***Eclipse Swordfish** — An Open Source SOA Runtime Framework for the Enterprise*

Oliver Wolf

Me

Email: oliver.wolf@sopera.de
Cell: +49-160-98931313
Twitter: @owolf

The company I'm working for

SOPERA

www.sopera.com

~~SOA~~

What makes Enterprise SOA...well..Enterprise SOA?

What Swordfish is and what it does

Extending Swordfish

Where we are and where we are heading

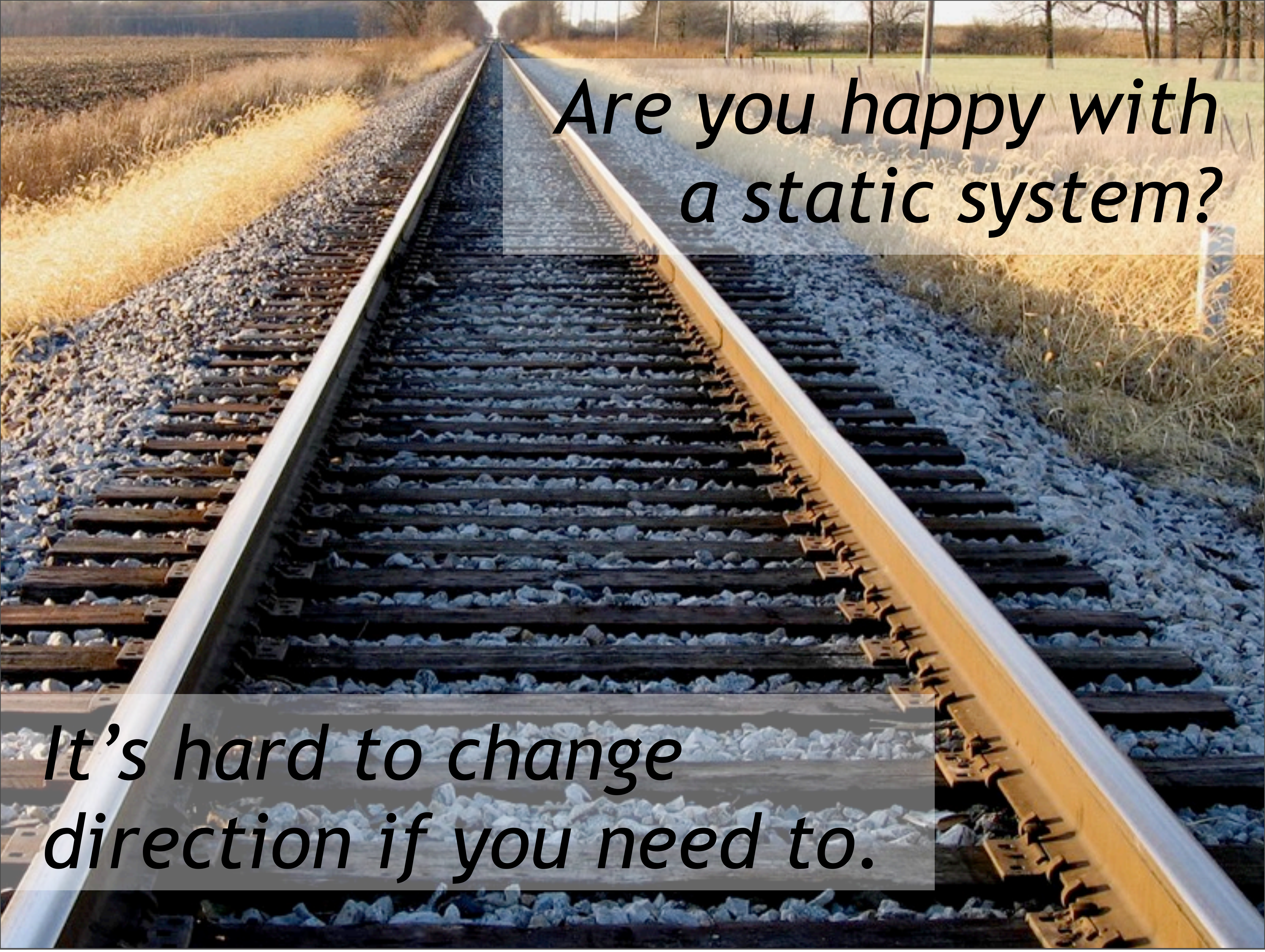
Your questions

Do you like spaghetti?

That's what you'll end up with JBOWS.

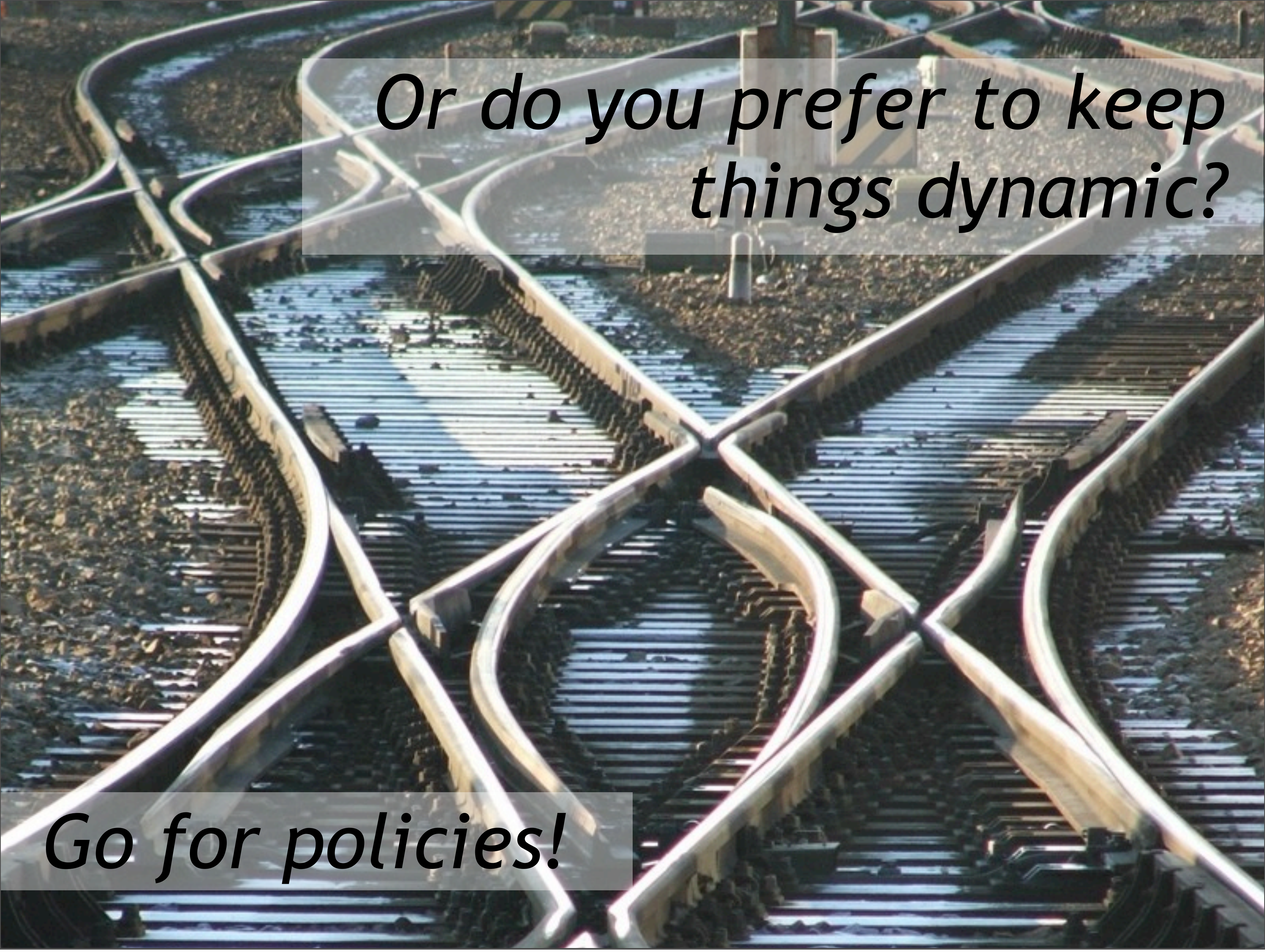
Or do you prefer to keep track of your services?

Use a service registry!



Are you happy with a static system?

It's hard to change direction if you need to.



Or do you prefer to keep things dynamic?

Go for policies!

Do you feel good when you don't see what's going on?

You might need to dig a bit deeper.

Or do you prefer to keep in control?

