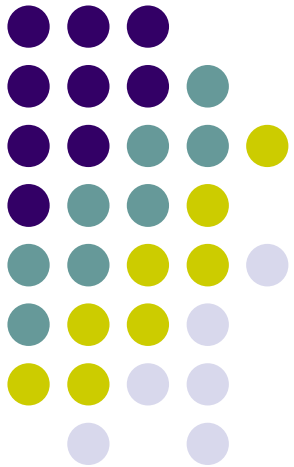
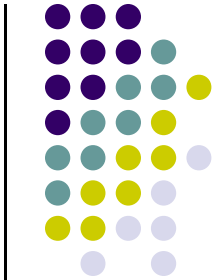
The Landscape of Service-Oriented Systems: Research Topics from the Perspective of Maintenance and Reengineering

Kostas Kontogiannis
NTUA



Motivation and Rationale

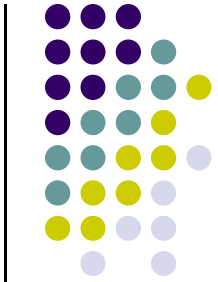


- Over the past years infrastructure technologies supporting SOA have become robust and dependable
- However, requirements and software systems complexity are growing at a higher rate than the supporting technologies can handle
- In order to be able to meet the requirements of the next generation Service Oriented systems we have to plan for new supporting technologies

"By 2011, early technology adopters will purchase 40 percent of their IT infrastructure as a service.
Gartner's outlook for IT organizations and users.
Feb. 2008

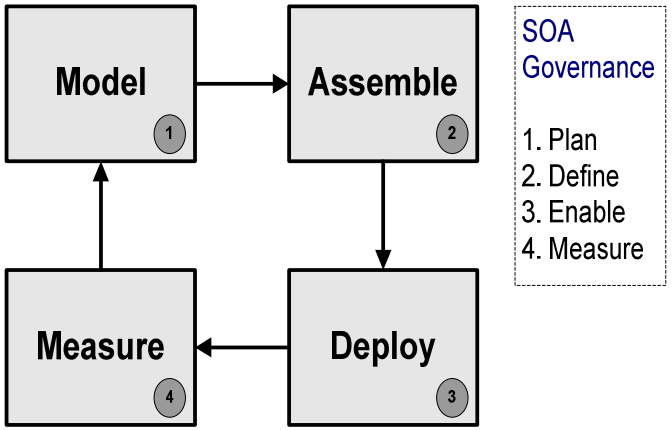
SOA adoption attempts on a number of surveyed corporations, indicated a 5% decline over the survey results of January of last year. One of the potential reasons is cited to be the failure to meet the expectation that SOA will save companies money, among other things
CIO Insight, "Future of IT" series, 2008

Solution Target



- Simplified development of *business* services
- Simplified assembly and deployment of business solutions built as *networks* of services
- Increased agility and flexibility
- Protection of business logic assets by shielding from low-level technology change
- Improved testability

