

Predicting Timing Failures in Web Services

Nuno Laranjeiro, Marco Vieira, Henrique Madeira

CISUC – Department of Informatics Engineering



**University of Coimbra
Portugal**

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

Outline



- The problem
- Approach goals and design
- Detection mechanism
- Prediction mechanism
- Usage mode
- Experimental evaluation
- Conclusion

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

2

The problem (1)



- Business critical environments
 - Rely on **SOA** standards (**web services**)
 - **Time-critical** services are executed
 - Web services → Wide-area or open environments
 - High or highly variable execution times ← difficult to deal with timing requirements

The problem (2)



- Late result can be **useless** to clients and adds **extra load** to servers
 - Can put a client unnecessarily on hold
 - Uses server resources for an inconsequent operation
- Services frequently deal with deadline violations (ad-hoc solutions)
- **Programming models lack mechanisms to easily assure timing failures detection and prediction**

The goal



- Practical way to deploy time-aware services
 - Programming model
 - Accurate timing failures detection and prediction
 - Bytecode instrumentation approach
- Low coding effort approach for creating new services
- Minimal changes in legacy services ← useful for providers

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

5

wsTFDP: quality properties



- Effective
 - Low **detection latency** (under 100ms)
 - Low **false-positive** and **false-negative** rate (under 5%)
- Non intrusive
 - Low performance **overhead** (under 100 ms)
- Easy to use
 - Minimal **changes** to current models or legacy code
- Generic
 - **Extensible** to other environments / technologies

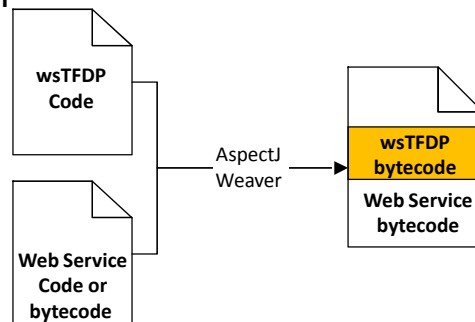
Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

6

wsTFDP: supporting technology



- wsTFDP is an isolated Java application
- Integration is supported by AOP
- AspectJ's compiler weaves wsTFDP into any application



Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

7

wsTFDP: mechanism's basics



- 2 basic components: **detection & prediction**
- AOP enables **cross-cutting concerns injection** in a non-intrusive way
- The same mechanism can be **transparently** used in multiple points

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

8

wsTFDP: AOP

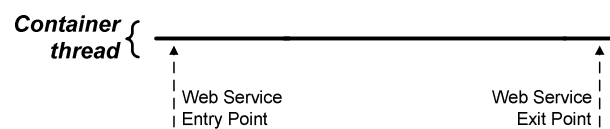


- wsTFDP uses a predicate to **intercept**:
 - Web service operations that have a *TimeRestriction* parameter [*@WebMethod*]
 - Any existing external service invocations [*@WebMethod*]
- There is also a predicate to **intercept** or **ignore** particular code points marked by the developer
- We use an around advice (control before and after interception points)

