NordiaSoft

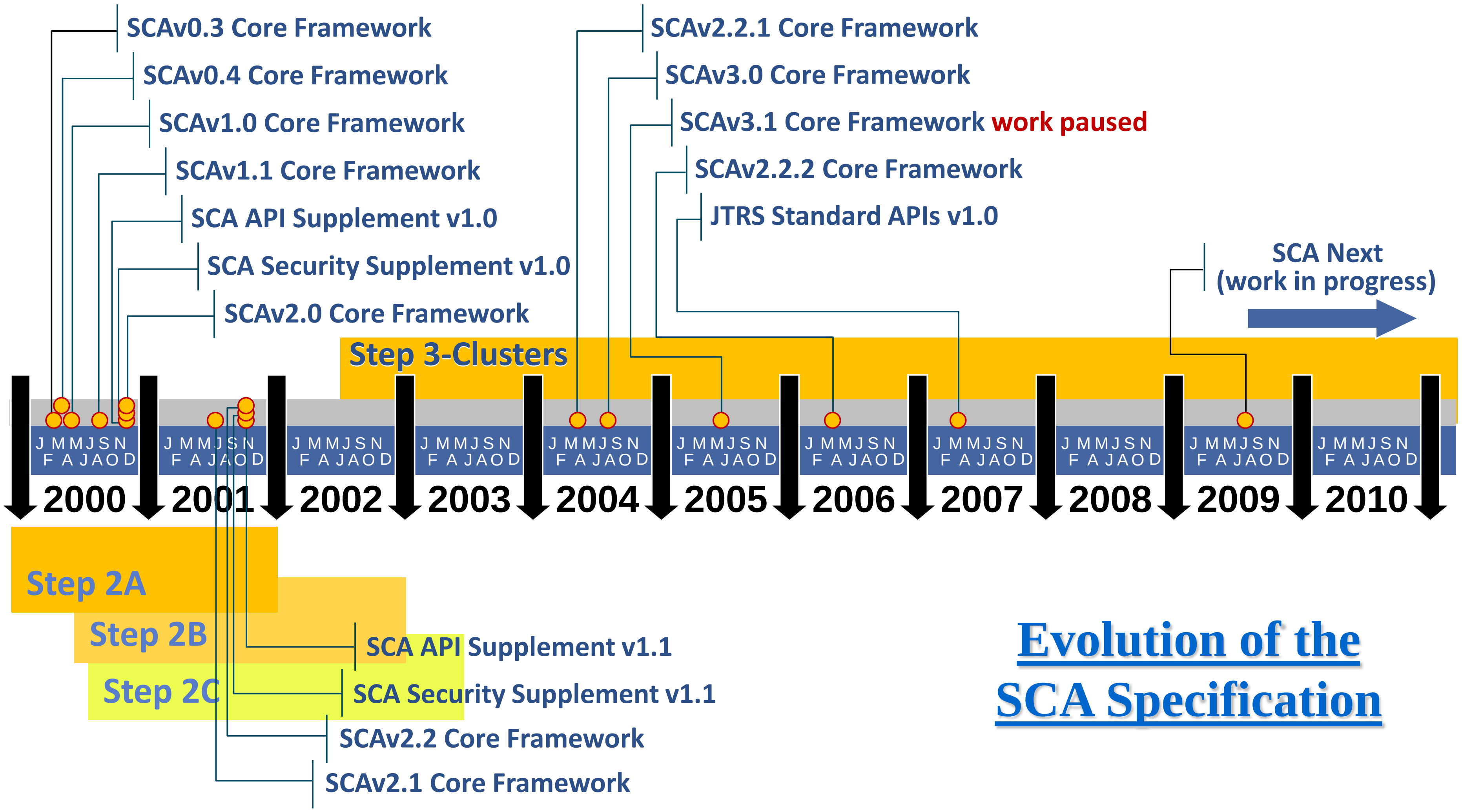
- Evolution of the SCA Specification: From v0.3 to v4.1 -

Official Launch of the Wireless Innovation Forum India Regional Committee.

