



**ORACLE®**

# SOA Patterns and Best Practices

Gregor Rayman, Technical Architect - SEESAT  
HrOUG – Rovinj 2007-10-17

# Safe Harbor Statement

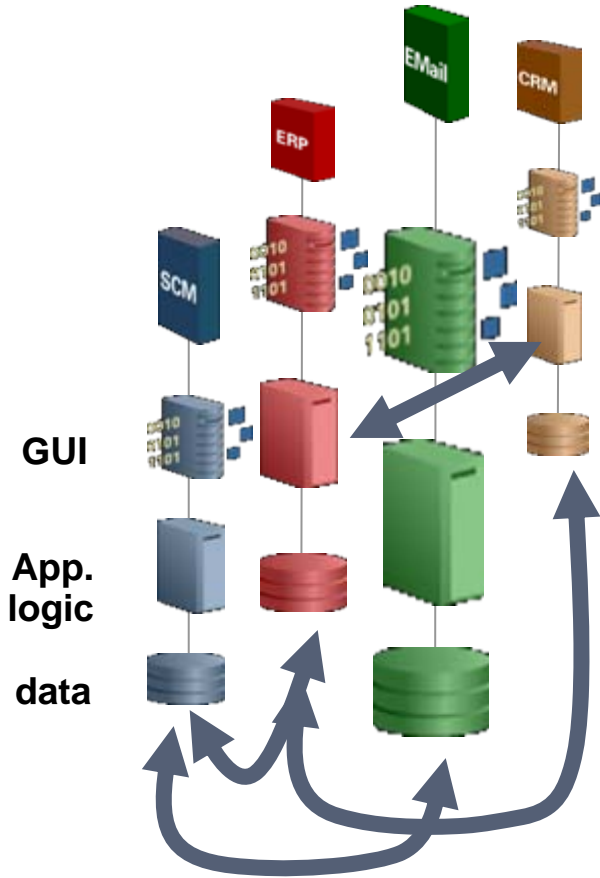
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



