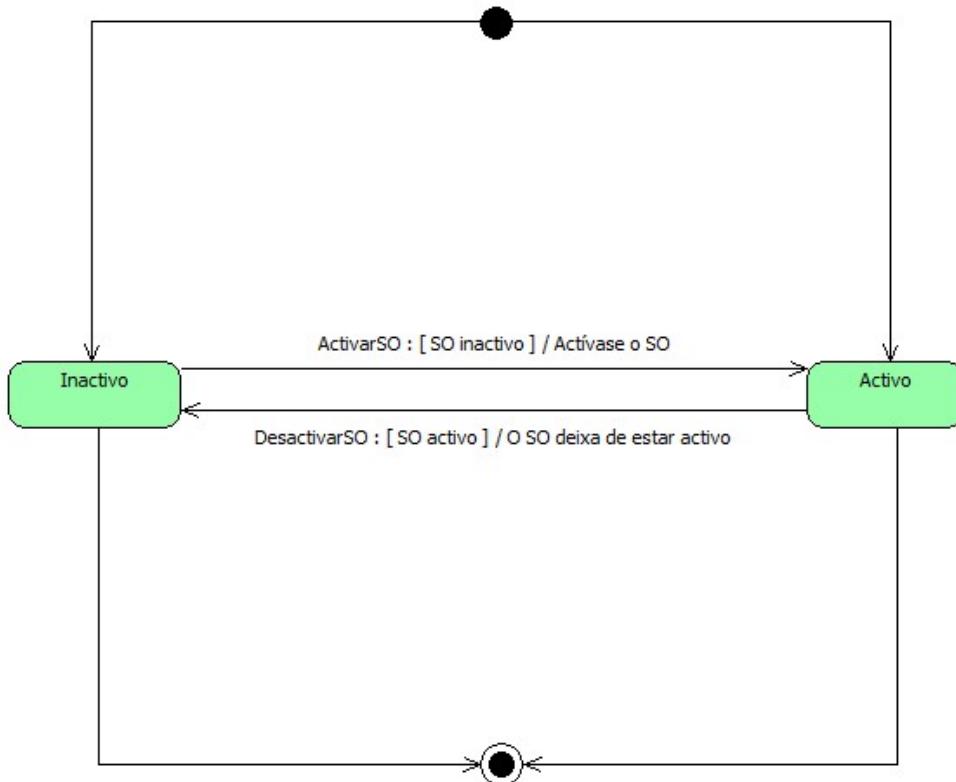


Figura 20: Diagrama de estados para o SO

Podemos observar en dito diagrama que tratamos de amosar os dous estados polos que cremos que pode pasar un SO: cando está seleccionado como activo e cando está seleccionado como inactivo. Ademais, obsérvanse as posíbeis transicións entre estados, que veñen dadas cando un SO concreto se selecciona e se cambia manualmente o estado (a través do caso de uso **Seleccionar SO**).

Este diagrama creámolo centrándonos na consideración do SO coma un obxecto, sen ter en conta a súa vida completa no sistema.

Deste xeito, entendemos que un obxecto SO se podería crear en calquera dos dous estados: depende de se é un novo SO introducido no sistema (creándose nese caso inactivo) ou se se crea a partir dos datos almacenados da base de datos (considerándose activo ou non dependendo da información que se teña na DID).

Ademais, este obxecto podería destruirse en calquera estado, posto que este non queda almacenado constantemente na aplicación mentres se teña rexistrado o SO, senón que só estará presente cando teña que ser xestionado ou empregado dalgún xeito.

Con isto temos xustificada a nosa presentación do diagrama de estados relativo á aplicación do patrón estado no SO, polo que agora nos centraremos nas clases engadidas. Intentamos aplicar o patrón seguindo os esquemas atopados pola rede e nos apuntamentos da materia, comezando por introducir unha primeira clase que será abstracta: `EstadoSO`, na que se terán métodos e atributos comúns aos dous estados. Algúns deses métodos serán abstractos, e estes serán nos que se reflexe ese comportamento variante en función do estado.

A maiores, nesa clase abstracta teranse asociacións con outras clases: `RegraDifusa`, `Sensor` e `Actuador`, que estaban inicialmente vinculadas ao SO. Facémolo así porque tamén os métodos que acceden ás regras, sensores e actuadores do SO e os manipulan se atopan nesta clase, polo que nos parece a representación máis coherente, como veremos tamén no diagrama de clases.

Para rematar coa descripción da aplicación deste patrón no noso modelo, indicar que se crearon dúas clases concretas para os dous estados: `Activo` e `Inactivo`. En cada unha delas, darase implementación a esos métodos abstractos da clase `EstadoSO` que varían en función do estado. Neles incluímos:

- A solicitude de información e o envío de comandos de actuación do SCFI ao SO: se o SO está activo poderase levar a cabo, pero se está inactivo este método non pode facer ningún envío.
- O engadido de regras difusas, sensores e actuadores: se o SO está activo poderanse engadir, mais se está inactivo non nos ten moito sentido dispoñer de regras que regulen o funcionamento porque ese SO non está traballando nese momento.

Grazas a colocar estas operacións nas clases que engadimos derivado da aplicación deste patrón, podemos evitar operacións con estruturas condicionais en función do estado. Ademais, resultaría sinxelo engadir novo comportamento, un estado diferente (por exemplo, podería haber un estado intermedio no que se puidese recuperar igualmente información do SO pero non se puidese configurar nin planificar).

8.3. Patrón Abstract Factory (Fábrica Abstracta)

Este patrón de creación é o que nos permite definir unha clase que é capaz de producir familias de obxectos relacionados, sen especificar as súas clases concretas.

Vimos óptimo aplicar este patrón no relativo aos transductores (é dicir, tanto sensores coma actuadores) asociados a un obxecto intelixente. Por un lado, o estado asociado ao SO está relacionado cunha `FabricaTransductores`, que será a interface que nos permitirá acceder á funcionalidade de crear sensores e actuadores para o propio SO. Así, existirá un método para crear cada tipo de transductor.

Pareceunos que o máis conveniente é incluir en cada método de creación dun sensor unha medición que ten tipo de dato `Float` (dado que sempre que creemos un sensor será porque ben se recuperou información actual deles do SO ou ben se accedeu ao contido da base de datos, onde tamén entendemos que se almacenan as medicións), inda que

Isto podería adaptarse en función da información que se queira recopilar/almacenar nun futuro.

Ademais, disporemos de fábricas concretas que implementarán estes métodos para crear transdutores. Decidimos representar tres (`FabricaLuz`, `FabricaHumidade` e `FabricaTemperatura`) a modo de exemplo, pero poderíanse incluír máis tipos diferentes segundo o tipo de transductor que fose necesario no sistema.

Como as clases `Sensor` e `Actuador` asociadas ao SO son abstractas, as fábricas anteriores serán as encargadas de instanciar os tranductores específicos de cada tipo (teremos unha clase filla de sensor e unha filla de actuador, polo tanto, por cada tipo de transdutor).

8.4. Patrón Proxy (Apoderado)

Este patrón estructural permite unha fronteira para controlar o acceso dun obxecto a outro.

A aplicación deste último patrón veu motivada pola validación do nome e da IP ao rexistrar un SO. Como hai que verificar certas condicións de formato en cada caso, e para así tamén quitarlle responsabilidades ás fachadas, pareceunos útil que esa comprobación se faga antes de acceder á fachada da base de datos nunha clase diferente. Así, ocorréuse-nos a aplicación dese patrón nese contexto, tendo outra opción de, nun futuro, meter novas comprobacións no acceso á base de datos noutros contextos de xeito sinxelo.

Deste xeito, dispoñemos dunha interface `BaseDatos` que conterá os métodos para acceder á base de datos (en xeral: FD, DID e SID). Cando facemos unha chamada a un deles para acceder, pasamos primeiro polo `Validador`, que actúa como proxy, e nos controla que os datos que pasamos, como por exemplo a IP ou o nome, teñan o formato correcto que necesitamos de forma que poidamos quitar carga de traballo a outras clases e preveñamos erros (isto só o faremos no noso caso cando se introducen datos novos, é dicir, no rexistro, porque no resto dos casos trataremos con datos xa almacenados que entendemos que non teñen ningún problema).

Se este validador nolo permite, pasaríamos á fachada da base de datos e, dende aí, iríamos ao DAO correspondente segundo a operación. Tanto o validador coma a fachada da base de datos herdan os métodos da interface `BaseDatos`, tendo o validador os métodos adicionais necesarios para as comprobacións que sexan precisas.

9. Diagrama de clases

Resúltanos máis coherente presentar primeiro o diagrama de clases, porque así podremos entender como se atopan relacionadas as clases no noso modelo e, a partir de aí, facilitaremos a comprensión dos diagramas de secuencia cos que remataremos a presentación do noso deseño.

O diagrama de clase amosado na Figura 21 é resultado de avaliar as clases que precisamos para o sistema do fogar intelixente. Ao mesmo tempo, hai que comentar que se foi elaborando de xeito concorrente cos diagramas de secuencia facendo, ao mesmo tempo, as modificacións pertinentes pola introdución dos patróns de deseño.

No mencionado diagrama temos distinguidas con cores as diferentes partes do modelo: en cor azul claro, temos a parte da interface gráfica, con todas as ventás e unha fachada da

que se poderá ir a calquera delas. Entendemos que con esa fachada é coa que interactúa o usuario.