SOA Life Cycle



SOA Life Cycle

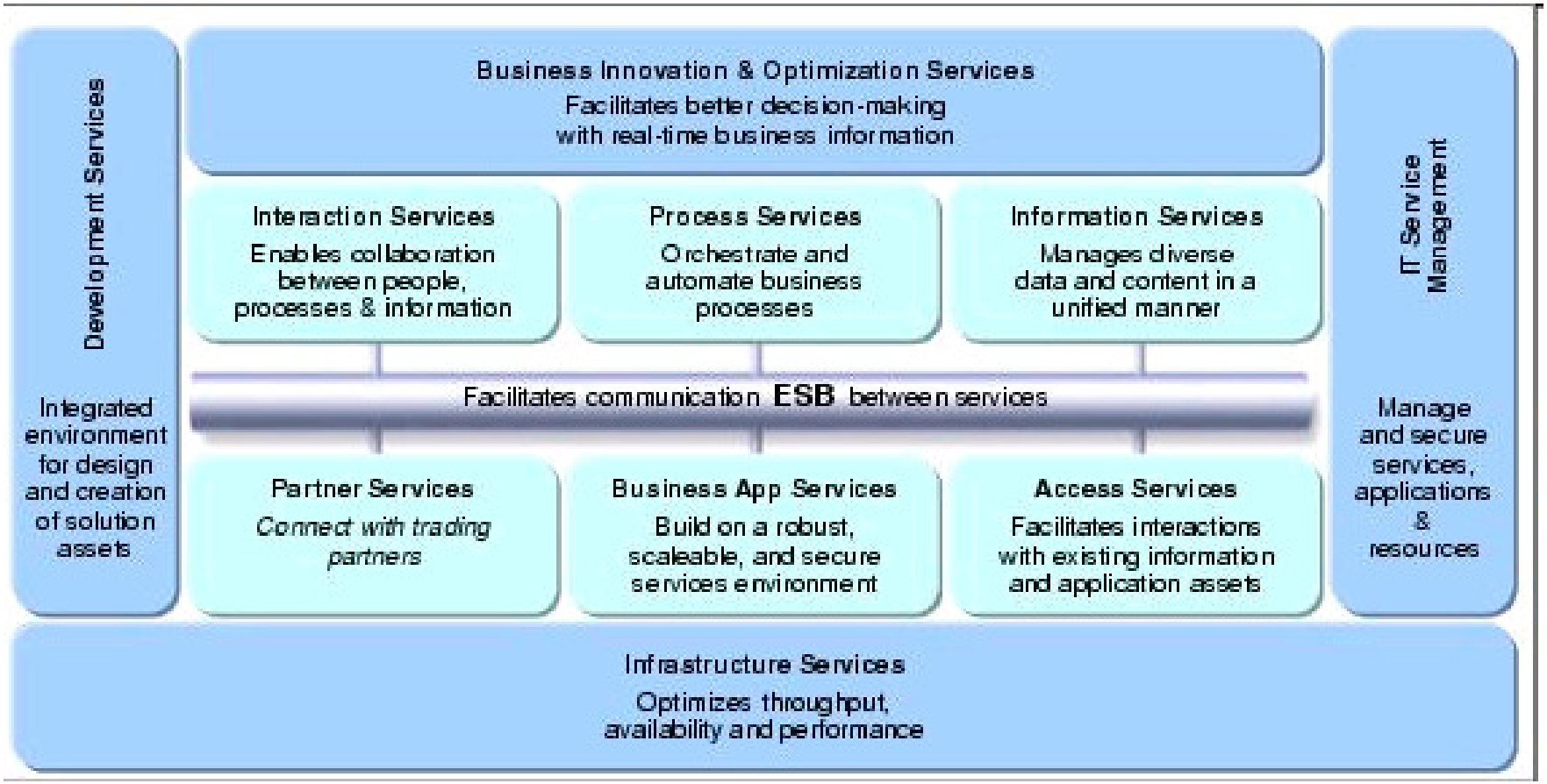
Modeling is the process of capturing business requirements, business goals and objectives, and transforming them into business process specifications—the *business model.*

Assembly is the phase that deals with the implementation issues by either reusing existing services or by creating new services.

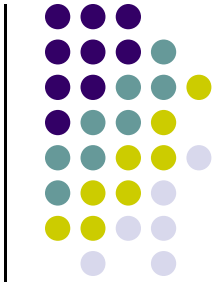
Deployment is the phase of resolving service dependencies, capacity planning, defining the hosting infrastructure, as well as system testing.

The *Management* phase refers to the operational activities that keep the applications running as well as the measurement and auditing.

SOA Logical Architecture Model

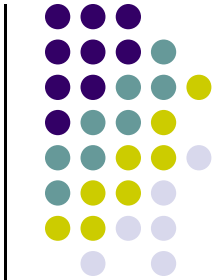


SOA Logical Architecture Model



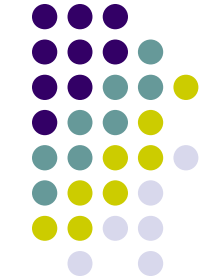
- **Interaction services** : Interaction services provide the capabilities required to deliver IT functions and data to users, meeting their specific preferences.
- **Process services** : Process services provide the control capabilities required to manage the flow and interactions of multiple services in ways that implement business processes.
- **Business application services** : Business application services are called by service consumers. Service consumers include other components in the logical architecture such as portal or a business processes.
- **Information services** : Information services provide the capabilities necessary to federate, replicate and transform disparate data sources.
- **Access services** : Access services provide bridging capabilities between core applications, prepackaged applications, enterprise data stores and the ESB to incorporate services that are delivered through existing applications into an SOA.
- **Partner services** : Partner services provide the document, protocol, and partner management capabilities for business processes that involve interactions with outside partners and suppliers.