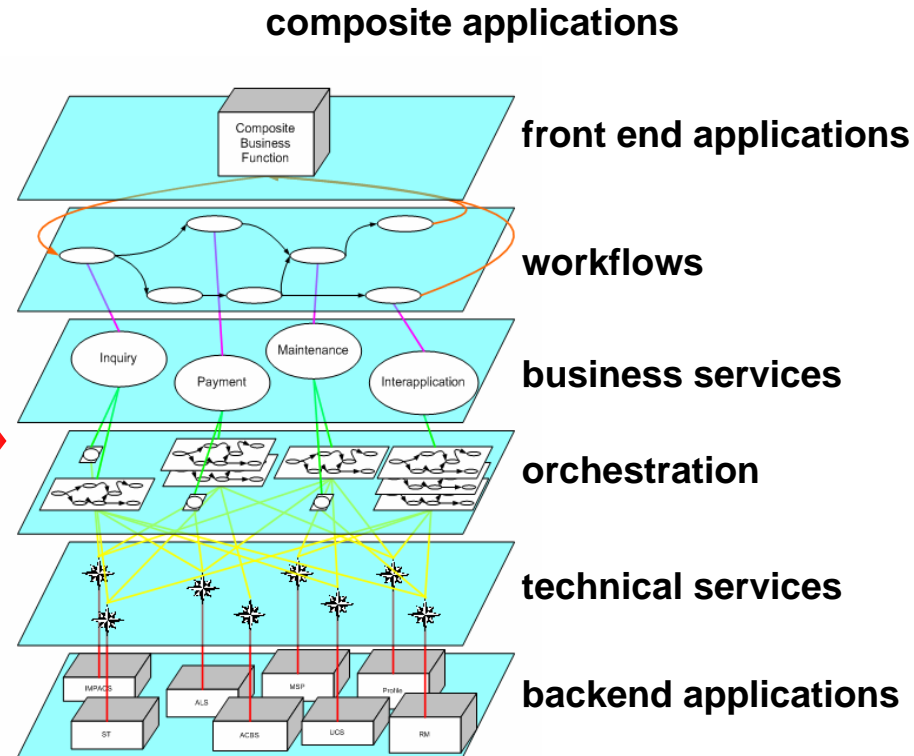
# Principles of SOA

## Application Silos



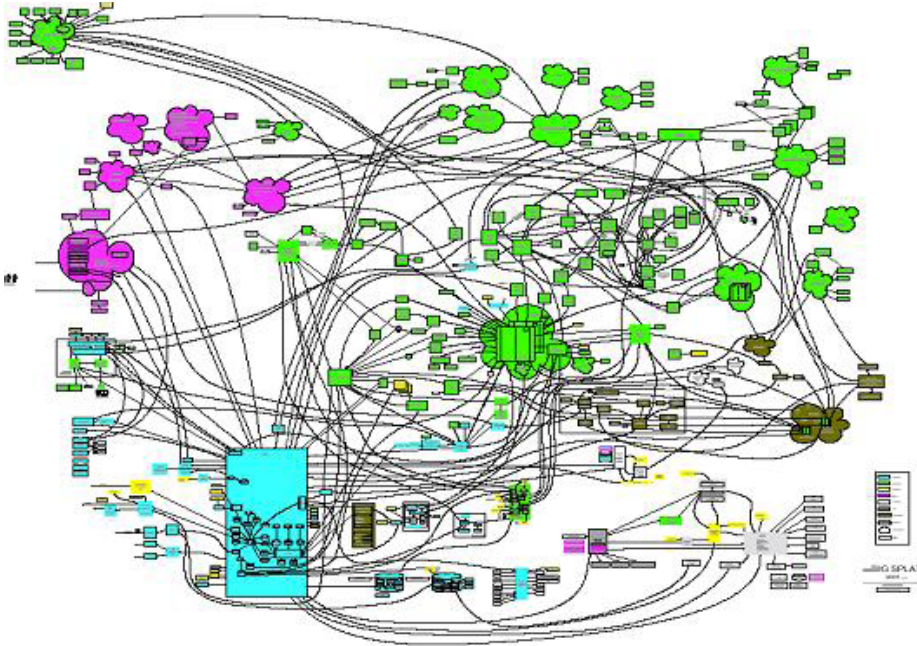
Point to point integration

## Service Oriented Architecture



# Caveat: Do not create 'process' silos

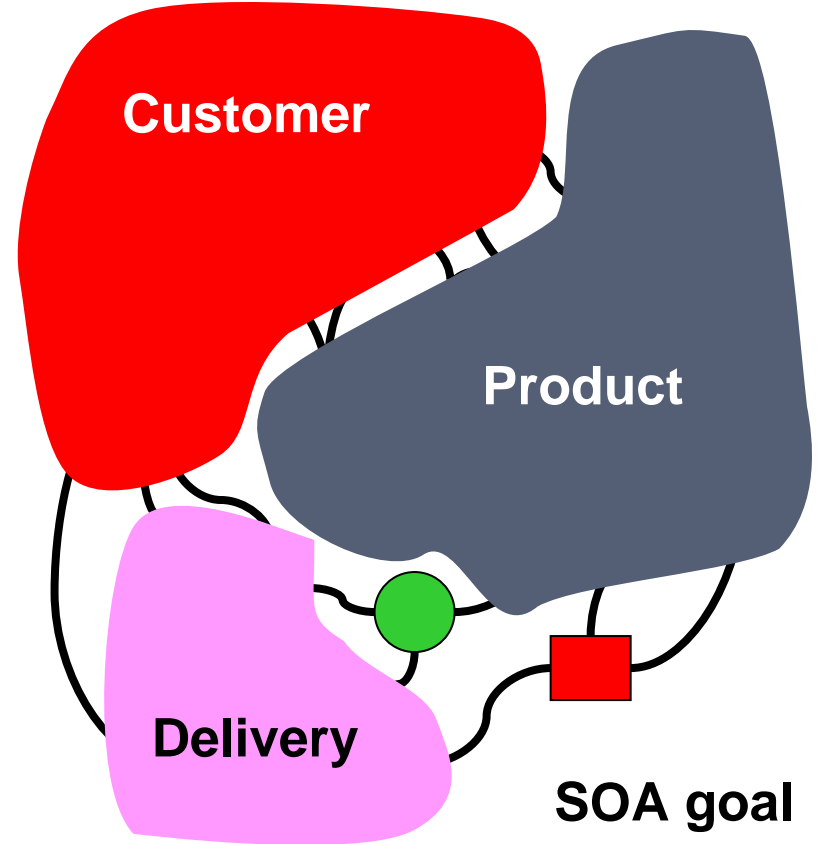
“Object soup”