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

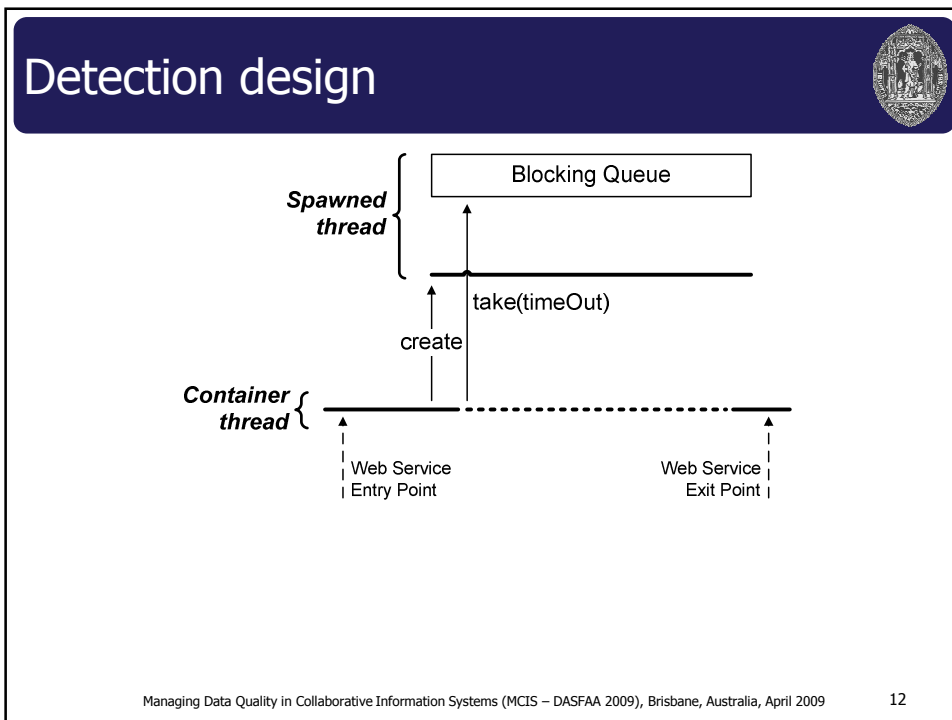
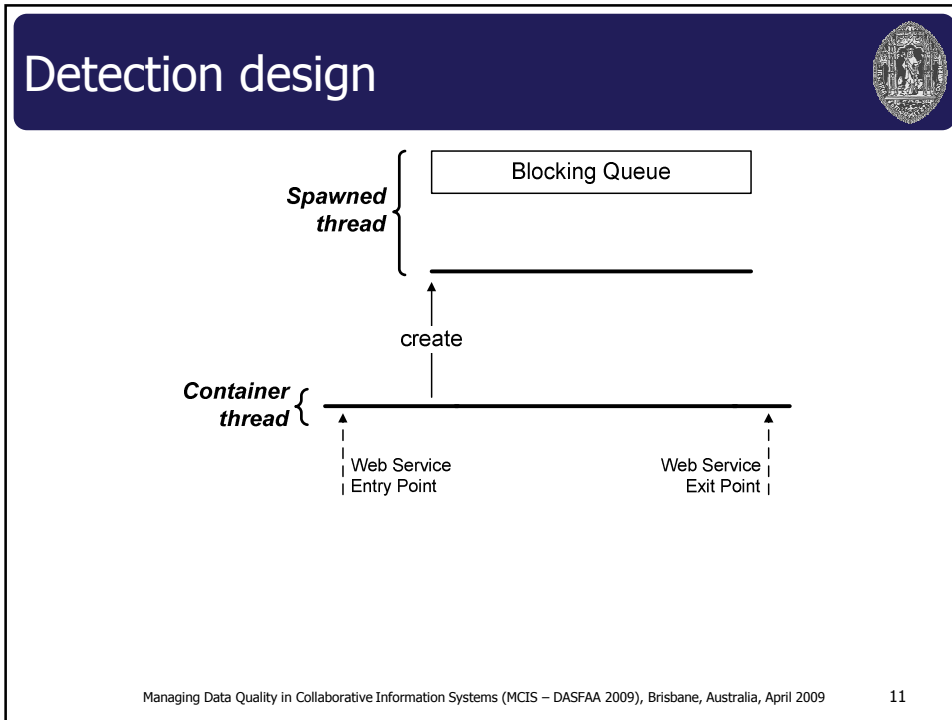
9

Detection design

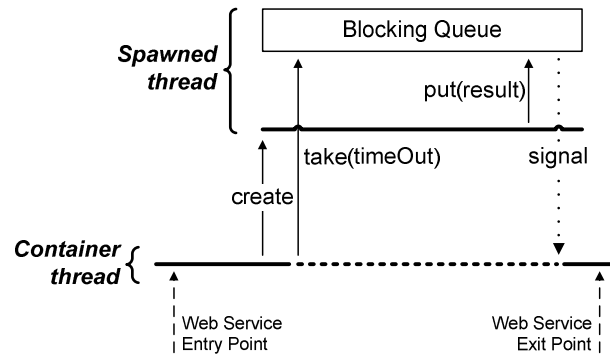


Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

10



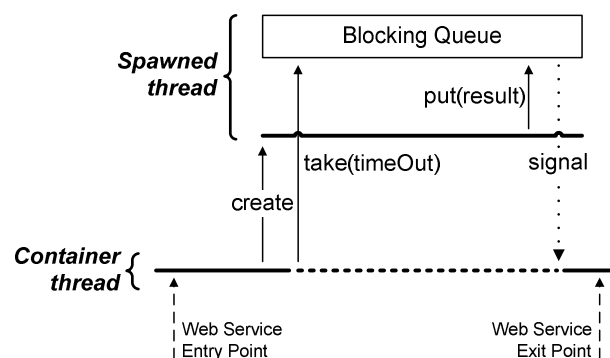
Detection design



Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

13

Detection design



- Parent thread waits for:
 - Signal to remove object (regular or exception)
 - Depletion of the waiting time