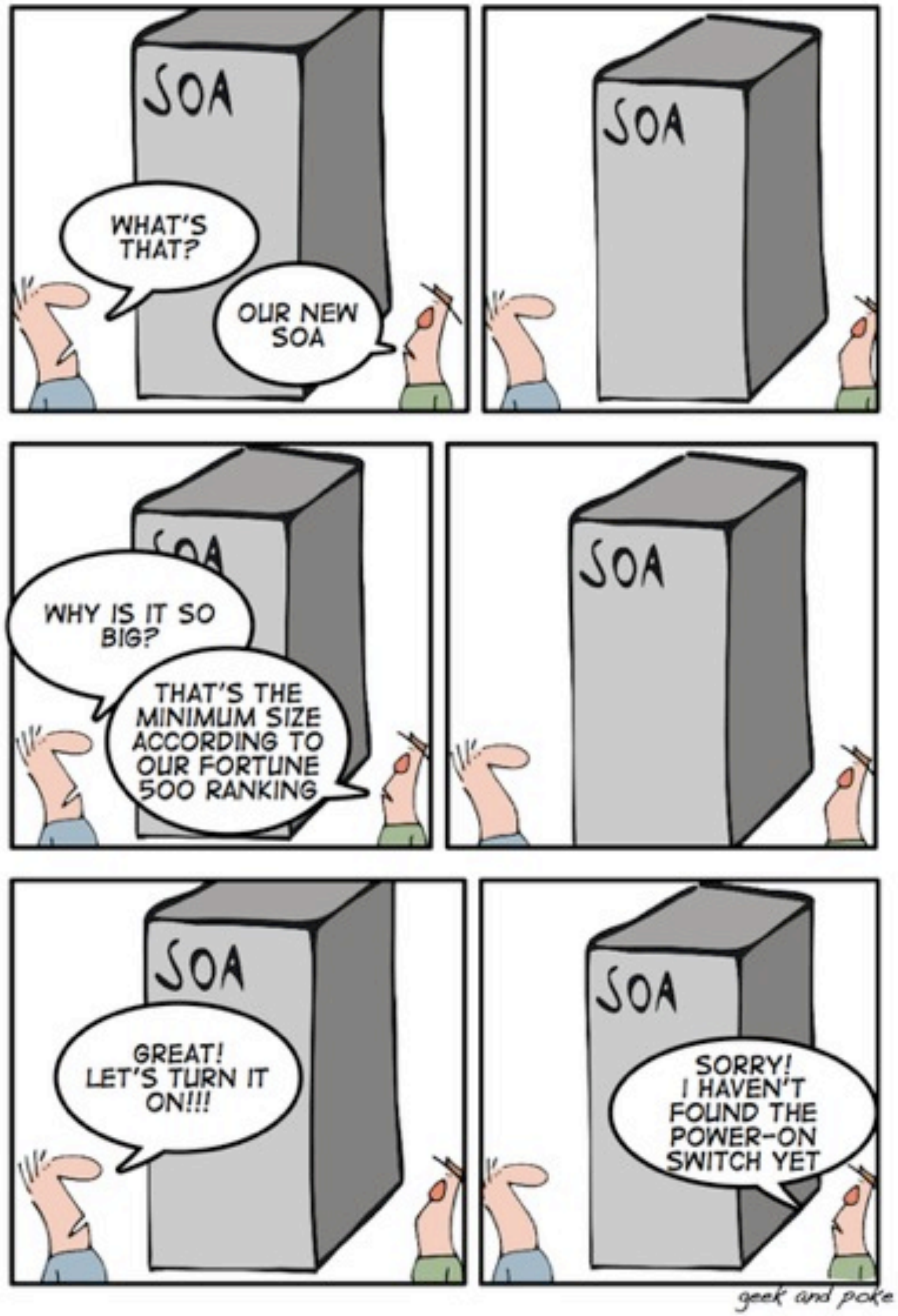
Service monitoring is essential!

Remote configuration

Policies Monitoring

Orchestration

Repository Registry


SOA
WHAT'S THAT?
OUR NEW SOA
SOA
SOA
WHY IS IT SO BIG?
THAT'S THE MINIMUM SIZE ACCORDING TO OUR FORTUNE 500 RANKING
SOA
SOA
GREAT! LET'S TURN IT ON!!!
SOA
SORRY! I HAVEN'T FOUND THE POWER-ON SWITCH YET
geek and poke

global market leader in logistics
annual turnover 54 billion Euro
500.000 employees worldwide

Deutsche Post DHL

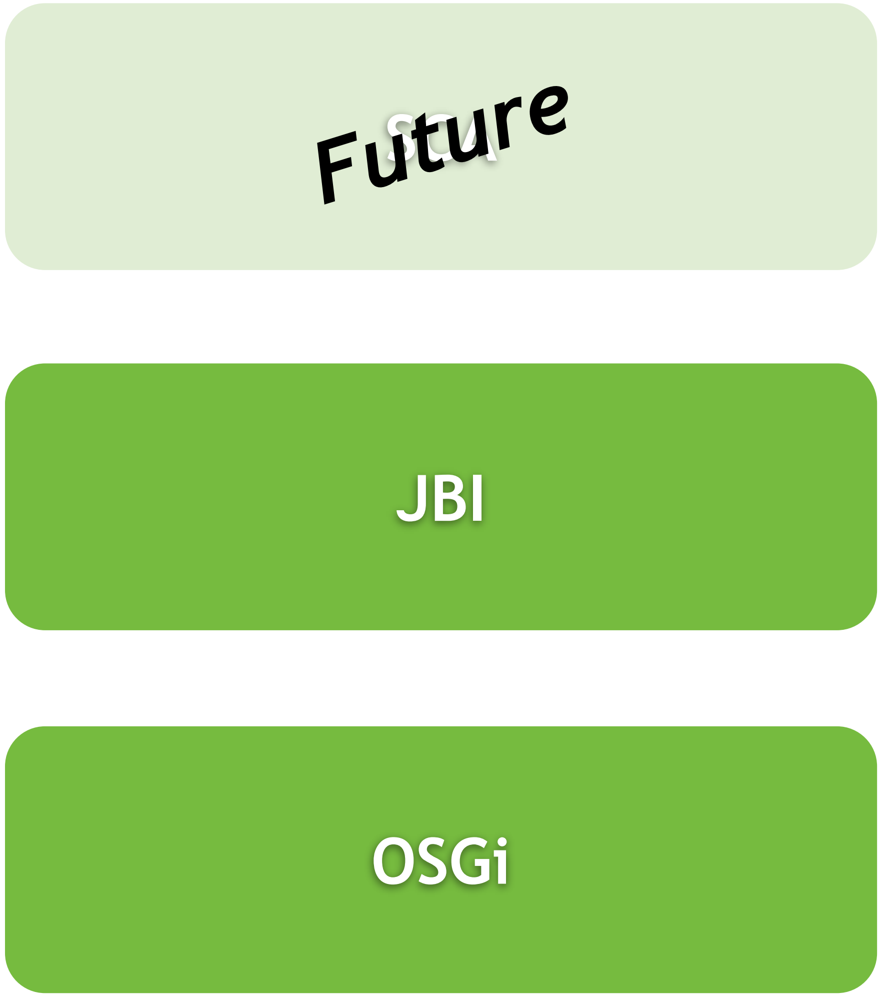
European SOA pioneer
started building a SOA in 1999
announced open source strategy in 2007

SWORDFISH

We’re on a mission

“The goal of the Swordfish project is to provide an ***extensible SOA runtime framework*** *based on the proven Eclipse Equinox runtime technology. The framework is designed to be complemented by additional open source components such as a service registry, a messaging system, a process engine etc. to form a comprehensive open source SOA runtime environment based on both established and emerging open standards.”*

Swordfish will eventually be based on all three relevant standards in the SOA space



Programming model
Assembly description format

Messaging abstraction
Message routing

Component model
Module deployment mechanism
Classloading

What’s that buzz about JBI? I thought it was dead?

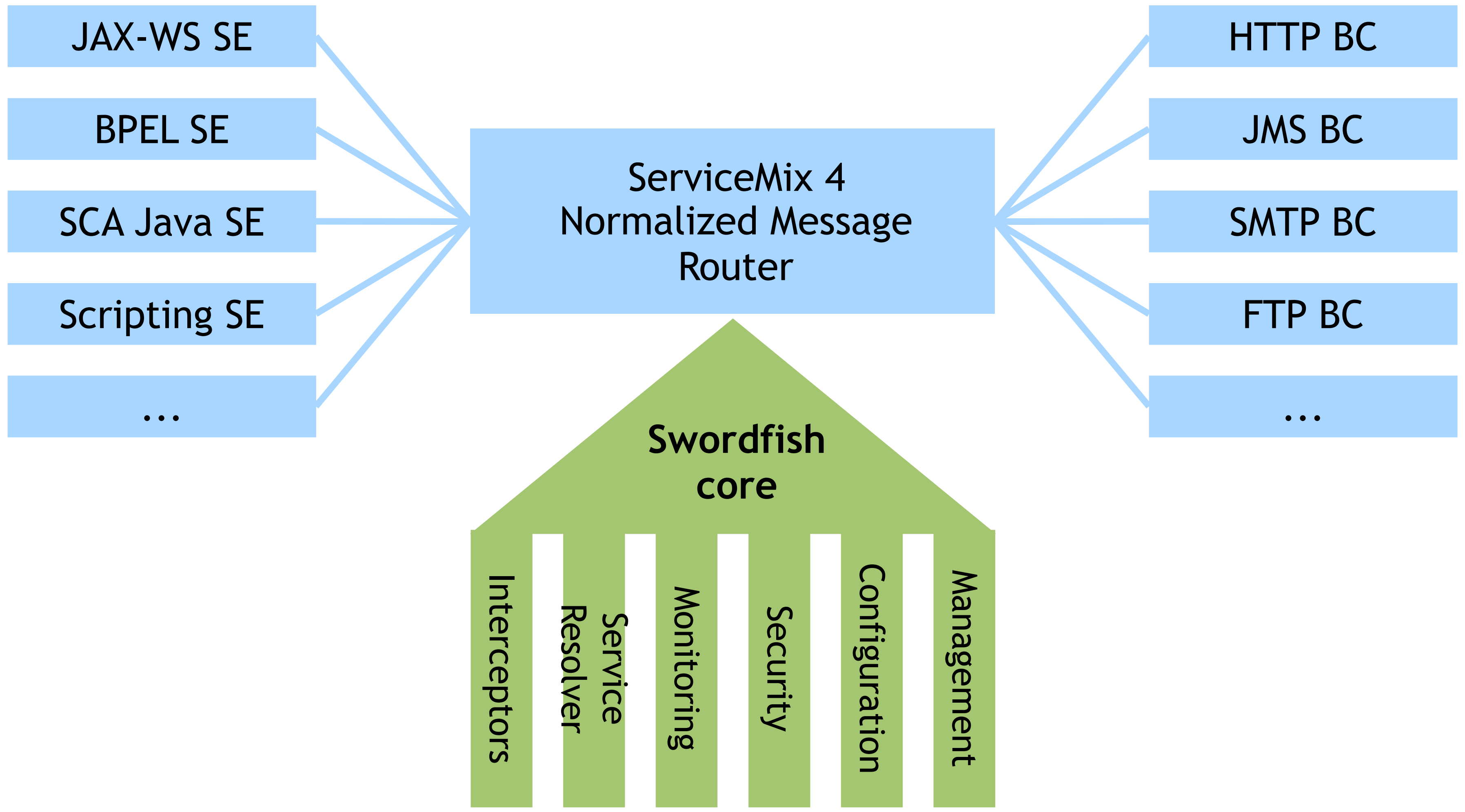
Then Apache, ow2 and Java.net are hosting zombies!

