An Extensible Architecture for Run-time Monitoring of Conversational Web Services

Konstantinos Bratanis, Dimitris Dranidis, Anthony J.H. Simons

South East European Research Centre (SEERC)
Research Centre of the University of Sheffield and CITY College
Thessaloniki, Greece
kobratanis@seerc.org, dranidis@city.academic.gr

Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
a.simons@dcs.shef.ac.uk

MONA+ 2010 - Cyprus - December 1, 2010

Outline

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Outline

Outline

Outline

Monitoring Conversation Web Services

- Even if a Web service was fault-free during testing, it could deviate during run-time, since its context of execution is subject to continuous change.
- A service provider could modify a Web service without prior notifying all consumers, or a Web service could be substituted within a composite Web service.
- Monitoring is the primary trigger for adaptation in service-based applications.
- Conversational Web services introduce added complexity when it comes to monitoring (conversational protocol / state per consumer).

Monitoring Conversation Web Services

- Even if a Web service was fault-free during testing, it could deviate during run-time, since its context of execution is subject to continuous change.
- A service provider could modify a Web service without prior notifying all consumers, or a Web service could be substituted within a composite Web service.
- Monitoring is the primary trigger for adaptation in service-based applications.
- Conversational Web services introduce added complexity when it comes to monitoring (conversational protocol / state per consumer).

The University Of Sheffield. CITY College An International Faculty Of The University. SOUTH-EAST EUROPEAN RESEARCH CENTRE SEERC

Monitoring Conversation Web Services

- Even if a Web service was fault-free during testing, it could deviate during run-time, since its context of execution is subject to continuous change.
- A service provider could modify a Web service without prior notifying all consumers, or a Web service could be substituted within a composite Web service.
- Monitoring is the primary trigger for adaptation in service-based applications.
- Conversational Web services introduce added complexity when it comes to monitoring (conversational protocol / state per consumer).


The University Of Sheffield.


CITY College
An International Faculty Of The University.


SOUTH-EAST EUROPEAN RESEARCH CENTRE

Monitoring Conversation Web Services

- Even if a Web service was fault-free during testing, it could deviate during run-time, since its context of execution is subject to continuous change.
- A service provider could modify a Web service without prior notifying all consumers, or a Web service could be substituted within a composite Web service.
- Monitoring is the primary trigger for adaptation in service-based applications.
- Conversational Web services introduce added complexity when it comes to monitoring (conversational protocol / state per consumer).

Motivation for an Extensible Monitoring Architecture

- Cross-layer adaptation requires different monitors at different layers to trigger adaptation.
- Correlation of cross-layer monitors could facilitate more efficient and effective adaptation.
- Numerous approaches for monitoring Web services exist in literature and they focus on a particular layer.

Motivation for an Extensible Monitoring Architecture

- Cross-layer adaptation requires different monitors at different layers to trigger adaptation.
- Correlation of cross-layer monitors could facilitate more efficient and effective adaptation.
- Numerous approaches for monitoring Web services exist in literature and they focus on a particular layer.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Motivation for an Extensible Monitoring Architecture

- Cross-layer adaptation requires different monitors at different layers to trigger adaptation.
- Correlation of cross-layer monitors could facilitate more efficient and effective adaptation.
- Numerous approaches for monitoring Web services exist in literature and they focus on a particular layer.

Logical Classification of Monitors

- We have identified two approaches for constructing monitors:
  1. *Heavy-weight* monitor: A single monitor supports the monitoring of different aspects of a Web service.
  2. *Light-weight* monitor: A single monitor supports the monitoring of one aspect of a Web service. Several light-weight monitors can be used to monitor diversified aspects of a service.

Logical Classification of Monitors

- We have identified two approaches for constructing monitors:
  1. *Heavy-weight* monitor: A single monitor supports the monitoring of different aspects of a Web service.
  2. *Light-weight* monitor: A single monitor supports the monitoring of one aspect of a Web service. Several light-weight monitors can be used to monitor diversified aspects of a service.

Logical Classification of Monitors

- We have identified two approaches for constructing monitors:
  1. *Heavy-weight* monitor: A single monitor supports the monitoring of different aspects of a Web service.
  2. *Light-weight* monitor: A single monitor supports the monitoring of one aspect of a Web service. Several light-weight monitors can be used to monitor diversified aspects of a service.