En segundo lugar, temos tres tonalidades de verde: na más clara representase a parte de xestión (que somentes ten a súa fachada), e na más escura a parte de configuración: a fachada correspondente, o configurador e o planificador. Ademais, xunto ao configurador temos a aplicación do patrón *Strategy*, tal e como explicamos no punto anterior, para a parte do Verificador de SCL e do Xerador Fuzzy.

Antes de mencionar o SO, tamén destacamos que, en cor morada temos a parte da base de datos, onde cabe sinalar a fachada da base de datos e o emprego do proxy para controlar o acceso a dita fachada, validando, cando é necesario, os campos que o precisen (en principio, ao rexistrar un novo SO, o nome e a IP do mesmo). Polo demais, hai que destacar que a fachada está vinculada aos DAOs (*Data Access Objects*) dos tres tipos de bases de datos que distinguimos: DID, SID E FD.

Nun verde intermedio está a parte do SO, unha parte delicada polas decisións que tomamos para que o deseño fose máis cómodo, intentando manter sempre a coherencia: como se pode ver polas dependencias que existen, resulta un obxecto moi importante. Primeiro centrarémonos en describir os elementos que componen esta parte do diagrama e logo, xustificaremos algunhas decisións.

Sobre os elementos desta parte en cuestión, hai que destacar os seguintes aspectos:

- A clase SO, que resulta a más importante e que estará vinculada, dalgún xeito, ao resto de clases da mesma cor (directa ou indirectamente). Nela hai moitas operacións, pero non hai que confundirse: moitas delas simplemente dan paso a outras clases nas que sí se resolve dita operación.
- A aplicación do patrón *State* sobre o SO, que se reflexa na aparición da clase abstracta EstadoSO, que ten algunas operacións non abstractas e outras que si o son (que varían en función do estado), e as clases concretas que herdan dela para modelar as dúas situacións nas que se pode atopar un SO determinado: Inactivo e Activo.
- A clase que representa as regras difusas que un SO pode ter asociadas. Entendemos que podería ter máis de unha regra, dependendo dos seus sensores e actuadores.
- Os sensores e actuadores que ten ese SO. Neste punto é onde aplicamos o patrón *Abstract Factory*, por iso incluimos unha fábrica de transductores como interface e unha fábrica concreta por cada tipo de transductor que consideramos, clases abstractas para sensor e actuador, e clases concretas para cada tipo de transductor. Os atributos que teñen os sensores e os actuadores cremos que son os más axeitados: un identificador e, no caso dos sensores, un Float para almacenar a medición tomada.

Sobre as decisións tomadas co SO, hai que destacar a cantidade de dependencias que existen con ese obxecto. A idea é a de ter un conxunto de obxectos nos que se encapsula a maior parte da información do SO e poidan ser empregados dende diferentes puntos do sistema, pero ao mesmo tempo é certo que non queremos que haxa dependencias con todas as clases, polo que intentamos buscar unha solución de compromiso que non aumentase moito a complexidade dos deseños e que, ao mesmo tempo, evitase demasiadas dependencias.

Deste xeito, hai unha dependencia do SO con todas as fachadas, e algunha compoñente máis, como o planificador ou o XeradorFuzzy, onde se precisa acceder a el para utilizar parte da súa información ou engadila. Preferimos facelo así e non pasar a información a través de parámetros das operacións porque serían moitos máis métodos e engadiríase moita máis complexidade no deseño, por exemplo, nos diagramas de secuencia. O mesmo ocorre na interface gráfica, pois neste caso estivemos pensando tamén que co noso deseño podería ser posible que se recuperaran máis dunha vez os mesmos datos dun SO no diagrama de secuencia, algo que se evita permitindo que chegue o SO a esas clases que representan as diferentes ventás. Hai que comentar tamén que non fixemos tanto fincapé nesa parte da interface para enfocarnos en maior detalle no resto de partes do sistema.

Onde si que se logrou evitar o emprego do SO é nos DAOs, pois temos a fachada por diante e nela podemos recoller a información exclusivamente necesaria para que os DAOs fagan o seu traballo, que estes devolvan o resultado e, se resulta preciso, a fachada almacene no SO a información (depende da situación). Tamén observamos deste xeito que as operacións posúen unha maior cantidade de argumentos, provocando que haxa menos dependencias pero tendo operacións más complexas (fixarse por exemplo no DAO da DID, onde hai unha gran cantidade de parámetros ao ter que pasar toda a información dinámica dos SOs por separado, en forma de conxuntos de datos). Ademais, ao non pasar o SO como parámetro, evitamos que clases como os DAOs non poidan acceder a unha posible funcionalidade ou modificar os propios obxectos intelixentes, tendo desta forma unha maior seguridade.

A clase SO, ademais, terá diferentes asociacións con outros obxectos que representan diferentes elementos relacionados cos SOs: os sensores e actuadores, as regras difusas creadas nas configuracións e o seu estado. Ademais, para evitar dependencias con estes obxectos, o SO terá operacións que invocarán directamente outras deses outros obxectos, e os métodos que recuperen información coma datos de regras difusas ou dos transdutores, proporcionarán directamente os datos separados (e con tipos de datos básicos: cadeas de caracteres, flotantes, arrays...), é dicir, as clases que usen o SO non terán que acceder para nada ao resto de obxectos cos que está asociado.

Para rematar, comentar algo de interese: moitas das operacións que en teoría non devolvían ningún tipo de parámetro facemos que devolvan un enteiro. Isto ten unha xustificación: hai moitas operacións que poderían fallar na base de datos, ou nunha validación. Para que dende a ventá se poida coñecer o estado, o que facemos é indicar que se devolvería un enteiro diferente para cada posible resultado de cada operación, de xeito que se poida avisar facilmente ao usuario de que todo foi correctamente ou de que, en caso de producirse un problema, poder saber cal foi o motivo exacto do mesmo. A maiores, inda que hai operacións que non devolven nada e que non deberían ter problemas segundo o deseño que fixemos, engadímosles ese enteiro por se o deseño cambiase e houbese que ter algunha outra consideración en conta. Dese xeito, dispone dunha maneira sinxela de avisar de problemas ou da corrección dos resultados producidos.

Polo demais, non temos que resaltar ningunha cuestión adicional deste diagrama, polo que con isto temos descrito con bastante detalle as decisións tomadas no mesmo, e agora poderemos comprender as clases que van sendo empregadas nos diferentes diagramas de secuencia que presentaremos no seguinte e derradeiro punto deste documento.

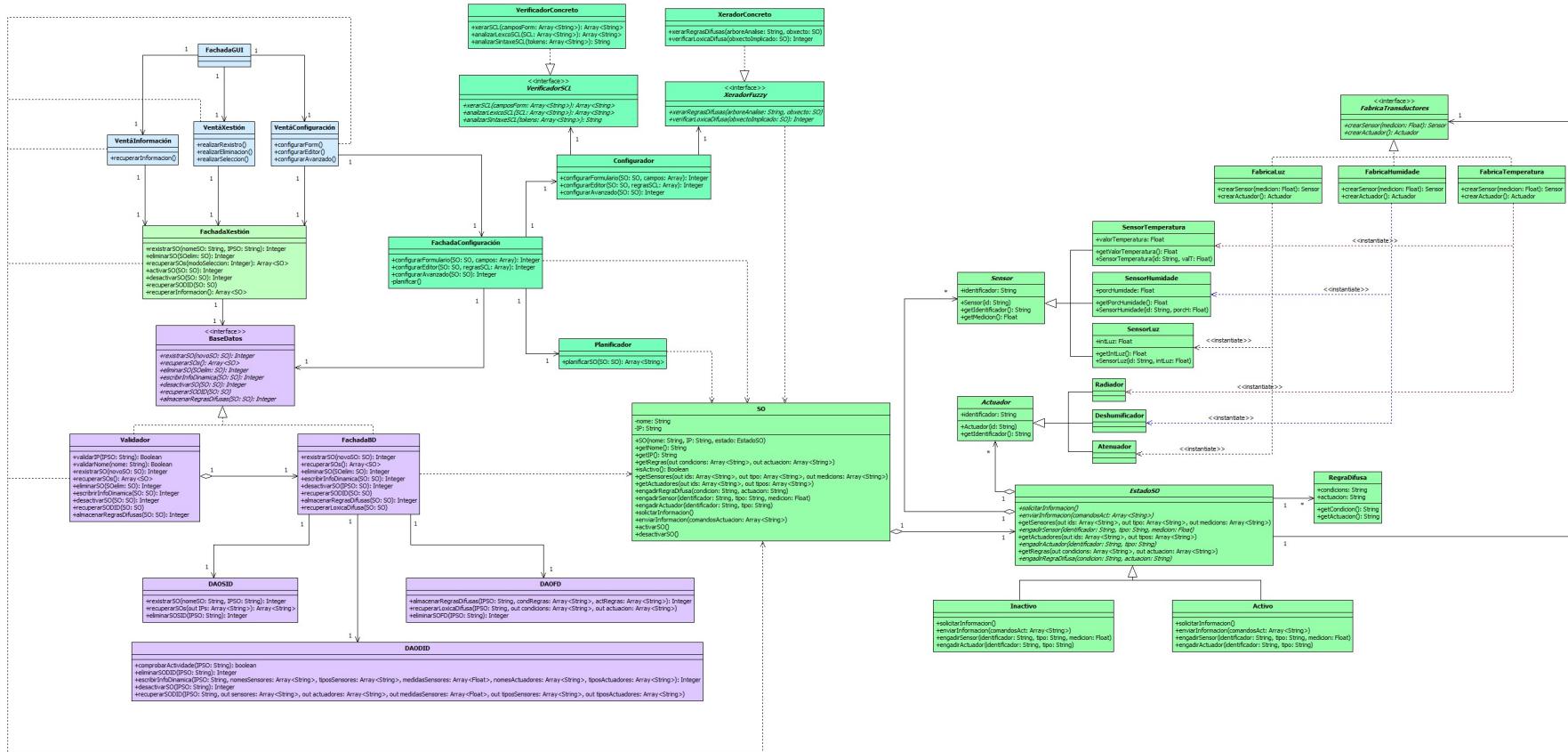


Figura 21: Diagrama de Clases

10. Diagramas de Secuencia

Para rematar, pasamos a amosar os diagramas de secuencia, unha vez que xa coñecemos as clases que poden participar (así será máis sinxelo entendelos). Para comezar con este apartado, pensamos que é importante ter en conta as seguintes consideracións que se describen:

- É relevante a lectura dos textos asociados a cada un dos diagramas, xa que se incorporan tanto a explicación das decisións tomadas coma as omisións realizadas, igualmente argumentadas para a comprensión completa deses diagramas ou entender algúns aspectos no que pensamos que pode haber dúbidas con moita facilidade.
- Incorporáronse certos diagramas que realmente non existen coma caso de uso, o que pretende xeralizar parte dos deseños evitando redundancias. Comezaremos precisamente por explicar ditos diagramas porque logo estes aparecerán referenciados en todos os demás.
- Do mesmo xeito, decidimos que os diagramas comenzañ a partir do estado no que os actores indicaron xa o que desexaban facer, o que representamos como un sinal que se envía do actor á primeira clase de cada diagrama. Isto non se aplica aos diagramas adicionais que xorden para evitar redundancias, onde se representa unha operación concreta que pode ser realizada repetidas veces en diferentes puntos do deseño.
- Intentamos incluir en todos os diagramas de secuencia os retornos máis interesantes, sobre todo cando hai algúns tipos de parámetro que se devolve ao rematar a operación, e así controlar mellor as secuencias que se amosan. No caso de que interveña o usuario ao comezo, sempre representaremos o retorno ata a Fachada da interface gráfica.

10.1. Diagramas Adicionais

Como comentamos na introducción, neste apartado amosaremos os comportamentos que serán recorrentes ao longo dos demás diagramas que representan os diferentes casos de uso, evitando así telos que reflexar completamente e repetidas veces neles.

A continuación, comezaremos comentando as cuestións más relevantes destes tres diagramas que xorden para representar ese comportamento habitual: *Recuperar SOs*, *Solicitar Información SO* e *Recuperar SO DID*.

10.1.1. Recuperar SOs

Este diagrama de secuencia empregarémoslo coma parte doutros diagramas asociados a distintos casos de uso, pola frecuencia coa que se teñen que recuperar todos os SOs da base de datos para poder facer algunha xestión con algún deles. Na Figura 22, pódese ampliar a información sobre o mesmo.

Salientar que o que se reflexa neste caso é un proceso mediante o cal, en primeiro lugar, se recuperan os SO da base de datos SID. Con eses datos vanse creando instancias de SOs e consultase se están activos ou non (accedendo á DID), o que se recolle en cada unha das instancias.

A idea é a seguinte: o SO créase por defecto en estado inactivo, pero se está activo o que se fai é o cambio de estado directamente (podemos dicir que pasa por un estado

inconsistente antes de poñerse adecuadamente). É por iso que consideramos que neste caso o SO se inicia de todos os xeitos no estado activo, e por iso o reflexamos así no diagrama de estados correspondente ao SO que tratamos na Figura 20.

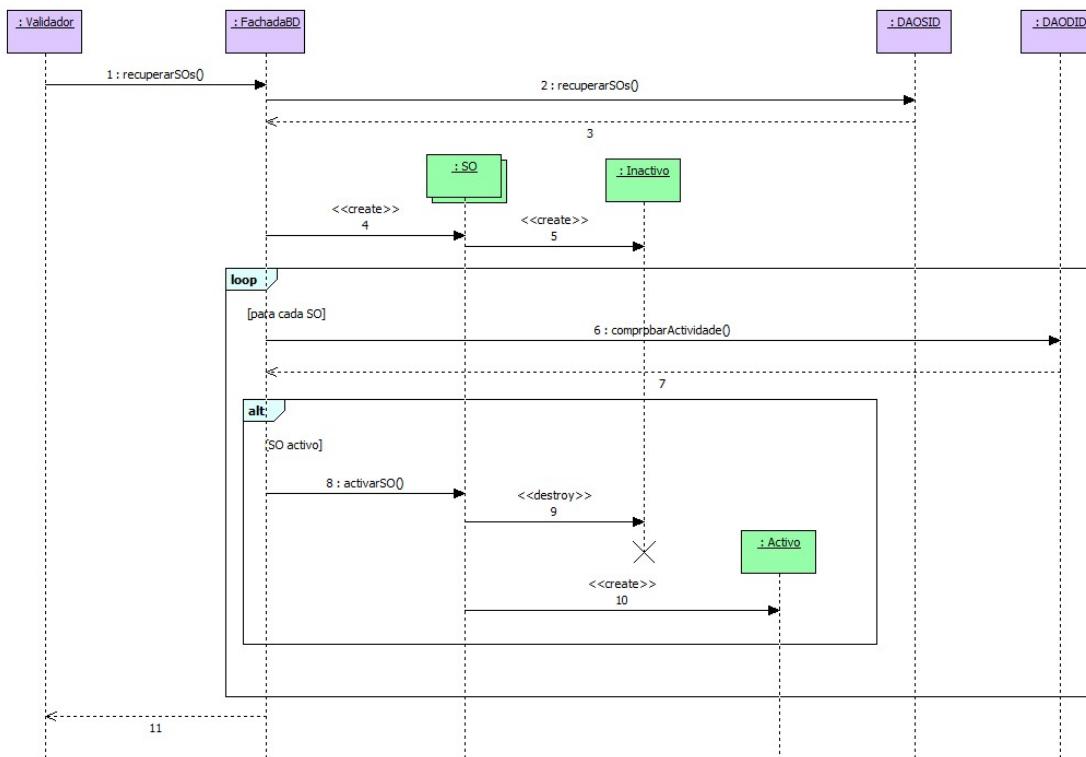


Figura 22: Diagrama de secuencia auxiliar 'Recuperar SO'

10.1.2. Solicitar Información SO

Este segundo diagrama empregarémoslo cando necesitemos solicitar a información dinámica dun determinado SO mediante o intercambio de mensaxes nos formatos *ActuatorML* e *SensorML*. Podemos consultar o diagrama completo na Figura 23.

Unha vez que contamos co SO en estado “Activo”, en varias ocasións desexaremos solicitar a información dinámica, e isto levámolo a cabo a través do envío de mensaxes ao obxecto intelixente correspondente (nos formatos que vimos ao comezo do documento).

O proceso modelámolo do seguinte xeito: solicítase a información e, cos datos recibidos, identifícanse un conxunto de sensores e de actuadores (transductores). Para cada un deles crearemos un obxecto Sensor ou Actuador segundo corresponda, o que representamos cun marco tipo *loop* e con outro *alt*.

Sinalar sobre este diagrama que os obxectos identificados como FabricaTransductores, Sensor e Actuador son as interfaces e as clases abstractas, non as clases concretas. Non entramos en maiores detalles porque é certo que o diagrama tería moita más complexidade e non se recollería ben o seu propósito: habería que crear a fábrica concreta axeitada ao tipo de transductor e, unha vez creada, chamar a

`crearSensor` ou `crearActuador` e logo levar a cabo a creación do sensor/actuador concreto do tipo que corresponde (reflexando así o comportamento completo do patrón *Abstract Factory*), pero isto provocaría que ese diagrama de secuencia involucrase unha grande cantidade de clases e se complicaría moito baixo o noso punto de vista modelalo e entendelo por parte do lector, polo que preferimos deixar a referencia ao que son exclusivamente as interfaces para entender a idea que se busca neste punto.

