Hacia un modelo más flexible para la implementación de la auto reparación de sistemas de software basada en Arquitectura

Tesis de Licenciatura en Ciencias de la Computación

Chiocchio, Jonathan Tursi, Germán Gabriel

Director: Santiago Ceria

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

10 de Junio de 2011

Contenido

Objetivo de la Tesis

Presentar un modelo de **Auto Reparación** más flexible, el cual involucre a los ***stakeholders*** en el proceso de definición de requerimientos de calidad, proveyéndoles a su vez mayor visibilidad.

Auto Adaptación / Auto Reparación

Requerimientos de los sistemas de software actuales:

Auto Adaptación / Auto Reparación

Requerimientos de los sistemas de software actuales:

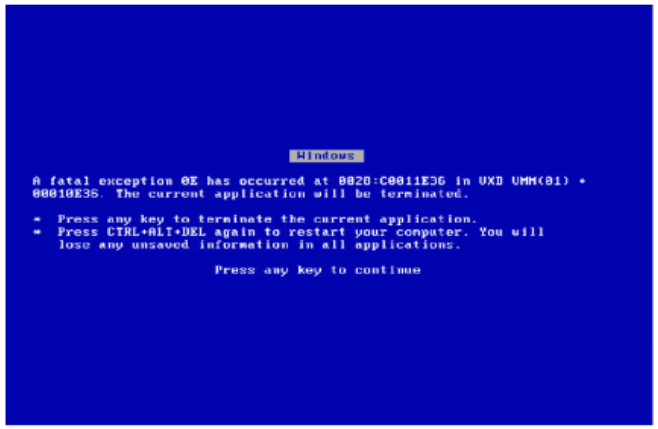
- Disponibilidad 99,9 %
- Minimizar el error humano
- Reducir costos operativos
- Implementar mecanismos para ajustar su comportamiento ante:
  - fallas de hardware,
  - requerimientos de calidad cambiantes,
  - cambios en el entorno en el que operan.

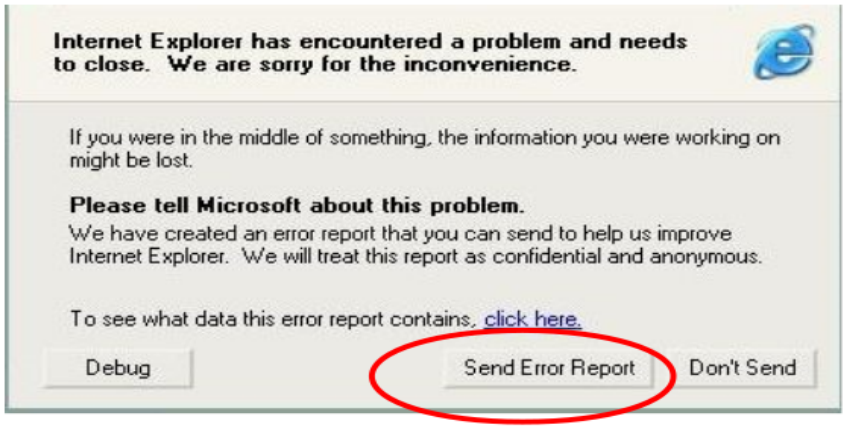
Auto Adaptación / Auto Reparación

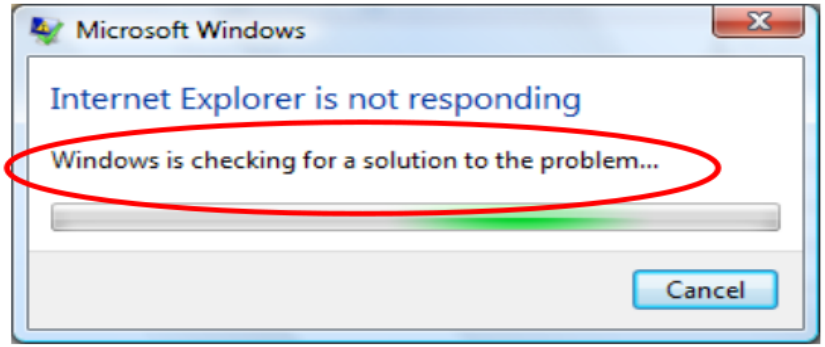
Requerimientos de los sistemas de software actuales:

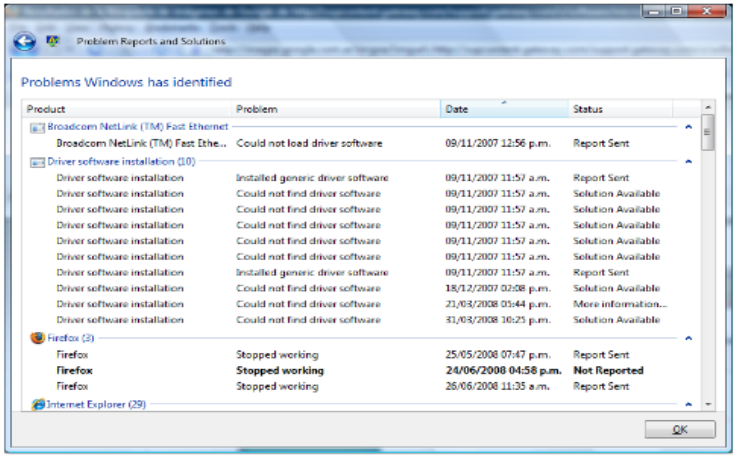
- Disponibilidad 99,9 %
- Minimizar el error humano
- Reducir costos operativos
- Implementar mecanismos para ajustar su comportamiento ante:
  - fallas de hardware,
  - requerimientos de calidad cambiantes,
  - cambios en el entorno en el que operan.
- ¡Y todo esto sin intervención humana!

Tendencia a los Sistemas Autónomos









Atributos de Calidad y Concerns

Los **atributos de calidad** (QAs) hablan de características específicas que debe tener el sistema, antes (mal) llamados "requerimientos no funcionales".

Los ***concerns*** permiten juzgar, especificar y medir los atributos de calidad. Usualmente, los requerimientos de atributos de calidad son expresados en términos de *concerns*.

Ejemplos

Atributo de Calidad	**Ejemplos de *Concerns***
Eficiencia	Comportamiento Temporal, Utilización de Recursos
Funcionalidad	Interoperabilidad, Seguridad
Mantenibilidad	Cambiabilidad, Facilidad de Prueba
Portabilidad	Adaptabilidad, Coexistencia
Usabilidad	Compresibilidad, Atractivo

Rainbow

- *Framework* de Auto Adaptación.
- Realizado en el marco del proyecto ABLE de la CMU.
- Trabaja con el modelo de la arquitectura (expresado mediante **Acme**).
- Externo al sistema a adaptar, no intrusivo.
- Soporta sistemas *legacy* y/o cuyo código no se encuentra disponible.