Monitoring Scalability

- We have identified two approaches for scaling monitoring capabilities:
  1. A single service acting as a message gateway for forwarding all requests/responses of the monitored services to specific monitors, which can be added and removed at run-time.
  2. A pool of different individual monitor services that are being attached to the monitored services at run-time.

Monitoring Scalability

- We have identified two approaches for scaling monitoring capabilities:
  1. A single service acting as a message gateway for forwarding all requests/responses of the monitored services to specific monitors, which can be added and removed at run-time.
  2. A pool of different individual monitor services that are being attached to the monitored services at run-time.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE SEERC

Monitoring Scalability

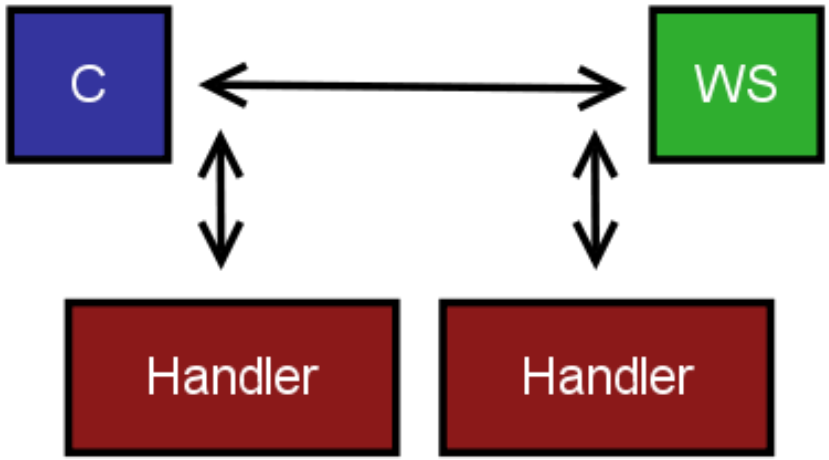
- We have identified two approaches for scaling monitoring capabilities:
  1. A single service acting as a message gateway for forwarding all requests/responses of the monitored services to specific monitors, which can be added and removed at run-time.
  2. A pool of different individual monitor services that are being attached to the monitored services at run-time.

Message Interception Techniques

1. Handler-Based Interception
2. Wrapper-Based Interception
3. Proxy-Based Interception

Handler-Based Interception

- A handler is attached to the monitored service and/or to the consumer.
- The request/response messages are processed first by the handler.

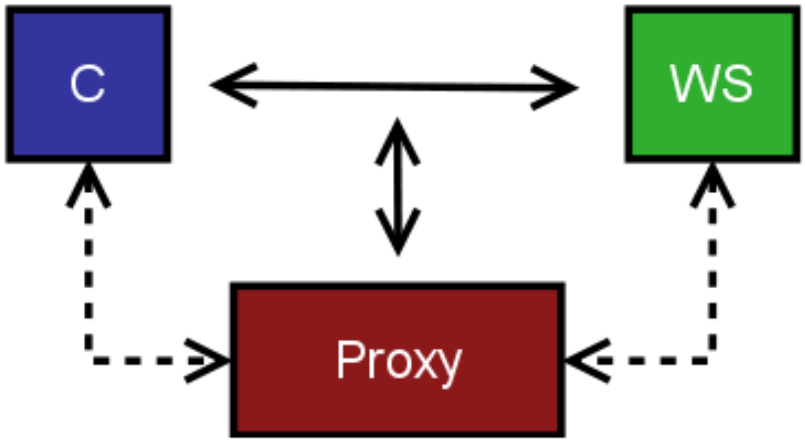


Wrapper-Based Interception

- The monitored service is wrapped within another service.
- The wrapper mediates between the service and the consumer.

