**ARQUITECTURA DEL MECANISMO DE WORKFLOWS DEL PATRONAT MUNICIPAL DE L’HABITATGE DE BARCELONA**

**Introducción**

Los programas conocidos como workflows son unos mecanismos informáticos que permiten la orquestación de distintas piezas de software de una Corporación, o cuando dicho software necesita interactuar con el exterior (por ejemplo, para incorporar un dato de Hacienda en un expediente).

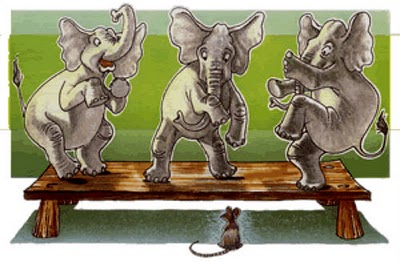
Los workflows son especialmente útiles cuando:

* Se trata de procesos complejos, con muchos pasos de realización, en contraposición a casos simples (calcular el ajut de un lloger)
* Estos procesos tienden a interactuar unos con otros (a diferencia de un proceso vertical que solo sabe de sí mismo, como emitir una factura)
* Se prolongan en el tiempo, a veces durante semanas o meses (a diferencia de enviar un email, por ejemplo)

Las ventajas de los wf

* El mecanismo fundamental es la capacidad para persistir –guardar- un proceso de larga duración (persistencia)
* Tiene mecanismos de Tracking, scheduling, entre otros.

**Una hipérbole: el ratón y los elefantes**



Es un cuento popular, creo, que los elefantes temen a los ratones ¿verdad que es una imagen ridícula, que un ser tan menudo haga ‘bailar’ al mayor de los mamíferos terrestres? Pues bien, algo así nos puede ocurrir sino efectuamos cierta abstracción necesaria. Hemos de huir de planificar una solución concreta en nuestra organización para ser capaces de concebir el entero edificio. Si queremos solucionar un problema dado, puede que –seguro que- luego nos encontraremos que nuestra solución hay que reinventarla una y otra vez, pues solo se pensó con un enfoque particular.

El elefante que moverá informáticamente nuestra organización es la arquitectura de la misma, sus cimientos, sus fundamentos, no el color de las puertas, ni el número de enchufes de una habitación. El ratón no debe orquestar al elefante.

**Arquitectura de los Workflows**

Diferenciaremos arbitrariamente dos piezas de software básicas que utilizaremos:

* Los aplicativos tradicionales
* Los workflows.

Las aplicaciones tradicionales consisten en un determinado programa atómico, encapsulado, que realiza una tarea específica, discreta y concreta. En el PMHB llevamos años diseñando dichos programas derivándolos del framework CSLA.

Los workflows, por otro lado, son un software cuya naturaleza es responder no tanto a qué se hace, sino a cómo y cuándo. Dicha pieza de software, en lugar de caracterizarse por su encapsulación es frecuente que interactúe con otros elementos, esperando sus resultados o comunicándole los mismos, llegando dichas comunicaciones a producirse incluso fuera de la organización.

**Elementos de Arquitectura propuesta para el PMHB**

* Aplicación: explicado arriba
* Workflow: explicado arriba
* Manager: el coordinador de workflows
* Token: el elemento testigo que usará el manager.

Lo anterior se plasma en el siguiente gráfico:

**Workflow**

**Aplicación**



**Wf Manager**

**Elementos fundamentales de la Arquitectura en el PMHB**

**Token**

**La aplicación.**

Nuestras aplicaciones realizan tareas concretas, como todo programa informático. Un paradigma del desarrollo de software de calidad que hemos seguido hasta ahora consiste en la programación orientada a objetos (OOP en inglés) del autor, mundialmente conocido, Rockford Lhotka: CSLA, (para Component Based Scalable Logical Application).

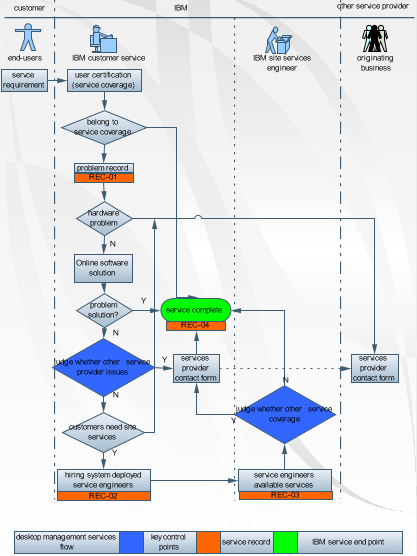
Este enfoque de programación dicta que un programa debe ser diferente a una “navaja suiza” multiuso. Es decir, debe saber hacer bien una parte específica de un proceso, y no depender del mundo exterior a fin de hacer bien eso para lo que fue concebido.