Ejemplo de una Arquitectura en Acme

```
Property MAX_RESPTIME : float = 1000.0;

Component s1 : Server {
  Property deploymentLocation;
  Port http0;
  Property fidelity;
  Property load;
}

Component c1 : Client {
  Property experRespTime : float << default : float = 100.0; >> ;
  Property requestRate;
  rule primaryConstraint = invariant self.experRespTime <= MAX_RESPTIME;
}
```

Tácticas y Estrategias de Reparación

Táctica: decisión de diseño que permite modificar el comportamiento del sistema ante determinados estímulos.

e.g. agregar un servidor al sistema.

Estrategia: es un **algoritmo que usa una o más tácticas** para cumplir los atributos de calidad en un nivel deseado.

Ejemplo

Invariante: el tiempo de respuesta experimentado por el usuario no debe superar el umbral determinado.

Estrategia que intenta reparar el sistema en caso de romperse el invariante:

1. agregar un servidor
2. si tiempo de respuesta $>$ umbral determinado $\Rightarrow$ disminuir la fidelidad
3. Estrategia exitosa $\iff$ tiempo de respuesta $<$ umbral determinado

Características de Rainbow a Mejorar

- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.

Características de Rainbow a Mejorar

- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.
- Duplicación en la configuración de invariantes
  $\Rightarrow$ *propenso a errores*
  $\Rightarrow$ *difícil de mantener*
  $\Rightarrow$ *overhead de procesamiento*

Características de Rainbow a Mejorar

- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.
- Duplicación en la configuración de invariantes
  $\Rightarrow$ *propenso a errores*
  $\Rightarrow$ *difícil de mantener*
  $\Rightarrow$ *overhead de procesamiento*
- Poco dinámico, debido a:

Características de Rainbow a Mejorar

- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.
- Duplicación en la configuración de invariantes
  $\Rightarrow$ *propenso a errores*
  $\Rightarrow$ *difícil de mantener*
  $\Rightarrow$ *overhead de procesamiento*
- Poco dinámico, debido a:
  - no se adapta al entorno de ejecución

Características de Rainbow a Mejorar

- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.
- Duplicación en la configuración de invariantes
  $\Rightarrow$ *propenso a errores*
  $\Rightarrow$ *difícil de mantener*
  $\Rightarrow$ *overhead de procesamiento*
- Poco dinámico, debido a:
  - no se adapta al entorno de ejecución
  - configuración compleja

Características de Rainbow a Mejorar

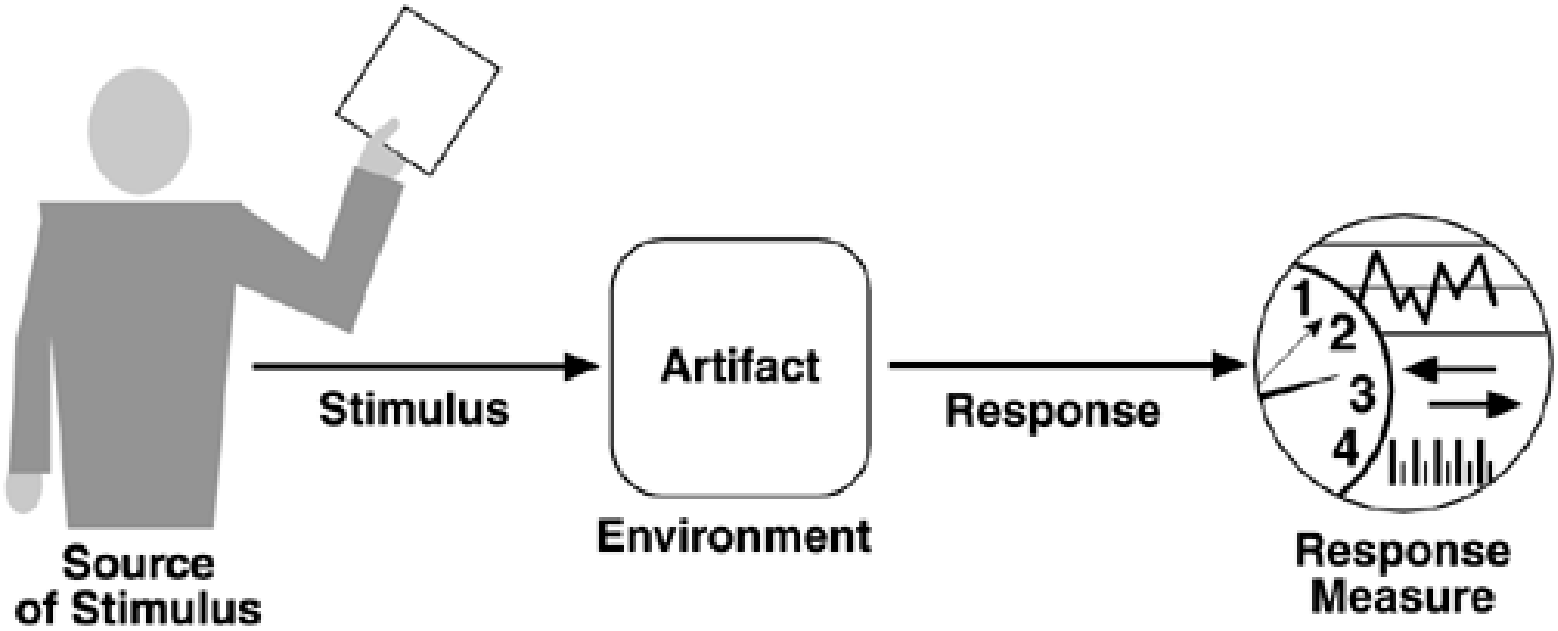
- No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.
- Duplicación en la configuración de invariantes
  $\Rightarrow$ *propenso a errores*
  $\Rightarrow$ *difícil de mantener*
  $\Rightarrow$ *overhead de procesamiento*
- Poco dinámico, debido a:
  - no se adapta al entorno de ejecución
  - configuración compleja
  - ausencia de mecanismo de actualización "en caliente"

Arco Iris

- *Framework* de Auto Reparación.
- Extensión de Rainbow implementando las mejoras antes mencionadas
- Incluye a los **Escenarios de Atributos de Calidad** para lograr un **modelo más flexible** de Auto Reparación.