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

14

Prediction steps



- 1) Build a runtime **graph** of the service's logical structure
 - 2) Gather runtime time-related performance **metrics**
 - 3) Use historical data to **predict** if a service can conclude on time
- **Graph structure:**
 - Nodes: Candidate points for failure prediction
 - Edges: Connections between nodes
 - Cost: Time for travelling between nodes

Graph structure



```
@WebMethod public int
buildInt()
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }

    result += invokeServiceD();

    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Graph structure



```
@WebMethod public int
buildInt()
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }

    result += invokeServiceD();

    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Default:

- 5 prediction points
(entry point + each service call)

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

17

Graph structure



```
@WebMethod public int
buildInt()
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }

    result += invokeServiceD();

    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Default:

- 5 prediction points
(entry point + each service call)

Goal:

- Ignore last service call
- Predict before DB query

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

18

Graph structure



```
@WebMethod public int
buildInt(TimeRestriction restrict)
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }

    result += invokeServiceD();

    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Default:

- 5 prediction points
(entry point + each service call)

Goal:

- Ignore last service call
- Predict before DB query

Graph structure



```
@WebMethod public int
buildInt(TimeRestriction restrict)
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }

    TimeChecker.ignore();
    result += invokeServiceD();
    TimeChecker.check();
    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Default:

- 5 prediction points
(entry point + each service call)

Goal:

- Ignore last service call
- Predict before DB query

Graph structure



```
@WebMethod public int
buildInt(TimeRestriction restrict)
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }
    TimeChecker.ignore();
    result += invokeServiceD();
    TimeChecker.check();
    result = queryDataBase();

    return result;
}
```

Structure:

- 4 service calls
- 1 database query

Default:

- 5 prediction points
(entry point + each service call)

Goal:

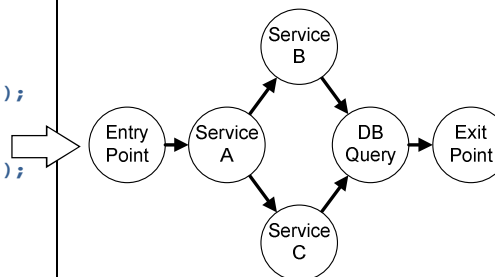
- Ignore last service call
- Predict before DB query

Graph structure



```
@WebMethod public int
buildInt(TimeRestriction restrict)
{
    int result = 0;
    result += invokeServiceA();
    if (result == 0)
    {
        result += invokeServiceB();
    } else
    {
        result += invokeServiceC();
    }
    TimeChecker.ignore();
    result += invokeServiceD();
    TimeChecker.check();
    result = queryDataBase();

    return result;
}
```



Prediction process



- Uses **historical edge data**
- Uses a **pessimist approach** to predict if service can conclude on desired time
- Uses **dijkstra's** shortest path computation algorithm
- At each prediction point:
 - Finds out the fastest way to the exit point
 - Calculates if the fastest way is more than the available execution time (chosen by the client)
- Service can be **aborted** or continue under a **degraded mode**

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

23

Metrics management



- Sorted list of historical data on each edge
- **Element removal strategy**
 - **Longest** duration value (higher accuracy)
 - **Random** removal (lower accuracy, but may capture typical behavior better)
- **Client-set confidence**
 - Used to discard a % of the lowest execution times
 - 95% confidence → 5% of the fastest executions happened in abnormal conditions

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

24

Usage mode (server-side)



- Add wsTFDP library to your project
- Add a *TimeRestriction* parameter to desired web service operations
- Optionally choose or ignore prediction points
- If using Maven as build tool, issue *mvn package* to compile and package your web service application

Usage mode (client-side)



- Add wsTFDP library to your project
- Invoke the target service
 - Specify desired duration
 - Specify a confidence value

