Diseño y Arquitectura de Servicios Escalables

Unidad 1

Replicación. Elasticidad

Índice

1. Replicación: Concepto y objetivos
2. Modelos de replicación
    1. Pasivo
    2. Activo
    3. Multi-máster
3. Elasticidad

Bibliografía

[BMST93] N. Budhiraja, K. Marzullo, F. B. Schneider, S. Toueg: “The Primary-Backup Approach”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 8, págs. 199-216. Addison-Wesley, 2ª edición, 1993.

[IBM05] International Business Machines, Corp.: “An Architectural Blueprint for Autonomic Computing”, 3ª ed., white paper, junio 2005 (primera edición en 2001).

[JT75] Paul R. Johnson, Robert H. Thomas: “The Maintenance of Duplicate Databases”, Network Working Group, RFC #677, enero 1975.

[Lam98] Leslie Lamport: “The Part-Time Parliament”. ACM Trans. Comput. Syst. 16(2): 133-169 (1998)

[Sch93] Fred B. Schneider: “Replication Management Using the State-Machine Approach”. En S. J. Mullender, editor, “Distributed Systems”, capítulo 7, págs. 166-197. Addison-Wesley, 2ª edición, 1993.

Índice

1. Concepto y objetivos

- La replicación es una técnica consistente en mantener más de una copia de un mismo componente o dato en una aplicación.
- El uso de cachés es un caso especial de replicación, que tiene como objetivo mejorar la escalabilidad (tanto de tamaño, como geográfica).

1. Concepto y objetivos

- En un sistema distribuido se utiliza la replicación para:
  - Evitar que un fallo en una de las réplicas provoque un fallo en el componente replicado.
    - Si las réplicas se han distribuido convenientemente será muy improbable que más de una falle simultáneamente.
      - Cada réplica en una máquina diferente.
  - Incrementar la capacidad de servicio.
    - Es un mecanismo necesario para lograr escalabilidad.
    - Muy fácil en las operaciones que no impliquen una modificación del estado del componente replicado.
    - Hay muchas aplicaciones con alto porcentaje de operaciones de lectura.

1. Concepto y objetivos

- Sin embargo, la replicación tiene un coste generalmente alto:
  - Hay que controlar con cierta frecuencia el estado de cada réplica.
  - Deben soportarse las incorporaciones de nuevas réplicas y el fallo de las existentes.
  - Deben propagarse las modificaciones a todas las réplicas, ofreciendo un modelo de consistencia adecuado.

1. Concepto y objetivos

- Por todo esto, deben diseñarse protocolos para:
  - Gestionar adecuadamente la consistencia. Esto implica:
    - Propagación de actualizaciones.
    - Filtrado o bloqueo de accesos que sólo impliquen lectura.
  - Añadir réplicas. Puede necesitar el bloqueo de las peticiones en curso.
  - Recuperar réplicas previamente caídas.
    - Depende de la cantidad de estado que tenga el componente replicado. Si hubiera poco, bastaría con descartarlo y propagarlo entero.
    - Los protocolos de recuperación (y/o reconciliación, según el modelo de particionado) no deben implicar una pérdida de disponibilidad.

Índice

2. Modelos de replicación

- Existen diferentes técnicas para replicar un determinado componente o servicio.
- Las más comunes son:
  - Replicación pasiva (o replicación primario-backup).
  - Replicación activa (o máquina de estados).
  - Replicación multi-máster.

2. Modelos de replicación

- Cada modelo se caracteriza por:
  - El número de réplicas que pueden servir directamente las peticiones de los clientes.
  - La forma de recibir las peticiones.
  - La forma de propagar las modificaciones.
  - La estrategia para soportar caídas de las réplicas.
  - Su capacidad para servir operaciones cuya ejecución no sea determinista.

Índice

2.1. Replicación pasiva
[BMST93]

- Todos los clientes envían sus peticiones a una misma réplica: réplica primaria.
- La réplica primaria es la única que sirve directamente la petición y modifica de inmediato su estado.
- No puede haber más de una réplica primaria en cada momento.
- Las peticiones recibidas (erróneamente) por otras réplicas deben ser:
  - Descartadas, avisando al cliente de ello, o...
  - Reenviadas a la réplica primaria.