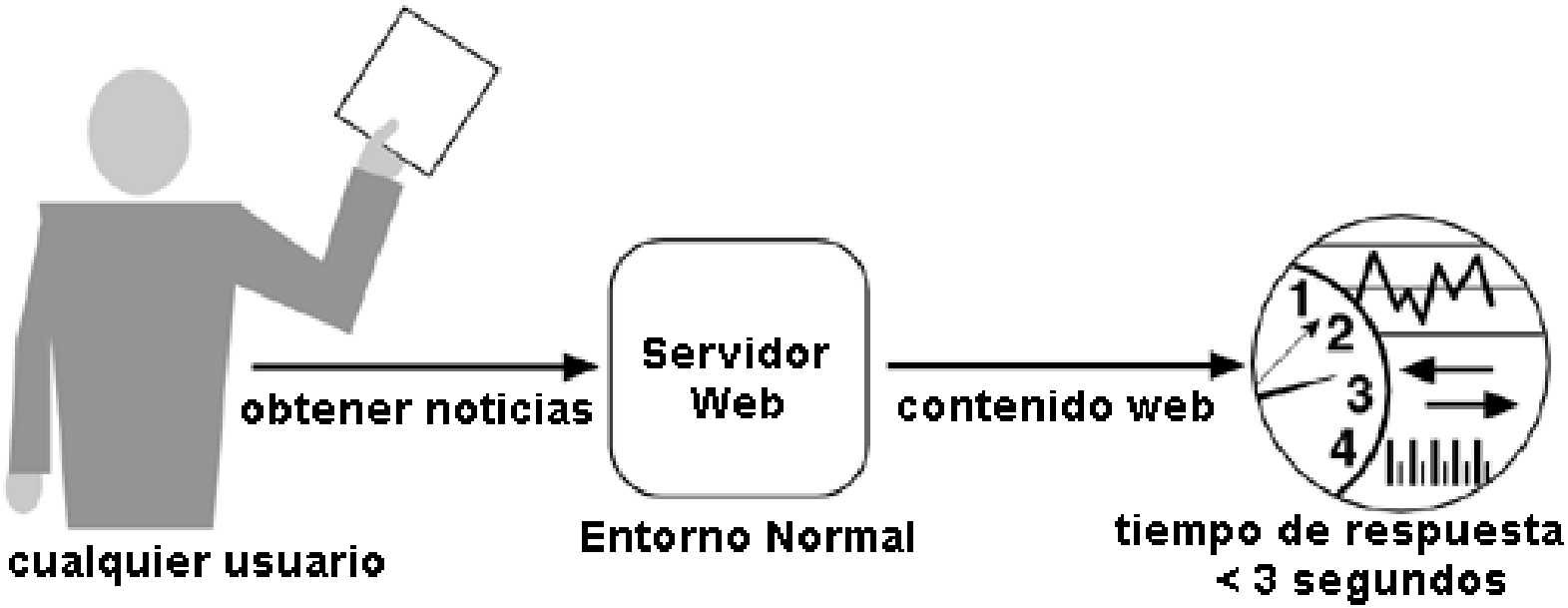
Escenarios de Atributos de Calidad

Los **Escenarios de Atributos de Calidad** o QAS (*Quality Attribute Scenarios*) permiten expresar cómo debería responder el sistema ante determinados estímulos y bajo ciertas circunstancias.



Escenarios de Atributos de Calidad(ejemplo)

Escenario de eficiencia



SHS: Extensión a QAS

Arco Iris define una extensión de los QAS denominada **Self Healing Scenario (SHS)**, agregando:

- Cada escenario tiene una prioridad determinada
- Estrategias que reparan al escenario
- Cada entorno define la importancia de los distintos *concerns*.

Por ejemplo, en un entorno de "Alta Carga":

$$\begin{aligned} \text{Tiempo de Respuesta} &= 60\,\% \\ \text{Fidelidad de la Información} &= 20\,\% \\ \text{Costo de Servidores} &= 20\,\% \end{aligned}$$

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.
- ~~Duplicación en la configuración de invariantes~~
  Invariantes sólo en los escenarios.

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.
- ~~Duplicación en la configuración de invariantes~~
  Invariantes sólo en los escenarios.
- ~~Poco dinámico, debido a:~~

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.
- ~~Duplicación en la configuración de invariantes~~
  Invariantes sólo en los escenarios.
- ~~Poco dinámico, debido a:~~
  - ~~no se adapta al entorno de ejecución~~
    Entornos dinámicos.

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.
- ~~Duplicación en la configuración de invariantes~~
  Invariantes sólo en los escenarios.
- ~~Poco dinámico, debido a:~~
  - ~~no se adapta al entorno de ejecución~~
    Entornos dinámicos.
  - ~~configuración compleja~~
    Configuración de escenarios mediante Arco Iris UI.

Arco Iris: Mejoras a Rainbow

- ~~No involucra a los *stakeholders* en el proceso de configuración de la auto reparación.~~
  Arco Iris los involucra.
- ~~Duplicación en la configuración de invariantes~~
  Invariantes sólo en los escenarios.
- ~~Poco dinámico, debido a:~~
  - ~~no se adapta al entorno de ejecución~~
    Entornos dinámicos.
  - ~~configuración compleja~~
    Configuración de escenarios mediante Arco Iris UI.
  - ~~ausencia de mecanismo de actualización "en caliente"~~
    Recarga de configuración dinámica

Arco Iris UI

Motivación principal

Facilitar el uso de Arco Iris a los distintos *stakeholders*.

Características Principales

- Herramienta de escritorio.
- Multi plataforma.
- Multi usuario.
- Lee y guarda configuración en XML.

¿Para qué una GUI?

En Rainbow se definen los invariantes en el modelo de la arquitectura:

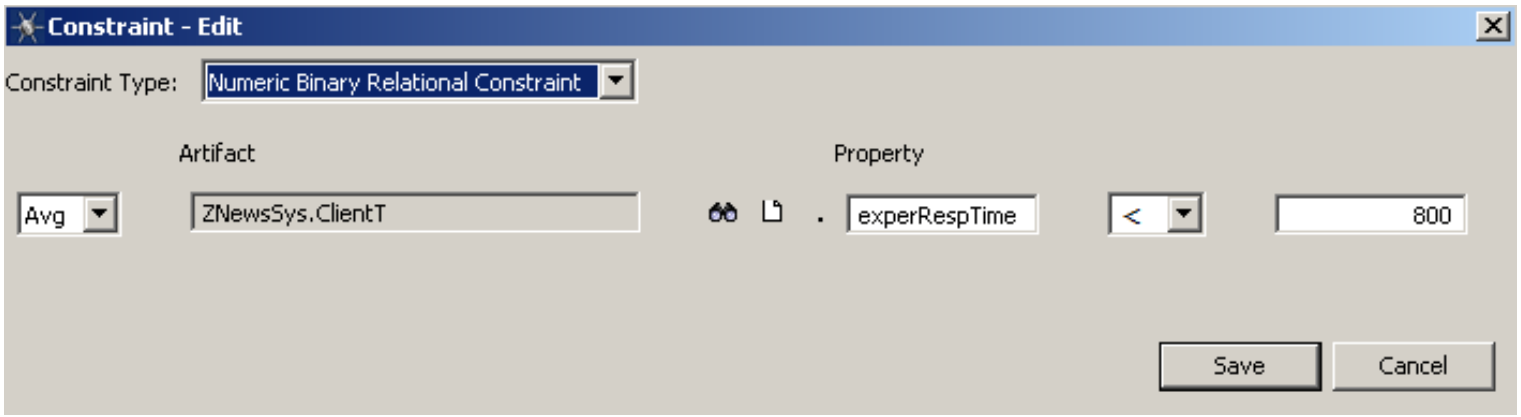
```
Property MAX_RESPTIME : float = 800.0;
...
Component Type ClientT = {
        Property experRespTime;
        rule primaryConstraint = invariant self.experRespTime <= MAX_RESPTIME;
}
```

Y se debe duplicar la lógica en la precondición de la estrategia:

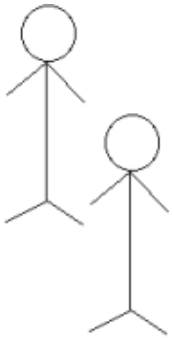
```
define boolean cViolation = exists c : T.ClientT in M.components | c.experRespTime > M.MAX_RESPTIME;
...
strategy SimpleReduceResponseTime
[ cViolation ] {
        t0: (cViolation) -> enlistServers(1) {
                t1: (!cViolation) -> done;
                t2: (default) -> TNULL;
        }
}
```

¿Para qué una GUI? (cont.)