PEtALS

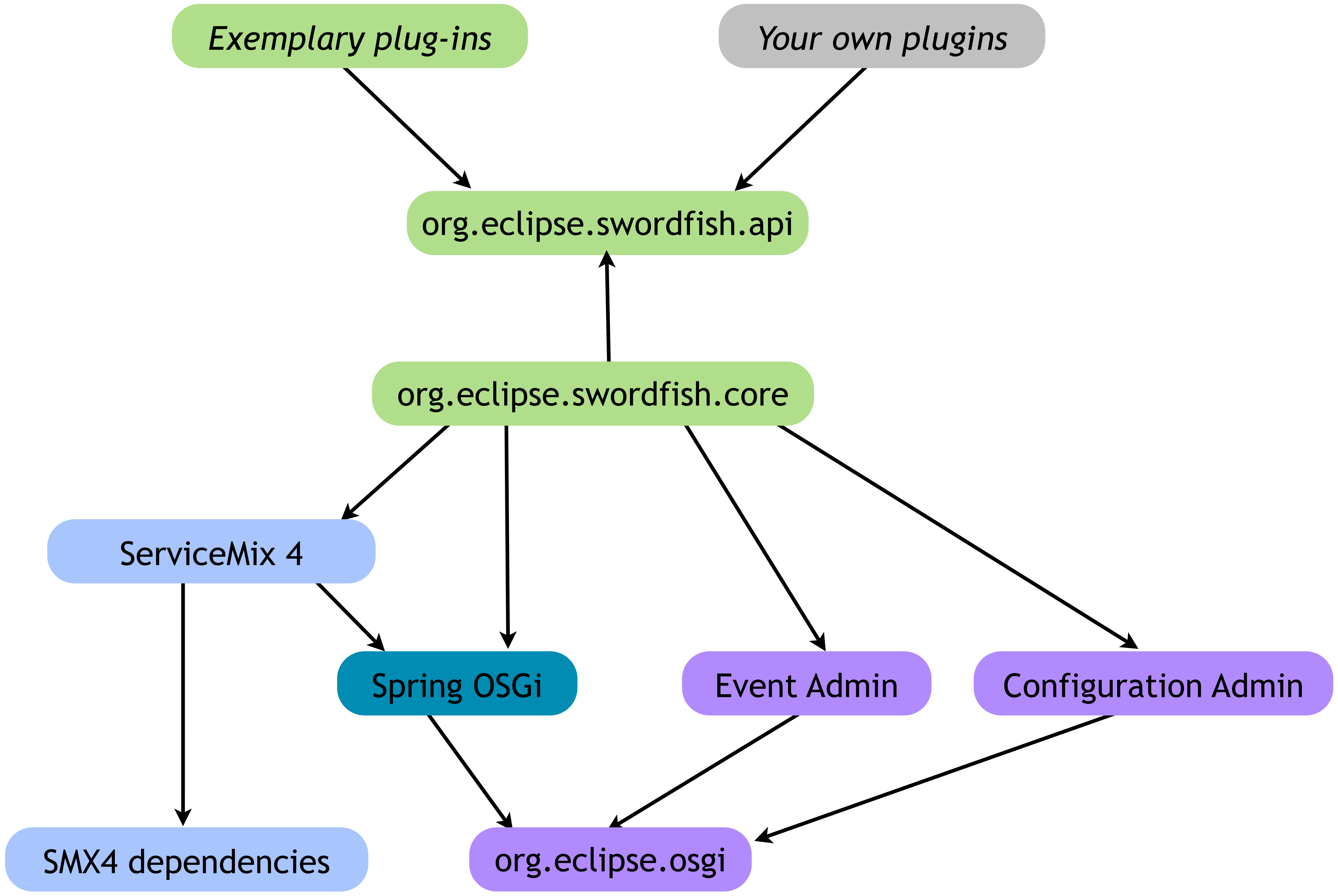
OpenESB
The *Open* Enterprise Service Bus

ServiceMix

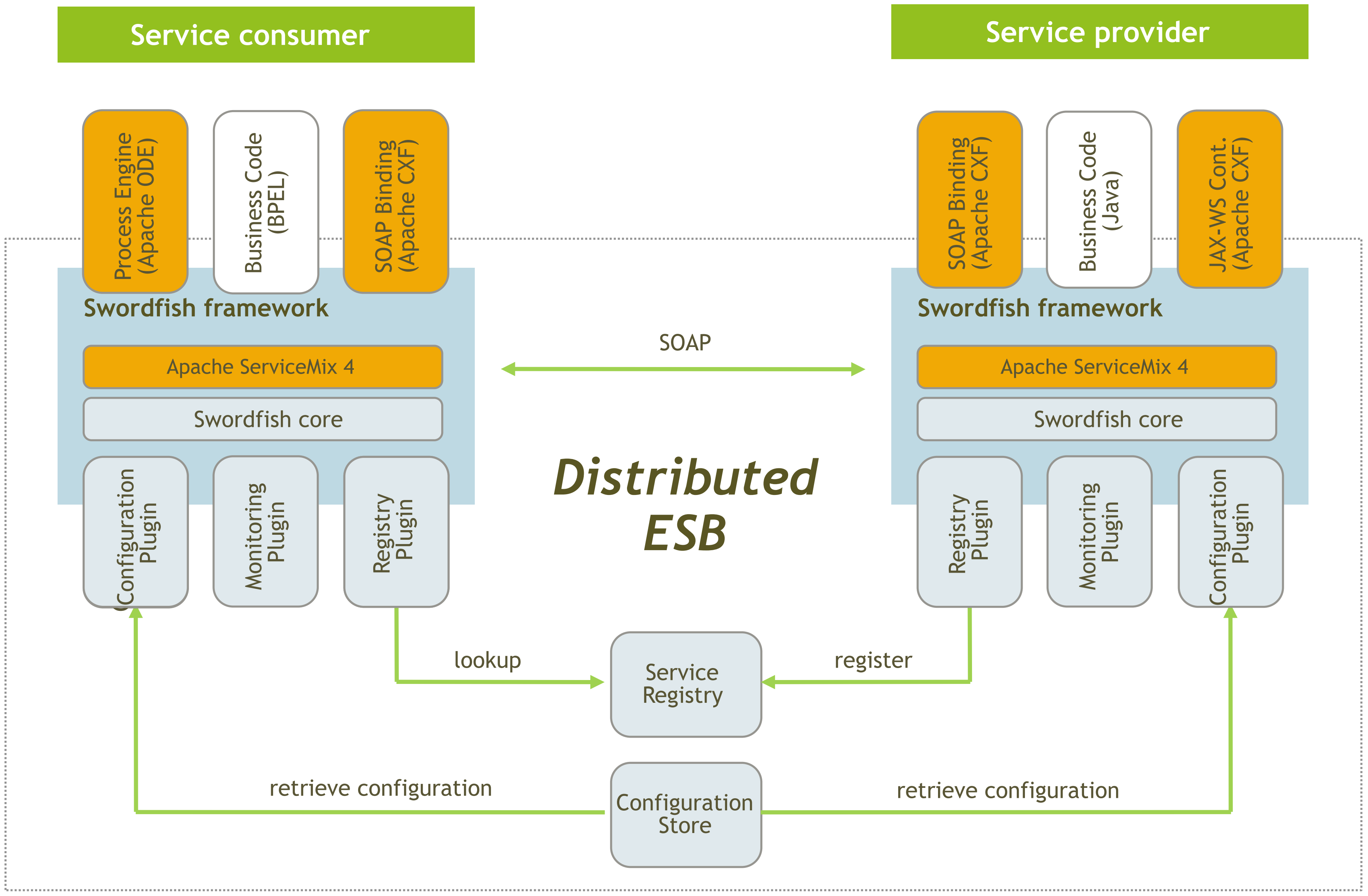
ServiceMix provides the routing facility and loads of components...