Sin embargo, en toda organización moderna hay que interconectar e interrelacionar los programas. La facturación es la base de la contabilidad; el programa de nóminas se relaciona con el programa de ventas; y un largo etcétera.

Y aquí todo se complica una vez más. Programas creados en diferentes épocas, por diferentes empresas, por distintas personas, con paradigmas contrapuestos, ahora tienen que convivir, relacionarse… y hacerlo bien. Y ahí aun el mejor de los programas nos muestra su debilidad: que sabe muy bien de lo suyo, pero le cuesta colaborar con el mundo exterior. Cuesta mucho tocar una pieza sin desmontar otra. Y no pocas veces se producen terroríficas reacciones en cadena.

**El workflow**

Veamos el gráfico que representa un flujo de trabajo cualquiera:



Cuando se mira el dibujo de un flujo de trabajo como el del gráfico anterior no puede uno menos que dejarse encandilar un poco. Poseen cierta belleza. Merecida belleza.

Un workflow representa una realidad que funciona (más o menos bien) en una organización, en una estructura, en un paradigma. Se resuelve un problema, se consigue un objetivo, distintas personas, en base al flowchart, deben entenderse.

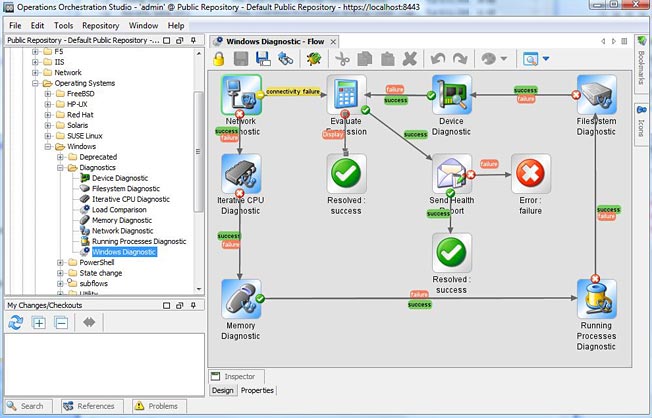
Un workflow es la Inteligencia que hace que se transforme una masa de personas en un equipo de trabajo, la fuerza que hace que algo o alguien efectúe cierto cometido, en determinado momento, durante un periodo dado. Y nos embelesa dicha Inteligencia.

Toda Organización que se precie debe contar con una serie de procesos orquestados y coordinados a fin de triunfar. El PMHB ya cuenta con muchos flujos así, bien informatizados o no.

Siendo que el desarrollo de software tiene que ver con la transformación de las realidades de la Sociedad en Información utilizada por sistemas electrónicos (paráfrasis para programas que hacen cosas útiles, como generar albaranes en una Empresa, por ejemplo) ¿Qué ocurre cuando tienen que interactuar muchos programas distintos entre sí? ¿Se complican las cosas como en la vida real? Cierto que sí.

Ahí entra en el cuadro una entidad del mundo del software: el workflow. No el workflow como un dibujo hecho con una herramienta de pintar cuadros y líneas, sino el workflow como un software polivalente que es capaz de encargarse de la orquestación de los aplicativos de una organización.

**La orquestación.**



El gráfico superior corresponde con una captura del software Operation Orchestration Studio, de HP. Solo con fijarnos en los iconos de la derecha captamos fácilmente el espíritu de la orquestación: hacer que un grupo heterogéneo de entidades se digan cosas útiles las unas a las otras a fin de conseguir un trabajo.

La razón de ser número uno del workflow es orquestar. Sí, como en una orquesta de música, que piezas dispares produzcan armónicos entre sí. Será responsabilidad del software del workflow controlar cuándo tiene que tocar un instrumento, esperar que calle otro. El orden de los trabajos, la duración de los mismos. Y no menos importante, qué se tienen que comunicar las aplicaciones entre sí.

**Windows Workflow Foundation.**

Microsoft ha desarrollado un framework para crear workflows: Windows Workflow Foundation. Este framework es tan versátil que los mismos ingenieros de Microsoft lo han utilizado en su última versión recién publicada, la cuatro, para producir el software de orquestación de la multinacional: Microsoft BizTalk.