En Arco Iris la definición de las restricciones de los escenarios puede ser llevada a cabo por cualquier *stakeholder*:

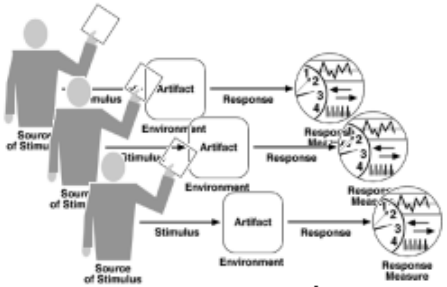


Flujo Conceptual



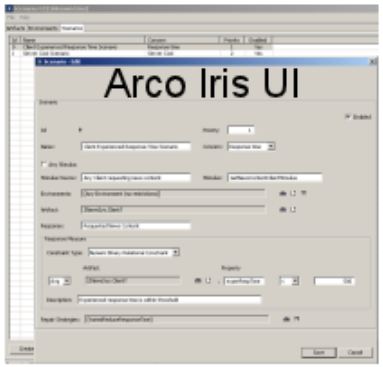
Stakeholders

crean



escenarios

mediante

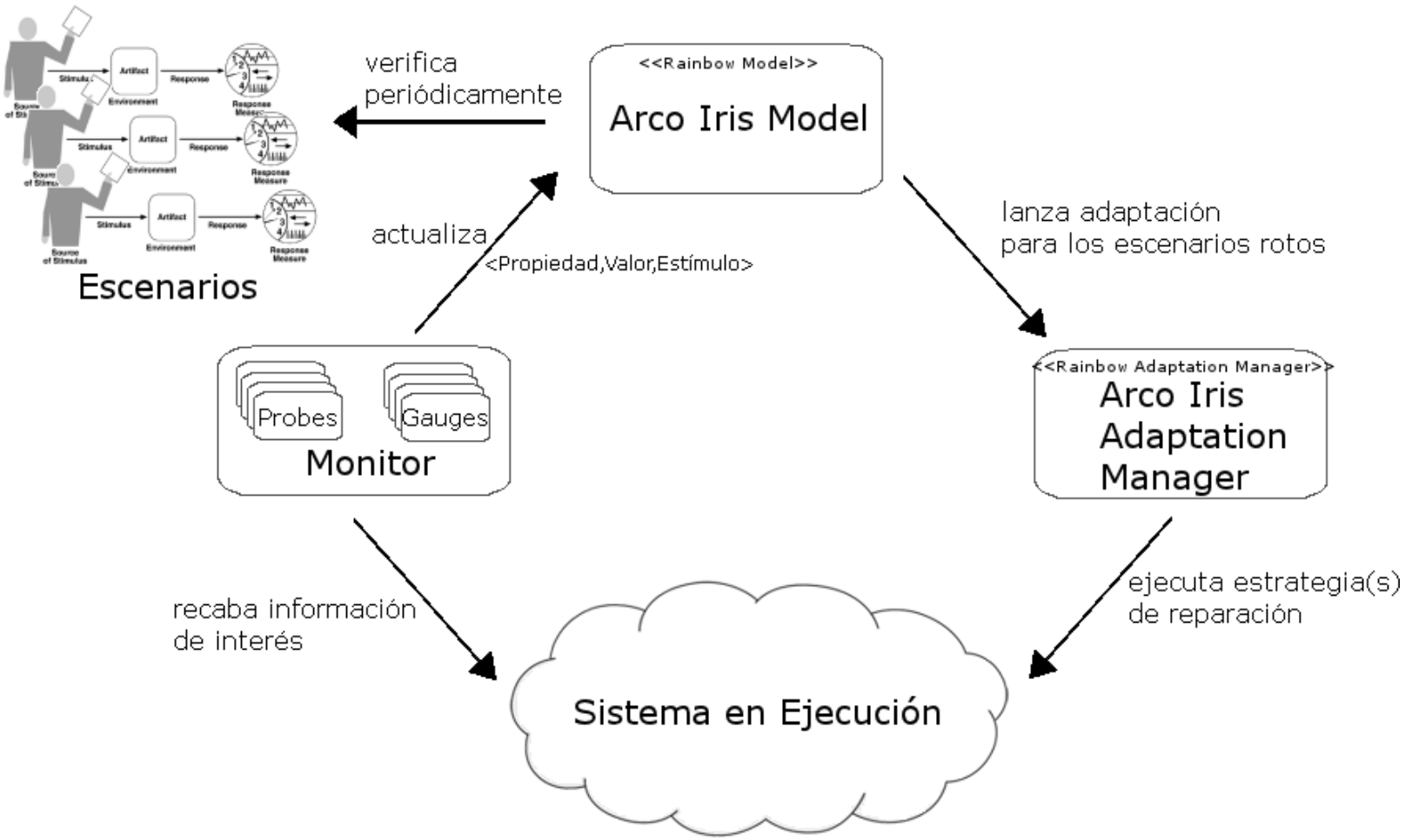


genera/actualiza

lee

```
<selfHealingConfiguration description="test">
  <artifacts>
    <artifact id="0" name="ClientT" systemName=
    <artifact id="1" name="ServerT" systemName=
  </artifacts>
  <environments>
    <environment id="0" name="NORMAL">
    ...
  <scenarios>
    <selfHealingScenario id="0" name="Client Ex
      <concern>RESPONSE_TIME</concern>
      <stimulus name="GetNewsContentClientStimu
      <environments>
    ...
</selfHealingConfiguration>
```

Arco Iris en Ejecución



Utilidad del Sistema

- Permite asignar una medida relativa de satisfacción (i.e. utilidad) sobre un sistema sobre el cual es necesario medir el impacto ante determinados cambios.
- 100 % de utilidad cuando:
  - no se viola ninguna restricción impuesta sobre el modelo (**Rainbow**)
  - todos los escenarios definidos se cumplen (**Arco Iris**)

Rainbow posee un mecanismo que permite predecir la utilidad del sistema luego de una supuesta ejecución de una estrategia.