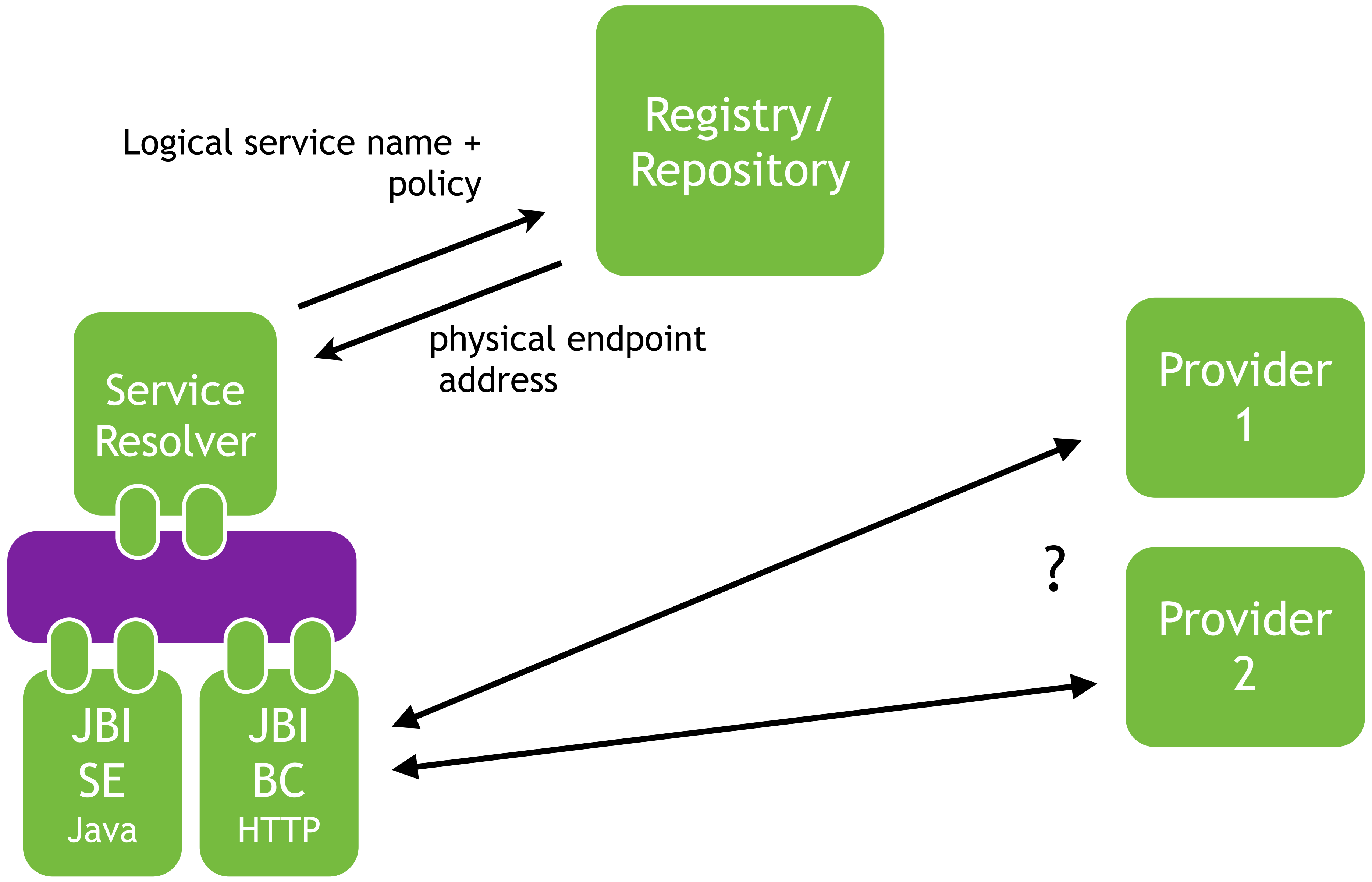
...and Swordfish adds a framework for the missing parts.


Exemplary plug-ins
Your own plugins
org.eclipse.swordfish.api
org.eclipse.swordfish.core
ServiceMix 4
Spring OSGi
Event Admin
Configuration Admin
SMX4 dependencies
org.eclipse.osgi

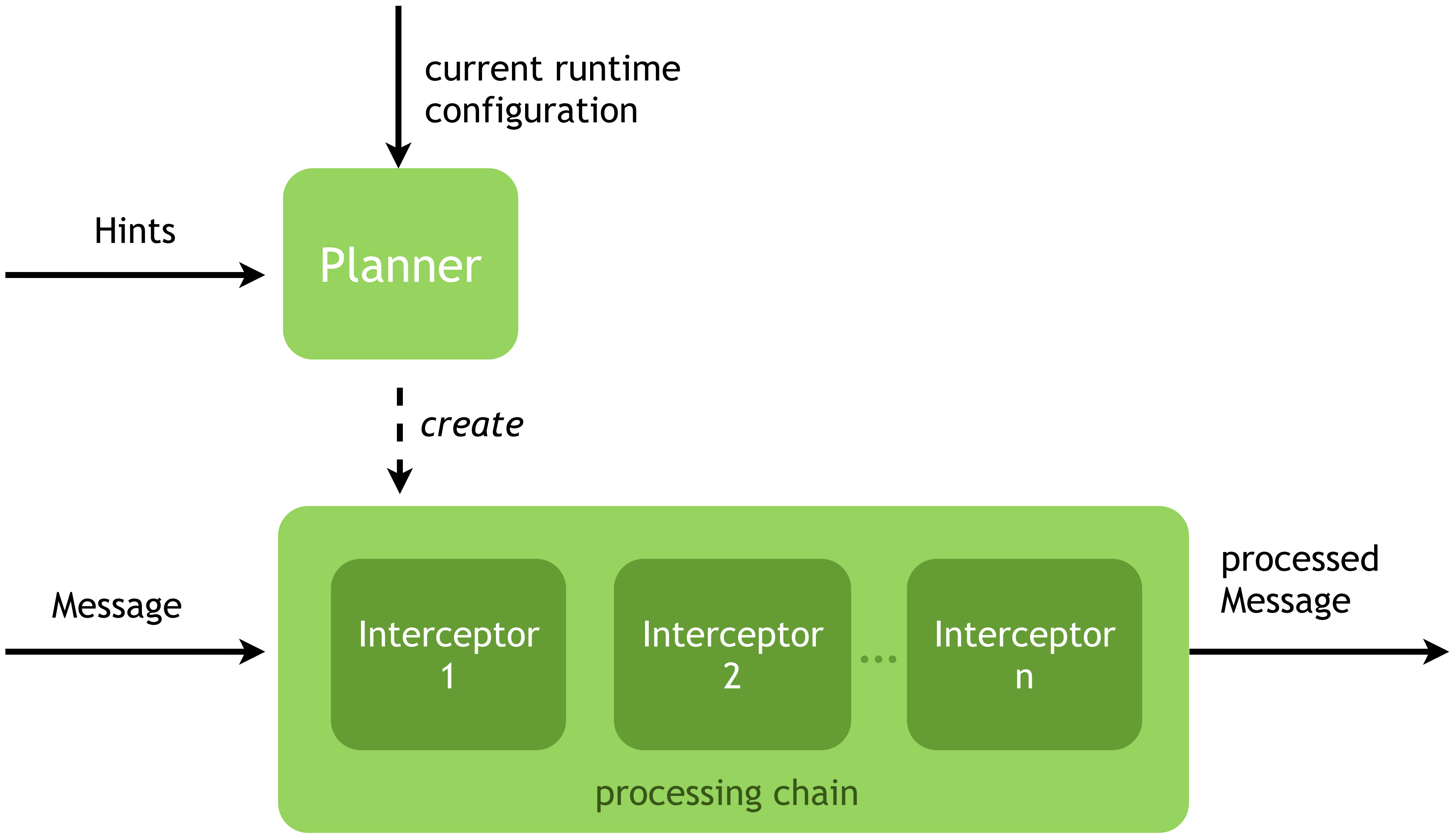
Swordfish is inherently based on the Distributed ESB pattern