2.1. Replicación pasiva

- Las réplicas no primarias se llaman réplicas secundarias.
- En algunos sistemas se admite que las réplicas secundarias procesen las peticiones de solo lectura.
- Antes de responder al cliente en cada petición que modifique su estado, la réplica primaria propaga las modificaciones a las demás réplicas.
- Cuando hay una caída de la réplica primaria debe elegirse de entre las secundarias una para reemplazarla y tomar el rol de primaria.

2.1. Replicación pasiva

- Pasos a seguir en el servicio de una operación:
  1. El cliente envía la petición a la réplica primaria.
  2. La réplica primaria procesa esa solicitud.
  3. Si hay modificaciones:
     1. Se difunden las modificaciones a las réplicas secundarias.
     2. Una vez recibidas y aplicadas las modificaciones, cada réplica secundaria contesta a la primaria.
     3. Una vez recibidas todas esas respuestas, la primaria continúa.
  4. La réplica primaria responde al cliente.

2.1. Replicación pasiva

- No será fácil asegurar que no se pierda ninguna petición en caso de fallo.
  - El cliente debe reintentar las peticiones que no sean contestadas por la réplica primaria.
  - Ese reintento se realizará sobre la nueva réplica primaria.
  - Los secundarios deben saber qué peticiones fueron atendidas, para no servirlas múltiples veces.

2.1. Replicación pasiva

- Ventajas:
  - No sobrecarga el sistema. Sólo una réplica procesa las peticiones.
    - Pero podría saturarse con facilidad.
    - Para evitar la saturación, los servicios deben ser ligeros.
      - Múltiples servicios deberían utilizar un mismo conjunto de réplicas.
      - Cada servicio usará un primario diferente.
      - Así se equilibrará la carga.
  - No necesita protocolos de difusión ordenada.
    - Sólo hay un emisor. El orden es fácil de establecer.
    - Basta con numerar consecutivamente los mensajes en que se propague estado a los secundarios.

2.1. Replicación pasiva

- Ventajas (continuación):
  - Soporta sin problemas la ejecución de operaciones no deterministas.
    - Esas operaciones las ejecuta únicamente la réplica primaria. La decisión que tome en una bifurcación no será discutida por otras réplicas.
    - Como propagamos el estado resultante, en lugar de la operación, los secundarios no podrán generar inconsistencias.
  - Mecanismos de control de concurrencia fáciles de implantar, pues son locales y se utilizan en la réplica primaria.

2.1. Replicación pasiva

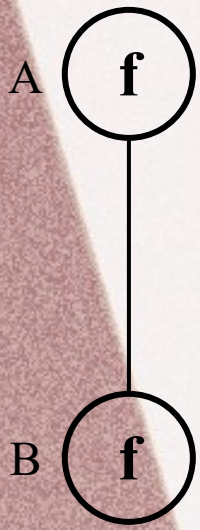
- Inconvenientes:
  - Reconfiguración costosa:
    - Necesidad de elección de un nuevo “líder”.
    - Cambio de rol en al menos una réplica.
    - Podrían perderse peticiones en curso.
      - El cliente reintentará la petición.
      - Se incrementa el tiempo de respuesta si hay fallos.
  - Código diferente para cada rol: primario y secundario se comportan de distinta manera.

2.1. Replicación pasiva

- Inconvenientes (continuación):
  - No se soportan los modelos de fallos más severos.
    - Los fallos arbitrarios no pueden ser gestionados.
      - Cuando el primario fallara de esta manera, el cliente no podría detectarlo.
    - Los fallos de omisión general no siempre se soportarán adecuadamente.
      - Si ocurren “f” fallos simultáneos de este tipo, se necesitan más de 2f réplicas para soportar esa situación sin promocionar erróneamente a un segundo primario.

2.1. Replicación pasiva

- **Omisión general**: n>2f.
  - Asumiremos que existen 2f réplicas que se han agrupado en dos conjuntos disjuntos A, B tal como muestra la figura.
  - Si todas las réplicas de A cayeran en cierta ejecución, una réplica de B sería elegida como primario.
  - Si todas las réplicas de B cayeran en cierta ejecución, una réplica de A sería elegida como primario.
  - Si las réplicas de A cometen fallos de omisión general con las réplicas de B (bien no emiten o no reciben sus mensajes), entonces esta ejecución será indistinguible de la primera para las réplicas de B y de la segunda para las de A, con ellos se elegirían dos primarios, uno en cada grupo.