Puntuación y Selección de una Estrategia

Puntuación: predicción de la utilidad del sistema luego de simular la ejecución de la estrategia.

```
scoreStrategia = 0
estimacionPorConcern = simular la aplicación de la estrategia como lo hace Rainbow y
                       obtener el valor resultante para los concerns
Por cada escenario habilitado
        escenarioSeSatisface = !escenario.applyFor(entornoActual) o
           (escenario.applyFor(entornoActual) y escenario.responseMeasure.holds(estimacionPorConcern))

        Si (escenarioSeSatisface)
                prioridadRelativa = calcular prioridad relativa del escenario
                pesoConcern = peso que el entorno actual asigna a escenario.concern
                puntajeEscenario = prioridadRelativa * pesoConcern
                scoreStrategia = scoreStrategia + puntajeEscenario
        Fin
Fin
```

Selección: se elige aquella estrategia que posea el puntaje máximo.

ZNN

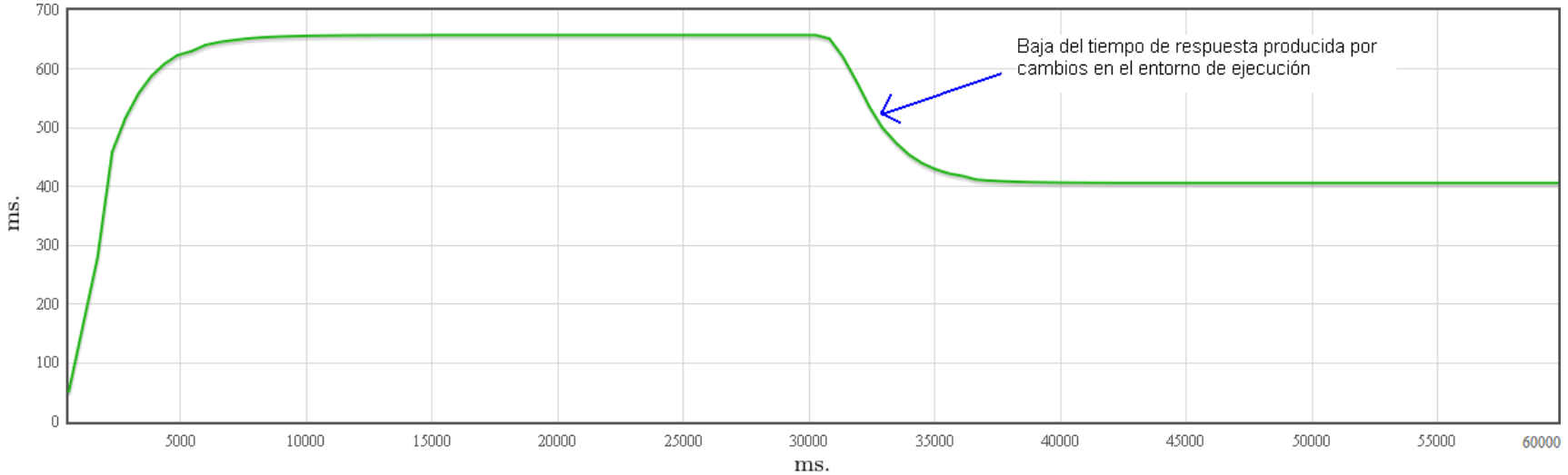
Para los casos de prueba se utilizó **ZNN**, un sistema que **simula un sitio web de noticias** presentado en una tesis de doctorado de la universidad de Carnegie Mellon.

ZNN **propone un marco para evaluar soluciones de Auto Adaptación**, evaluando concretamente a Rainbow.

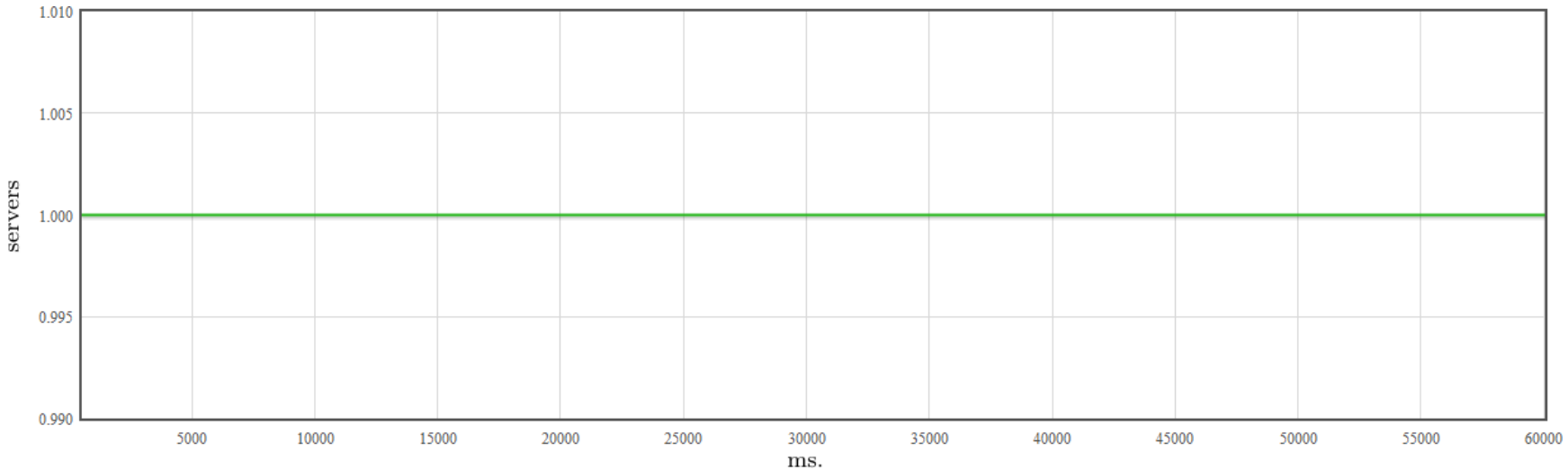
Utiliza una simulación de una **arquitectura cliente-servidor**.