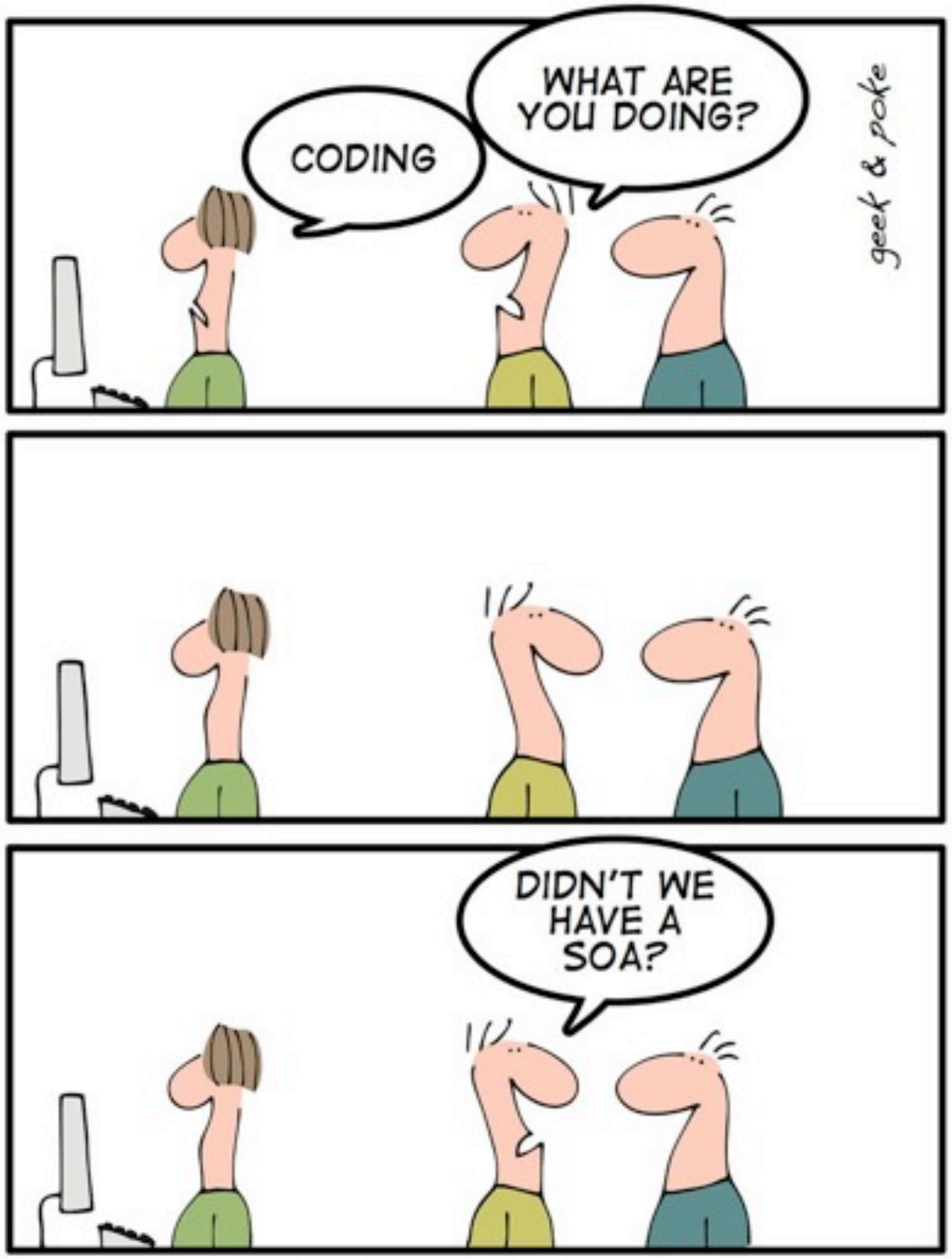


Service registry lookup



Policy-driven message processing





TROUGH OF DISILLUSIONMENT

Extending Swordfish

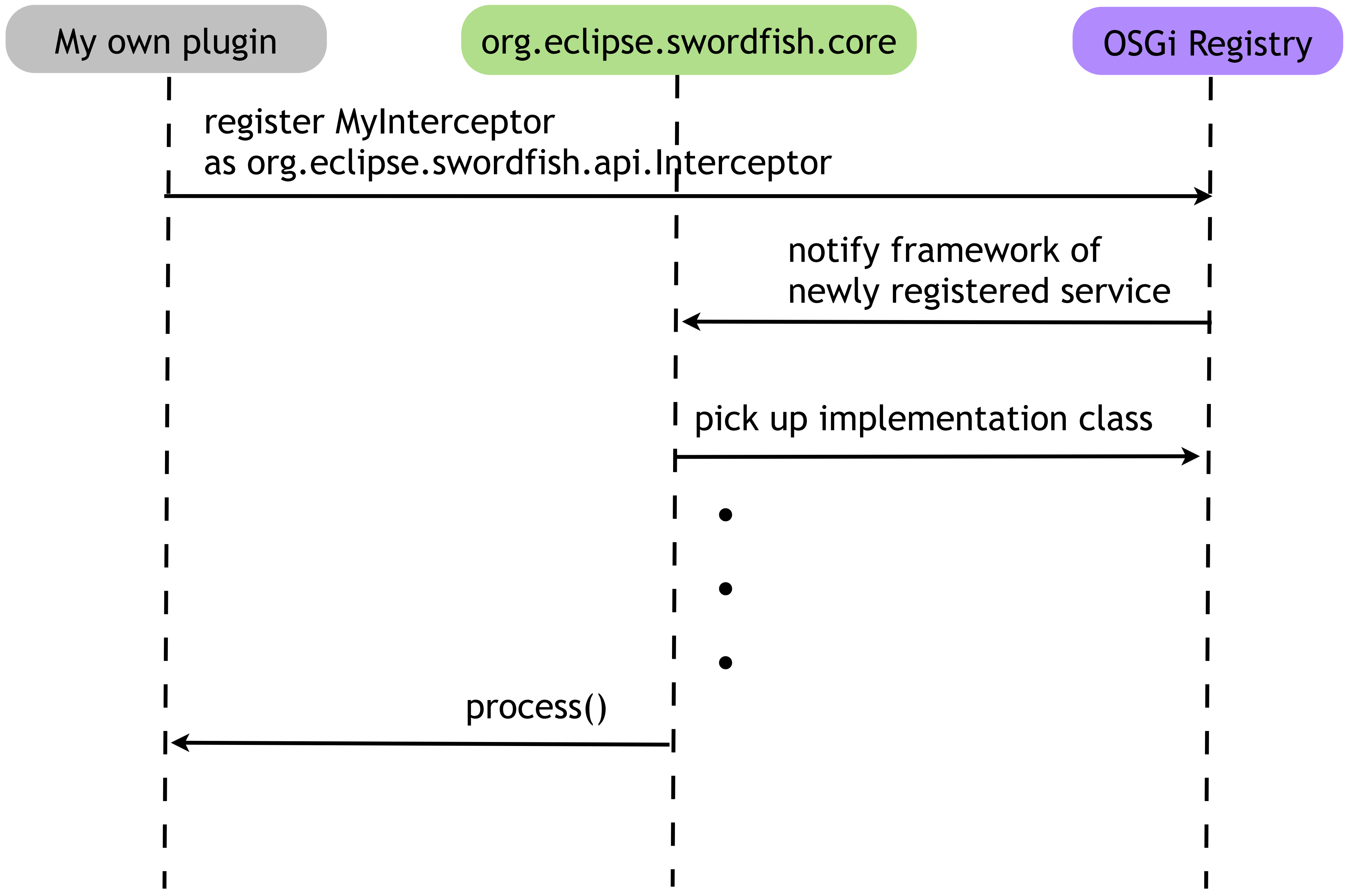
Slides are getting kinda dull now.
Yikes!

I apologize.

There are some things to know about Swordfish extensions

- Extensions in Swordfish are not extensions in the classical Eclipse sense – we don’t use the plugin registry
- Just register your extension component as an OSGi service, there are multiple ways to do it:
  - manually implement an Activator and intantiate and register your OSGi service there
  - use Spring DM
  - use OSGi DS (now comes with Equinox, good tooling in PDE)
  - use any other service activation framework (iPOJO, ...)

Bottomline: Use whatever makes most sense to you!


My own plugin
org.eclipse.swordfish.core
OSGi Registry
register MyInterceptor
as org.eclipse.swordfish.api.Interceptor
notify framework of
newly registered service
pick up implementation class
•
•
•
process()

The Swordfish framework currently provides four APIs for custom implementations

General interceptor API

Service resolver API

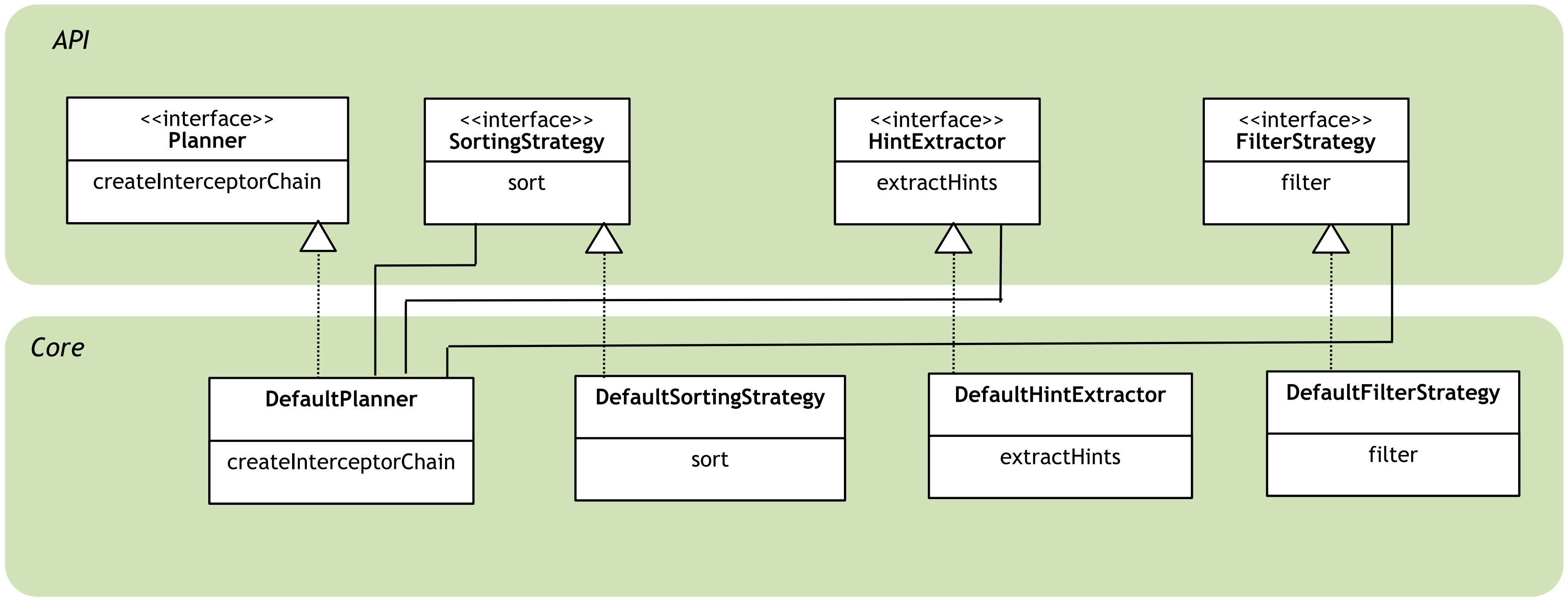
Event API

Configuration API

The General interceptor API

- Swordfish hooks into the Normalized Message Router in Servicemix, intercepts all messages flowing back and forth and feeds them into an interceptor chain
- Interceptors in that chain can do (almost) anything with a message: read and modify payload, read and modify properties, re-route etc.
- The interceptor chain is dynamically built from all registered instances of *Interceptor*
- A component called *Planner* is responsible for building the chain
- A default implementation of *Planner* (*DefaultPlanner*) is part of the Swordfish core