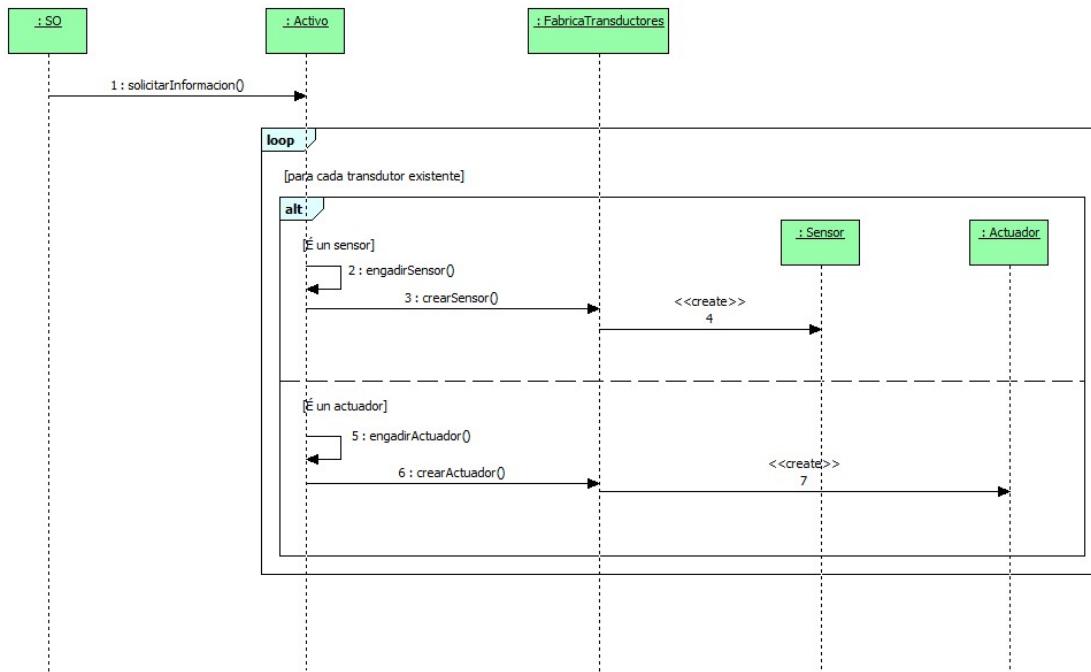


Figura 23: Diagrama de secuencia auxiliar 'Solicitar Información SO'

10.1.3. Recuperar SO DID

Por último, tamén incluimos un diagrama adicional ao que vamos facer referencia cuando necesitemos recuperar a información dinámica do SO da DID. Na figura 24 pódese consultar máis información acerca deste proceso.

Destacar que, pese ás semellanzas con "Solicitar información SO", o diagrama anteriormente exposto recupera a información solicitánto-a aos propios SOs (comunicándose con eles) e non da base de datos, o que ocorre neste novo caso. Fixarse, por exemplo, que agora o que se fai é recibir unha serie de arrays con datos de sensores e outros cos datos dos actuadores, de xeito que a partir de aí se irán creando transdutores segundo corresponda e asociándoos ao SO (concretamente ao estado, pois xa xustificamos previamente que os asociabamos a el porque en función deste pódese ter información dos transdutores ou non).

De novo, buscamos que o diagrama de secuencia tivese sinxela comprensión, búsquedas que nos levou a representar nel somentes a interface da fábrica de transdutores e as clases abstractas Sensor e Actuador, en lugar de amosar paso a paso como se iría aplicando o patrón *Abstract Factory* coas classes concretas. Isto decidímoslo facer así unha

vez más para evitar ter un diagrama demasiado complejo de entender e que non sería representativo.

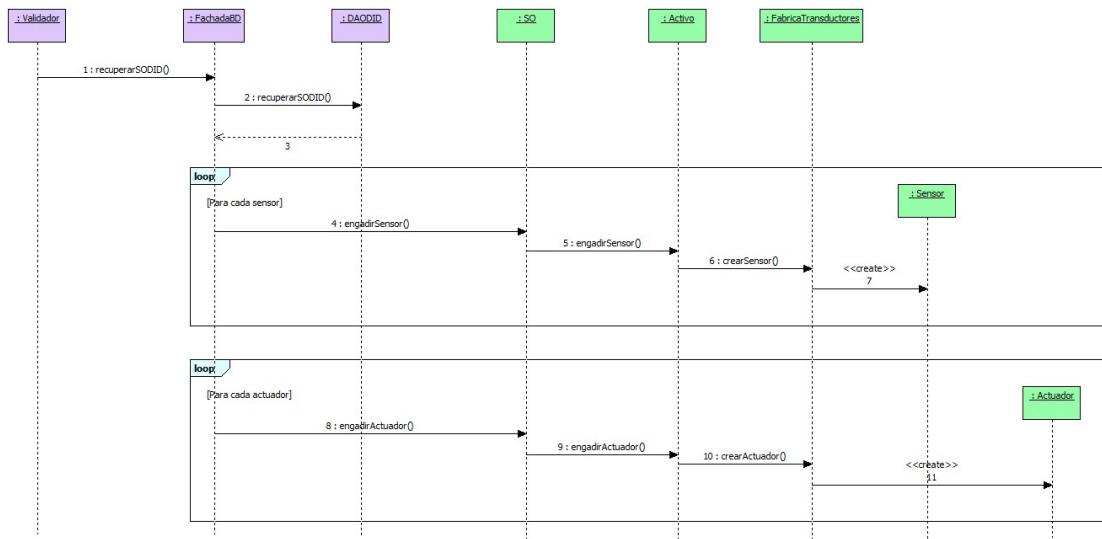


Figura 24: Diagrama de secuencia auxiliar 'Recuperar SO DID'

10.2. Rexistrar SO

Pódese consultar na Figura 25 o diagrama asociado ao caso de uso homónimo. A continuación destacaremos algunas consideracións tomadas no seu deseño e que é preciso ter en conta:

- Na VentaXestion realizase unha primeira comprobación xenérica sobre os campos, como sería o caso de que non se pasen parámetros baleiros. É a que se reflexa mediante o primeiro marco opt
- Hai un segundo marco opt, empregado para indicar o caso no que os datos introducidos sexan incorrectos (en formato) e polo tanto dar a posibilidade de pedilos de novo ao usuario. Neste sentido habería a opción de indicar un bucle, xa que na medida na que a información obtida de novo non sexa correcta reiteraríase a acción, porén consideramos que debe haber máis énfase no feito da alternatividade respecto da situación esperada. Para comprobar o remate correcto ou incorrecto da operación, introduciuse ese parámetro de tipo enteiro que se devolve como resultado.
- Observar como se engadiron os diferentes patróns: cando se crea o SO, directamente faise no estado inactivo (ao ser novo non se puido ter activado), e unha vez creado pásase á base de datos, onde se chamará en primeira instancia ao validador, que logo de verificar o nome e a IP chamará á fachada da base de datos dende a que se recuperará a información necesaria para o rexistro dun SO e se pasará ao DAO da base de datos estática. Esta descripción correspón dese coa aplicación dos patróns *State* e *Proxy*.

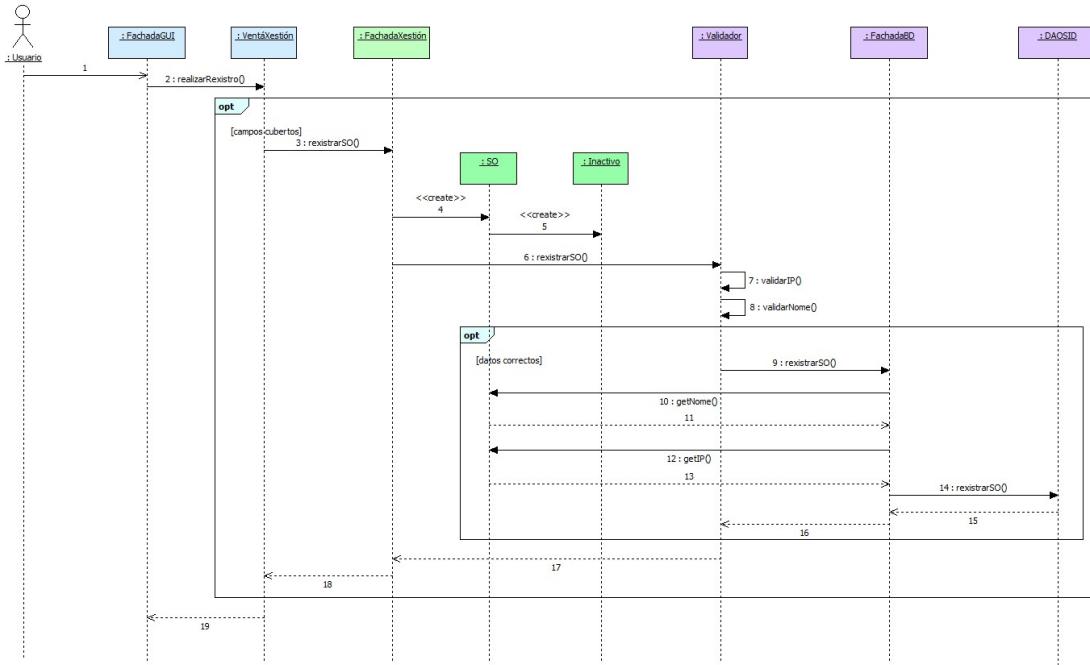


Figura 25: Diagrama de secuencia do caso de uso 'Rexistrar SO'

10.3. Eliminar SO

Entre as suposicións feitas no diagrama representado na Figura 26 está o feito de que consideramos a IP coma a clave primaria do obxecto (é dicir, que na base de datos podremos identificar a información dun obxecto intelixente únicamente pola IP de dito obxecto).

Neste diagrama obsérvase como se fai referencia a '*Recuperar SO*', o diagrama adicional exposto ao comezo desta sección (posto que se necesita coñecer todos os SOs e seleccionar deles os que non se encontran activos, o que non resulta un problema dado que un dos parámetros do método RecuperarSOs é o modo de selección dos obxectos).

Resaltar que neste diagrama se pode comprobar a xustificación do uso da clase FachadaBD. Mediante a operación `eliminarSO`, chámase a cada un dos DAOs, que representan cada unha das bases de datos para levar a cabo a eliminación de toda aquela información dun SO do sistema (estática, dinámica e asociacións a regras difusas).