Comportamiento del Sistema sin Auto Reparación

Tiempo de respuesta



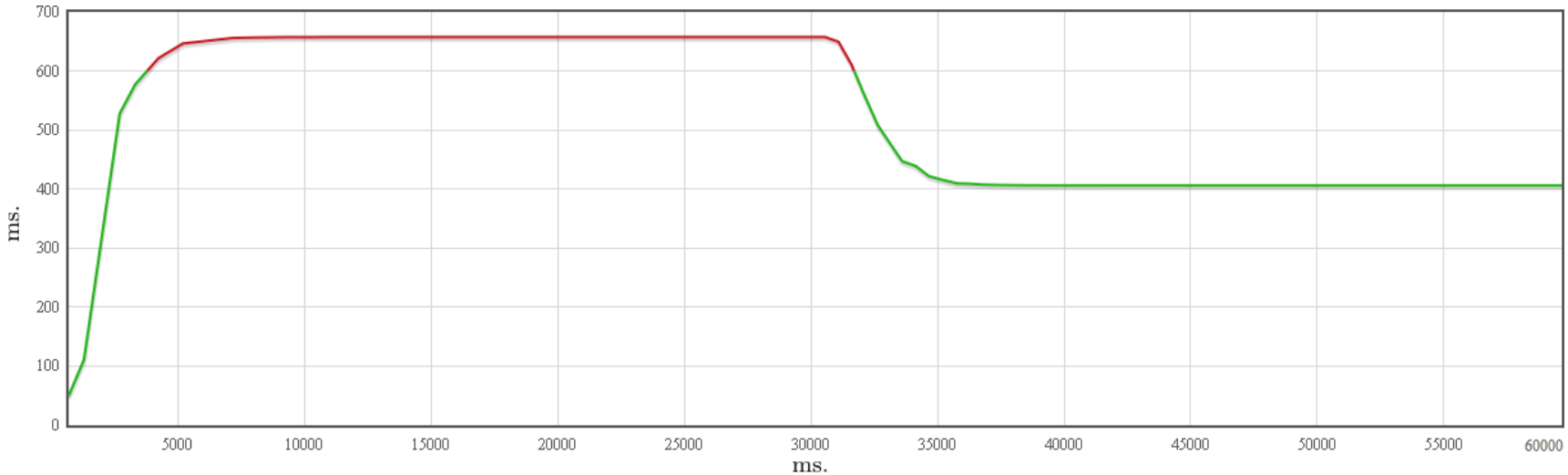
Costo de servidores



Comportamiento con un Escenario, Sin Estrategias

Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	–

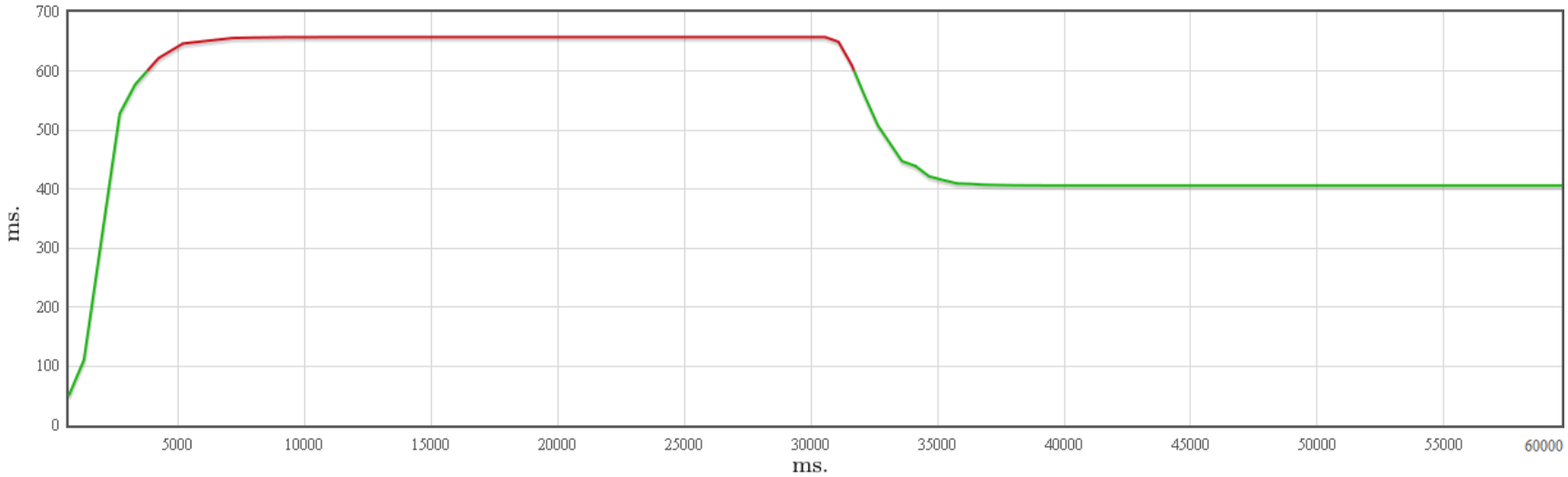
Tiempo de respuesta



Comportamiento con un Escenario, Sin Estrategias

Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	–

Tiempo de respuesta

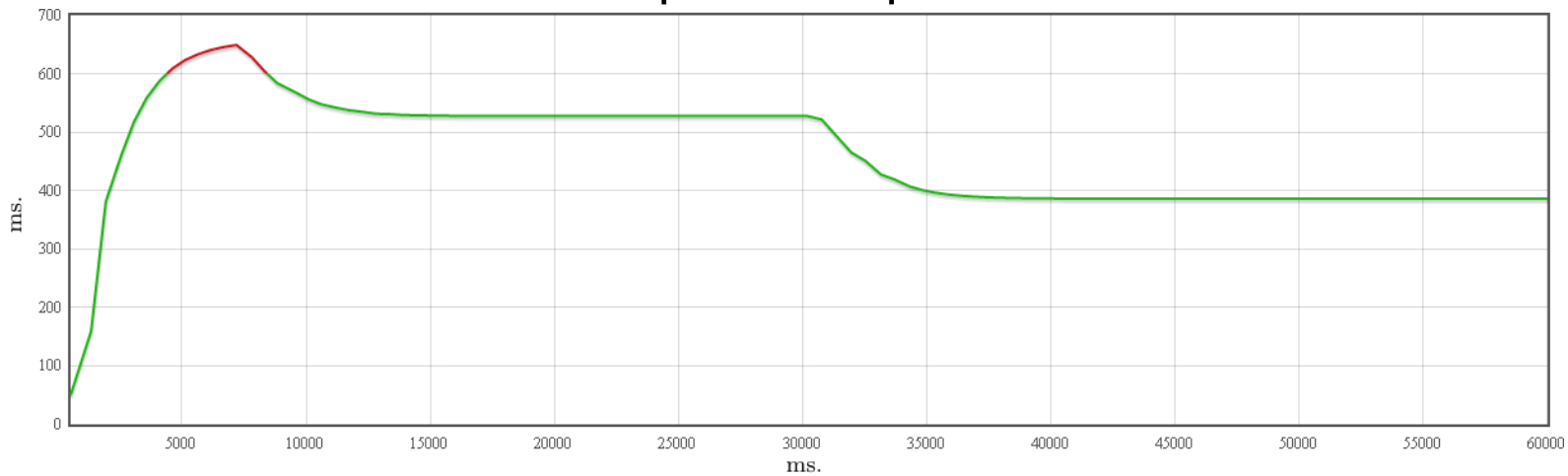


¡Arco Iris no puede reparar el sistema porque el escenario no tiene estrategias asociadas!