New Delhi, India. June 19-20, 2014

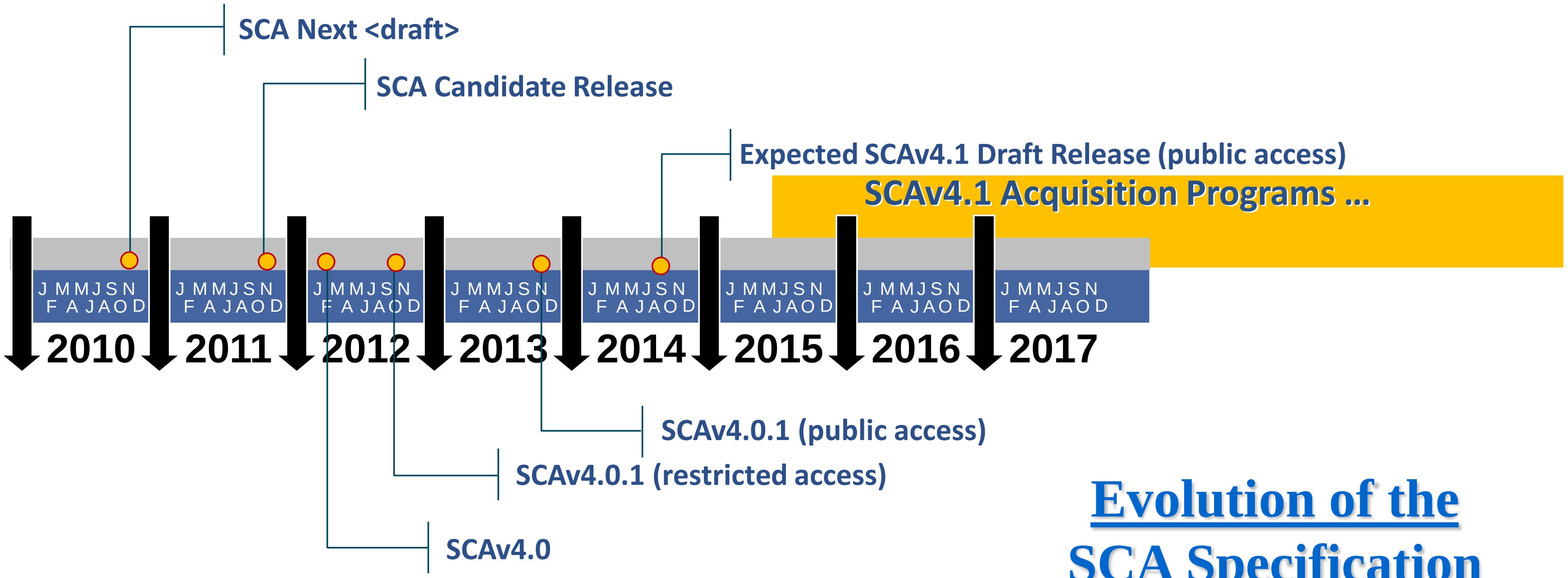
Juan Pablo Zamora Zapata, Ph.D.

Evolution of the SCA specification



Evolution of the SCA Specification

Evolution of the SCA specification


SCA Next <draft>
SCA Candidate Release
Expected SCAv4.1 Draft Release (public access)
SCAv4.1 Acquisition Programs …
J M M J S N F A J A O D
2010
2011
2012
2013
2014
2015
2016
2017
SCAv4.0.1 (public access)
SCAv4.0.1 (restricted access)
SCAv4.0

Evolution of the SCA Specification

Evolution of the SCA specification

- **The SCA specification is over 14 years old!**
  - The largest acquisition program in the world is the US JTRS program and it relies on SCAv2.2 and SCAv2.2.2
  - The ESSOR program relies on SCAv2.2.2 with some additions
  - SCAv2.2 and SCAv2.2.2 are the most popular versions of the SCA specification world wide
  - Currently, there hundreds of thousands of SCAv2.2.2 deployed
- **The SCA specification has always been about Software Components for real-time heterogeneous embedded systems**
  - SCAv4.1 adds a number of new features such as component scalability for smaller footprints and push registration for faster boot times
  - SCAv4.1 reintroduces backwards compatibility with SCAv2.2.2 applications