Ese mismo software nos servirá para hacer algo más humilde, pero no menos eficaz: la Arquitectura de Orquestación de Programas del Patronat. (AOP)

**El manager.**

* Ya tenemos el programa informático, la aplicación, que realiza una tarea dada.
* El workflow coordina dicha tarea y la relaciona con otros programas

Pero, ¿qué ocurre cuando hay que coordinar no solo programas entre sí, sino workflows entre sí? Ese es el cometido del Manager. Entre sus atribuciones estarán:

* Ofrecer un punto de activación a los usuarios/aplicaciones para ejecutar workflows
* Saber qué workflows están disponibles en un repositorio de los mismos
* Supervisar los workflows que se ejecutan: si están en memoria, persistidos en un almacén, cancelados, con errores o terminados. En qué momento deben lanzarse o saber cuándo deben terminar.
* Si es necesario, invocar otros workflows, pasarles los datos que precisen, detener el workflow que invoca y esperar la respuesta del invocado.

**El token (testigo)**



Todos hemos visto en las carreras de relevos que un corredor debe pasar a otro el testigo (token) a fin de que este siga. De no hacerse adecuadamente el corredor es descalificado.

De nuevo, podemos tomar como ejemplo un paradigma que funcionó en las redes token-ring. Famosas tiempo atrás, la más destacable característica de dichas topologías es que las estaciones que se conectaban se reconocían entre sí a través del token.

Aunque explicado exhaustivamente más adelante, el token de nuestro framework será la pieza de información que usará el manager para saber con quién quiere comunicarse un workflow concreto.

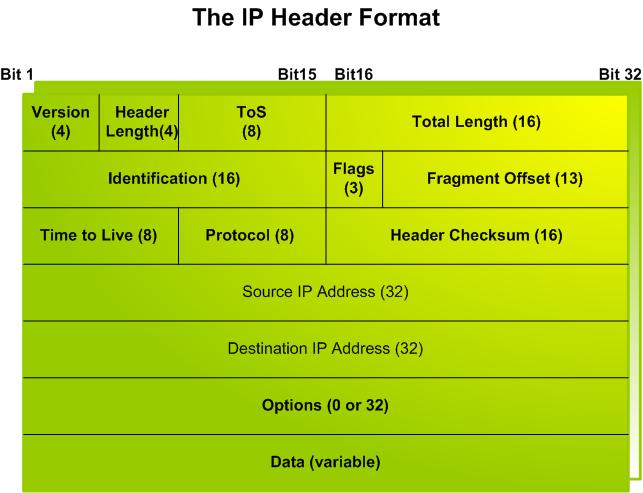
La inconmensurable ventaja del uso de token entre workflows es conseguir, digamos esto entre signos de admiración: ¡la encapsulación de los workflows! Si antes decíamos que los workflows se comunican muchas veces entre sí, eso presupondría que un workflow debe conocer de su vecino. Hasta cierto grado eso los haría débiles, tan débiles como que el cambio de un vecino los convirtiera en inservibles. Pero ¿qué hay si encapsulamos todo el conocimiento del entorno en un elemento, el token? De este modo, los workflows serán ajenos del mundo exterior, delegando dicho conocimiento a una parte concreta de sí mismos. Desarrollemos esto un poco más.

**Cómo funcionará todo: la historia del velcro y el cardo.**

A diferencia del cuento del ratón y los elefantes, se cuenta que en 1948, tras venir de un paseo por el campo con su perro, el ingeniero suizo George de Mestral descubrió lo complicado que resultaba desenganchar de sus pantalones y del pelo de su perro los frutos del cardo. Tras su estudio inventó el velcro.

Este no es el único ejemplo, ni muchísimo menos, de cómo, pensando en cierto elemento que funciona muy bien para determinada actividad se puede extrapolar a otra totalmente distinta. Para nuestra arquitectura haremos algo similar. No reinventaremos la rueda, sino más bien nos vamos a fijar en un proceso que lleva ya décadas funcionando de maravilla. Lo inventaron los ingenieros militares norteamericanos durante la Guerra Fría, y respondía ya en su concepción a la necesidad de comunicar datos entre programas informáticos muy dispares y de organizaciones distintas, como estamentos militares, universidades y corporaciones. Además, como se temía que un ataque nuclear desbaratara las rutas de comunicación se crearon rutas alternativas de transmisión de datos. Dentro de cada ruta, ciertos elementos que luego se llamaron routers sabrían cómo reenviar un paquete de información por uno u otro camino.