Así mesmo, resulta preciso indicar que co primeiro sinal que envía o usuario simbolizamos que este accedeu á opción de eliminar un SO na parte de xestión, e co segundo que este selecciona un SO para ser eliminado.

Neste caso para rematar gustaríanos engadir un debate que tivemos durante o modelado do sistema. Aquí poderíamos prescindir de empregar o SO como obxecto dado que estamos almacenando, manexando é movendo certa información que non se chega a empregar, podendo ser sustituido este polo nome e pola IP separados, mesmo poderíamos usar únicamente a IP que é o necesario para poder identificar unívocamente o SO na base de datos. Porén, esta alternativa supón unha ruptura do estándar que seguimos no resto da funcionalidade manexada polo usuario como a activación e desactivación, creación ou configuración dos SOs, polo que decidimos manter una idea xeral común e evitar saírnos

dentro do posíbel dela.

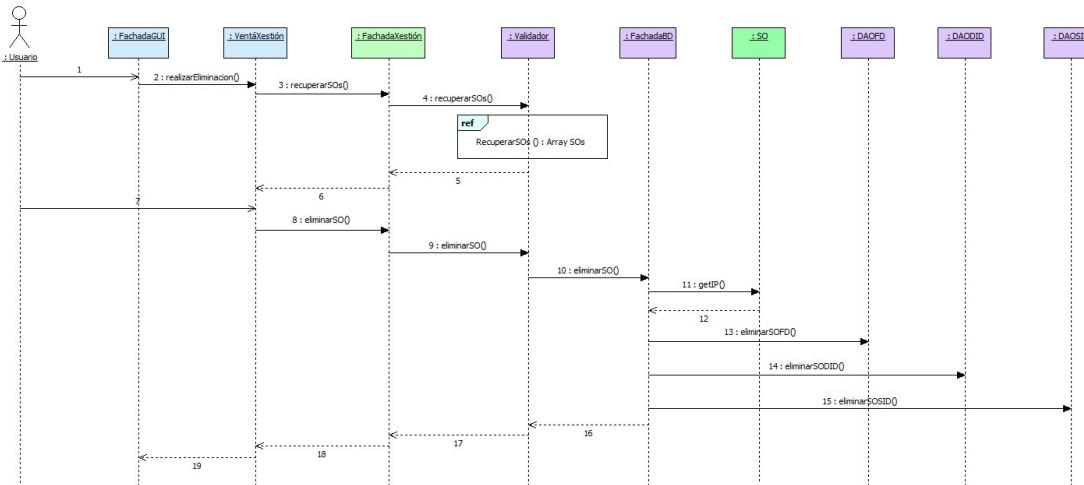


Figura 26: Diagrama de secuencia do caso de uso 'Eliminar SO'

10.4. Seleccionar SO

Para representar este caso de uso, cremos que o máis conveniente é o emprego de dous diagramas de secuencia: “Activar SO” e “Desactivar SO”, dado que entendemos que se atopen o suficientemente diferenciados (e así podemos facilitar a súa comprensión) e porque se trata de dúas alternativas que consideramos realmente preciso distinguir e representar por separado.

10.4.1. Activar SO

No diagrama da Figura 27, no que se representa o proceso de activación dun SO, debemos destacar as seguintes decisións que fomos tomando ao longo do seu desenvolvemento:

- Neste diagrama temos dous accesos á parte da base de datos. O primeiro deles, a través da referencia ao diagrama “Recuperar SO”, para recoller todos os SOs da base de datos, amosalos por pantalla, e poder seleccionar o que se quere activar (se está desactivado, inda que isto supoñemos que se comproba antes de dar a opción a activalo ao usuario).
- Ademais, faise a referencia ao diagrama “Solicitar info SO”, no que se representa a solicitude de datos sobre os sensores e actuadores do SO. Tras esa recuperación de información é cando ven o segundo acceso á base de datos, neste caso á DID.
- Respecto da información dinámica, esta recuperárase sobre os SOs directamente e será na base de datos onde se obteña para ser pasada por partes aos DAOs.
- Hai que ter en conta tamén que na fachada da base de datos representamos que se recuperan os sensores e os actuadores, e poderíamos entrar en maior rango de detalle indicando que nesa recuperación se van obtendo os diferentes datos de cada un deses transdutores (xa mencionamos antes que operacións que recuperan datos de obxectos asociados ao SO non devolven o obxecto directamente, senón

que nos proporcionan conxuntos de datos cos tipos básicos), pero cremos que con iso a complexidade do diagrama aumentaría con algo que se pode supoñer que se realiza antes de obter eses datos, polo que non se incluiu.

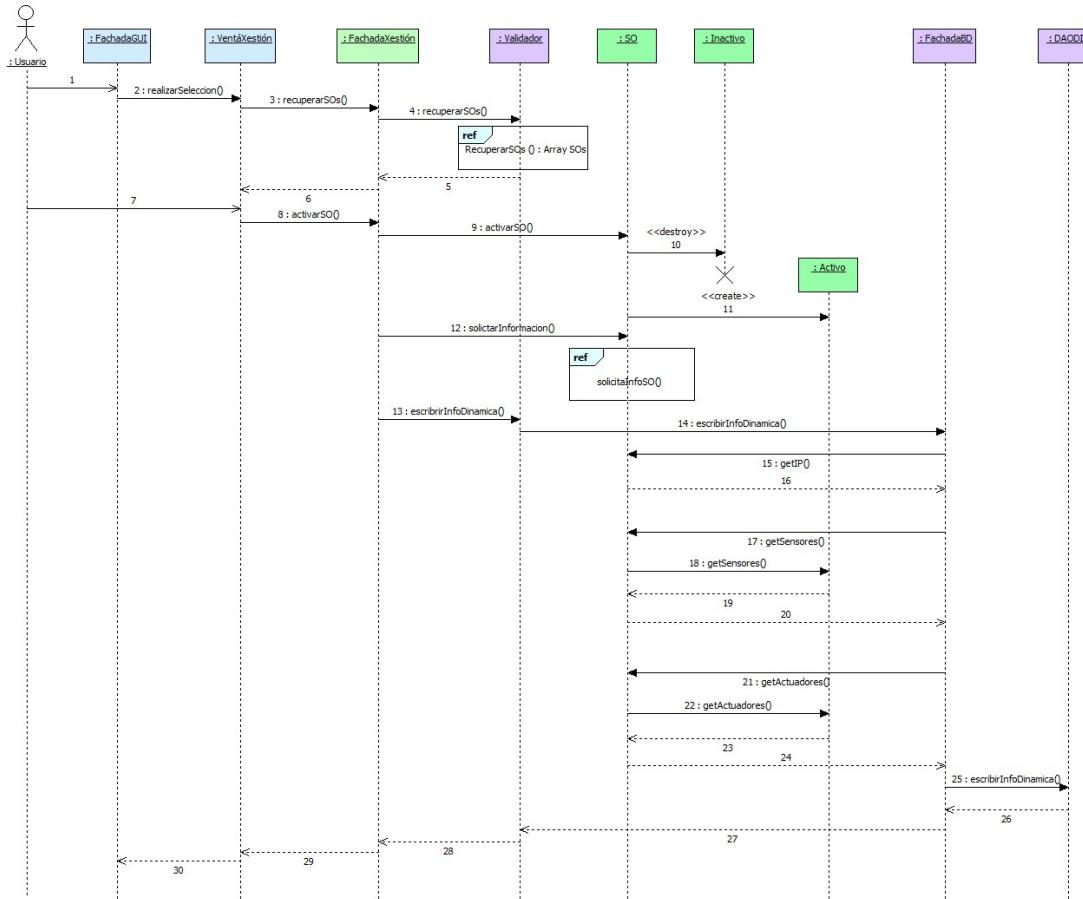


Figura 27: Diagrama de secuencia 'Activar SO' dentro do caso de uso 'Seleccionar SO'

10.4.2. Desactivar SO

O único que convén salientar respecto do diagrama da Figura 28, é que se almacena o feito de que o SO deixe de estar activo na DID.

Ademais, cabe sinalar que se cambia o estado do propio obxecto SO de activo a inactivo, para reflexar tamén sobre este que se desactiva (chamando á operación axeitada). En principio, non lle damos máis uso ao SO, inda que podería ser que futuros cambios provoquen a realización dalgúnha operación máis, parecéndonos axeitado por tanto reflectilo.

Polo demais, mencionar que existe algunha similitude co caso anterior no comezo (ao recuperarse tamén os datos dos SOs), e ideas que xa se foron comentando noutros diagramas coma a aplicación de diferentes patróns ou a utilidade da fachada da base de datos, polo que non se redundará nelas.