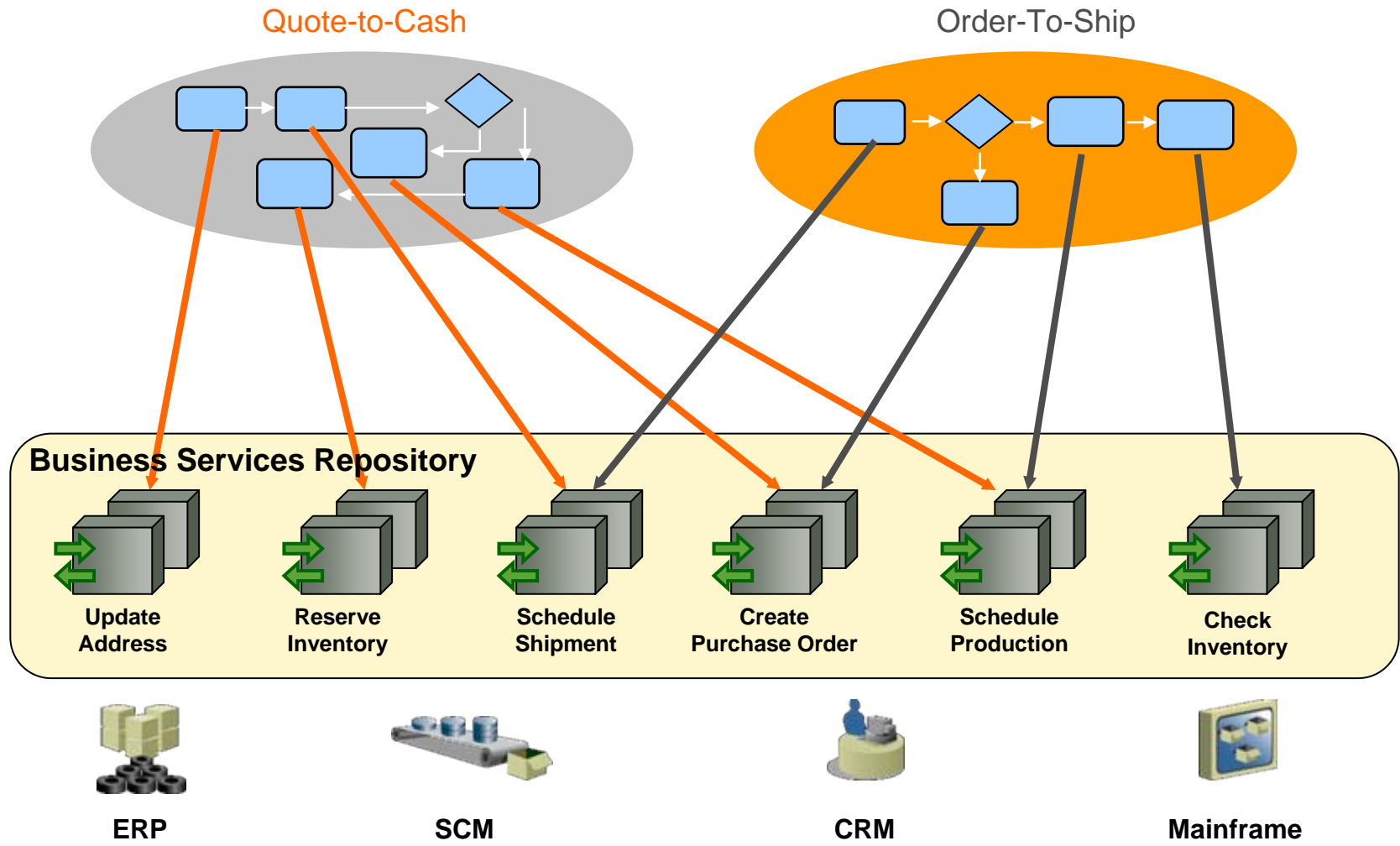
Typical point to point integration

Too many “services”

Coarse grained services



# Process driven app. composition



# More systems – more risks

## Operational risks

- **Service not available**
  - Do we need it now?
  - Can we use other service?
- **Service has an external effect**
  - Can we call it as a part of a transaction?
- **Failures**
  - Business exceptions
  - Technical failures
  - Can we try it again?
  - Do we know it?