La razón de que el router supiese la senda a utilizar se encerró dentro de lo que conocemos como datagrama IP:



Nos interesan en particular los siguientes campos de la cabecera IP:

* Source IP Address
* Destination IP Addres
* Data

Pensemos que esta humilde y pequeña pieza de software (la cabecera IP de un “paquete” de datos de internet) es aún hoy, un axioma de eficacia –si bien mejorada en la versión IPV6 debido al gigantesco tamaño actual de la Red-. Dicho mecanismo nos lleva y trae terabytes de información arriba y abajo mundialmente, con eficacia, segundo a segundo. ¿Podríamos copiar de dicho instrumento algunos paradigmas para nuestro framework? En concreto, para diseñar los viajeros entre workflows: los token.

Nos fijaremos, a fin de resumir, en las prestaciones Source IP Address, Destination IP Addres, Data; y los routers que hacen uso de todo.

**Source IP (dirección IP de origen)**

Es el usuario que inicia el proceso, que le pide al manager que lance un workflow para él. Cuando el workflow “sepa” la respuesta, se la dirá al usuario debido a que conoce su origen. Además, debido a la trazabilidad, cuando un workflow se comunique con otro, y este con un tercero, el token siempre llevará indeleblemente escrito su origen, y así en todo proceso quedará registrado por petición de quién se hace qué.

**Destination IP Addres (dirección IP de destino)**

Corresponde al identificador único (Guid) de un workflow que existe en un almacén (una base de datos con workflows). Cuando el token llega al manager este lee dicho campo y “levanta” del almacén el workflow, lo ejecuta en beneficio del usuario.

**Data (datos)**

Parecería que este campo es arbitrario, y lo es. Lo que es digno de mención es que todos y cada uno de los parámetros que hay que enviar a un workflow, y entre estos, irán encapsulados bajo el token. Así todos se abrirán con la misma llave.

**El interfaz IToken**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo de dato |  |
| Id | Guid | El identificador único del token |
| Usuario | Guid | El identificador único del usuario |
| Workflow | Guid | El identificador único del usuario en el almacén que se quiere ejecutar |
| StartIn | Date | Scheduling: cuándo hay que ejecutar el wf |
| EndIn | Date | Scheduling: el TTL del workflow |
| StartDate | Date | Momento en que arranca el wf |
| EndDate | Date | Momento de finalización |
| Parameters | Object | Datos para procesar el wf |
| Data | Object | Datos que devuelve el wf |

**El router: las rutas externas del workflow**

Cada enrutador lleva incorporada una tabla de rutas, a fin de conocer a quién enviarle un datagrama en concreto que busca un destino. Una decisión arquitectónica que pensamos deficiente, sería escribir una ruta de ejecución en el inicio del proceso.

En efecto: aparentemente parecería lógico que antes de iniciar un viaje hay que conocer el destino. Bien, sí y no.

Digamos que un usuario (o un programa informático, para el caso es lo mismo), le da a un botón de una pantalla que dice: “facturación del mes”. ¿Debería ser que, debajo de dicho botón estuvieran escondidas todas las rutas por dónde ha de pasar el proceso de facturación? En los programas tradicionales, esto es así. Un programa al uso de facturación suma las facturas del periodo deseado, lo hace en un momento, y se consigue el resultado.

Para nuestro símil, digamos que el destino final único que quiere conocer el botoncito es: cuánto hemos vendido en el mes. Pero el botón usa un token ¿Cómo funciona entonces nuestra arquitectura?

|  |  |  |
| --- | --- | --- |
| Campo | Tipo de dato |  |
| Token. Id | Guid | 111222—333—33—3-344 |
| Token.Usuario | Guid | Usu-22255-3 |
| Token.Workflow | Guid | 556699—993 |
| Token.StartIn | Date | 01/04/2010 00:00:01 |
| ~~EndIn~~ | ~~Date~~ | ~~no relevante~~ |
| ~~StartDate~~ | ~~Date~~ | ~~no relevante~~ |
| ~~EndDate~~ | ~~Date~~ | ~~no relevante~~ |
| Token.Parameters | Object | (de 1/3 a 31/3) |
| Token. Data | Object | 5.000,00€ |

Fijémonos en dos cosas en especial. La última de todas, el destino del mapa, es Token. Data. Todo va a empezar porque queremos ese “destino”. ¿Y qué ruta suministramos al token? Solo el siguiente punto: Token.Workflow. Nada más.