Índice

2.2. Replicación activa
[Sch93]

- El cliente debe hacer llegar la petición a todas las réplicas.
  - En algunos sistemas, la petición llega a una sola réplica servidora, que antes de iniciar su proceso la envía a todas las demás (incluida ella) en orden total.
- Todas las réplicas sirven la petición: son activas.
- Todas las réplicas devuelven una respuesta.
  - Debe efectuarse un filtrado de respuestas en el *stub* cliente.
- Consistencia: Se necesita un protocolo de difusión fiable de orden total.

2.2. Replicación activa

- Pasos a seguir en el servicio de una operación:
    1. El cliente difunde la operación a todos los procesos servidores.
    2. Cada proceso servidor procesa la operación y devuelve respuesta.
    3. El cliente filtra esas contestaciones y continúa cuanto tenga suficientes.
        - El número de respuestas necesarias dependerá del modelo de fallos asumido.

2.2. Replicación activa

- Ventajas:
  - Todas las réplicas comparten un mismo programa y un mismo estado.
  - La caída de una o más réplicas no requiere un cambio de rol en las réplicas restantes.
  - La caída de una o más réplicas no retarda la entrega de una respuesta al cliente.
  - Bajo este modelo, los programas servidores pueden escribirse de forma muy similar a la de un servicio no replicado.

2.2. Replicación activa

- Ventajas (continuación):
  - Pueden soportarse los fallos arbitrarios.
    - Si pueden darse “f” fallos simultáneos, se necesitarán al menos “2f+1” réplicas.
      - Depende de si podemos verificar quién envía cada mensaje.
      - Si no es posible, se necesitan “3f+1” réplicas.
    - El cliente necesitará esperar una mayoría de respuestas coherentes.
      - ¿Cómo puede preverse el número de fallos simultáneos?

2.2. Replicación activa

- Inconvenientes:
  - Hay que garantizar que las peticiones lleguen a todas las réplicas: protocolos de difusión fiable.
  - Hay que garantizar que diferentes peticiones con interacción entre sí lleguen en el mismo orden a todas las réplicas: la difusión requiere orden total.
    - Como la difusión atómica se reduce al problema de consenso, éste debe poder resolverse en el modelo de sistema asumido.
    - Paxos [Lam98] es un ejemplo de protocolo para dar soporte al modelo activo, considerando todos los mecanismos necesarios (consenso, comunicación, orden, procesos, persistencia...).

2.2. Replicación activa

- Inconvenientes (continuación):
  - Resulta difícil atender operaciones no deterministas, pues en cada réplica podrían originar resultados diferentes.
    - Esto provoca inconsistencias.
    - Por tanto, el código de este tipo de servicios debe ser determinista.
  - Resulta difícil la gestión de las peticiones iniciadas por un objeto replicado bajo este modelo (múltiples peticiones que deberían ser identificadas como una sola por el servicio invocado).
  - Necesidad de filtrado de peticiones y filtrado de respuestas.

Índice

2.3. Replicación multi-máster [JT75]

- Los dos modelos anteriores tienen una consistencia fuerte.
  - Eso requiere un alto grado de coordinación entre réplicas.
  - Difícilmente escalarán.
- ¡Se necesita algún modelo de replicación alternativo!
  - Con consistencia relajada.
  - Replicación multi-máster.

2.3. Replicación multi-máster

- Interacción entre réplicas similar a la del modelo pasivo.
  - Pero se contesta de inmediato al proceso cliente.
  - La propagación de modificaciones, si las hay, se realiza a posteriori.
    - De manera asincrónica.
- Pero cada cliente puede elegir una réplica delegada (máster para esa interacción) diferente.

2.3. Replicación multi-máster

- Ventajas:
  - Sobrecarga mínima.
    - Similar a la del modelo pasivo.
    - Pero se puede responder mucho antes a los clientes.
  - Fácilmente escalable.
    - No hay control de concurrencia.