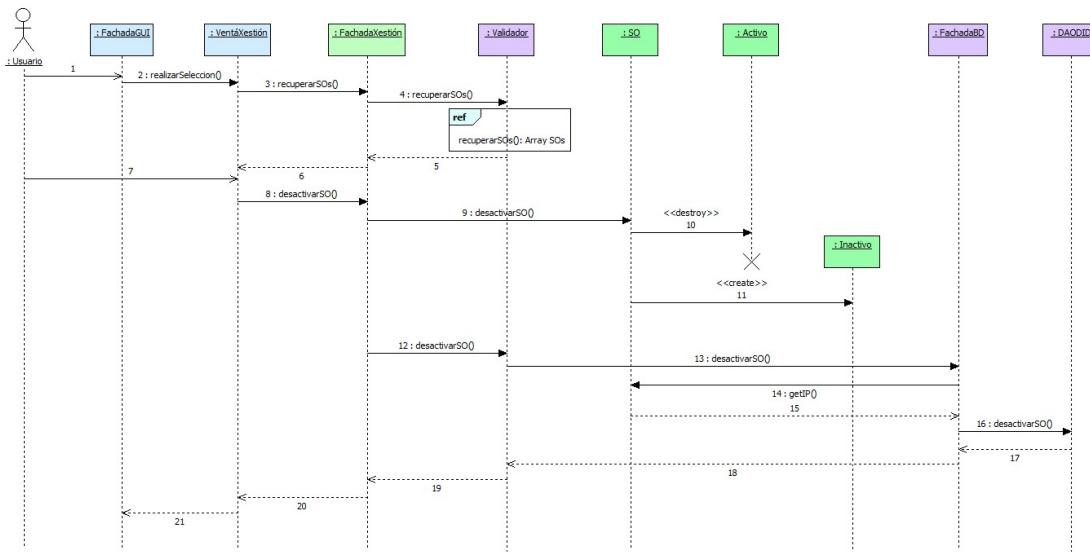


Figura 28: Diagrama de secuencia 'Desactivar SO' dentro do caso de uso 'Seleccionar SO'

10.5. Configurar SO

Para modelar a configuración dos SO empréganse tres diagramas de secuencia, un por cada extensión do caso de uso correspondente: "Configurar Formulario", "Configurar Editor" e "Configurar Avanzado", dado que albergan suficiente complexidade como para non ser viable unha representación en conxunto, coma fixemos no diagrama de actividade.

Unha vez máis, e na liña de evitar unha sobrecarga visual, referirémonos sistemáticamente aos diagramas adicionais das Figuras 22, 23 e 24, para tarefas recorrentes coma a obtención de información dos SOs.

10.5.1. Configurar SO por Formulario

Este diagrama de secuencia recollido na Figura 29 precisa aclarar que valoramos extraer a interacción coa clase FachadaXestión, dado que só a empregamos para recuperar os SOs e poderíamos facer o equivalente na FachadaConfiguración ao ter a referencia á Base de Datos, pero dado que ambos estarían assumindo unha mesma responsabilidade respecto da VentaConfiguración entendemos que debemos ser consistentes e por tanto só os recuperara a FachadaXestión (ademas de que lle damos valor a esta clase como a fachada da parte que se centra na xestión dos SOs, o que se relaciona con recuperar a súa información).

Incorporamos algunas anotacións sobre a Figura 29 a continuación:

- Os datos que poderían implicar unha maior complexidade de xestión coma tokens ou as árbores de análise decidimos xestionalos coma *strings* ou *arrays de strings* atendendo a cada situación. Non queremos entrar, para este tipo de detalles, en cuestiós moi concretas sobre implementación (dentro da clase que manexa esta información saberase como xestionala).

- Realizáronse dous accesos á base de datos: o primeiro faise para recuperar todos os SOs e poder amosarlle ao usuario os SOs activos para poder ser configurados, e un segundo acceso lévase a cabo xusto antes de amosar o formulario para poder crealo en función dos datos dinámicos do SO. Logo, poderíase precisar outro acceso á base de datos antes do planificador, pero como decidimos pasar a referencia ao SO por diferentes clases e xa se fai un acceso á DID previamente, non é preciso volver a facelo. Deste xeito reducimos a repetición do código e o tempo de execución, inda que haxa algunha dependencia máis.
- Cando se xeran as regras difusas a partir da árbore de análise, modelamos cun bucle a creación das mencionadas regras, dado que o que se fai é chamar a un método do SO para engadir unha regra difusa, que á súa vez accederá á clase que representa o estado activo onde realmente se levará a cabo o engadido da mencionada regra. Como isto se fai unha a unha, segundo se van xerando, non nos pareceu coherente representar as regras neste caso coma un conxunto de obxectos, para reflexar ese proceso de creación.
- Tamén ocorre algo semellante ao que nos pasou previamente ao recuperar sensores e actuadores na fachada da base de datos. Recupéranse as regras difusas para ser almacenadas, xa descompostas en partes para non ter que acceder ao obxecto que representa (e evitar dependencias), pero poderíamos precisar tamén que para cada regra asociada ao SO se vai recuperando as súas condicións e actuacións, o que engadiría inda máis complexidade a un diagrama que xa resulta bastante longo. Así mantemos de novo esa idea de non amosar como se recuperan exactamente os datos de cada regra dentro da operación correspondente.

Mencionar, para rematar coa descripción deste diagrama, que aquí se accede ás clases `VerificadorConcreto` e `XeradorConcreto`, clases que apareceron pola aplicación do patrón *Strategy*.

10.5.2. Configurar SO por Editor

Este diagrama que se pode ver na Figura 30, resulta moi semellante ao anterior (Figura 29, ainda que non se xera a sintaxe SCL, pois xa é introducida directamente polo usuario, nin se fai a recuperación inicial de datos do SO porque non hai que elaborar ningún formulario).

Deste xeito, á DID accederase xusto antes de chamar ao configurador, para ter toda a información recuperada antes de acceder a esa clase (e comenzar coa verificación SCL e a xeración de regras) de maneira que así o Xerador Fuzzy dispoña de todo o necesario para poder xerar a lóxica difusa sen ningún tipo de problema.

Polo demais, o resto de cuestións son xa semellantes ao caso de configurar por formulario, polo que non redundaremos nelas.

10.5.3. Configurar SO no modo avanzado

Neste diagrama, que está recollido na Figura 31 é salientable que desparece unha parte da complexidade reflexada nos anteriores, dado que agora non hai que xerar as regras

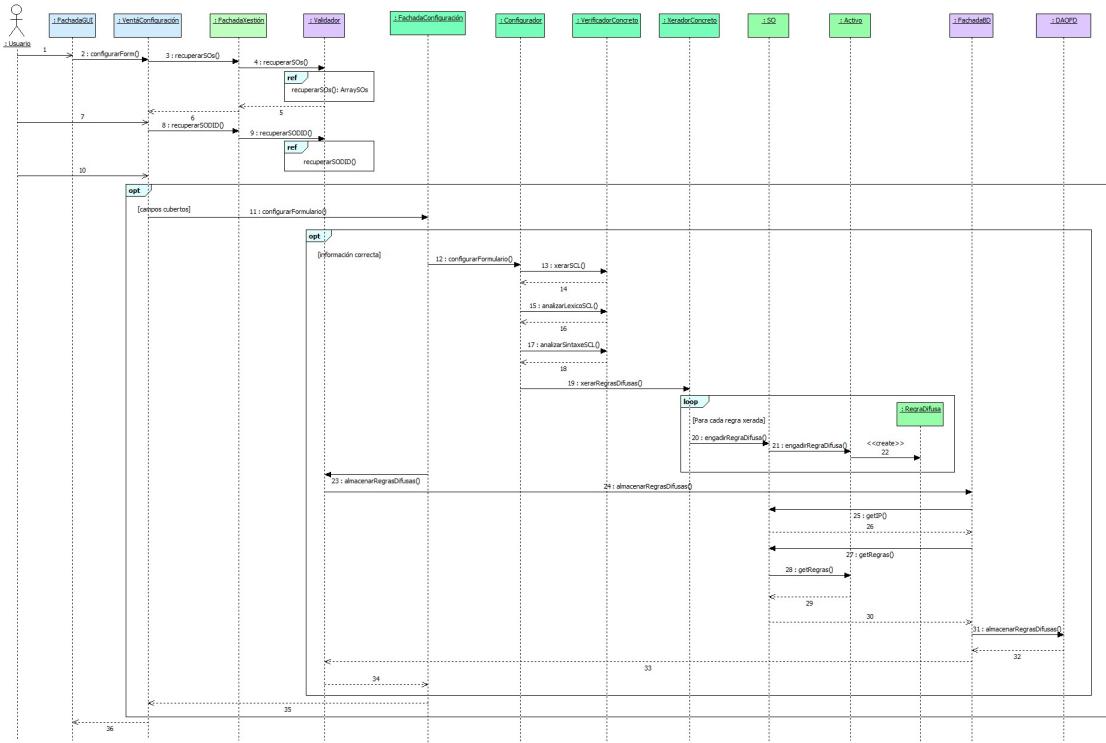


Figura 29: Diagrama de secuencia do caso de uso 'Configurar SO por Formulario'