SCAv4.0.1 is Not Backwards Compatible

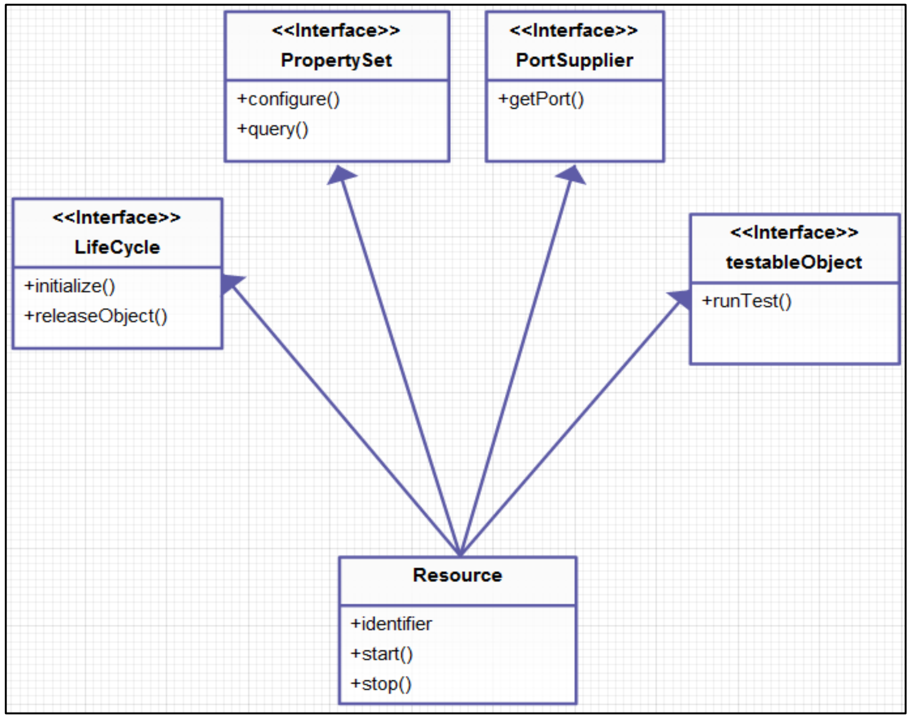
- **SCAv4.0.1 did not offer any level of backwards compatibility with SCAv2.2.2**
  - NordiaSoft was first to underline the lack of backwards compatibility
  - Presented at the SCAv4.1 Workshop held in November 2013 in San Diego
  - Requested support for backwards compatibility for SCA applications

- **SCAv4.0.1 introduced a number of new features that break backwards compatibility. Component Scalability is one example. Push registration for components is another example.**