The default planner implementation uses a three step process to create the interceptor chain

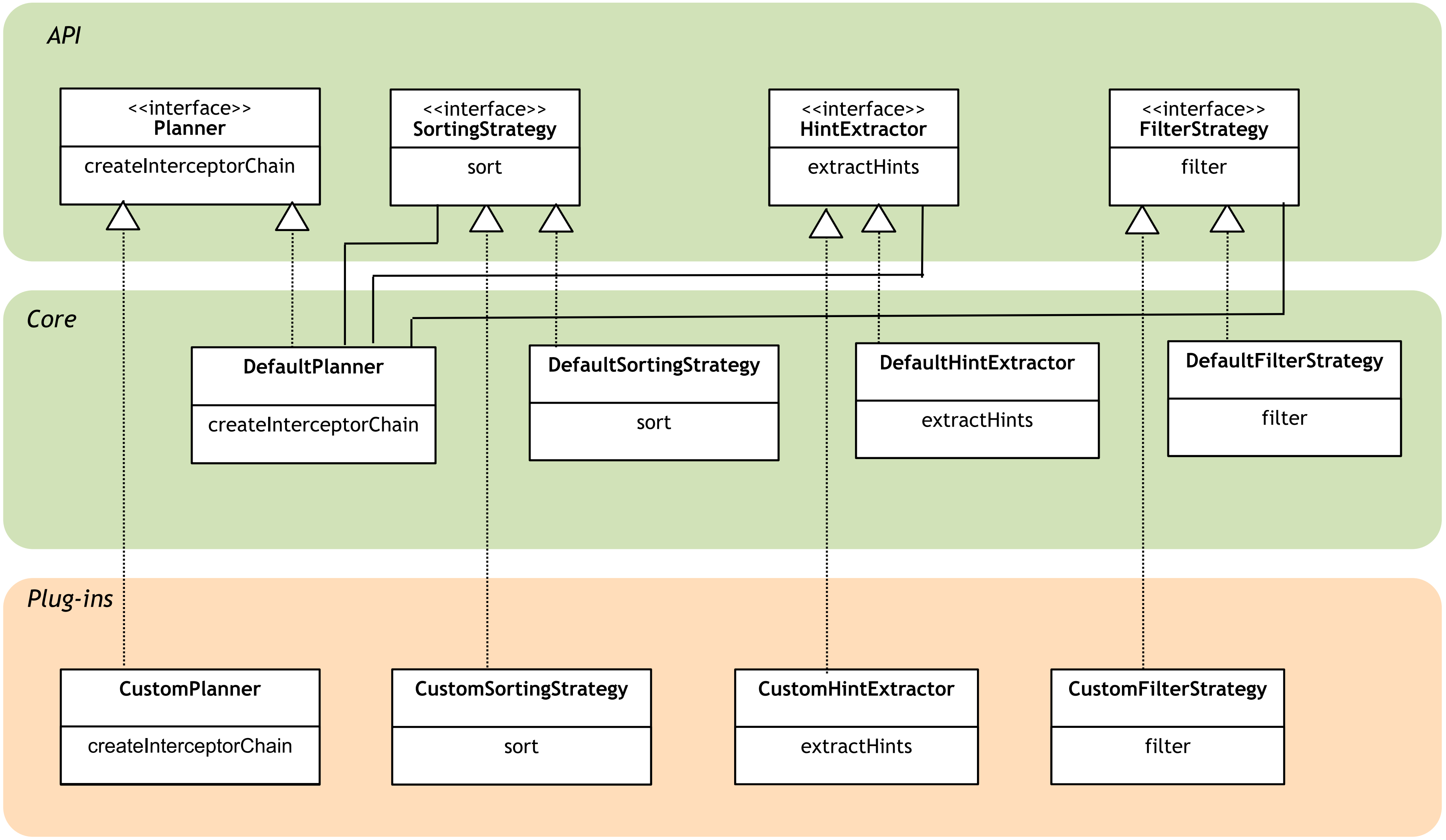


Step 1: Apply a *SortingStrategy* to sort the unordered set of Interceptors according to some criteria (e.g. priority or satisfaction of pre- and post-conditions)

Step 2: Apply a *HintExtractor* to extract any number of *Hints* from the MessageExchange that help define how the message should be processed (e.g. Policies)

Step 3: Apply a *FilterStrategy* to remove Interceptors from the ordered list that are not required according to the *Hints* and possibly other criteria

The interceptor framework can be extended in a number of ways



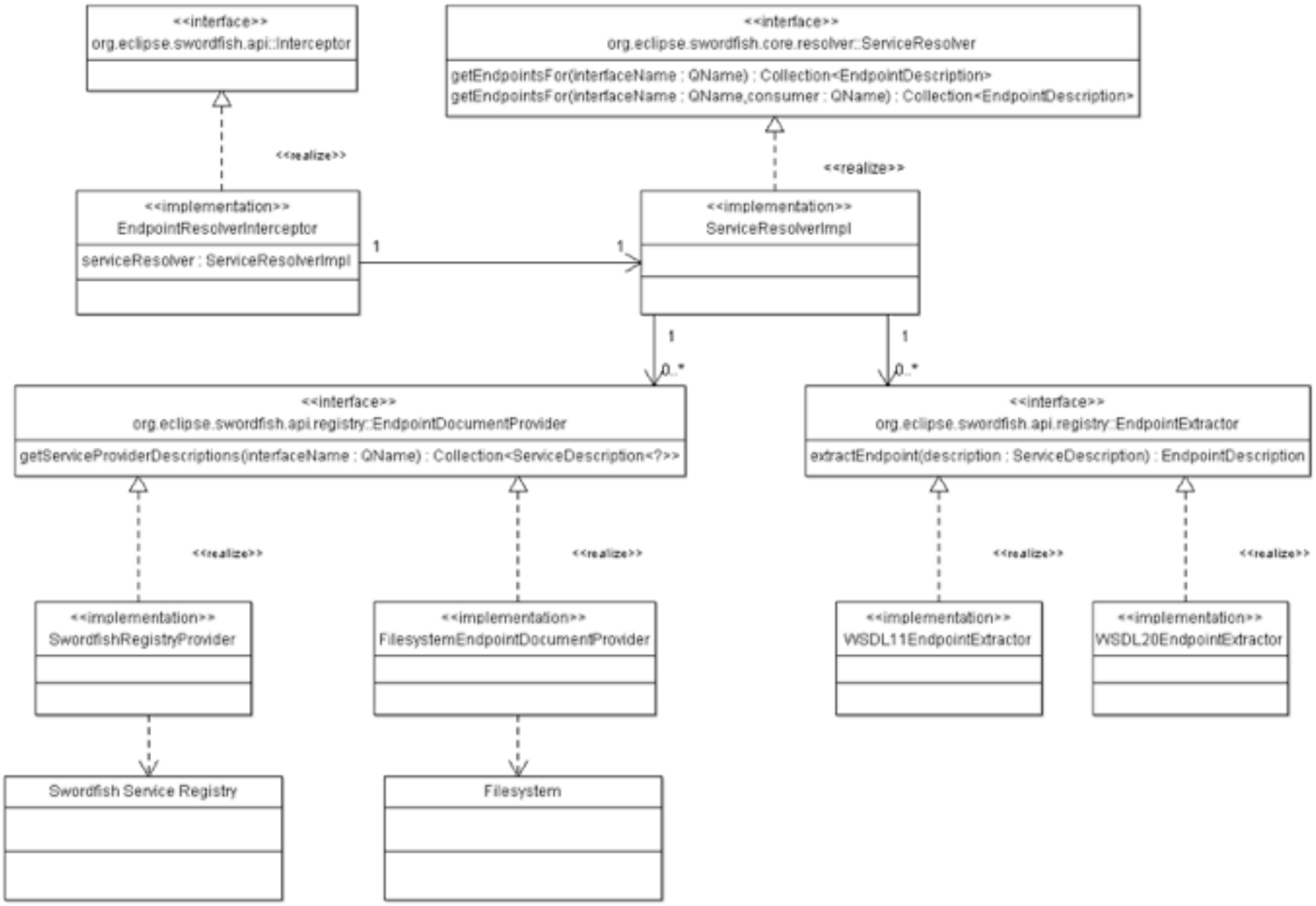
The Service resolver API

- The Service Resolver API builds upon the general interceptor API
- A special kind of interceptor (*ServiceResolverInterceptor*) is provided as part of the Swordfish core
- The *ServiceResolverInterceptor* is reponsible for translating the (logical) service interface name carried inside a message exchange into a physical endpoint address where the message exchange will be ultimately sent to
- The *ServiceResolverInterceptor* collaborates with a number of other (replaceable) components in order to perform its task