Proxy-Based Interception

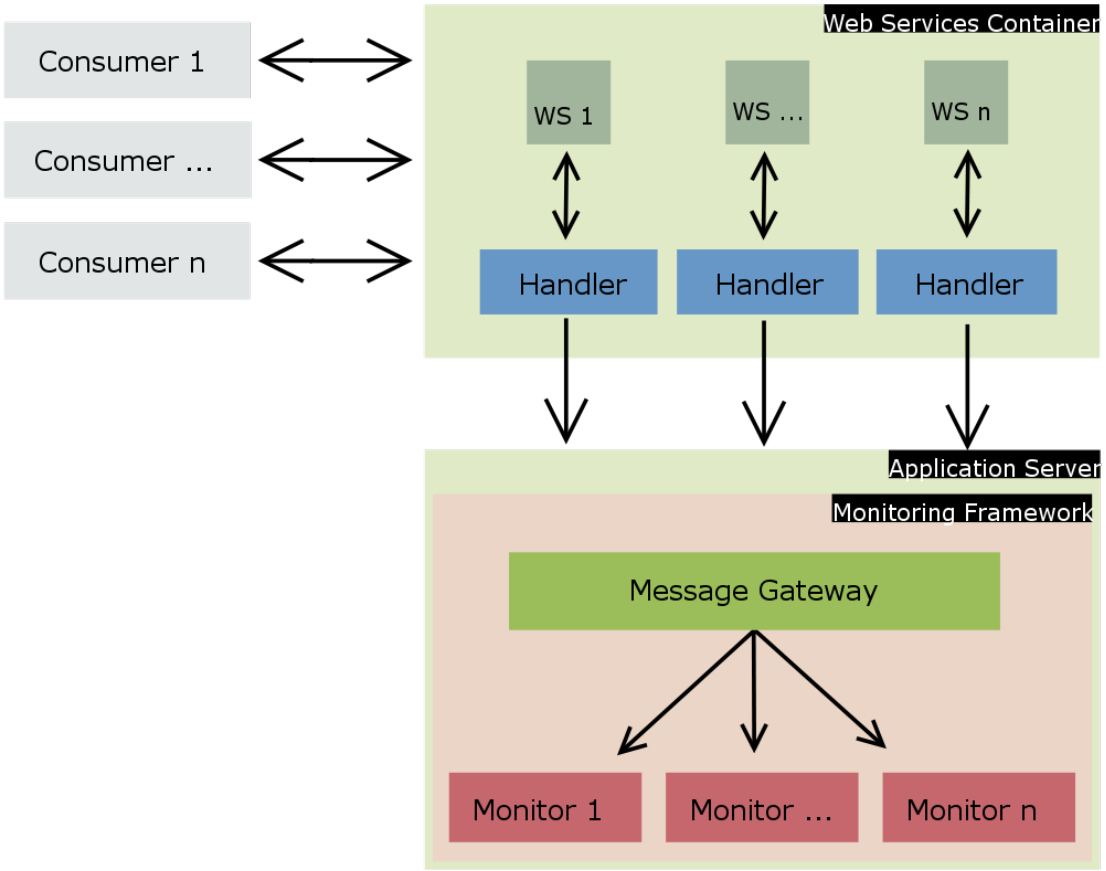
- An intermediate node acts as a network proxy.
- It transparently intercepts the messages.



Monitoring Session Handling

- For each instance of the monitored Web service, a new monitoring session is created.
- The monitoring session is uniquely identified by a monitoring session identifier (MSID).

The Monitoring Architecture



The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Integration of a Behavioural Conformance Monitor

- JSXM is a model-based testing tool which uses the Stream X-Machine (SXM) formalism, in order to generate test cases and animate an SXM model.
- The functionality of JSXM was exposed through an API and it was integrated in the architecture as a monitor.
- The JSXM-based monitor is able to check the behavioural conformance of a Web service using an SXM model as an oracle.


The University Of Sheffield.


CITY College An International Faculty Of The University.


SOUTH-EAST EUROPEAN RESEARCH CENTRE

Integration of a Behavioural Conformance Monitor

- JSXM is a model-based testing tool which uses the Stream X-Machine (SXM) formalism, in order to generate test cases and animate an SXM model.
- The functionality of JSXM was exposed through an API and it was integrated in the architecture as a monitor.
- The JSXM-based monitor is able to check the behavioural conformance of a Web service using an SXM model as an oracle.

Integration of a Behavioural Conformance Monitor

- JSXM is a model-based testing tool which uses the Stream X-Machine (SXM) formalism, in order to generate test cases and animate an SXM model.
- The functionality of JSXM was exposed through an API and it was integrated in the architecture as a monitor.
- The JSXM-based monitor is able to check the behavioural conformance of a Web service using an SXM model as an oracle.

