A New Approach to Information Modeling for Manageable Component-Based, Active, and Autonomous Service Provisioning

Vladimir Tosic, Bernard Pagurek
Department of Systems and Computer Engineering
Carleton University, Ottawa
e-mail: vladimir,bernie@sce.carleton.ca

Abstract: We present a new approach to information modeling for application and service management of computer systems and communication networks. The motivation for our work is that existing approaches to service management show limitations in addressing challenges of emerging trends in service provisioning. One of the challenges is that service provisioning is becoming increasingly component-based with possibly complex and dynamic relationships between involved service providing entities. There is also a strong trend for provisioning of differentiated services to different classes of clients. Further, features like distributed and dynamic service composition and autonomous reconfiguration require additional flexibility and adaptability of service provisioning systems, as well as further reduction of human involvement in service provisioning and management. Also, as computer systems and communication networks converge, there is a strong need for unified management of computer applications and communication services. The core of our approach is to extend object-oriented information modeling of services with the specification of *views*. In our approach, a service component provides some service functionality that is described using standard object-oriented concepts. In addition, every service component can offer one or more views describing conditions on usage of this functionality. A view contains formal specification of functional constraints (preconditions, postconditions, and invariants), quality of service (QoS) constraints (QoS required from underlying services and QoS guarantees to clients), and authorization policies. As more than one view can be specified for a service component, views can be used for specification of differentiated services to different classes of clients. In order to achieve easier and more consistent specification of views, we organize views of the same service component into single-inheritance hierarchies. We also introduce several view-related features that support dynamic (e.g., run-time) adaptation to changes in the environment (e.g., QoS of underlying services). First, clients can dynamically switch between different views of the same service component. Second, new views for the existing service functionality can be created dynamically. Third, a service component can dynamically deactivate (i.e., disable) a view and, eventually, re-activate it later. Compared with alternatives, our approach promises both easier dynamic composition of differentiated services from service components and increased flexibility and adaptability of resulting solutions. We are working further on defining new flexible and adaptable application and service management architectures and algorithms that are based on information modeling using the concept of views.

Biography: Vladimir Tosic is currently a Ph.D. student at the Department of Systems and Computer Engineering, Carleton University, Ottawa working under supervision of Prof. Bernard Pagurek in the area of application and service management. His research interests also include component-based software engineering, management of distributed computer and communication systems, mobile and intelligent agents. Vladimir received his Dipl.Ing. degree in Electrical Engineering (Computer Science and Engineering stream) from University of Nis, Yugoslavia. After graduation he worked for one year as an intern at the OpenView Software Division of Hewlett-Packard Company in Germany. He completed his M.Eng. thesis in Electrical Engineering (Computer Science and Engineering stream) through a collaboration of University of Nis, Yugoslavia and the OpenView Software Division of Hewlett-Packard Company. For his academic successes Vladimir received a number of awards, including Epstein Scholarship in 2000, the "1996 Nikola Tesla Youth Creativity Award" from Nikola Tesla Foundation, Yugoslavia, and the "Best Graduating Student of the University of Nis in 1995/1996" awards. Vladimir authored or co-authored about twenty papers. He is a student member of the IEEE (including Computer and Communications societies) and the ACM. For further information about his research contact him at vladimir@sce.carleton.ca or visit his WWW page http://www.sce.carleton.ca/~vladimir/index.html.

A New Approach to Information Modeling for Manageable Component-Based, Active, and Autonomous Service Provisioning

Vladimir Tosic, Bernard Pagurek
Department of Systems &
Computer Engineering
Carleton University, Ottawa
{vladimir,bernie}@sce.carleton.ca

Outline

- Introduction - trends in service provisioning
- Motivation for this work
- Description of the concept of views
- Features increasing flexibility and adaptability
- Why not alternatives?
- Implementation issues
- Conclusions and future work

Some Trends in Service Provisioning

- Convergence of computer applications and communication services
  - the need for unified management
- Component-based service provisioning
  - may result in complex and dynamic relationships
- Differentiated services for different classes of clients
  - computer systems: personalization

Some Trends in Service Provisioning (cont.)