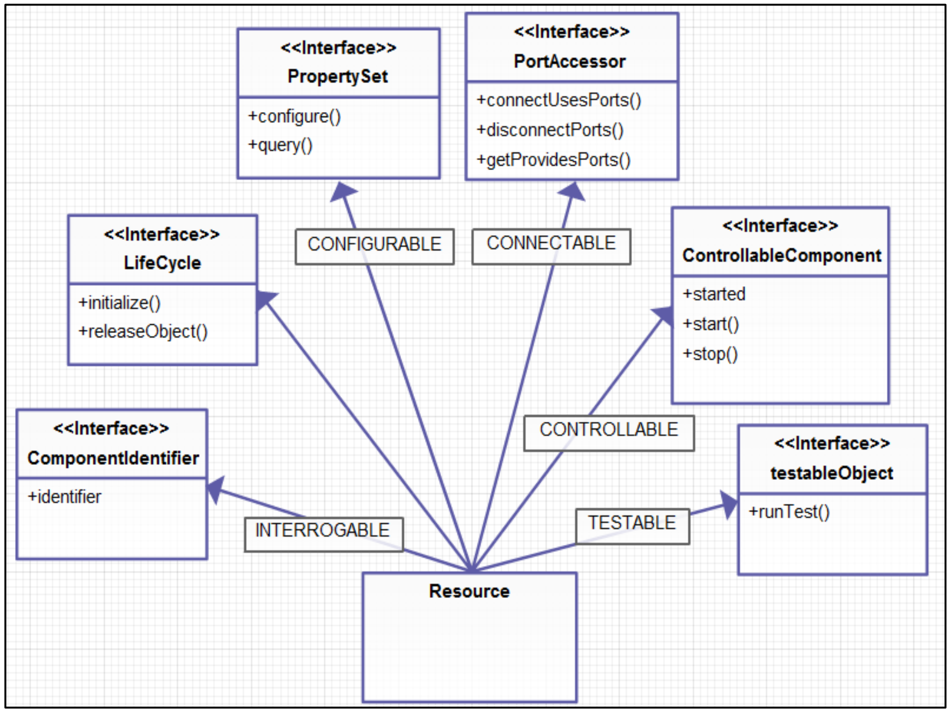
SCAv4.0.1 is Not Backwards Compatible

NordiaSoft

- **Component Scalability relied on Conditional Inheritance**
  - The basic Resource interface definition has been changed
  - No possible backwards compatibility with this approach
  - Only option is to port source code of all SCAv2.2.2 components



SCAv2.2.2



SCAv4.0.1

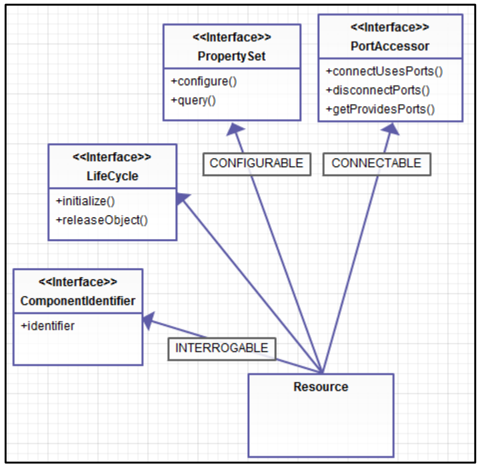
SCAv4.0.1 is Not Backwards Compatible

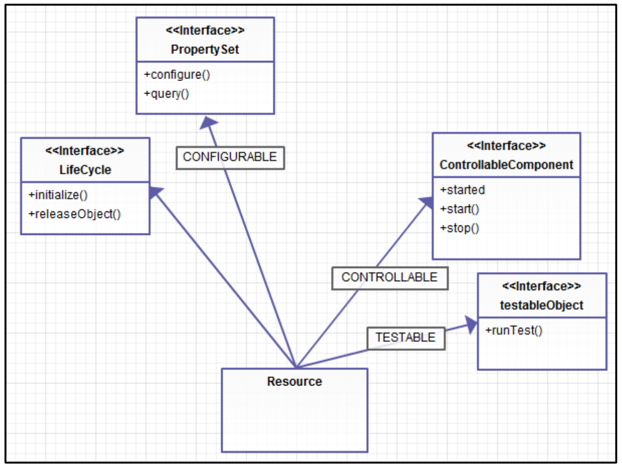
- **Component Scalability relies on Conditional Inheritance…**
  - **Approach is not UML compliant**
  - **IDL does not support conditional interfaces**
  - **In fact, no programming language supports conditional inheritance**