Imaginemos un complicado proceso de facturación, de una compañía de seguros. Eso no es como facturar las neveras de una tienda. Pero todo puede que empiece igual, un señor le da a un botoncito que dice “facturar el mes”.

En la compañía de seguros, el botoncito activa a un programa que empieza por determinar los servicios facturables: seguros vendidos por los comerciales, servicios no cubiertos por las pólizas, recargos y un sinfín más.

Quizás el software tenga que enviar peticiones de información a otras compañías concertadas “¿Se le ha prestado algún servicio al Sr. Pérez este mes? Y puede que estas tarden en responder segundos, horas o días. El departamento de marketing debe consultarse, pues puede que determinadas promociones apliquen a la facturación. Y en función de decisiones de gerencia quizás se establece que hay recargos o descuentos sobre algunos tipos de servicios que se facturan. Consultemos con gerencia. Quizás se decida compensar las prestaciones a abonar en detrimento de los servicios a cobrar. Por cierto, legalmente no se debiera pagar al asegurado si no ha hecho efectivas las cuotas. Consultemos con cobros. Y tampoco pagaremos si Hacienda nos dice que es deudor del fisco. Si así es, el software debe hacer nuevas consultas ¿A quiénes? Una de dos. O lo sabe todo y todo lo hace él, o delega en otros programas especializados que sepan hacer mejor dicho cometido. Lo dejamos aquí.

Repreguntemos: ¿debe el botón “facturación del mes” conocer todas las rutas por dónde ha de viajar nuestro token, nuestra petición, a fin de obtener el resultado? Debiera sernos claro que no, que eso convertiría a nuestro software en muy frágil, muy difícil de mantener. ¿Cuál es entonces la solución de nuestra AOP?

Como en las redes token ring, nuestro token solo conoce la siguiente estación. El workflow que le dará el resultado final, y este ya sabrá a quien invocar. Y este último el siguiente, y así.

Pero aún nos queda determinar quién juega el papel de router, con las tablas de rutas, en nuestra solución. Este papel queda para otro interfaz de los workflows del Patronat: el Interfaz IPaths.

**El Interfaz IPaths**

Decíamos que un workflow se comunicará con el mundo exterior quizás varias veces, conforme se desciende, de arriba abajo, en el flujo de sus tareas. Quizás invoque una aplicación tradicional externa, o un servicio web de otra organización, o quizás otro workflow necesario. Ahora bien, toda esta comunicación debe, forzosamente, y siguiendo los criterios arquitectónicos del framework, caracterizarse por lo siguiente:

* La comunicación será asíncrona.
* La salida se hará por una puerta llamada bookmark
* La entrada se hará por la misma puerta que se salió, y solo tras entrar por ahí se podrá cruzar al siguiente proceso.

¿Cómo sabrá el workflow, cuando el flujo le pida saber algo de fuera, por qué puerta salir y dónde ir?: debido al interfaz IPaths. Todo workflow del PMHB contará con dicho interfaz.

|  |  |  |
| --- | --- | --- |
| Campo | Tipo de dato |  |
| BookMarkName | String | PuertaDeAverías |
| WorkflowTarget | Guid | WfAverias (34234-43242-2df) |

Cuando el flujo de nuestro workflow llegue a un punto en que se le diga: no sigas para abajo, toma la puerta “PuertaDeAverías” y ve al Workflow WfAverias(34234-43242-2df) , y espera a que vuelva con la información de las reparaciones realizadas para los inquilinos.

Y cuando vuelve, al minuto, a la hora, o al mes siguiente, nuestro workflow sigue. Y ahora se encuentra con que aparece otra puerta: “PuertaDeRecargos”, que apunta al Workflow: WfRecargos(3332s), y se repetirá el proceso anterior.

Se pretende ilustrar que el workflow del que estamos hablando cuenta con un mapa de rutas de sus puertas y los vecinos de al lado. Desconoce si estos, a su vez, saldrán por sus respectivas puertas a preguntar a sus vecinos. Veamos todo esto dibujado.

App externa

t

token

start

parada 1

parada 2

End

token

t

token

start

parada 1

End

App externa

token

Workflow facturación

Workflow reparaciones

Destaquemos que el token que se pasan los procesos y los workfows es el mismo. Solo que cada vez contiene más información. Hasta que regresa al señor del botoncito “facturar el mes”, con la información Token. Data = 5000€.

**Resumen:**

Un programa informático, nuestra casa: 

Nuestro futuro framework de workflows, la ciudad:

