# Choreos Middleware

Andrés Merlo Trujillo

Universidad de Granada (UGR)

**Abstract.** The abstract should briefly summarize the contents of the paper in 150–250 words.

**Keywords:** First keyword · Second keyword · Another keyword.

## 1 Introducción

El aumento de tamaño de internet y el uso de web services ha destapado graves problemas para la incorporacion de servicios en sistemas web. Este problema se puede resolver mediante estandares, haciendo uso de la orquestracion de servicios. Sin embargo, conforme ha ido creciendo, se han descubierto graves problemas de escalabilidad y puntos criticos de fallo, debido a la naturaleza centralizada de la orquestracion. [1]

Al haber crecido tanto internet, se ha convertido en una red heterogenea, movil y adaptable donde tiene mas sentido reutilizar servicios existentes de manera distribuida y descentralizada para los programas, en vez de realizar programas monoliticos[2].

Esta solucion resolveria la centralizacion y los problemas subsecuentes anteriormente mencionados, con la desventaja de ser mas dificiles de gestionar por su naturaleza distribuida[2][1].

A este concepto de reutilizar servicios de manera distribuida se le denomina "Coreografias". Una coreografia permite realizar servicios distribuidos, formalizando la forma en la que los demas servicios interactuan, indicando el comportamiento esperado de los servicios participantes.

Es por eso que aparece Choreos, un middleware para componer coreografias de servicios a gran escala orientado principalmente al Internet del Futuro (FI) [1].

## 2 Objetivos

El objetivo principal de Choreos es el de ofrecer una plataforma para diseñar, desplegar y ejecutar las coreografías para sistemas orientados a servicios (SOS) y a gran escala. Para ello, Choreos ofrece un IDRE (Integrated Development and Runtime Environment) para modelar y especificar estas coreografias. Además tiene herramientas de validación que permiten verificar que la especificación realizada tenga un mínimo de calidad. [2]

Otro objetivo muy importante es el de mantener la escalabilidad, la interoperabilidad y el QoS (Quality of Service)[3]. Esto es algo también muy importante, ya que tienen pensado su uso para el Internet Futuro (FI) de ultra gran escala (ULS)[2].

Por último, otro objetivo que se puede deducir de los anteriores, es que han querido crear una capa de abtracción sobre la que trabajar para facilitar la creación de coreografias con unos minimos de calidad. Estos minimos se comprueban mediante el IDRE en la etapa de validación y verificacion, si la calidad no es satisfactoria, se vuelve a la etapa inicial de especificación [2].

## 3 Diseño

El diseño de la arquitectura que proponen, se basan en otras tecnologias middleware: bus de servicio distribuido (DSB) y grid y cloud computing.[1]

Para la parte del DSB, hacen uso de una solución middleware denominada "PEtALS", la cual se basa en una red Peer to Peer (P2P). Esto permitirá hacer un DSB muy escalable, que permitirá coreografias de servicios muy heterogeneos, muy indicado para el FI (Future Internet)[1].

El DSB se encarga principalmente de la conexión con servicios distribuidos de distintas organizaciones. Además, tambien se usara para el IDRE para poder descubrir los servicios distribuidos y poder definir la interaccion con estos. [10]

Otro aspecto de diseño es el uso de una arquitectura de grid y cloud computing, que permite tener una escalabilidad y un rendimiento excelente.

Además, todos los servicios que esten disponibles para las coreografias, dispondran de un adaptador para solventar las diferencias de comunicacion y, a su vez, de un "Coordinator Delegate" (CD), que se encargará de realizar la coordinación entre los servicios elegidos para cada coreografia. Estas entidades, se intercambian la informacion de coordinacion para evitar interacciones incorrectas. [2]

## 4 Características

Las principales caracteristicas que tendra este middleware son las siguientes:

- **Rendimiento:** Gracias al uso del Grid y Cloud Computing, se pueden servir a millones de usuarios, realizando miles de peticiones simultaneas.[1]
- **QoS-aware:** Haciendo uso de una subcomponente denominada "Predictor", el middleware puede estimar comportamiento de la calidad de servicio frente al tiempo[10], haciendo que otras subcomponentes puedan tomar decisiones.
- **Escalabilidad:** La arquitectura especificada permite que se puedan lanzar muchos servicios sin repercutir demasiado en el rendimiento. Además, haciendo uso del "Predictor" y un analizador de escalabilidad (Scalability Analyser), se puede estimar la evolución del QoS, algo directamente relacionado con la escalabilidad. [10]

- **Tolerante a fallos:** Debido a la naturaleza distrbuida de los servicios, estos pueden estar replicados y poder continuar el funcionamiento del sistema sin problema. Además, el middleware posee un mecanismo de tolerancia a fallos [20], que se encarga de mantener el servicio siempre activo.
- **Simplicidad en la modelacion de coreografias:** Gracias al uso del IDRE, se puede modelar coreografias de servicios de manera sencilla, permitiendo especificar los requisitos y modelar las coreografias. La coreogradia es creada automaticamente, asegurando los criterios de calidad y escalabilidad[30].

## 5 Mecanismos

El principal mecanismo para la coordinacion de los servicios es mediante el uso de paso de mensajes en los CDs. Los CDs se encargan de enviar informacion para asegurarse en todo momento de que se esten realizando las coreografias correctamente.

Los mensajes propios de los servicios para realizar la tarea deseada en un sistema web, se intercambian siguiendo el protocolo peticion/respuesta (Request/Response RR). Además, estos mensajes deben ser enviados por los propios CDs [2], luego pasar por el adaptador del servicio, para traducirlo y que sea entendible por el mismo, y luego realizar la operación deseada.

Además, en caso de caída de un servicio, el middleware debe ser capaz de reasignar a las coreografias que usen dicho servicio a otra replica, siendo transparente para los clientes.

## 6 Protocolos y servicios

Para la interconexión de las distintas componentes de Choreos, se ha usado una API REST sobre HTTP, ya que es superior en terminos de rendimiento, velocidad y fiablidad que otros protocolos.[128]

Esto permite tener una interfaz uniforme entre todos los componentes, haciendo que la comunicación y la depuración sea más sencilla frente a otras opciones como DPWS. [128]

Este middleware consta principalmente de un servicio, denominado “Enactment Engine” (EE). Este se encarga de manejar los fallos de los componentes externos, muy importante en sistemas a gran escala [128].

Este servicio consta de los siguientes componentes:

- **Choreography Deployer:** Expone la API la dar soporte al despliegue de coreografias. El cliente debe suministrar la especificacion de la coreografia, esta contiene la localizacion de los servicios [128] y la forma de interaccion de los mismos.
- **Deployment Manager:** Se encarga de desplegar servicios en un entorno cloud. Recibe la especificiacion de la coreogradia en un script que realiza todas las tareas necesarias para lanzar el servicio (procesos, preparacion de servicios, etc)[128].

Estas son las componentes que el servicio “Enactment Engine ” tiene por defecto. Además incluye otras componentes de terceros como “Chef Solo”, encargada de instalar la configuracion del middleware y sus componentes en un nodo concreto, y el “Cloud Gateway”, encargado de crear y destruir máquinas virtuales (nodos)[128] para el despliegue de los servicios.

## 7 Propiedades

Las principales propiedades de Choreos, como ya se ha dicho en secciones anteriores, son las siguientes:

- **Escalable:** La naturaleza distribuida de los servicios, hace que se puedan tener mas nodos en momentos dados para hacer frente a la demanda de un servicio concreto. Además, el middleware tiene un mecanismo de escalabilidad que permite estimar el rendimiento de un servicio en función del tiempo de respuesta de los mismos.
- **Tolerante a fallos:** Al igual que antes, debido a la naturalzea distribuida, se pueden tener replicas de servicios, los cuales pueden ser cambiados por el EE en caso de fallo.
- **Fiable y sencillo:** Ofrece un IDRE que permite crear coreografias de una manera mas sencilla que si no se utilizase el middleware. Además se comprueba que tenga unos criterios de calidad minimos antes de poder ser desplegado.
- **Interoperabilidad:** Al hacer uso de adaptadores, los distintos servicios heterogeneos pueden comunicarse sin problema, al traducir sus mensajes a un lenguaje comun y luego ser vueltos a traducir en el otro extremo.[30]

## 8 Ejemplos de funcionamiento

La organizacion ha realizado tres ejemplos de uso de Choreos. “Passenger-Friendly Airport”, “Adaptive Customer Relationship Booster” y “DynaRoute”.[30] Yo me centraré solo en el primer ejemplo.