```
NewCustomerInput myServiceInput = new NewCustomerInput();
TimeRestriction timeRestriction = new TimeRestriction();
timeRestriction.setTimeInMillis(1000L);
timeRestriction.setConfidenceValue(0.95D);
NewCustomer proxy = new
NewCustomerService().getNewCustomerPort();
proxy.newCustomer(timeRestriction, myServiceInput);
```

Experimental evaluation



- Does the mechanism introduce a noticeable **delay** in services?
- **How fast** can the mechanism detect failures and how does this evolve under higher loads?
- Is it able to provide **low false-positive and false-negative rates** during the detection and prediction process?
- Can developers **easily use** the mechanism?

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

27

Experimental setup



- Set of TPC-App services (performance benchmark)
- We added a variable delay (1-2 sec) to an existing external service call
- 12 different tests x 3 repetitions (20 min each test)
- State restored between each experiment

Node	Software	Hardware
Server	Windows server 2003 R2 Enterprise x64 Edition service pack 2 & JBoss 4.2.2.GA	Dual Core Pentium 4 3Ghz 1.46GB RAM
DBMS	Windows server 2003 R2 Enterprise x64 Edition service pack 2 & Oracle 10g	Quad Core Intel Xeon 5300 Series 7.84 GB RAM
Client	Windows XP pro SP2	Dual Core Pentium 4, 3GHz, 1.96GB RAM

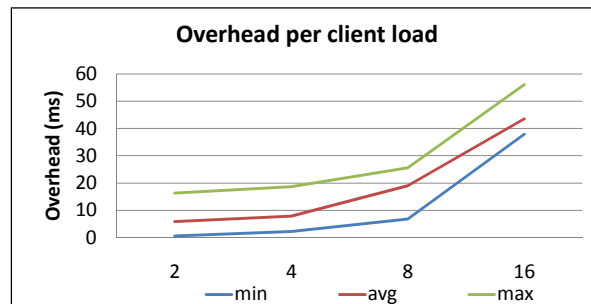
Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

28

Results: performance overhead



- Baseline server application **Vs** wsTFDP-enabled application
- 2, 4, 8, and 16 emulated clients (3 repetitions)
- High constant value for service deadline



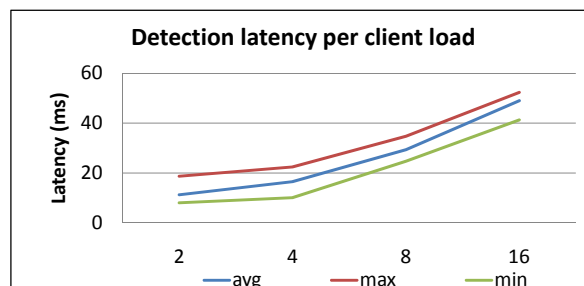
Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

29

Results: detection latency



- Difference between expected and observed time.
- Extra load (2 threads in an infinite loop)
- Client-side variable duration
 - Minimum to 2x maximum observed historic values



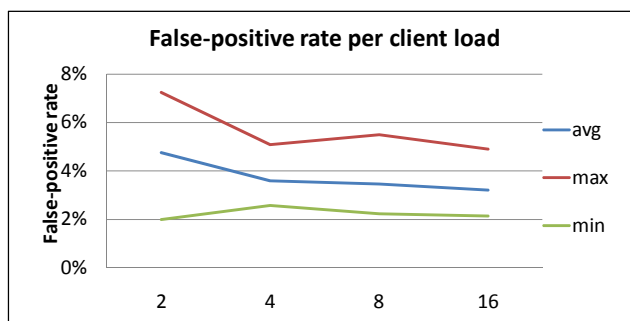
Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

30

Results: false-positive rate



- Target: low false-positive rate (5%)
- Moderate configuration tuning



Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

31

Results: false-negative rate & integration procedure



- No configuration tuning → 26% false-negative rate
- Integration by a developer:
 - We used an existing TPC-App implementation
 - Main developer tasks:
 - Merging the application's project descriptor (use ajc)
 - Adding a *TimeRestriction* parameter to each service
 - Total duration of about 10 minutes

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

32

Conclusion



- Business operations use **SOA** and are **time critical**
- Service programming models lack mechanisms to easily assure temporal properties
- Practical way to deploy time-aware services
- wsTFDP provides:
 - Low false-positive/negative rates
 - Low performance impact in our first prototype
 - Generic and easy to use
- Future Work
 - Study and integrate more complex prediction algorithms
 - Decrease the amount of wsTFDP specific code

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009

33

<http://cisuc.dei.uc.pt>



?

Nuno Laranjeiro, Marco Vieira, Henrique Madeira
{**cnl**, mvieira, henrique}@dei.uc.pt

Managing Data Quality in Collaborative Information Systems (MCIS – DASFAA 2009), Brisbane, Australia, April 2009