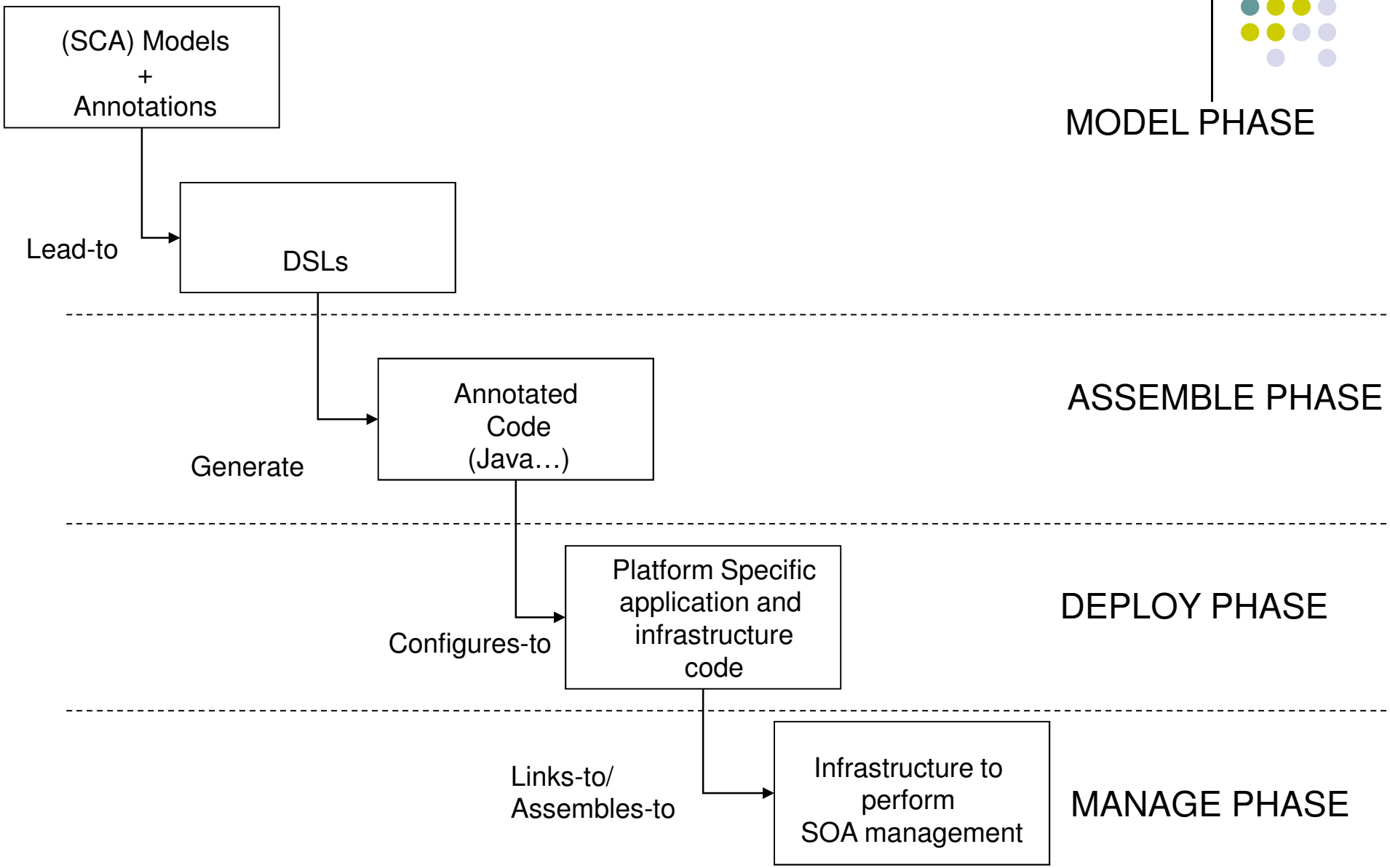
SOA Programming Models



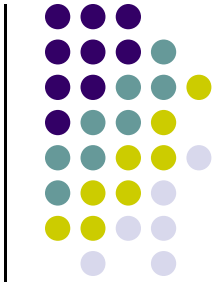
- A SOA Programming Model is a collection of models, techniques, methodologies and tools for implementing services and assembling them into solutions and should provide:
  - A simplifying abstraction that allows programmers to concentrate principally on business logic while building / assembling service oriented systems
  - A uniform representation for messages that interact with services
  - ESB technology and issues related to auditing, logging, routing, adaptation of mismatched interfaces, security,- at an enterprise-wide level
  - Abstractions for service composition languages
  - Business analysis and simulation tools (*business state machines)* to be integrated in the system's development process
  - Infrastructure to facilitate component reuse and templates with *points of variability*
  - Enablement of Human Roles in Service Oriented Systems

Programming Models in Context



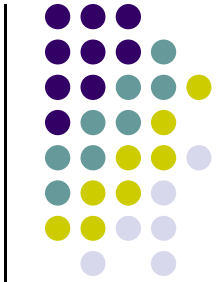


Programming Model Issues



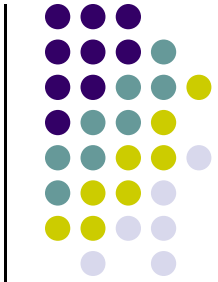
- Definition of a service (including semantics and metadata for discovery, invocation and maintenance purposes)
- Simplified data access models
- Component types to simplify development
- Process components (e.g. BPEL and business state machines)
- Customizing services: design patterns, templates, and points of variability
- Service-oriented user interfaces
- Development tools for the SOA software life cycle
- Policies in the SOA

SOA System Management and Run-Time Infrastructure Issues



- Root Cause Analysis
- Business Process View of Logs
- Compliance and Governance issues
- Assurance and Security (above the transport layer)
- What will be the challenges for the next generation of SOA Run time infrastructure?

SOA Maintenance and Reengineering



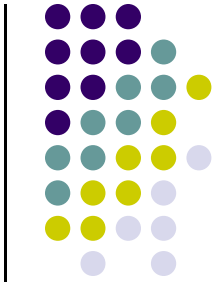
- Service Oriented Systems and ULS Systems of Systems pose unique challenges for their maintenance and evolution

- These challenges stem from the seer size and complexity of such systems, their multi-platform, multi-language nature, and from the number and diversity of stakeholders involved

- Some initial issues related to a research agenda for Service Oriented Systems from the maintenance / reengineering perspective include:

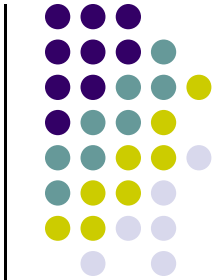
  - Evolution patterns
  - Tools, techniques and environments to support maintenance activities, including dependency and impact analysis, change control, and management
  - Multilanguage system analysis and maintenance
  - Reengineering processes
  - Tools for the verification and validation of compliance with constraints
  - Round-trip engineering in service-oriented systems

SOA Requirements Engineering and Project Management



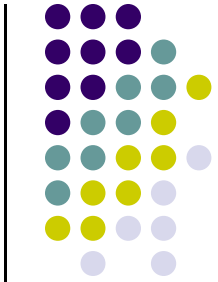
- Modeling dynamic configurations of large scale systems
- Modeling and managing conflicting NFRs
- Understanding and modeling the impact specific design decisions may have to specific NFRs, quality engineering
- Techniques for risk assessment and mediation
- Infrastructures for change control and management
- Global issues related to enterprise planning, strategic reuse, operations support

SOA Evolution Patterns



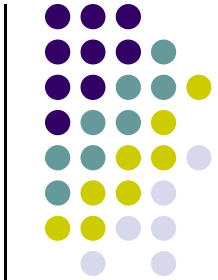
- As a research community we have investigated significantly the evolution pattern of various applications (see open source systems analysis)
- These applications pertain mostly to stand-alone systems that are statically configured at compile time
- However, there is limited work on analyzing the evolution pattern of SOA type of systems where these systems are essentially systems of evolving systems (perpetual beta), involving a much greater and much more diverse number and type of stakeholders
- Questions include:
  - Are these SOA systems have specialized evolution patterns that differ from the ones we have seen so far?
  - How do we model the evolution patterns for SOA systems
  - How do we reason, compare, evaluate, assess, use these evolution patterns to determine / forecast system quality and plan system extensions