*Passenger-Friendly Airport* se concentra en los servicios suministrados a los pasajeros. Describe el uso de la informacion que aparece en un aeropuerto para adaptar los servicios ofrecidos a los biajeros, ofreciendo una mejor calidad. Chroeos se usa para obtener acceso a los sensores, tanto del aeropuerto como en los dispositivos de los pasajeros. Por ejemplo, se usan microfonos para ajusta automaticamente el volumen de las megafonias, dependiendo de que nivel de sonido haya (si hay mucha gente, se sube el nivel de la megafonia automaticamente).[128]

Además, en los dispostivios de los pasajeros se instala un serivico de localizacion, para darles indicaciones de donde deben ir, haciendo que la navegacion por el aeropuerto sea más sencilla.[128]

## 9 Análisis crítico

Este middleware presenta una capa de abstraccion muy interesante. El poder realizar servicios distribuidos a patir de otros servicios, tambien distribuidos. Ademas, incluye un entorno en el que definir las interacciones con otros servicios necesarias, haciendo que sea mucho mas simple y general para definir.

Otro aspecto muy interesante es el hecho de tener una tolerancia a fallos en el propio middleware, algo muy importante cuando se trabaja en entornos distribuidos. También es agnostico de la arquitectura y protocolos subyacente de los servicios, proveiendo una interfaz general para que puedan interactuar entre ellos.

Además, parece ser que va a ser el dominante, ya que han aparecido más estándares, incluso antes que Choreos, pero que nunca han llegado a implementarse del todo, haciendo que este middleware sea el único en su campo.

La eficacia de este middleware esta ya probada y documentada, haciendo que sea una muy buena opcion para desarrollar una aplicacion de servicios escalable y tolerante a fallos.

## 10 Conclusiones

Como conclusión, se puede decir que este middleware tiene mucho potencial para ser el dominante en su ambito. Además, se puede ver que han tenido en cuenta cuestiones de heterogeneidad, deteccion de errores, escalabilidad, etc. Esto hace que tenga un rendimiento excelente, como aparece en algunos artículos.

Además, no solo proveen el propio middleware, también proveen un entorno de desarrollo (IDRE), haciendo que sea mucho mas facil el modelado y la verificacion de los servicios.

Por último, como la tendencia es hacer uso cada vez más de los servicios, puede que sea una excelente herramienta para desarrollar aplicaciones basadas en servicios, o incluso desarrollar más servicios a partir de existentes.

## 11 First Section

### 11.1 A Subsection Sample

Please note that the first paragraph of a section or subsection is not indented. The first paragraph that follows a table, figure, equation etc. does not need an indent, either.

Subsequent paragraphs, however, are indented.

**Sample Heading (Third Level)** Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

**Table 1.** Table captions should be placed above the tables.

| Heading level | Example | Font size and style |
|---|---|---|
| Title (centered) | **Lecture Notes** | 14 point, bold |
| 1st-level heading | **1 Introduction** | 12 point, bold |
| 2nd-level heading | **2.1 Printing Area** | 10 point, bold |
| 3rd-level heading | **Run-in Heading in Bold.** Text follows | 10 point, bold |
| 4th-level heading | *Lowest Level Heading.* Text follows | 10 point, italic |

*Sample Heading (Fourth Level)* The contribution should contain no more than four levels of headings. Table 1 gives a summary of all heading levels.
Displayed equations are centered and set on a separate line.

$$x + y = z \tag{1}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. 1).

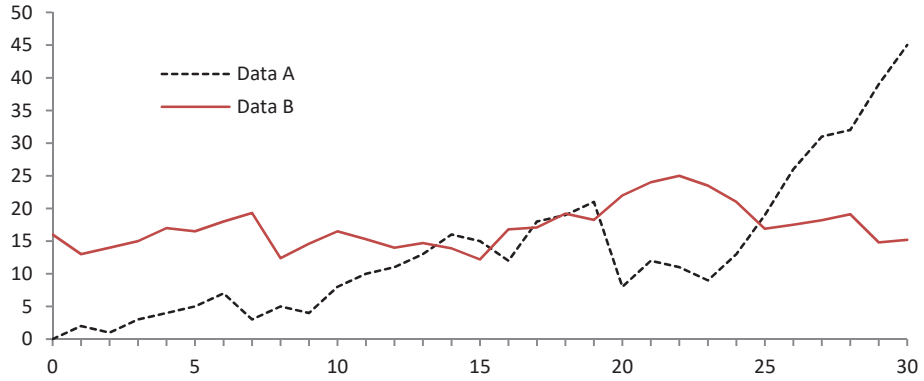


**Fig. 1.** A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

**Theorem 1.** *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

*Proof.* Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4], and a homepage [5]. Multiple citations are grouped [1–3], [1, 3–5].

**Acknowledgements** Please place your acknowledgments at the end of the paper, preceded by an unnumbered run-in heading (i.e. 3rd-level heading).

## References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017