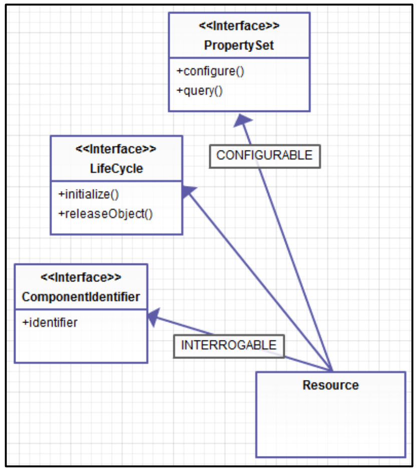
- **How Can Conditional Inheritance be Implemented?**
  - **Need to process the IDL interfaces with a specific set of pre-processor switches (#ifdef) to produce a specific set of IDL interfaces**
  - **This works but every time the IDL is processed with a different set of switches, the IDL interfaces are redefined. This creates several definitions of the same interfaces**
  - **Implementing a Core Framework that supports all the variations is very difficult if not impossible**
  - **This does not support the evolution of requirements. A new radio with a larger profile will not be able to run old waveforms of a smaller profile**

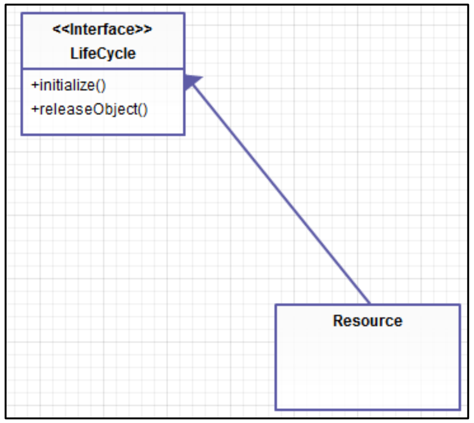
SCAv4.0.1 is Not Backwards Compatible

❖ **How Can Conditional Inheritance be Implemented?**









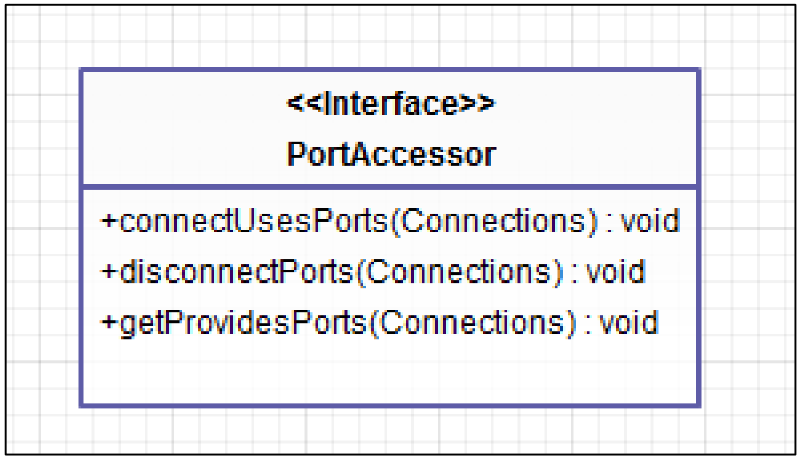
Over 100 more possibilities…

SCAv4.0.1 is Not Backwards Compatible

- **The new approach for component registration also breaks backwards compatibility**
  - Push registration was introduced to accelerate the boot time
  - The SCARI Core Framework makes extensive use of push registration already.
- **How is push registration implemented in SCAv4.0.1?**
  - First, the CORBA Naming Service has been eliminated
  - Application components now register back with their launcher; Just like Devices always have done
  - At registration time, all components provide information that does not need to be parsed by the DomainManager anymore

SCAv4.0.1 is Not Backwards Compatible

- **How has push registration been implemented in SCAv4.0.1?**
  - **Push registration applies to both application and node components**
  - **It uses a unified registration API**
  - **Allows components to register their ports involved in connections**
  - **Using the registered ports, the SCAv4.0.1 Core Framework can perform all connections in one call**



```
<<Interface>>
PortAccessor
+connectUsesPorts(Connections) : void
+disconnectPorts(Connections) : void
+getProvidesPorts(Connections) : void
```

```
struct ConnectionType
{
   ConnectionIdType portConnectionId;
   Object portReference;
};

typedef sequence <ConnectionType> Connections;
```