2.3. Replicación multi-máster

- Inconvenientes:
  - Cuando un máster falle, se puede perder la operación en curso.
  - Comparte con el pasivo la gestión limitada de modelos de fallos que puede soportar.
    - El arbitrario no puede gestionarse.
  - Consistencia muy débil entre réplicas.
    - “Eventual” si se utilizan operaciones conmutativas…
    - ...y se propagan estas, en lugar de sus resultados.

Índice

3. Elasticidad

- Un servicio distribuido es elástico cuando cumple estas tres propiedades simultáneamente:
  - Escalabilidad.
  - Adaptabilidad.
  - Autonomía.
- Objetivo: Que el servicio sea potencialmente escalable, pero adaptándose siempre a la carga que haya en cada momento y sin la necesidad de que lo gestione un administrador de sistemas.

3. Elasticidad

- La escalabilidad y la adaptabilidad se estudiarán en temas posteriores.
- La gestión autónoma de los servicios distribuidos fue propuesta por IBM en 2001 [IBM05].
  - Creando el concepto de computación autónoma.
  - Los servicios son capaces de autoadministrarse para lograr ciertos objetivos.

3.1. Computación autónoma

- En [IBM05] se define la “computación autónoma” como...
  - “*A computing environment with the ability to manage itself and dynamically adapt to change in accordance with business policies and objectives.*
  - *Self-managing environments can perform such activities based on situations they observe or sense in the IT environment rather than requiring IT professionals to initiate the task.*
  - *These environments are self-configuring, self-healing, self-optimizing, and self-protecting.*”

3.1. Computación autónoma

- El sistema debe administrarse por sí mismo...
- … y debe ser adaptativo.
- También debe proporcionar buena calidad de servicio (QoS).
- Ejes de la calidad de servicio:
  - Rendimiento.
  - Disponibilidad.

3.1. Computación autónoma

- Atributos de sus componentes:
  1. Auto-configuración. Adaptación dinámica a los cambios en su entorno.
  2. Auto-curación. Los componentes deben percibir cualquier defecto e iniciar las acciones correctivas pertinentes sin interrumpir el funcionamiento normal del sistema.
  3. Auto-optimización. Los componentes monitorizarán y sintonizarán sus recursos adecuadamente para garantizar un rendimiento óptimo.
  4. Auto-protección. Los componentes se anticiparán, detectarán, identificarán y se protegerán frente a las amenazas, tomando las acciones correctivas necesarias para minimizar sus vulnerabilidades.