Integration of a Behavioural Conformance Monitor

- JSXM works as follows:
  1. JSXM transforms the actual Web service requests and responses to SXM inputs and outputs.
  2. Then, it animates the SXM model using these inputs and produces the expected SXM output.
  3. Finally, if the transformed actual response does not match the expected SXM output, there is a violation in the behaviour of the monitored Web service.

Integration of a Behavioural Conformance Monitor

- JSXM works as follows:
  1. JSXM transforms the actual Web service requests and responses to SXM inputs and outputs.
  2. Then, it animates the SXM model using these inputs and produces the expected SXM output.
  3. Finally, if the transformed actual response does not match the expected SXM output, there is a violation in the behaviour of the monitored Web service.

Integration of a Behavioural Conformance Monitor

- JSXM works as follows:

1. JSXM transforms the actual Web service requests and responses to SXM inputs and outputs.
2. Then, it animates the SXM model using these inputs and produces the expected SXM output.
3. Finally, if the transformed actual response does not match the expected SXM output, there is a violation in the behaviour of the monitored Web service.

Integration of a Behavioural Conformance Monitor

- JSXM works as follows:

1. JSXM transforms the actual Web service requests and responses to SXM inputs and outputs.
2. Then, it animates the SXM model using these inputs and produces the expected SXM output.
3. Finally, if the transformed actual response does not match the expected SXM output, there is a violation in the behaviour of the monitored Web service.

SOUTH-EAST
EUROPEAN
RESEARCH
CENTRE

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2


The University Of Sheffield.


CITY College An International Faculty Of The University.


SOUTH-EAST EUROPEAN RESEARCH CENTRE

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

Evaluation setup

- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Evaluation setup

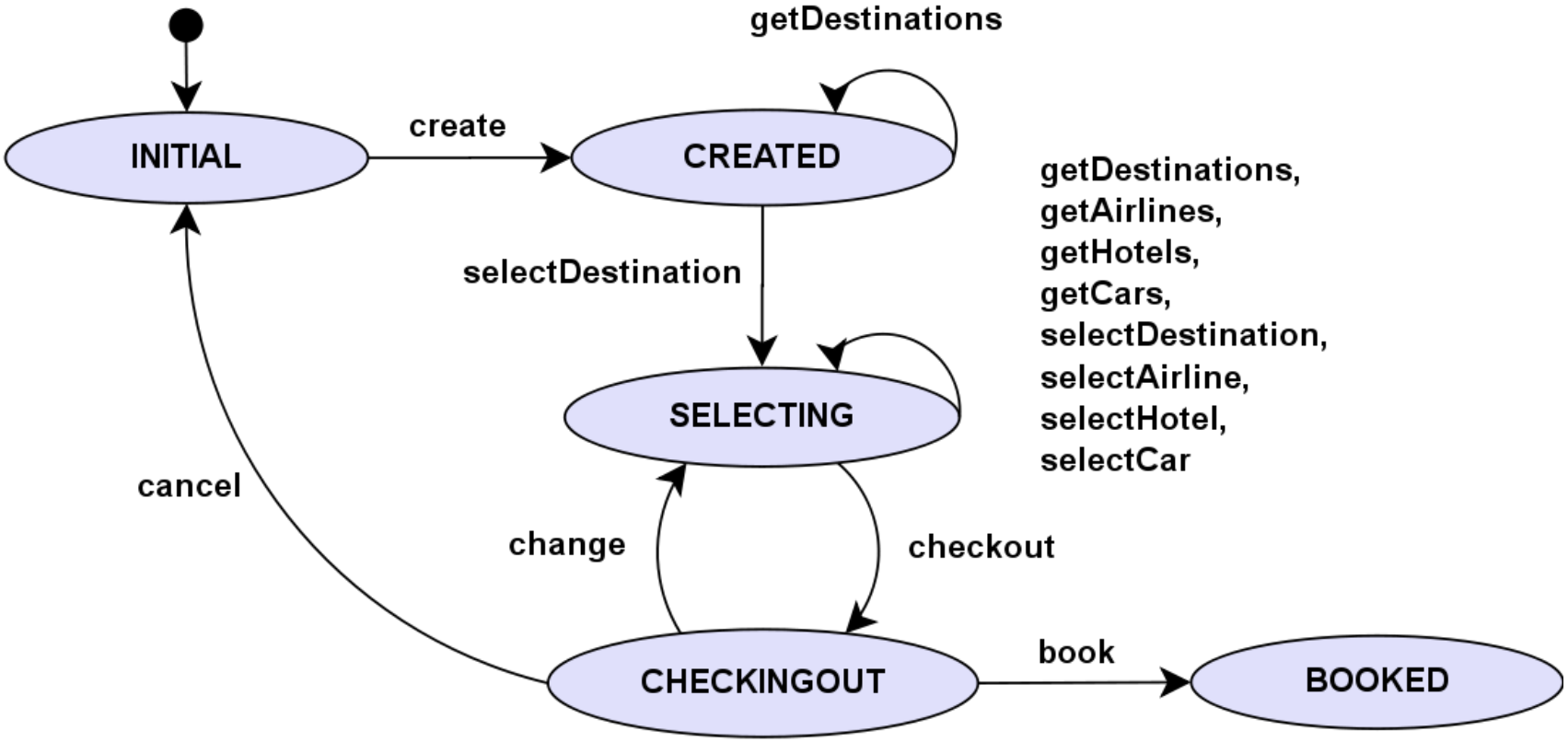
- The evaluation involved a linearly increasing number of consumers accessing concurrently a monitored service.
  - 4 different interaction scenarios were used by the consumers.
- A conversational service was deployed in JBoss AS, in order to be the service under monitoring.
- The monitoring framework was deployed in the same server.
  - Intel Core 2 Quad Q8400
  - 4GB Ram
  - openSUSE 11.2