SCAv4.0.1 is Not Backwards Compatible

- **How does push registration break backwards compatibility?**
  - **SCAv2.2.2 components do not register using the unified API**
  - **SCAv2.2.2 component do not realize the new PortAccessor interface**
  - **All SCAv2.2.2 components must be ported to run on a SCA4.0.1 Core Framework**
  - **Late registration of a Device or Service is not possible for a number of reasons**

SCAv4.0.1 is Not Backwards Compatible

- **The Wireless Innovation Forum was asked by JTNC to create a task group in order to investigate backwards compatibility**
  - The WinnF created the "SCAv4.1 Backwards Compatibility Task Group" in December 2013
  - Goal: Investigate how SCAv4.0.1 breaks backwards compatibility with SCA 2.2.2, define potential solutions to address specific issues, gather consensus and submit change proposals to the JTNC for SCAv4.1

The Need for Backwards Compatibility

- **Steve Bernier, CTO at NordiaSoft, was asked to chair the new WinnF Task Group**
  - Goal is to preserve investment made in SCAv222 world wide; millions of lines of source code

- **Schedule adopted:**
  - [Milestone 1] – Jan 14 2014 – Prioritized list of issues, relative levels of disruptiveness, associated requirements
  - [Milestone 2] – Mar 14 2014 – List of potential solutions
  - [Milestone 3] – Apr 14 2014 – Final report containing selected solutions
  - [Milestone 4] – June 30 2014 – Submit change proposals for JTNC

The Need for Backwards Compatibility

❖ **The Task Group created 6 main documents that contain several change proposals each**

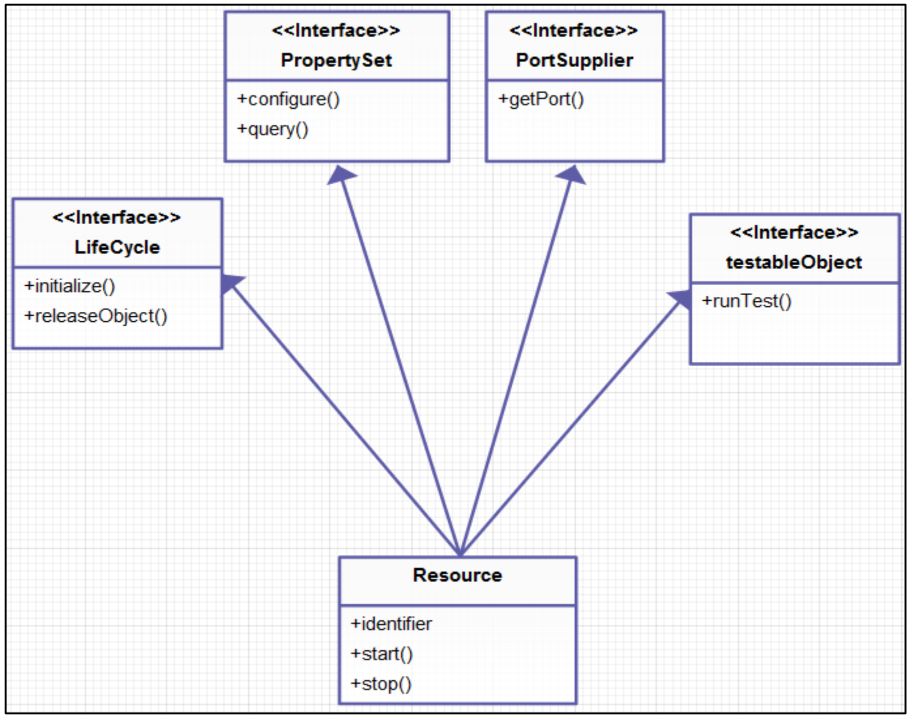
1. SCA Application Backwards Compatibility
2. SCA Application Mixture Backwards Compatibility
3. Scalable Components
4. Scalable Managers
5. Naming Convention
6. Device Registration