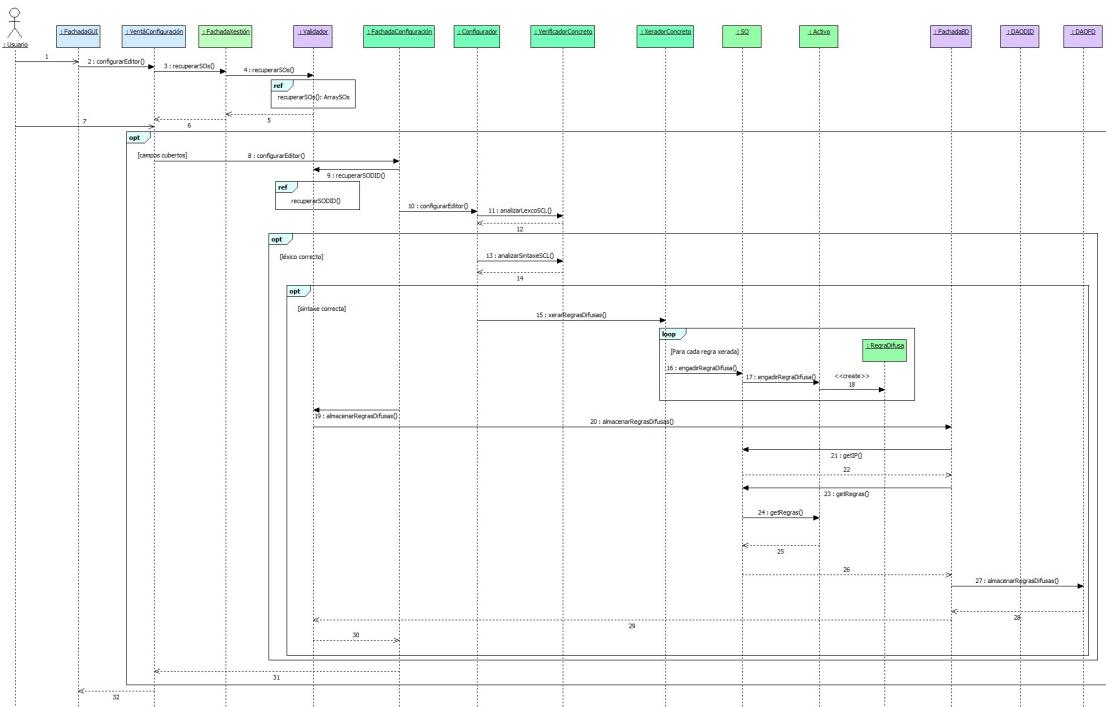


Figura 30: Diagrama de secuencia do caso de uso 'Configurar SO por Editor'

difusas, senón que as facilita o usuario. Polo tanto, tampouco será preciso acceder á DID para recuperar a información dinámica.

Inda así, o que si se introduciu é unha comprobación da corrección das regras, para o que temos unha operación no XeradorConcreto: `verificarLoxicaDifusa`. Unha vez que a lóxica difusa se verificou, poderase acceder á base de datos como en casos anteriores (é dicir, á FD), e almacenar as regras que introducirá o usuario nela.

Hai que resaltar tamén que agora, na ventá de configuración, engádense directamente ao SO seleccionado as regras introducidas polo usuario. De aí que, tras validar que se introducirá algúna regra, haxa un frame tipo `loop` no que se crean regras difusas e se engaden ao correspondente SO.

Para rematar, indicar que tamén hai algunha similitude cos diagramas dos outros casos extendidos de 'Configurar SO', pero neste último caso son menos.

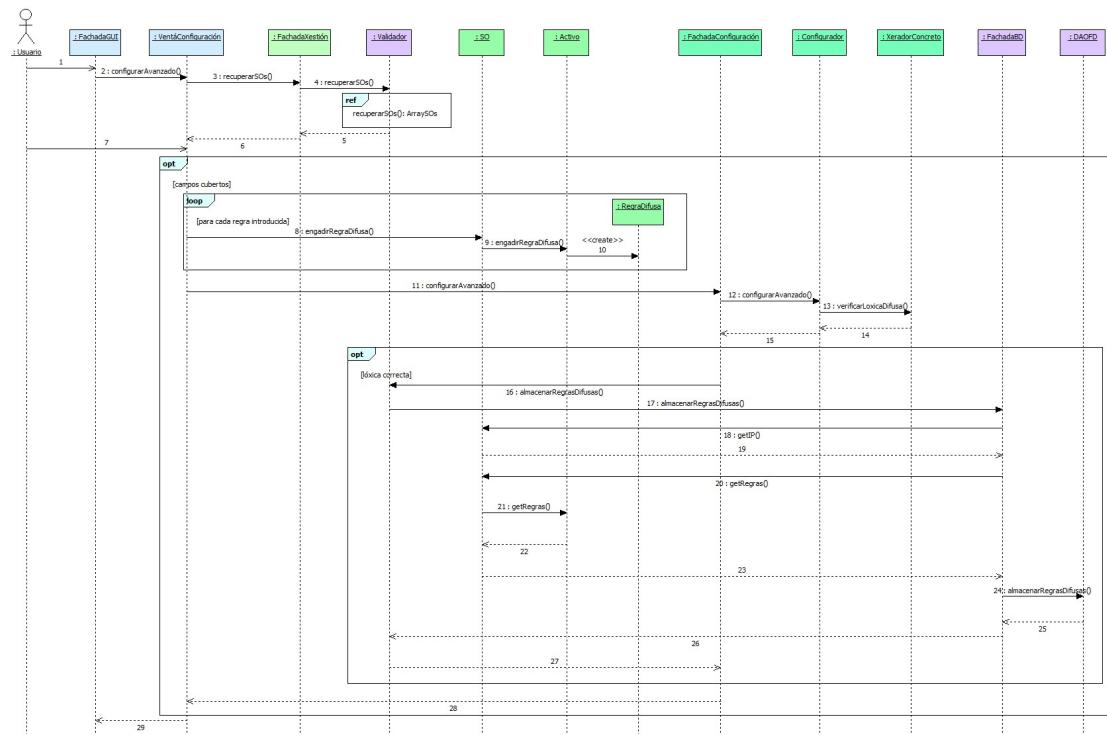


Figura 31: Diagrama de secuencia do caso de uso 'Configurar SO no modo avanzado'

10.6. Planificar actuación

Representado na Figura 32, esta planificación expone cun bucle para ser realizada sobre todos os SOs activos. Este é, ademais, o único diagrama no que se produce a participación do actor que representa a alarma.

Empézase recuperando todos os SOs, identifícanse aqueles que están activos, e vanse facendo as mismas accións para cada un deles: solicítase a información ao SO, gárdase na DID, accédese á FD para recuperar a lóxica difusa (que vemos que se vai engadindo ao SO correspondente dende a fachada da base de datos) e planifícase o SO xerando, se é preciso, comandos de actuación.

Neste sentido, engadimos un marco opt para enviar comandos de actuación ao SO en caso de que haxa que facelo, de maneira que se conseguem reflexar así as dúas alternativas tamén recollidas no caso de uso 'Planificar Actuación'.

Tampouco enviamos mensaxe de confirmación nin se produce en ningún momento a participación de ningunha compoñente da parte de interface gráfica posto que no caso de uso homónimo non existe participación algunha por parte do actor que representa ao usuario.

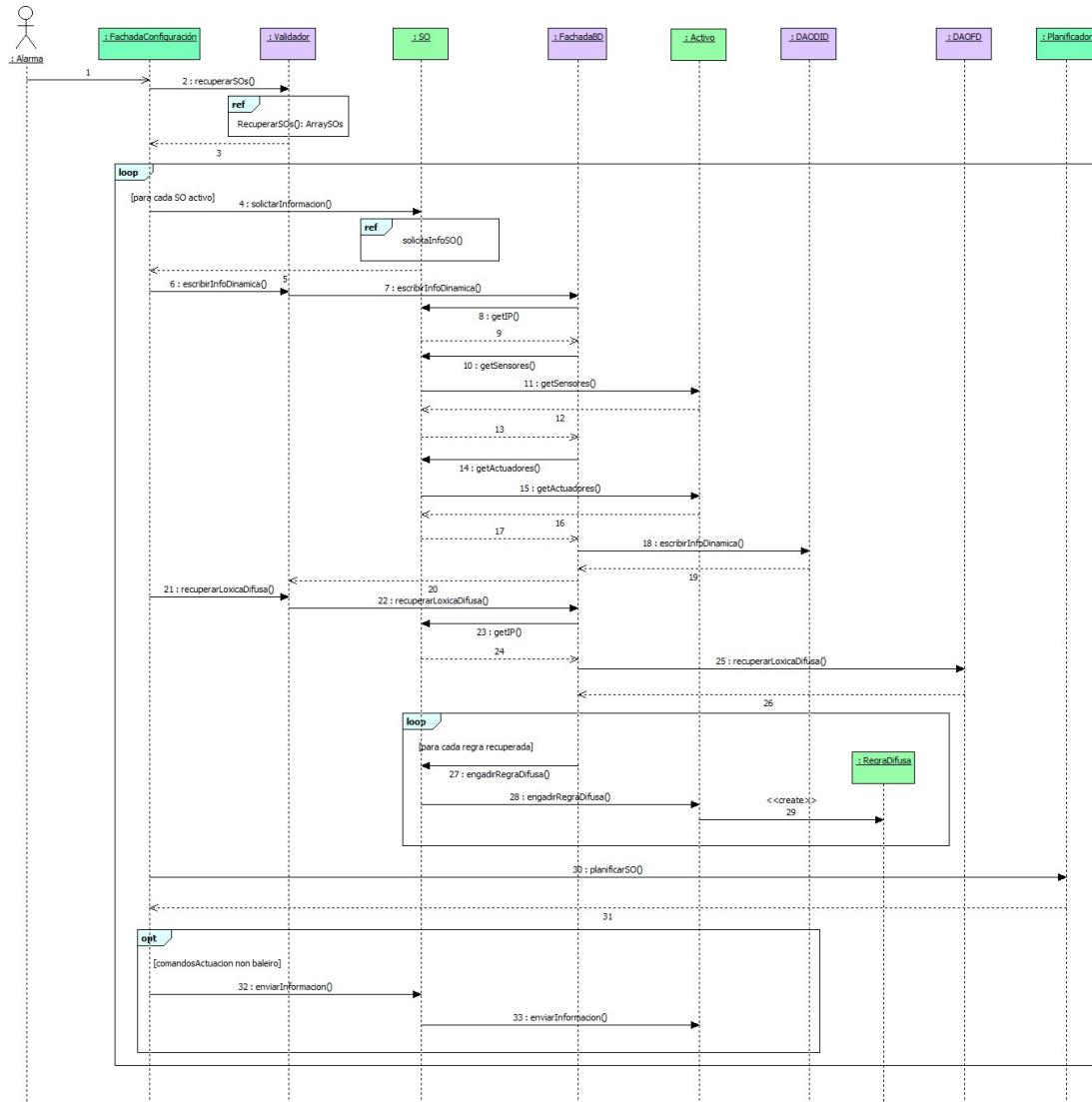


Figura 32: Diagrama de secuencia do caso de uso 'Planificar Actuación'

10.7. Visualizar información

Dentro do diagrama recollido na Figura 33, cómpre salientar o bucle de petición de información aos SO e o almacenamento na base de datos da mesma (motivado pola recepción de información), o que facemos de forma moi semellante á abordada en diagramas anteriores, polo que non redundaremos na súa explicación.