SOA Tools, and Environments



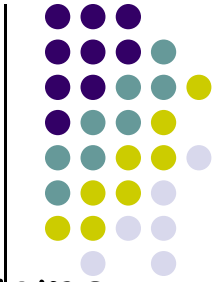
- Better integration of life-cycle activities (requirements, design, implementation, testing, deployment, maintenance)
- Better support for use of architectural styles and design patterns (in reverse and forward engineering)
- Model consistency management, synchronization and co-evolution
- Model re-factoring and model weaving
- Robust model transformation technology (meta-models, languages, run-time environments) and domain specific languages and meta-programming

Multi-language and Multi-platform System Analysis



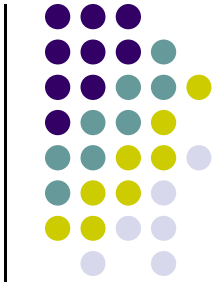
- Issues pertaining to Modeling, Extraction, Representation
  - Formalization and modeling
  - Extraction, discovery, and storage
- Issues pertaining to Analysis, Exploration, Visualization
  - Exploration, queries, and knowledge management
  - Presentation and usability
- Tool Support, Process, Organizational Issues
  - Tool support to understand, analyze and maintain ML software
  - Impact on software maintenance processes

SOA Systems Reengineering Processes



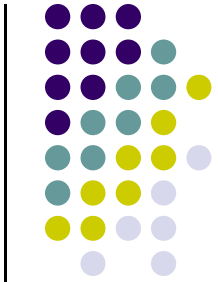
- In the software reengineering community we have done some work on investigating reengineering processes (horse-shoe, SRAH, OAR, SMART etc.)
- There is also work performed on defining processes for reengineering legacy systems to network-centric environments
- However, we have not done any significant work on investigating reengineering processes for SOA systems (reengineering / evolving SOA systems)
- Questions include:
  - Are reengineering processes for SOA systems different from the ones we have identified so far (horse-shoe, SRAH, OAR, SMART)
  - How do we model and denote these processes
  - How do we evaluate / select different reengineering processes for SOA systems (for example given an application domain, or system characteristics), and what qualitative / quantitative methods to use

Verification and Validation - Compliance with Constraints



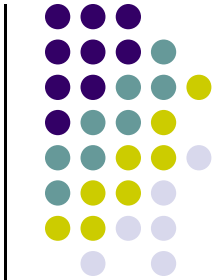
- Reverse engineering not only for understanding but also for compliance checking and property validation
- Model checking may play an important role
- For SOA systems we can assume three levels of abstraction:
  - Infrastructure level
  - Source code / Component/API level
  - Business process level
- It is interesting and challenging to investigate how to bridge the gap (compliance wise) between these levels
- Questions include:
  - How constraints / invariants / policies are modeled in each level of abstraction
  - How do we identify / trace, constraints / invariants / policies between levels of abstractions in the light of an evolving SOA system
  - How all these relate to recent initiatives on IT governance and compiiance regulations

Round-trip Engineering in Service-Oriented Systems



- Extraction and identification of model dependencies between models associated with Service oriented system entities
- Design and implementation of model partitioning techniques so that the scope and the effect of dependencies / changes between models can be localized to specific parts of the models involved
- Incremental model transformation techniques
- Design and implementation of techniques for maintaining consistency between models so that specific system quality constraints, relations and policies are guaranteed through system evolution
- Round trip engineering in the presence of possibly partial models
- Dealing with Perpetual beta

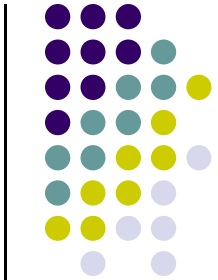
Top Research Areas (1)



- **Service Versioning**
  - Models and techniques that allow for the automatic or semi-automatic identification of mismatches or potential mismatches between the service as specified with the service as deployed beyond pure syntactic differences and focus on behavioral and operational aspects as well
  - Techniques that allow for global service update management in the form of models and tools for what-if analysis during componentry substitution
  - Data management related programming model extensions for specifying, denoting and validating, whether data sources and schemas are complete and adequate with respect to the data required for a composition of deployed services in a SOA system.
  - Abstractions, models and techniques to manage service connectivity (e.g. wiring properties) throughout the life cycle, without the need of altering the overall service assembly and supporting infrastructure.