3.2. Computación autónoma. Arquitectura

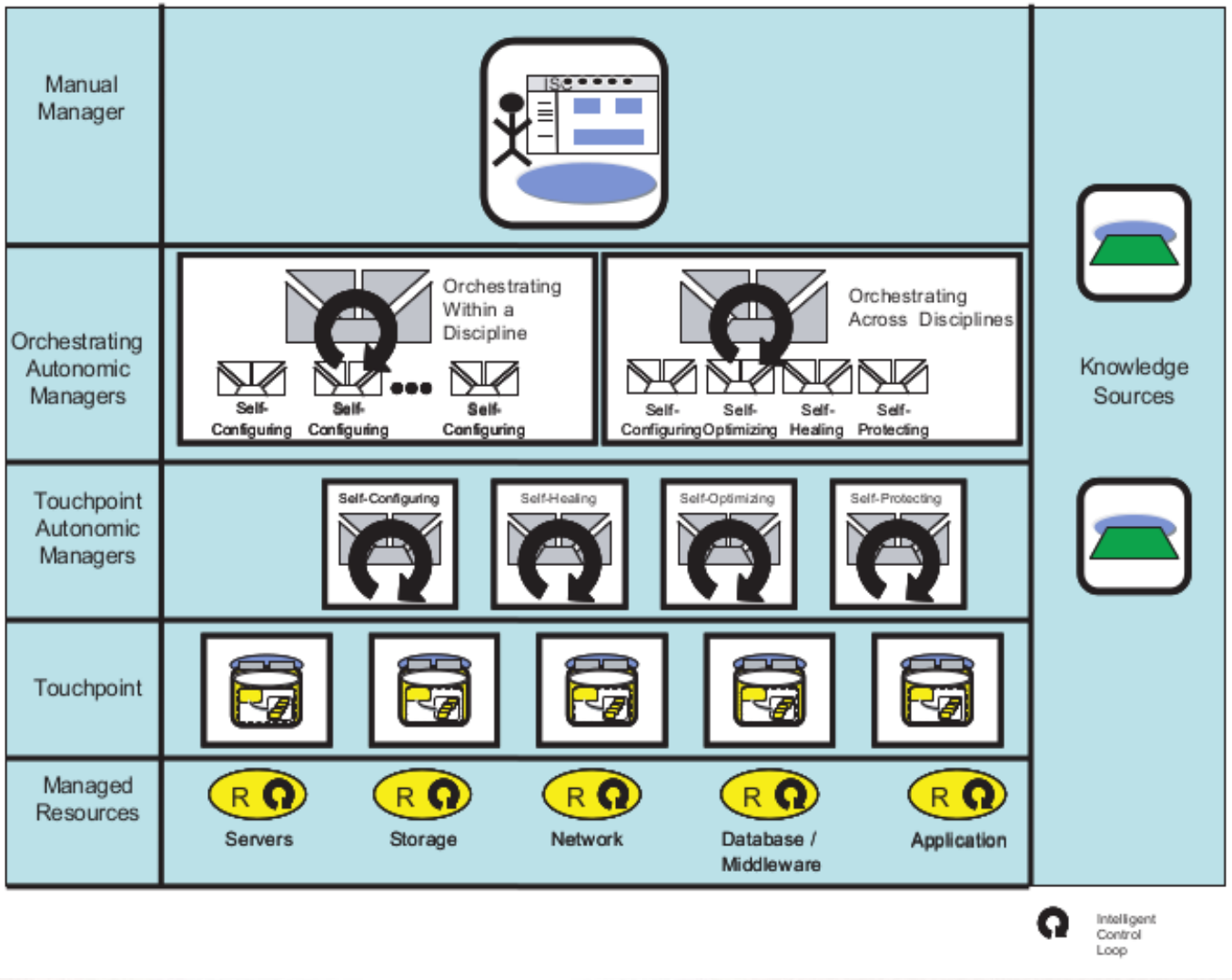


Ilustración tomada de [IBM05]