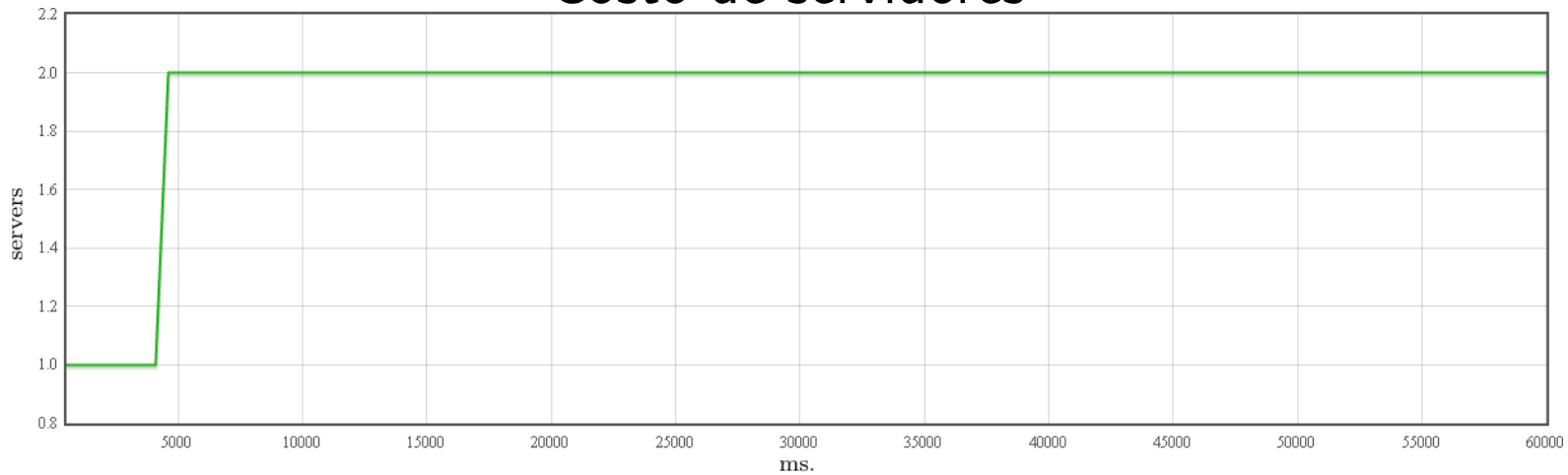
Comportamiento con un Escenario y una Estrategia

Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	EnlistServerResponseTime

Tiempo de respuesta



Costo de servidores

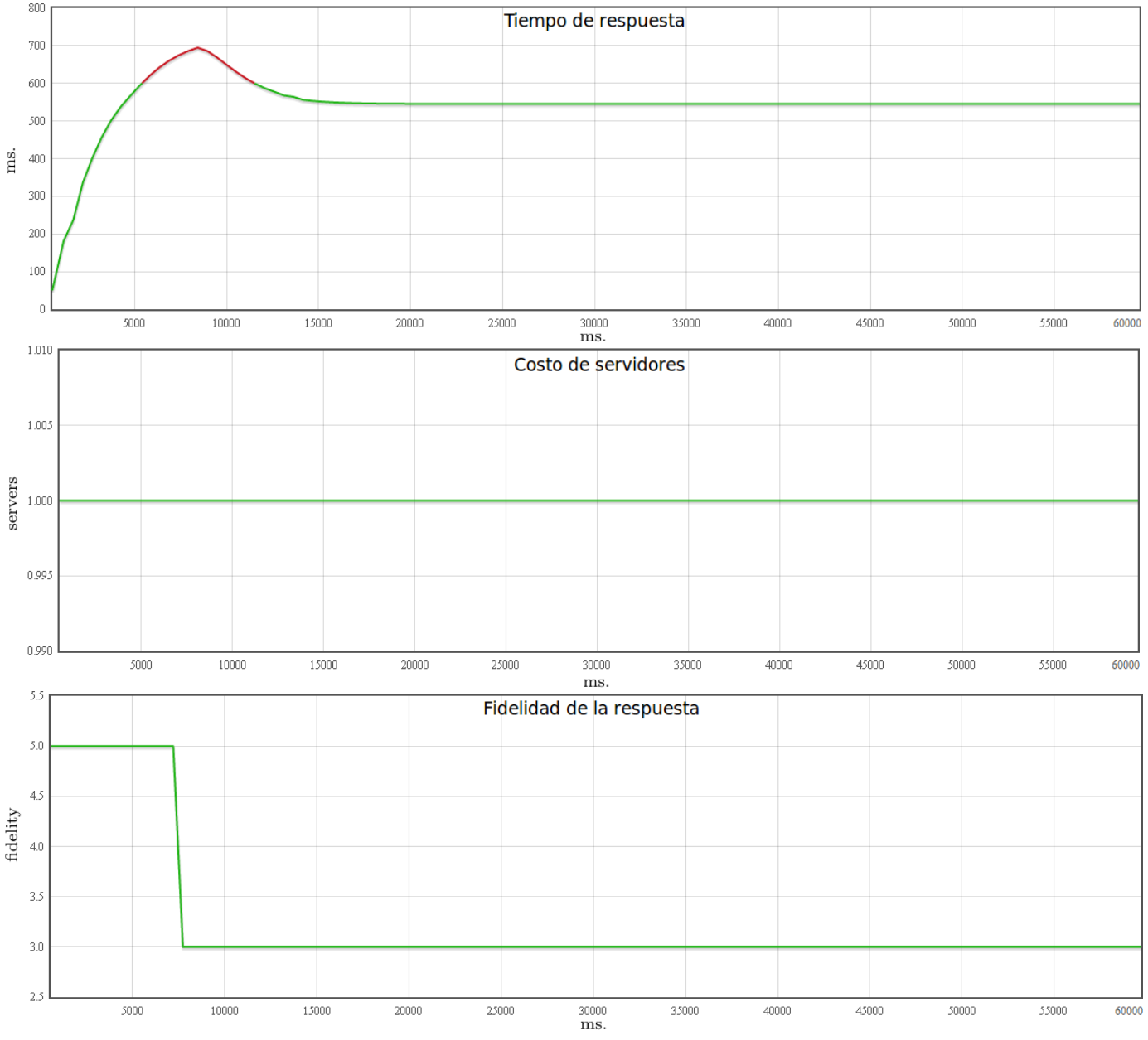


Tradeoff Entre Estrategias

Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	EnlistServerResponseTime, LowerFidelityReduceResponseTime
2	Costo de Servidores	Cualquier Entorno aplica	costo $<$ 2	–

```
strategy LowerFidelityReduceResponseTime {
  t0: (true) -> lowerFidelity(2, 100) @[5000 /*ms*/] {
    t1: (!RESP_TIME_STILL_BROKEN) -> done;
    t2: (RESP_TIME_STILL_BROKEN) -> lowerFidelity(2, 100) @[8000 /*ms*/] {
      t2a: (!RESP_TIME_STILL_BROKEN) -> done;
      t2b: (default) -> TNULL;  // in this case, we have no more steps to take
    }
  }
}
```

Tradeoff Entre Estrategias (cont.)