- Increased distribution and dynamism
  - usage of active technologies
  - frequent disturbances in wireless systems
- The need for increased flexibility and adaptability (with reduction of human involvement)
  - distributed and dynamic service composition
  - autonomous reconfiguration
  - QoS management at the application level

Motivation for This Work

- Existing approaches to service composition and service management show limitations in addressing challenges of emerging trends in service provisioning
- We believe that a new approach to information modeling of software and service components can help to enable easier service composition and more flexible and adaptable service management

Service Components

- A service component is a nearly independent and replaceable part of the system that provides a set of well-defined services
- Examples: distributed objects, software components, telecommunication services encapsulated in software modules, ...
- Functionality can be described using object-oriented and component-based concepts
- Service component instances and service component types (classes)

The Concept of Views

- There is also a need to describe non-functional aspects, i.e. conditions of usage
- A *view* contains a formal specification of:
  - functional constraints (pre- and postcondtions, invariants)
  - QoS constraints (QoS required from underlying services and QoS guaranteed to clients)
  - authorization policies (defining access rights)

Specification of Differentiated Services Using Views

- Views are specified separately from the functionality to which they refer
  - more than one view can be specified for the same service component type
  - a client may use different views for different instances of the same service component type
- A client can open sessions with a service component
  - possibly more than one session (maybe with different views) with the same service component
- Some of the constraints can be used in view trading or in component instance trading

Hierarchical Organization of Views

- In order to achieve easier and more consistent specification of views
- Single-inheritance hierarchy per service component type
- Inheritance and redefinition of constraints from superviews
  - "OR" for preconditions, QoS requirements, authorization policies
  - "AND" for postconditions, invariants, QoS guarantees

Example - Object Trader

```
interface Trader {
  Object[] find(Object attribute);
  void add(Object attribute, Object item);
  /* ... other methods */
}; // end interface Trader

view TraderClient on Trader {
  find.Authorization: true;
  find.QoSParameters: responseTime;
  find.Postcondition: responseTime.name
     == "response time (including service)"
     && responseTime.unit == "ms"
     && responseTime.value<=5;
  add.Authorization: false;
 /* ... constraints for other methods */
}; // end view TraderClient
```

```
view TraderPriorityClient extends
    TraderClient {
  find.Postcondition:
    responseTime.value<=2;
}; // end view TraderPriorityClient

view TraderManager extends
    TraderPriorityClient {
  add.Authorization: true;
  add.Precondition: not (item in
    find(attribute));
  add.Postcondition: item in
    find(attribute);
  /* ... constraints for other methods */
}; // end view TraderManager
```

Features Increasing Flexibility and Adaptability I

- Support for dynamic adaptations (e.g., to changes in the environment like QoS of underlying services)
- A client may be allowed to dynamically switch between views of the same service component without changing the session
- Example: A client dynamically decides that better QoS is needed
- The service component checks restrictions

Features Increasing Flexibility and Adaptability II

- Dynamic creation of new views for existing service component instances
- This is not creation of new functionality, but of new offerings of existing functionality!
- Due to possible conflicts, a strict control of dynamic creation of views is needed
- Example: Subviews differing in QoS can be defined dynamically from the same superview that defines functional constraints and authorization policies

Features Increasing Flexibility and Adaptability III

- A service component instance need not support all views defined for its service component type all the time
- Example: High guaranteed QoS cannot be achieved all the time
- A service component instance may be allowed to dynamically activate/deactivate support for some views
- If a view is deactivated, its subviews are *not* automatically deactivated

Using Views: Deactivation of a View Used by Some Clients

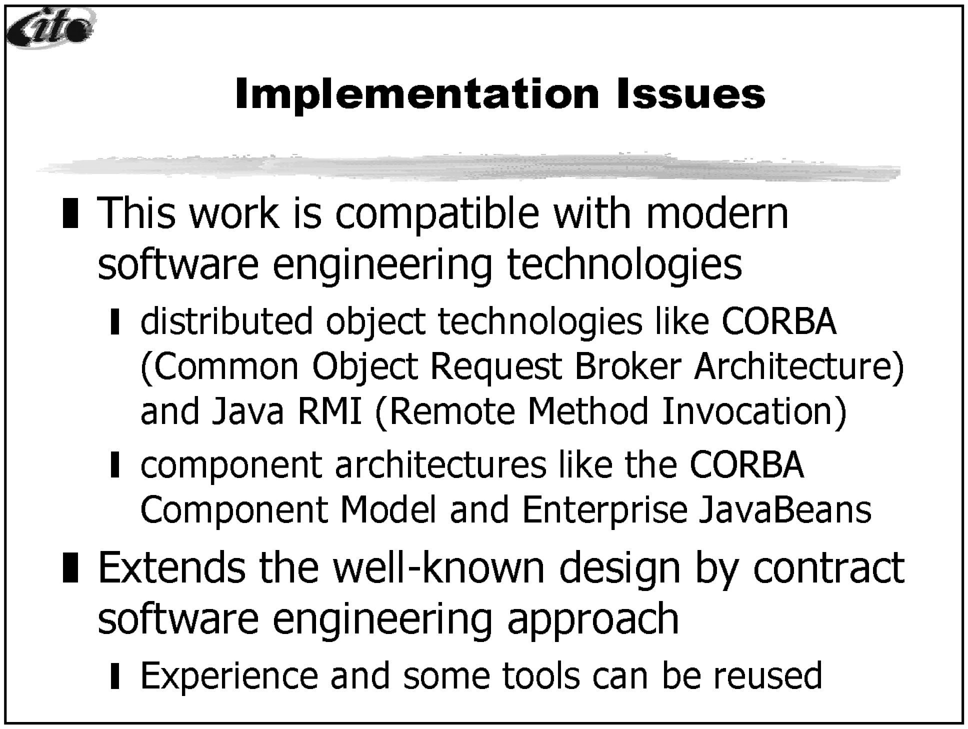
- Automatically switch to the first active superview
- Emit an event to all affected clients
- Clients can decide what to do next
- When the original view is reactivated, if a client has not explicitly switched views, that view should be restored
- This algorithm helps in fast autonomous adaptation