The Service resolver API — Components

- *EndpointDocumentProvider*: Provides instances of *EndpointDocument* of some type for a given service interface name
- *EndpointExtractor*: Extracts the physical endpoint information from a specific type of *EndpointDocument*
- *PolicyDefinitionProvider*: Provides instances of *PolicyDefinition* of some type for a given logical service provider name
- *PolicyExtractor:* Extracts the policy information from a specific type of PolicyDefinition
- *PolicyProcessor:* Match policies of some type and calculate an effective agreed policy if possible

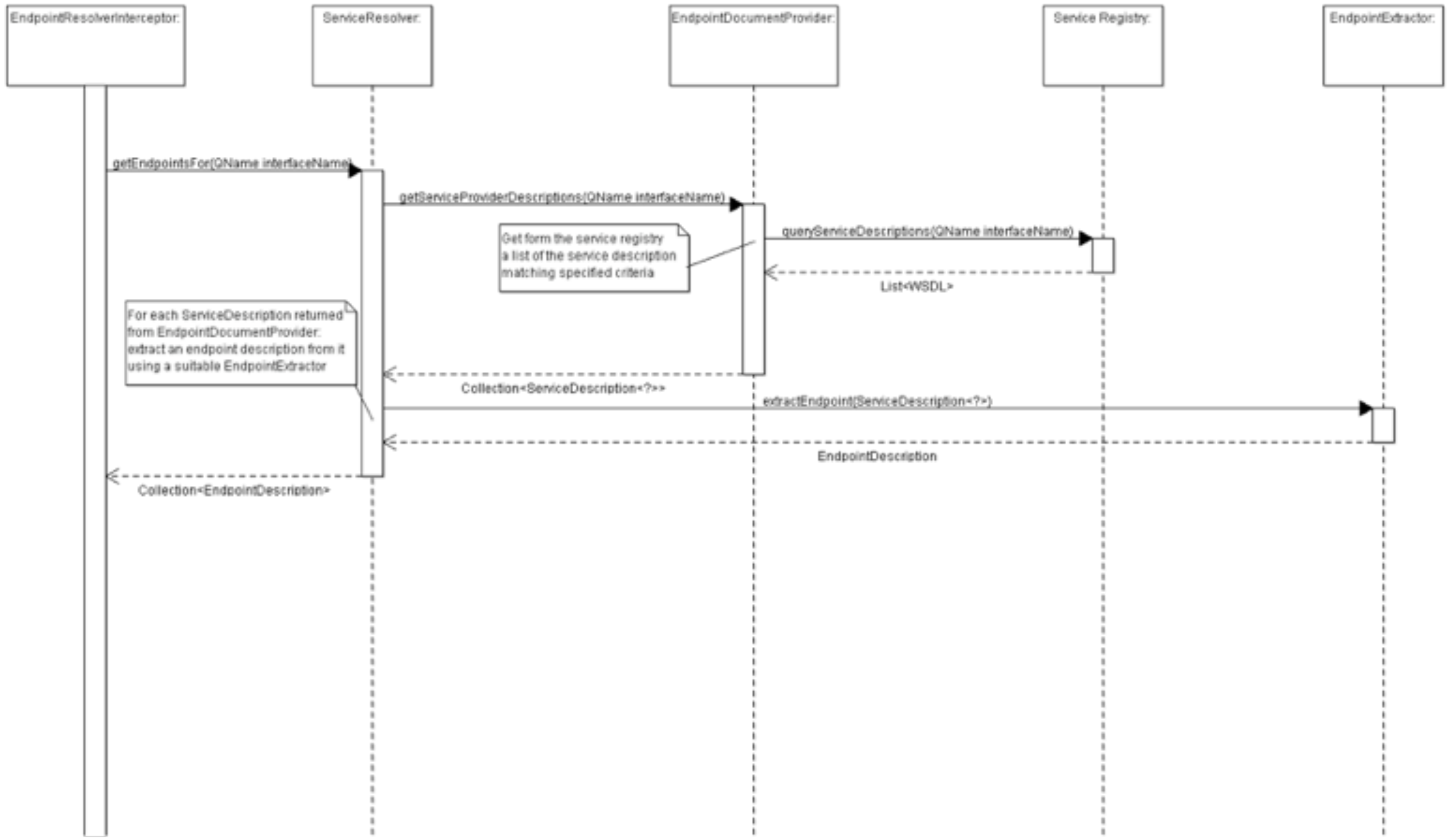
The Service resolver API — class diagram



<<interface>>
org.eclipse.swordfish.api::Interceptor
<<interface>>
org.eclipse.swordfish.core.resolver::ServiceResolver
getEndpointsFor(interfaceName : QName) : Collection<EndpointDescription>
getEndpointsFor(interfaceName : QName,consumer : QName) : Collection<EndpointDescription>
<<realize>>
<<realize>>
<<implementation>>
EndpointResolverInterceptor
serviceResolver : ServiceResolverImpl
1
1
<<implementation>>
ServiceResolverImpl
1
0..*
1
0..*
<<interface>>
org.eclipse.swordfish.api.registry::EndpointDocumentProvider
getServiceProviderDescriptions(interfaceName : QName) : Collection<ServiceDescription<?>>
<<interface>>
org.eclipse.swordfish.api.registry::EndpointExtractor
extractEndpoint(description : ServiceDescription) : EndpointDescription
<<realize>>
<<realize>>
<<realize>>
<<realize>>
<<implementation>>
SwordfishRegistryProvider
<<implementation>>
FilesystemEndpointDocumentProvider
<<implementation>>
WSDL11EndpointExtractor
<<implementation>>
WSDL20EndpointExtractor
Swordfish Service Registry
Filesystem

http://wiki.eclipse.org/Swordfish_Documentation:_Architecture:_Service_Registry_Plugin

Default resolver — sequence diagram



EndpointResolverInterceptor:
ServiceResolver:
EndpointDocumentProvider:
Service Registry:
EndpointExtractor:
getEndpointsFor(QName interfaceName)
getServiceProviderDescriptions(QName interfaceName)
queryServiceDescriptions(QName interfaceName)
Get form the service registry a list of the service description matching specified criteria
List<WSDL>
For each ServiceDescription returned from EndpointDocumentProvider: extract an endpoint description from it using a suitable EndpointExtractor
Collection<ServiceDescription<?>>
extractEndpoint(ServiceDescription<?>)
EndpointDescription
Collection<EndpointDescription>

http://wiki.eclipse.org/Swordfish_Documentation:_Architecture:_Service_Registry_Plugin

Service resolution extension options

- Replace the whole *ServiceResolverInterceptor* with a custom implementation that potentially applies a completely different strategy
- Implement a *ServiceResolver*
- Use the default *ServiceResolver* and just implement the components it collaborates with (as shown before)

The Event API

- The Event system is the basis for Swordfish's monitoring capabilities
- The Event API builds upon the OSGi EventAdmin Service
- Swordfish core generates Events of type *TrackingEvent* for each MessageExchange flowing through the NMR
- Other pre-defined *Event* types include *OperationEvent* and *ConfigurationEvent*

Event system extension options

- Implement your own types of Events
- Implement Event sinks that forward events to a backend system, e.g. for complex event processing (CEP)

The Configuration API

- The Configuration API builds upon the OSGi ConfigurationAdmin service
- A *ConfigurationAgent* receives configurations from a *ConfigurationSource* (possibly a remote one) and provides them to other components through the OSGi Configuration Admin service
- A component that implements *ConfigurationConsumer* receives updated configurations as they become available

Configuration system extension options

- Implement a custom *ConfigurationAgent*
- Use the provided default *ConfigurationAgent* and implement a *ConfigurationSource* that retrieves configurations from somewhere (file system, remote repository, bundles,...)

Where we are and where we’re heading

SWORDFISH

7 committers (5 full time)

5 contributors

release 0.9 out as part of Galileo

next release 1.0 planned for fall 2009

Swordfish is part of a larger initiative called Eclipse SOA

	Galileo June 2009 Release 0.9	Eclipse Summit Europe November 2009 Release 1.0	EclipseCon March 2010 Release 1.1	Eclipse Release Summer 2010 Release 2.0
ESB	Swordfish framework basic plugins basic tool support	Integration with open source security framework	Test support	Full enterprise ESB
Process Orchestration	BPEL process engine	Integration with open source BPM suite (Spagic)		
Registry & Repository	Basic runtime registry	Advanced runtime registry/service locator basic repository	Service repository w/service lifecycle management	Full enterprise service repository
Management	JMX-based management	Plugin for integration into Hyperic HQ		
Data Integration		Plugins for open source ETL tools (e.g. Talend)	Plugins for open source EDI tools (e.g. Smooks)	
Service/Business Activity Monitoring		Service activity reporting	Complex event processing	Full business activity monitoring

The goal of the initiative is a common and extensible SOA platform

Goals of the Eclipse SOA initiative

- Deliver a common and extensible SOA platform (tooling and runtime) based on Equinox
- Foster adoption of this platform by vendors and system integrators
- Achieve interoperability between products provided by the participating vendors