## Organization risks

- **Visibility, discoverability**
  - Can we find it?
- **Service overlap, duplication**
  - Can we reuse it?
  - What do other departments do?
  - Can we adjust, extend it?
- **Transparency**
  - What exactly does it do?
  - How does it run?
- **Service dependencies**
  - What system does it use?
  - Who needs it?
  - Can we change it?



# Growth of reusable assets



# Organizational best practices

- **Stay close to business**
  - Remember: SOA is about business, not about technology
  - Create cross specialization teams
  - Pay attention to business and IT alignment
- **Be agile – long run consists of small steps**
  - Approach SOA evolutionary – no big bang projects
  - Use standards, shield proprietary interfaces, embrace change
  - Streamline the business processes – use SOA as optimization enabler
- **Security**
  - Security is not an “ad on”. Plan it from the beginning
- **Define SOA governance strategy early**
  - It’s not about one application.





# SOA governance strategy

- **Design time aspect**
  - requirement documentation
  - rules, standards, patterns, conventions
- **Run time aspect**
  - Service visibility and discoverability
  - Rollout strategy
  - Service and process monitoring, SLA, BAM
  - Error handling



# SOA design time best practices

- **Standards**
  - SOAP, WS-I, JMS, Industry standards data model (e.g. eTOM)
  - Use adaptors when accessing proprietary protocols
- **Naming conventions** – define and enforce them
- **Focus on interfaces**
  - No dependencies on implementations
  - Clearly define service contracts
- **Catalog and categorize your services**
  - Create and maintain the portfolio of your services
- **Establish an enterprise service bus**
  - Avoid point to point integration
  - Use service virtualization
- **Keep your systems coupled loosely**
  - But know, where to split them



# Avoid the anti-patterns

- **Don't let technology drive you**
  - Remember, it's about business
  - Be alarmed, if a project is called e.g. "the ESB project" instead of e.g. "near real-time billing project"
  - Be alarmed, if the "SOA approach" forces every application to change
- **Don't overuse web services (SOAP)**
  - Another communication protocols can be better suited
  - Use XML, but don't overuse it
- **Chose the right granularity**
  - One service can have more operations
  - Develop only needed services
  - Develop reusable services. Be alarmed, if similar business processes use different similar services
- **Chose the right coupling**
  - Sometimes tight coupling is the better choice



# Best practice: Have a Service registry

## Important asset in SOA governance

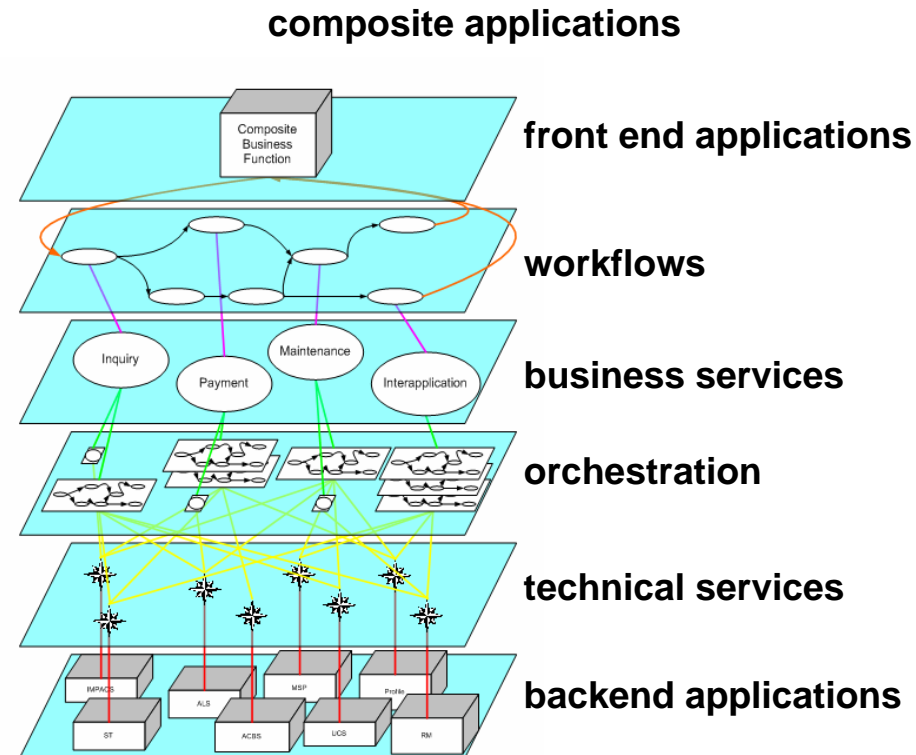
- **Makes the services searchable, discoverable**
- **Contains service documentation**
  - Categorization – technical service, business service, workflow?
  - Technical– interfaces, protocols, data model, dependencies
  - Risks – is it: read-only, idempotent, expensive, compensable?
  - Operational – availability, SLA, alternative services
  - Organizational – status, who uses it, who approves changes
- **Is a collaboration platform**
  - Documents user roles
  - Contains also not enterprise wide and wanted services
  - Governed by an enterprise SOA strategy