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

TravelAgency Web Service



The University Of Sheffield.

CITY College An International Faculty Of The University.

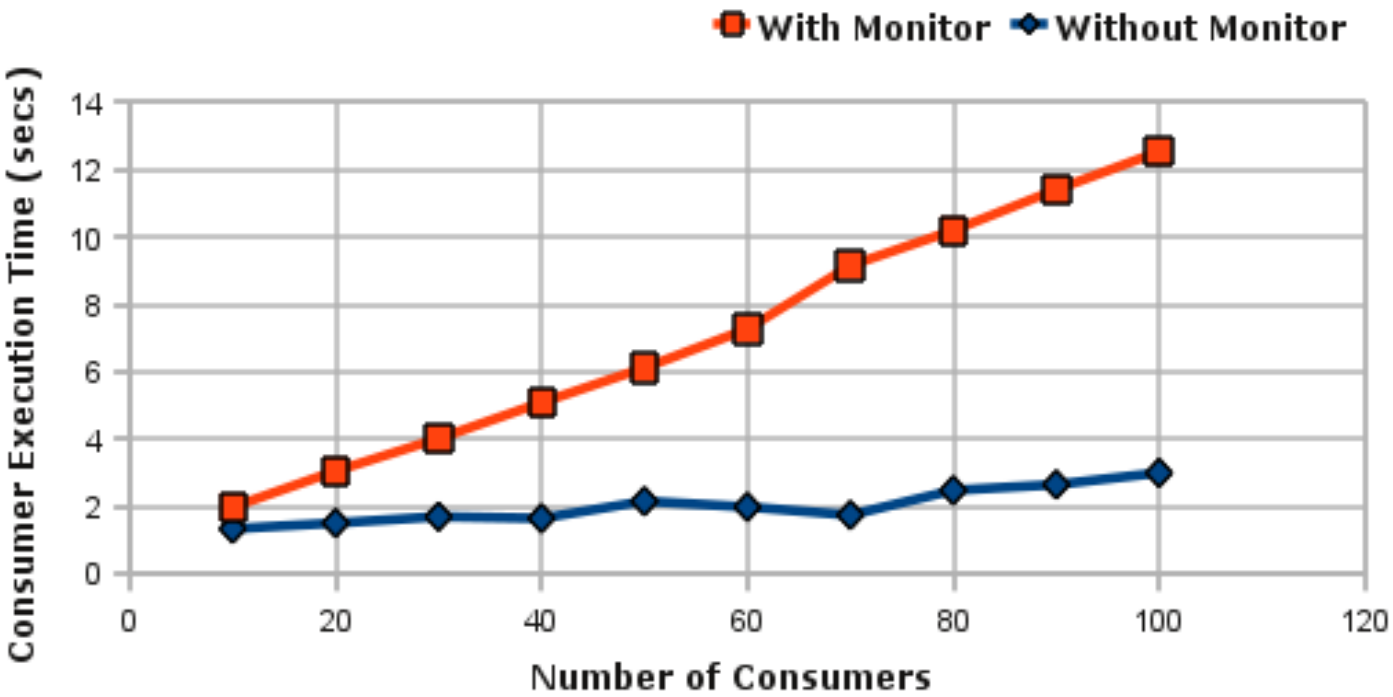
SOUTH-EAST EUROPEAN RESEARCH CENTRE

Evaluation Scenarios

Interaction scenarios containing different execution sequences.