The Need for Backwards Compatibility

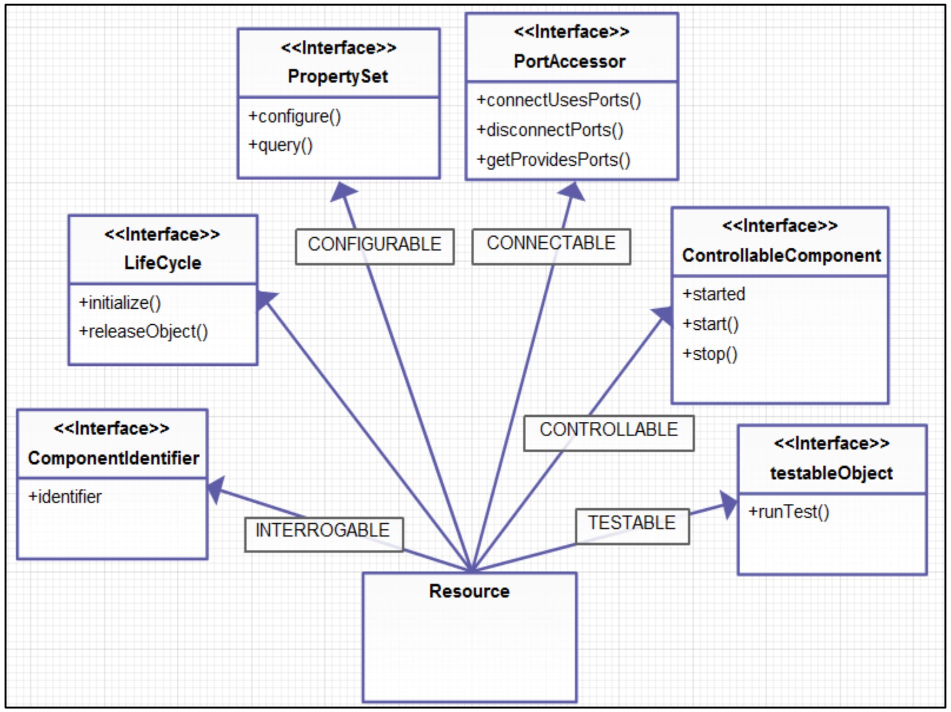
- **1. SCA Application Backwards Compatibility**
  - Goal: ensure that an unmodified SCAv2.2.2 application, as a whole, can run on a SCAv4.1 Core Framework
- **NordiaSoft Proposal**
  - Avoid redefining SCAv2.2.2 interfaces. Rename the SCAv4.0.1 Resource to a different name from SCAv2.2.2
  - Reintroduce the SCAv222 interfaces and requirements for SCA applications into SCAv4.1
  - Make the support for this change proposal an optional unit of functionality (UOF) for SCAv4.1

The Need for Backwards Compatibility

- ❖ **1. SCA Application Backwards Compatibility**
  - ➢ How different is the definition of Resource between SCAv2.2.2. and SCAv4.0.1?