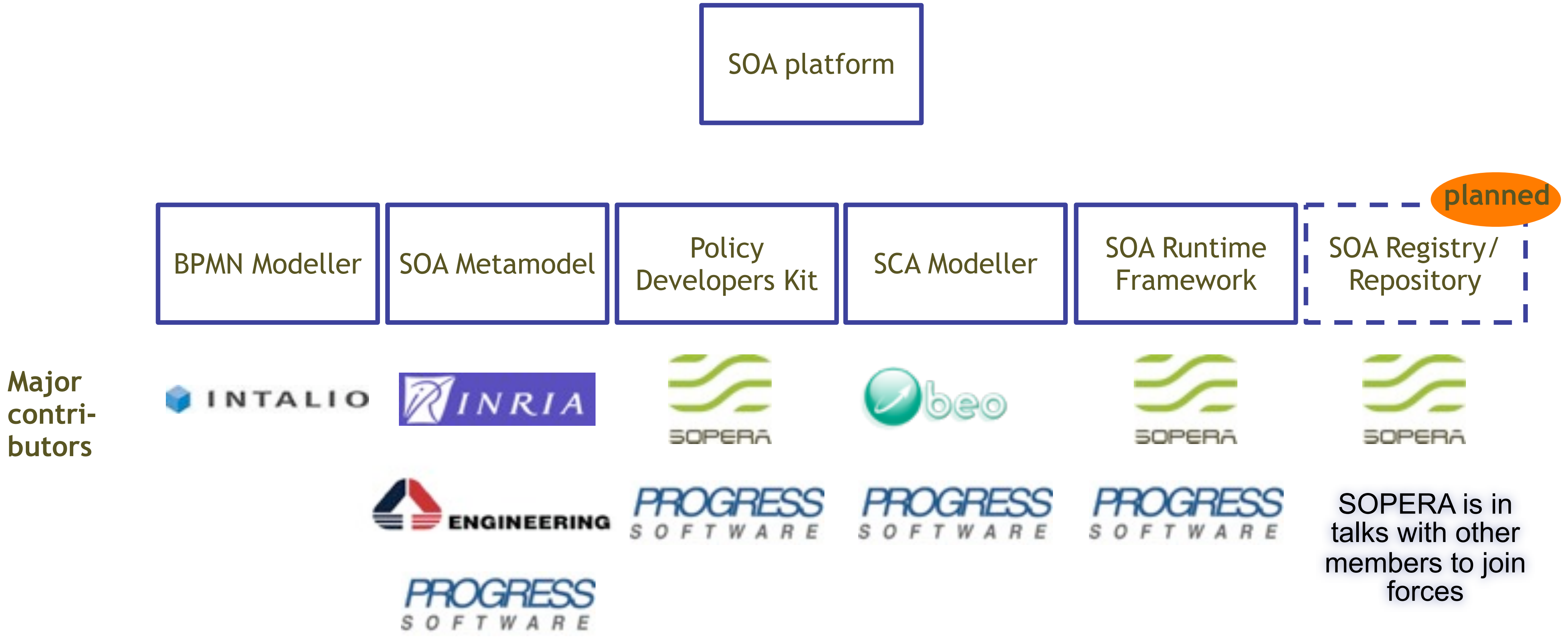
Workstream ①: Forming a new top-level project

The goal is to be the home for all SOA relevant projects in order to ease the development to a platform including tooling and runtime

Workstream ②: Forming a new industry working group

The goal is to establish a new brand and to define criteria under which conditions vendors and system integrators are allowed to use the brand

The new top-level project becomes the home for all SOA-related projects



The new IWG will develop and govern the use of a new brand for vendors and SIs

Deliverables of the SOA IWG

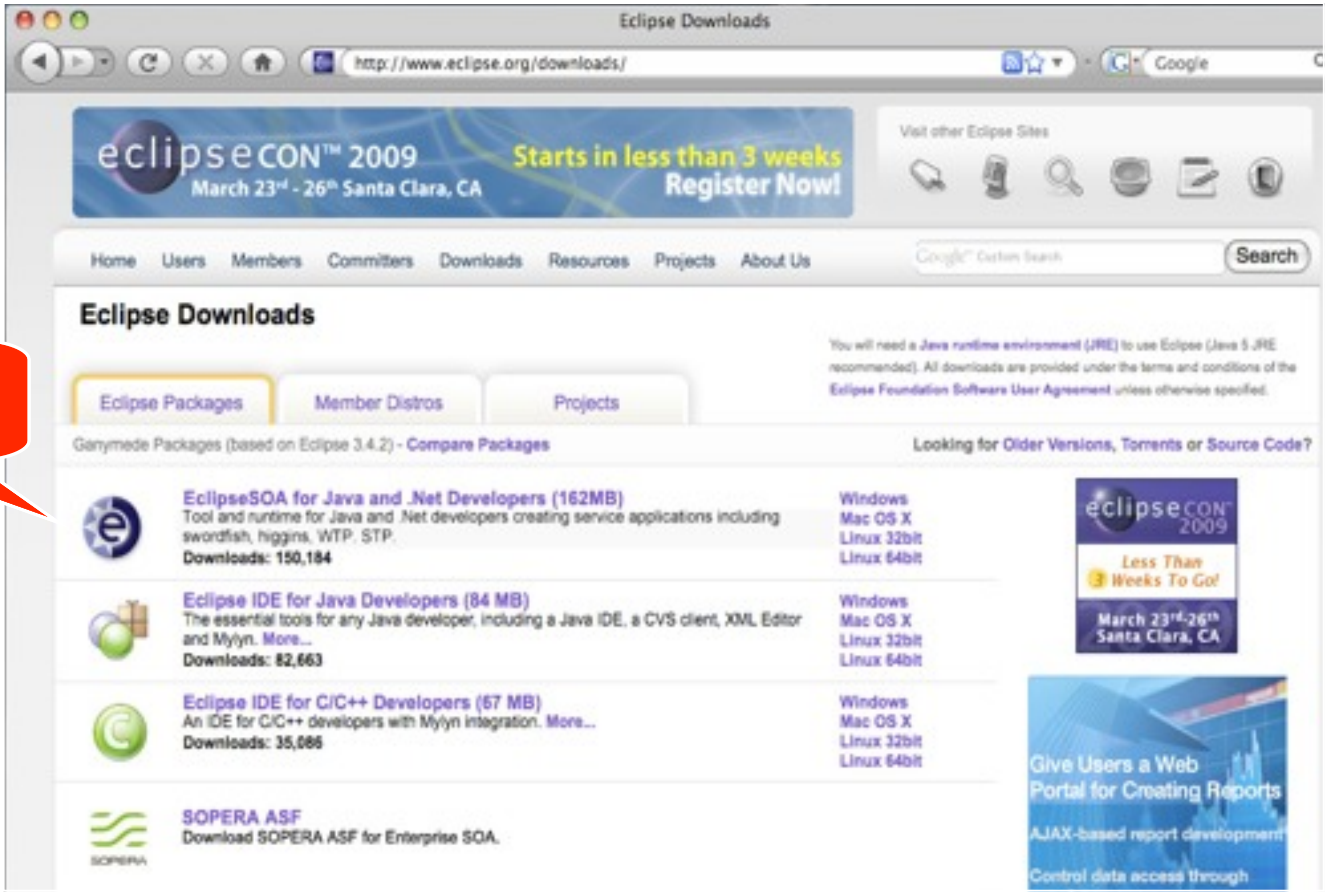
- Defines requirements to be implemented by the resources of the participants in the relevant projects
- Develops a new brand
- Defines criteria for the use of the brand

Members of the SOA IWG

- Steering Committee Member:
  - Strategic or Enterprise Developer Member of the Foundation
  - Minimum 3 developers implementing the defined requirements
- Member Participant
  - Solution or Committer Member of the Foundation
  - Minimum 1 developer implementing the defined requirements

The brand will be used for an Eclipse package as well

Eclipse SOA will be available as a separate package


Eclipse Downloads
http://www.eclipse.org/downloads/
Google
eclipsecon™ 2009
March 23rd - 26th Santa Clara, CA
Starts in less than 3 weeks
Register Now!
Visit other Eclipse Sites
Home
Users
Members
Committers
Downloads
Resources
Projects
About Us
Google™ Custom Search
Search
Eclipse Packages
Member Distros
Projects
eclipsecon 2009
Less Than 3 Weeks To Go!
March 23rd-26th Santa Clara, CA
Give Users a Web Portal for Creating Reports
AJAX-based report development
Control data access through

Eclipse Downloads

You will need a Java runtime environment (JRE) to use Eclipse (Java 5 JRE recommended). All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Ganymede Packages (based on Eclipse 3.4.2) - Compare Packages

Looking for Older Versions, Torrents or Source Code?

EclipseSOA for Java and .Net Developers (162MB)
Tool and runtime for Java and .Net developers creating service applications including swordfish, higgins, WTP, STP.
Downloads: 150,184
Windows
Mac OS X
Linux 32bit
Linux 64bit

Eclipse IDE for Java Developers (84 MB)
The essential tools for any Java developer, including a Java IDE, a CVS client, XML Editor and Mylyn. More...
Downloads: 82,663
Windows
Mac OS X
Linux 32bit
Linux 64bit

Eclipse IDE for C/C++ Developers (67 MB)
An IDE for C/C++ developers with Mylyn integration. More...
Downloads: 35,086
Windows
Mac OS X
Linux 32bit
Linux 64bit

SOPERA ASF
Download SOPERA ASF for Enterprise SOA.

Learn it!

http://www.eclipse.org/swordfish/
http://wiki.eclipse.org/Swordfish

Get real with Swordfish!

Try it!

http://www.eclipse.org/swordfish/downloads/

Get involved!

Mailing list: swordfish-dev
Daily developer group chat on Skype

SWORDFISH