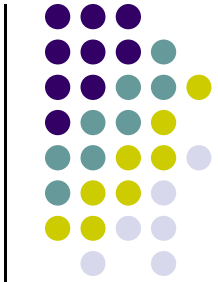
- **Property Tracing in SOA Systems**
  - Techniques to allow for the acquisition of data that can assist analysts and developers determine how quickly a service can be provisioned to accept new consumers and implement with minimal impact the required adaptations
  - Techniques and models that allow for the acquisition of data that can assist analysts and developers determine the impact of a change in a service or wiring between services (e.g. change of transaction policies, end point implementations, connectivity properties) to other services and to the system behavior in general
  - Techniques and models to assess the vulnerability of a SOA system with emphasis on security issues (e.g. denial of service attacks, access controls), as well as techniques and models to assess the risk and impact of service unavailability (e.g. what is the risk and what will happen if a service becomes unavailable).

Top Research Areas (2)



- **Smart service infrastructure**
  - Techniques, to denote and dynamically identify boundaries and scope of transactions for the purposes of maintaining integrity or recovering / compensating from the inability to successfully perform the service or any required service.
  - Techniques, to denote context-aware semantic and operational type service descriptions to allow for reaching the maximum set of consumers by allowing through adaptation the best match or offering the maximum utility in a specific consumer's "desired" service characteristics request.
  - Techniques, models, formalisms and, framework extensions to allow for different conversation policies to be
- **Logging, Monitoring and Diagnostics in SOA environments**
  - Techniques for the non-intrusive collection of events in a customized or adaptive manner (e.g. increase or decrease the level of monitoring according to some policies, perceived risks and vulnerability criteria) but also facilitates the processing and analysis of collected events in an efficient manner (e.g. efficient event reconciliation, log filtering, and log amalgamation).
  - Techniques for the acquisition of data that can assist analysts and developers perform root cause analysis in large loosely coupled SOA systems. Specific areas of problems may include testing for failures for services to terminate properly and failure to meet the specified target SLAs or QoS.
  - Techniques to allow for preflight and in-flight checks.
  - Techniques for seamless and continuous cross cutting monitoring of SOA systems at infrastructure, application and business process abstraction layers.

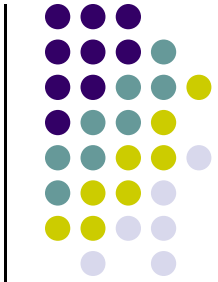
Top Research Areas (3)



- **Data access services, data handling and, data validation**
  - Techniques and infrastructure to attain global awareness of all data changes across services, regardless of the source of change and to maintain the system's state and data.
  - Techniques and programming model abstractions for efficient caching and distribution of data into a distributed data-cloud type of architecture.

- **Tools for supporting the SOA life cycle**
  - Robust and customizable code generators from industry segment specific models. Challenges here include the need for the generated code to be efficient (performance wise) and well structured (design patterns wise),
  - Model management tooling infrastructure with emphasis on a) transformations of models and code generation, b) model co-evolution and synchronization c) model round-tripping (forward and backward transformations) to achieve traceability.

Conclusion - Thoughts



- There are conflicting views with respect to the usefulness of service-orientation and its related implementations and standards (e.g. Web Services)

- Service Oriented systems pose unique challenges for maintenance, reengineering, analysis for compliance checking and property validation

- Despite the slower than desired adoption of service- oriented systems in industry, the outlook is very positive, as SOA systems provide a higher degree of agility and adaptation of existing processes and services to new ones in order to meet new customer needs

The Landscape of Service-Oriented Systems: Research Topics from the Perspective of Maintenance and Reengineering

Kostas Kontogiannis
NTUA