# Service categories

- **Component Services** – allow access to the underlying applications.
  - atomic
  - one system only
  - stateless
- **Composite (Business) Services** – implement a business function.
  - atomic
  - orchestrate component services
  - stateless
  - no long running transaction
- **Conversational (Workflow) Services** – control business processes
  - stateful
  - complex transactions

## Service Oriented Architecture



# Best practice: Enterprise Service Bus

- **Connect**

- Should be able to connect to all systems in the enterprise
- Standards: WebServices, Messaging, Database, Files, FTP...
- Adapters: For established legacy applications and proprietary protocols

- **Transform and Enrich**

- Talks to each system in the system's "language"
- Interpreter in the service provider – service consumer conversation

- **Distribute**

- Delivers the message to the appropriate end systems
- Handles error conditions



# Pattern: Asynchronous invocation

**Do we need the service immediately?**

**If not:**

- We can “order” the service. It will be delivered when available.
- We can better schedule the load.
- Our process can continue.

**Used communication patterns**

- One way message
- Request / Reply



# Best practice: Failure classification

## Business exceptions

- Specific for business process
- Have to be handled by the business process
- Examples: Customer is on a blacklist; Warehouse run out of stock; ...

## Technical failures

- Independent from the business process
- Best handled by the engine, policy driven
- Sometimes can be handled by repeating the invocation later
- Examples: Network broken; System overload; Bugs





# Error handling goals

- **If possible, reach the goal anyway**
  - Repeat the invocation
  - Use alternative service or approach
  - Sometimes human intervention inevitable
- **If not possible**
  - Offer alternative process path
  - Undo the partial process progress

## Best practice – be proactive, avoid errors

- **Technical errors:** high availability, redundancy, clustering
- **Business errors:** careful process design



# Undo: Transactions, Compensations

## • ACID transactions

- are not effective until completed
- can be rolled back
- provide better consistency across the applications
- are simpler to be used
- lock participating resources
- should be short
- require tight coupling
- often not possible

## • SOA “transactions”

- effective immediately
- cannot be rolled back, have to be compensated
- consistency cannot be always achieved without side effects
- have to be planned more carefully
- do not lock participating resources
- can be long running
- loose coupling is enough
- often the only choice




# Pilot project criteria

## A pilot project for SOA should:


1. Address a significant, well understood, but not critical business need
2. consider issues of Governance (relating to the scope chosen)
3. have SOA related infrastructure requirements
4. require an achievable stretch beyond current capabilities where gaps exist (skills, processes etc.)
5. be something you will put into production and deliver ROI




# Pilot project examples





**I outsourced a non-critical business service.**




**I expanded my market by putting an industry standard interface on my proprietary application.**



**I enabled multi-channel access to a key business service.**



**A service in front of my Loyalty System lets my customers consume points through partners.**



# Pilot project aspects

- Value
- Service identification
- Governance
- Development Methodology
- Service description
- Service provider
- Service requestor and business logic
- Service registry and discovery
- Security
- Service provider and requestor platform and products
- Production Monitoring
- Skills



Q & A





**ORACLE IS THE INFORMATION COMPANY**