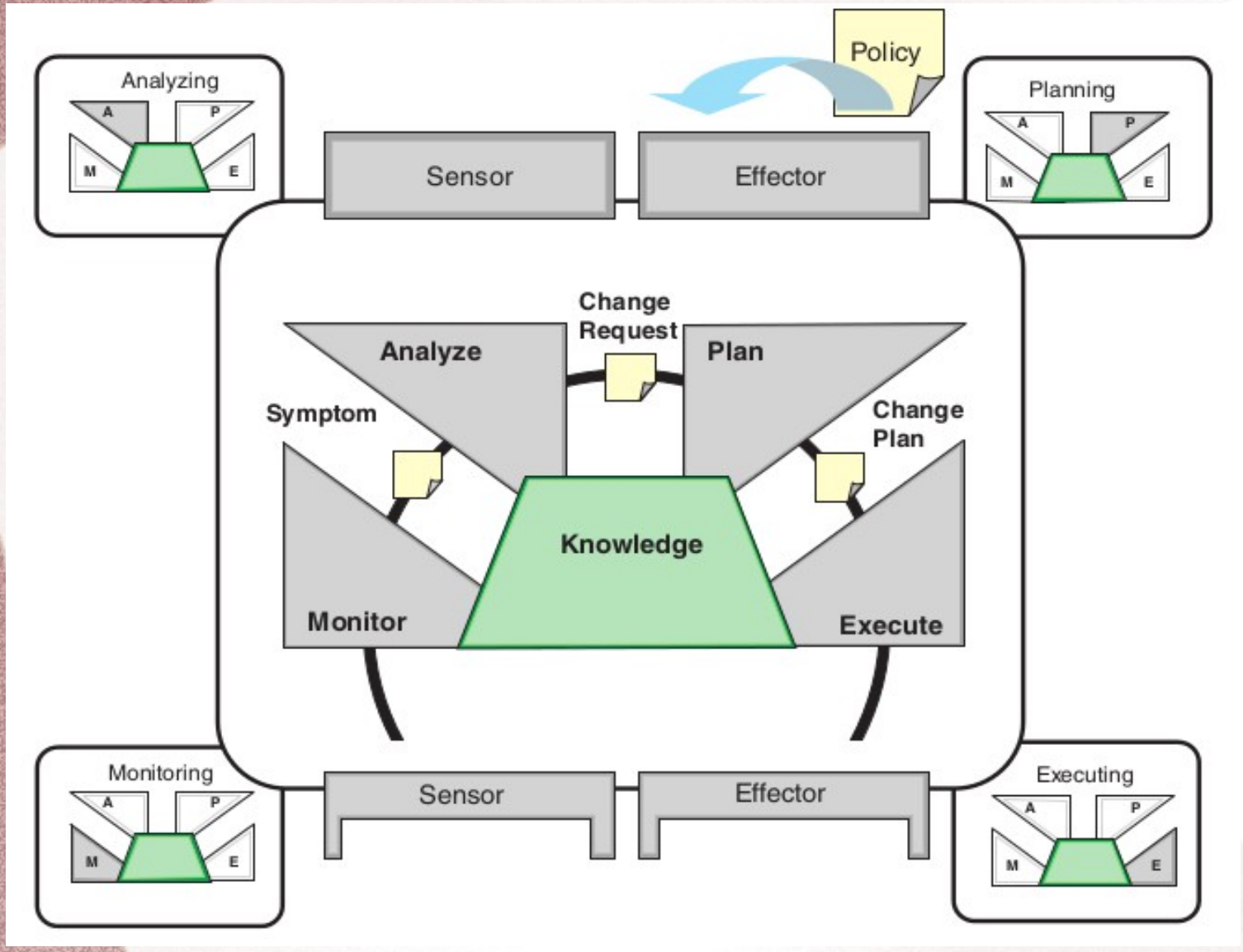
3.2. Computación autónoma. Arquitectura

- Elementos de la arquitectura:
  - Recurso gestionado. Cada uno de los elementos del sistema que podrá ser utilizado en un servicio.
  - "*Touchpoint*". Interfaz de acceso a un recurso gestionado. En esa interfaz habrá sensores y actuadores.
  - Gestor autónomo. Componentes que regularán el ciclo de control de los recursos (o de otros gestores de bajo nivel) con el objetivo de automatizar sus tareas y alcanzar sus objetivos.
  - Fuente de conocimiento. Implantación de cierto tipo de almacén (BD, registro, diccionario...) que facilita el acceso al "conocimiento" según las interfaces definidas en la arquitectura.
  - "Conocimiento": Síntomas, políticas, peticiones de cambios, planes de adaptación...

3.3. Gestor autónomo

- Se establece el modelo MAPE-K para el ciclo de control desarrollado por estos gestores:
  - Monitorización.
  - Análisis.
  - Planificación.
  - Ejecución.
  - K-> Conocimiento.

3.3. Gestor autónomo



3.3. Gestor autónomo

1. Monitorización:
    - Recoge la información relevante desde los sensores emplazados en cada uno de los recursos gestionados.
    - Filtra o realiza resúmenes a partir de esa información.
    - Deposita “síntomas” en la fuente de conocimiento.
        - A procesar en la fase de análisis.

3.3. Gestor autónomo

2. Análisis:

- Facilita herramientas para procesar la información recolectada en el paso de monitorización.
    - Modelos de colas, series temporales...
- Con ellas se extraen conclusiones sobre la situación actual.
    - A tomar como base para predecir el comportamiento inmediato del sistema...
    - … y aplicar las medidas correctoras pertinentes en las etapas siguientes del ciclo de control.
- Depositará “peticiones de cambio” en la fuente de conocimiento.

3.3. Gestor autónomo

3. Planificación:
    - Facilita los mecanismos necesarios para desarrollar las acciones que permitan alcanzar los objetivos previstos.
    - Estos mecanismos utilizan información sobre las políticas para guiar su trabajo.
    - Se genera un plan de actuación / adaptación.

3.3. Gestor autónomo

4. Ejecución / Actuación:
   - Esta etapa proporciona mecanismos para controlar la ejecución del plan resultante de la etapa anterior.
   - Las modificaciones a realizar son dinámicas.
     - No interrumpen el servicio.
     - El intervalo necesario para aplicar estas actuaciones debe ser breve.
   - La “reacción” aplicada en este punto debe estar en consonancia con el histórico de monitorizaciones acumulado hasta el momento.