A maiores, mencionar que, unha vez que se devolve toda a información dos SOs sobre os que hai que visualizala (os SOs activos), na ventá correspondente terase que procesar o SO debidamente para poder amosar os resultados que se queiran. Ademais, ao ter os SOs dispoñibles nesa clase, pensamos que podería ser unha boa oportunidade para que o usuario personalice a forma de utilización dos datos, cuestión na que non profundizaremos moito máis porque non nos parece relevante.

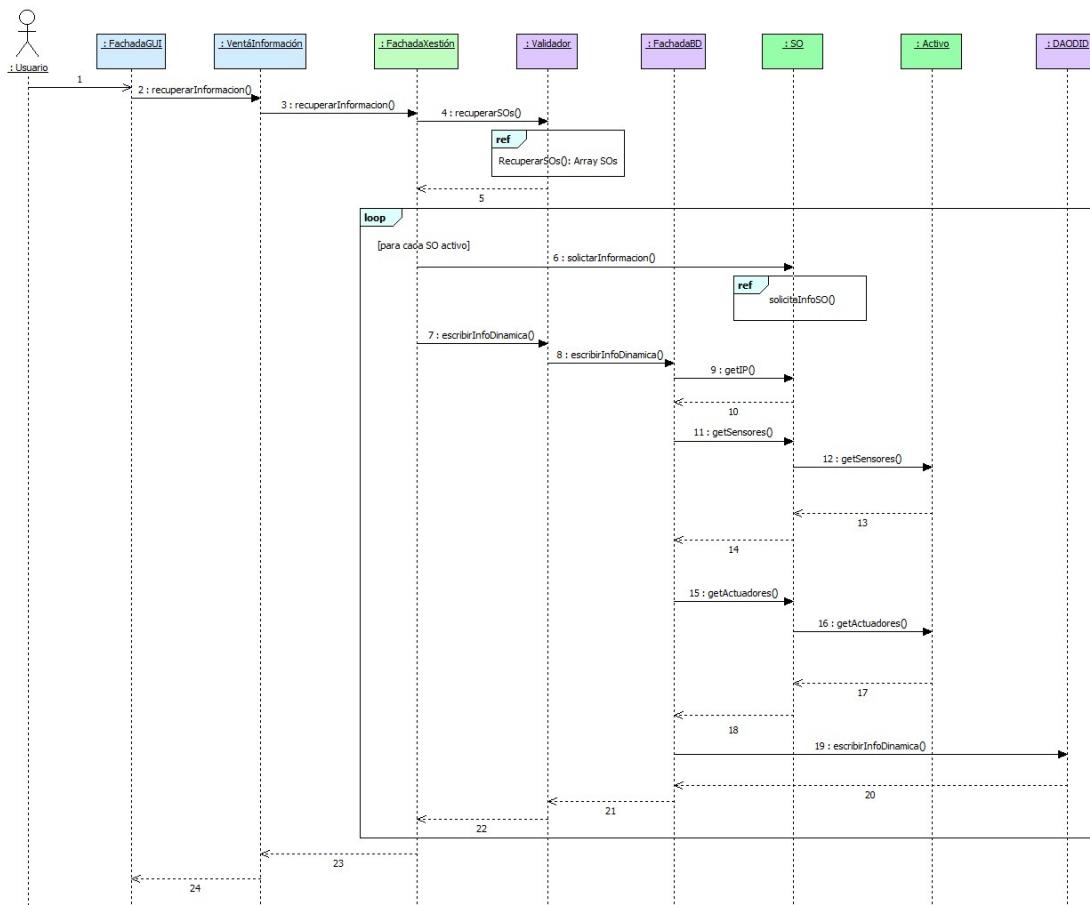


Figura 33: Diagrama de secuencia do caso de uso 'Visualizar información'

11. Conclusións

Rematamos xa o noso informe cunha pequena reflexión final sobre o traballo presentado ao longo do mesmo, a modo de conclusións.

Neste proxecto, tentamos dar o noso enfoque ao deseño dun Sistema de Configuración dun Fogar Intelixente, deseño que fixemos a través de diferentes etapas nas que se foron desenvolvendo diferentes actividades e, ao mesmo tempo, volvendo sobre as xa feitas para mantelas actualizalas e corrixir aquelas cuestións que nos deixaron de convencer ou nos parecían finalmente incorrectas.

O desenvolvemento do proxecto segue dalgunha maneira a estrutura do documento: empezouse intentando comprender o problema e adaptalo ao noso caso con algunha cuestión que non quedara clara na descripción inicial, logo pasouse a tratar os casos de uso: elaborouse un diagrama de casos de uso, describiríronse en detalle cada un deles, desenvolvéronse diagramas de actividade para representalos e consideráronse diferentes casos de proba. Neste segundo paso tamén se introduciu o desenvolvemento da interface gráfica da aplicación dende a cal xestionar o sistema.

O seguinte paso consistiu en desenvolver un diagrama de componentes no que representamos diferentes subsistemas e componentes do SCFI, xunto coas interfaces e as relacións entre eles. Outro paso centrouse en comezar a desenvolver os diagramas de secuencia, tendo en conta xa os patróns DAO, MVC e Fachada, e ao mesmo tempo ir construindo o diagrama de clases correspondente.

No último paso levouse a cabo unha labor de realimentación por todas as partes do deseño para facelo coherente e, ao mesmo tempo, introducíronse novos patróns de deseño para completalo: *Strategy*, *Abstract Factory*, *State* e *Proxy*. A aplicación dos patróns levou tamén a modificar o diagrama de clases e todos os de secuencia, incluso á introdución dalgún máis dos que tiñamos propostos inicialmente (dos que están no grupo de 'Diagramas Adicionais').

Con isto describimos, máis ou menos, as diferentes etapas polas que pasou o noso deseño. Cabe sinalar que, como en todo deseño dun sistema, estas alternativas que presentamos son resultado dun consenso entre todos os membros do grupo, e somos conscientes de que habería moitas outras propostas que se poderían presentar a este problema. De todos os xeitos, o resultado final pareceunos a todos o máis axeitado.

Referencias

- [1] IEEE, “SITE: The Simple Internet of Things Enabler for Smart Homes“ [en liña].
Dispoñible en: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7831376> [data de consulta 20/02/2020].
- [2] Microsoft, “Model-View-Controller“ [en liña].
Dispoñible en: <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649643> [data de consulta 17/03/2020].
- [3] Best Practice Software Engineering, “Facade Pattern“ [en liña].
Dispoñible en: <http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/facade.html> [data de consulta 17/03/2020].
- [4] Oracle, “Core J2EE Patterns - Data Access Object“ [en liña].
Dispoñible en: <https://www.oracle.com/technetwork/java/dataaccessobject-138824.html> [data de consulta 17/03/2020].
- [5] Visual Paradigm, “What is Component Diagram“ [en liña].
Dispoñible en: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/> [data de consulta 22/03/2020].
- [6] IBM, “The sequence diagram“ [en liña].
Dispoñible en:
<https://developer.ibm.com/articles/the-sequence-diagram/> [data de consulta 27/03/2020].
- [7] IBM, “Tips for writing good use cases.“ [en liña].
Dispoñible en: <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RW14023-USEN-00.pdf> [data de consulta 17/03/2020].
- [8] Fowler, M. (2004). UML distilled: A brief guide to the standard object modeling language. Boston: Addison-Wesley. [data de consulta 02/03/2020].
- [9] Refactoring GURU, “State.“ [en liña].
Dispoñible en: <https://refactoring.guru/design-patterns/state> [data de consulta 23/05/2020].
- [10] Refactoring GURU, “Strategy.“ [en liña].
Dispoñible en:
<https://refactoring.guru/design-patterns/strategy> [data de consulta 23/05/2020].
- [11] Refactoring GURU, “Abstract Factory.“ [en liña].
Dispoñible en: <https://refactoring.guru/design-patterns/abstract-factory> [data de consulta 23/05/2020].
- [12] Refactoring GURU, “Proxy.“ [en liña].
Dispoñible en: <https://refactoring.guru/design-patterns/proxy> [data de consulta 24/05/2020].