Scenario	Effect	Execution
1	Booked	Succeeds
2	Booked with car	Succeeds
3	No hotel is selected	Fails
4	No checkout is done	Fails

Experimental Result



The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Future directions

- Improve the implementation of the monitoring architecture to support monitoring of long-running (persistent) Web service interactions.
- Investigate extensions for supporting the integration of monitors for both non-functional and functional aspects of conversational as well as non-conversational Web services.
- Integrate existing monitoring tools from the literature to the monitoring architecture.
- Continue evaluating the monitoring architecture using more complex monitoring scenarios.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Future directions

- Improve the implementation of the monitoring architecture to support monitoring of long-running (persistent) Web service interactions.
- Investigate extensions for supporting the integration of monitors for both non-functional and functional aspects of conversational as well as non-conversational Web services.
- Integrate existing monitoring tools from the literature to the monitoring architecture.
- Continue evaluating the monitoring architecture using more complex monitoring scenarios.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE SEERC

Future directions

- Improve the implementation of the monitoring architecture to support monitoring of long-running (persistent) Web service interactions.
- Investigate extensions for supporting the integration of monitors for both non-functional and functional aspects of conversational as well as non-conversational Web services.
- Integrate existing monitoring tools from the literature to the monitoring architecture.
- Continue evaluating the monitoring architecture using more complex monitoring scenarios.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE

Future directions

- Improve the implementation of the monitoring architecture to support monitoring of long-running (persistent) Web service interactions.
- Investigate extensions for supporting the integration of monitors for both non-functional and functional aspects of conversational as well as non-conversational Web services.
- Integrate existing monitoring tools from the literature to the monitoring architecture.
- Continue evaluating the monitoring architecture using more complex monitoring scenarios.

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE SEERC

Conclusions

- A primary objective of the implemented monitoring architecture was to provide a platform for integrating cross-layer monitoring approaches for Web services.
- Implementing a monitoring architecture for Web services involves issues such as message interception, session handling and concurrency.
- The presented monitoring architecture is agnostic to the interception mechanism used.
- Finally, the architecture is not bound to a specific vendor and thus it can be fully utilised within service-based applications.

Conclusions

- A primary objective of the implemented monitoring architecture was to provide a platform for integrating cross-layer monitoring approaches for Web services.
- Implementing a monitoring architecture for Web services involves issues such as message interception, session handling and concurrency.
- The presented monitoring architecture is agnostic to the interception mechanism used.
- Finally, the architecture is not bound to a specific vendor and thus it can be fully utilised within service-based applications.


The University Of Sheffield.


CITY College An International Faculty Of The University.


SOUTH-EAST EUROPEAN RESEARCH CENTRE

Conclusions

- A primary objective of the implemented monitoring architecture was to provide a platform for integrating cross-layer monitoring approaches for Web services.
- Implementing a monitoring architecture for Web services involves issues such as message interception, session handling and concurrency.
- The presented monitoring architecture is agnostic to the interception mechanism used.
- Finally, the architecture is not bound to a specific vendor and thus it can be fully utilised within service-based applications.


The University Of Sheffield.


CITY College An International Faculty Of The University.


SOUTH-EAST EUROPEAN RESEARCH CENTRE

Conclusions

- A primary objective of the implemented monitoring architecture was to provide a platform for integrating cross-layer monitoring approaches for Web services.
- Implementing a monitoring architecture for Web services involves issues such as message interception, session handling and concurrency.
- The presented monitoring architecture is agnostic to the interception mechanism used.
- Finally, the architecture is not bound to a specific vendor and thus it can be fully utilised within service-based applications.

Thank you for your attention! ...Questions?

The University Of Sheffield.

CITY College An International Faculty Of The University.

SOUTH-EAST EUROPEAN RESEARCH CENTRE