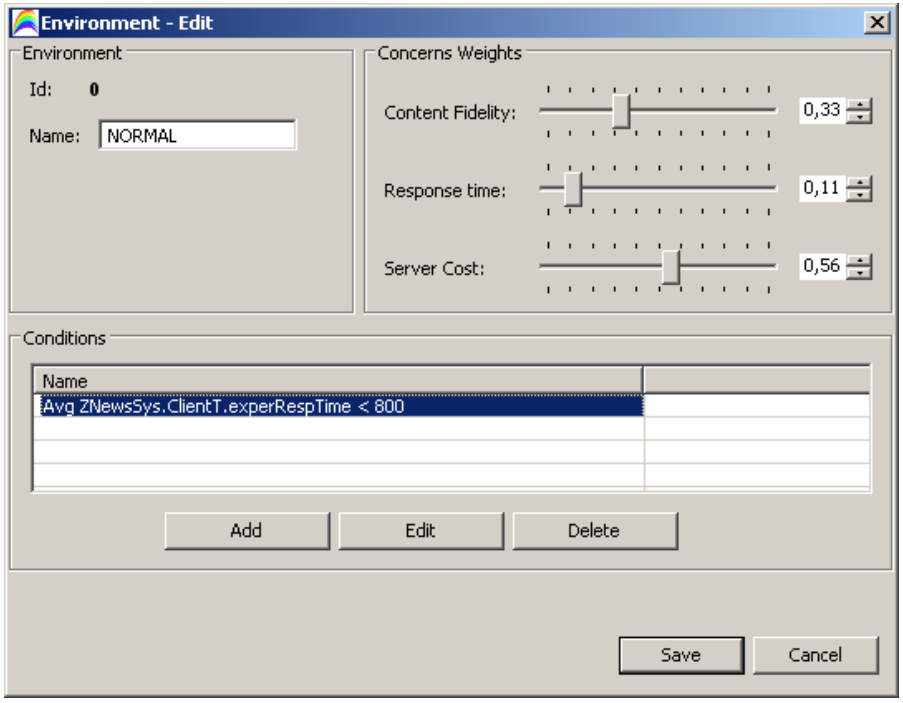


Tradeoff según Prioridades

Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	EnlistServerResponseTime
2	Costo de Servidores	Cualquier Entorno aplica	costo $<$ 2	ReduceOverallCost

Tradeoff entre Escenarios Según Concerns

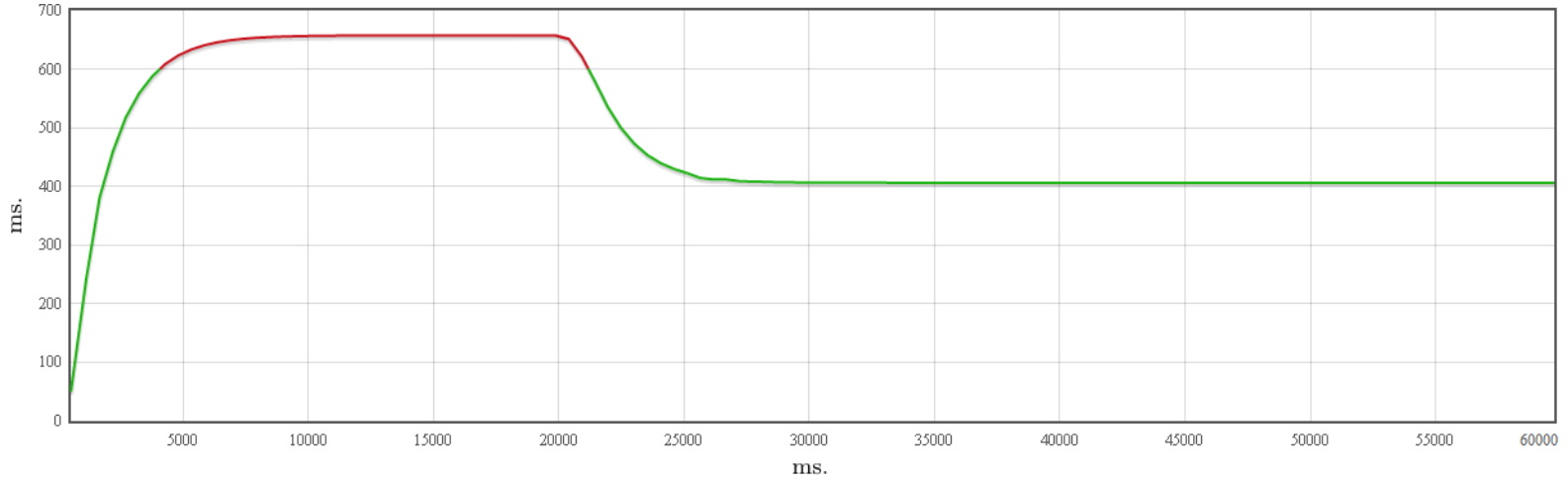
Prioridad	*Concern*	Entorno	Cuantificación Respuesta	Estrategias
1	Tiempo de Respuesta	Cualquier Entorno aplica	tiempo de respuesta $<$ 600 ms.	EnlistServerResponseTime
1	Costo de Servidores	Cualquier Entorno aplica	costo $<$ 2	–



Tradeoff entre Escenarios Según Concerns (cont.)

```
Current environment: NORMAL
Computing Current System Utility...
Server Cost Scenario NOT BROKEN for [ESum] 1.0
Experienced Response Time Scenario BROKEN for [EAvg] 602.25
Current System Utility (Score to improve): 0.555
Evaluating strategy EnlistServerResponseTime...
  Scoring EnlistServerResponseTime...
  Server Cost Scenario BROKEN after simulation for Server Cost ([ESum] 2.0)
  Experienced Response Time Scenario NOT BROKEN after simulation for Response time ([EAvg] 458.25)
  Score for strategy EnlistServerResponseTime: 0.111
NO applicable strategy, adaptation cycle ended.
```

Tiempo de respuesta



Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).

Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).
- Extensión de tipos de restricciones soportadas.

Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).
- Extensión de tipos de restricciones soportadas.
- Atributos de Calidad y *Concerns* configurables por el usuario.

Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).
- Extensión de tipos de restricciones soportadas.
- Atributos de Calidad y *Concerns* configurables por el usuario.
- Mejorar heurísticas de predicción del potencial impacto de estrategias.

Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).
- Extensión de tipos de restricciones soportadas.
- Atributos de Calidad y *Concerns* configurables por el usuario.
- Mejorar heurísticas de predicción del potencial impacto de estrategias.
- ¿Ausencia u obsolescencia del modelo de la arquitectura?

Trabajo a Futuro

- Ampliación de la recarga dinámica de configuración (idealmente: nunca detener Arco Iris para reconfigurar).
- Extensión de tipos de restricciones soportadas.
- Atributos de Calidad y *Concerns* configurables por el usuario.
- Mejorar heurísticas de predicción del potencial impacto de estrategias.
- ¿Ausencia u obsolescencia del modelo de la arquitectura?
  - `http://www.cs.cmu.edu/~able/research/discotect.html`

Conclusiones

- *Stakeholders* con mayor participación en el proceso de Auto Reparación:
  - En mayor parte, Arco Iris no requiere habilidades técnicas para su configuración
  - Información sobre restricciones fuera del modelo de la arquitectura y unificadas
  - Arco Iris UI
- Dinamismo agregado:
  - Recarga automática de configuración
  - Adaptación dinámica de soluciones de acuerdo al entorno de ejecución
- El sistema conociendo sus requerimientos vs. implementación de soluciones *ad hoc*

¿Preguntas?

“Dígame Licenciado”