Why Not Alternatives?

- Multiple service component instances differing in constraints - redundancy, not always applicable, less flexible
- Adapters (including roles, contracts, wrappers, proxies) with different constraints - more error-prone, harder to manage, without automatic code generation
- Multiple functional interfaces - no formal specification of constraints, possibility of conflicts, less flexible

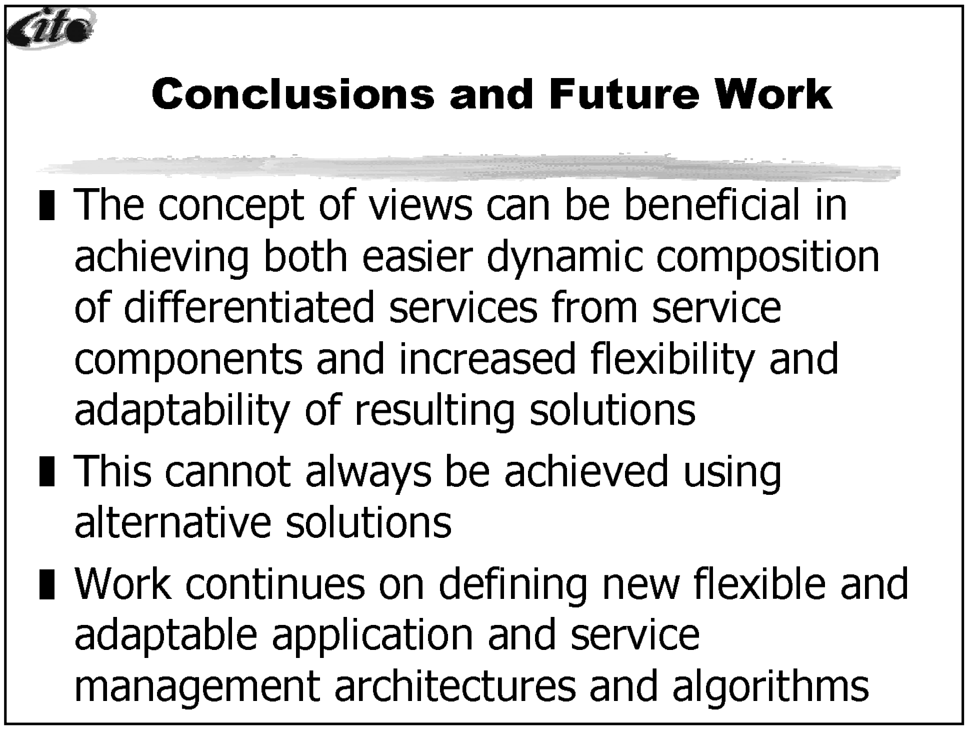
Why Not Alternatives? (cont.)

- QoS specification languages (e.g., QML, QDL) - address only QoS issues, no wide acceptance
- Policy-driven management - does not address functional and QoS constraints, top-down approach, policy refinement and policy conflicts are serious problems
- TINA (Telecommunications Information Networking Architecture) - too complex, not fully component-based, multiple interfaces, no formal specification of constraints (except QoS)



Implementation Issues

- This work is compatible with modern software engineering technologies
  - distributed object technologies like CORBA (Common Object Request Broker Architecture) and Java RMI (Remote Method Invocation)
  - component architectures like the CORBA Component Model and Enterprise JavaBeans
- Extends the well-known design by contract software engineering approach
  - Experience and some tools can be reused



Conclusions and Future Work

- The concept of views can be beneficial in achieving both easier dynamic composition of differentiated services from service components and increased flexibility and adaptability of resulting solutions
- This cannot always be achieved using alternative solutions
- Work continues on defining new flexible and adaptable application and service management architectures and algorithms