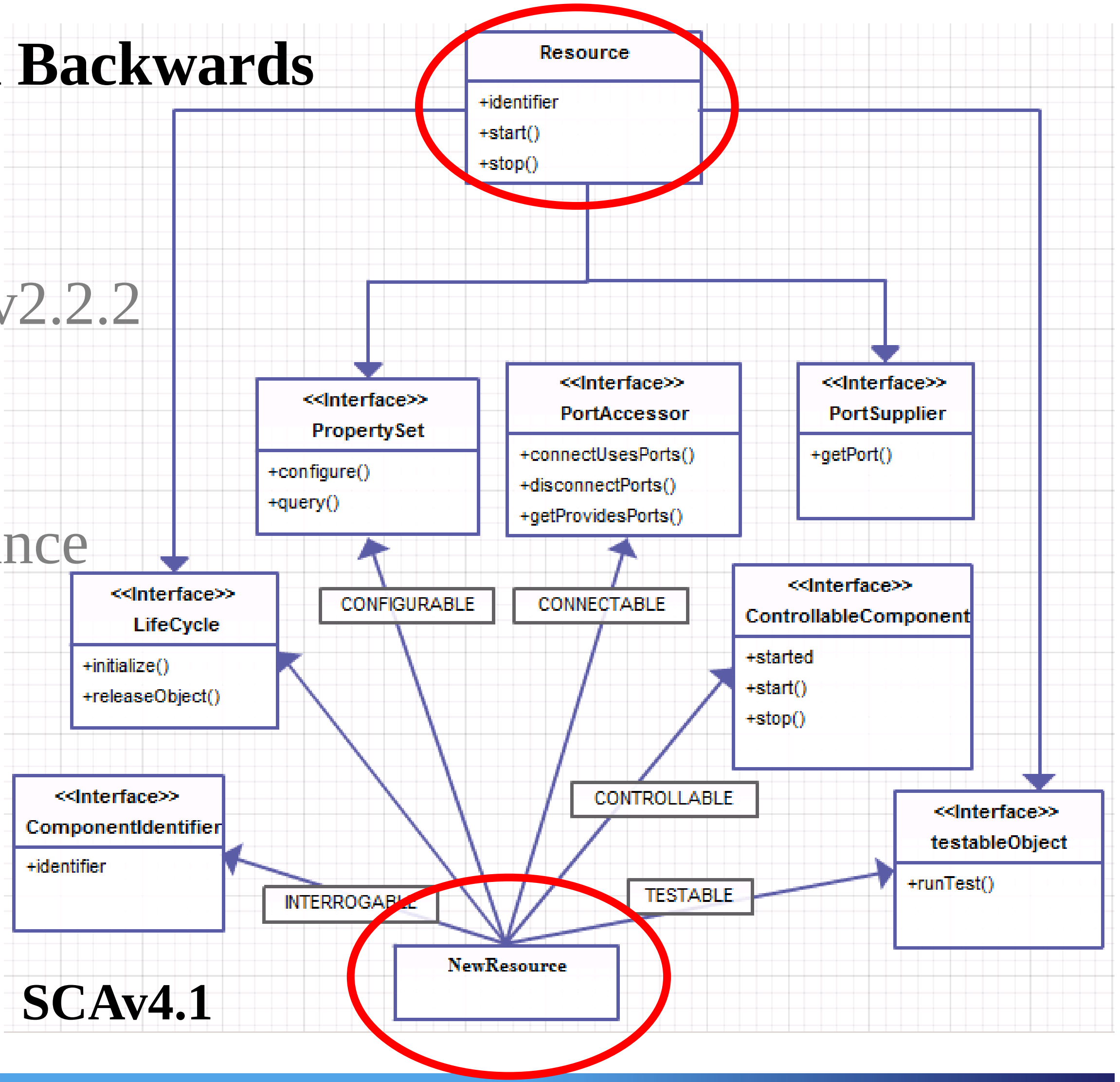
SCAv2.2.2



SCAv4.0.1

The Need for Backwards Compatibility

- ❖ **1. SCA Application Backwards Compatibility**
  - ➢ This initial solution offers duality: SCAv2.2.2 and SCAv4.1
  - ➢ But it still relies on Conditional Inheritance



The Need for Backwards Compatibility

- **2. SCA Application Mixture Backwards Compatibility**
  - Change Proposal from NordiaSoft
  - Ensure that an application made of a mixture of SCAv2.2.2 and SCAv4.1 components can run side-by-side on a SCAv4.1 Core Framework
    - Dependent upon the optional unit of functionality for Application Backwards Compatibility
  - Allows a gradual migration from SCAv2.2.2 to SCAv4.1
    - Offer an alternative to the “migrate everything or stay at SCAv2.2.2” approach
    - Allow an incremental migration by creating new SCAv4.1 applications containing any number of SCAv2.2.2 components
  - Change proposal to be balloted by end of June

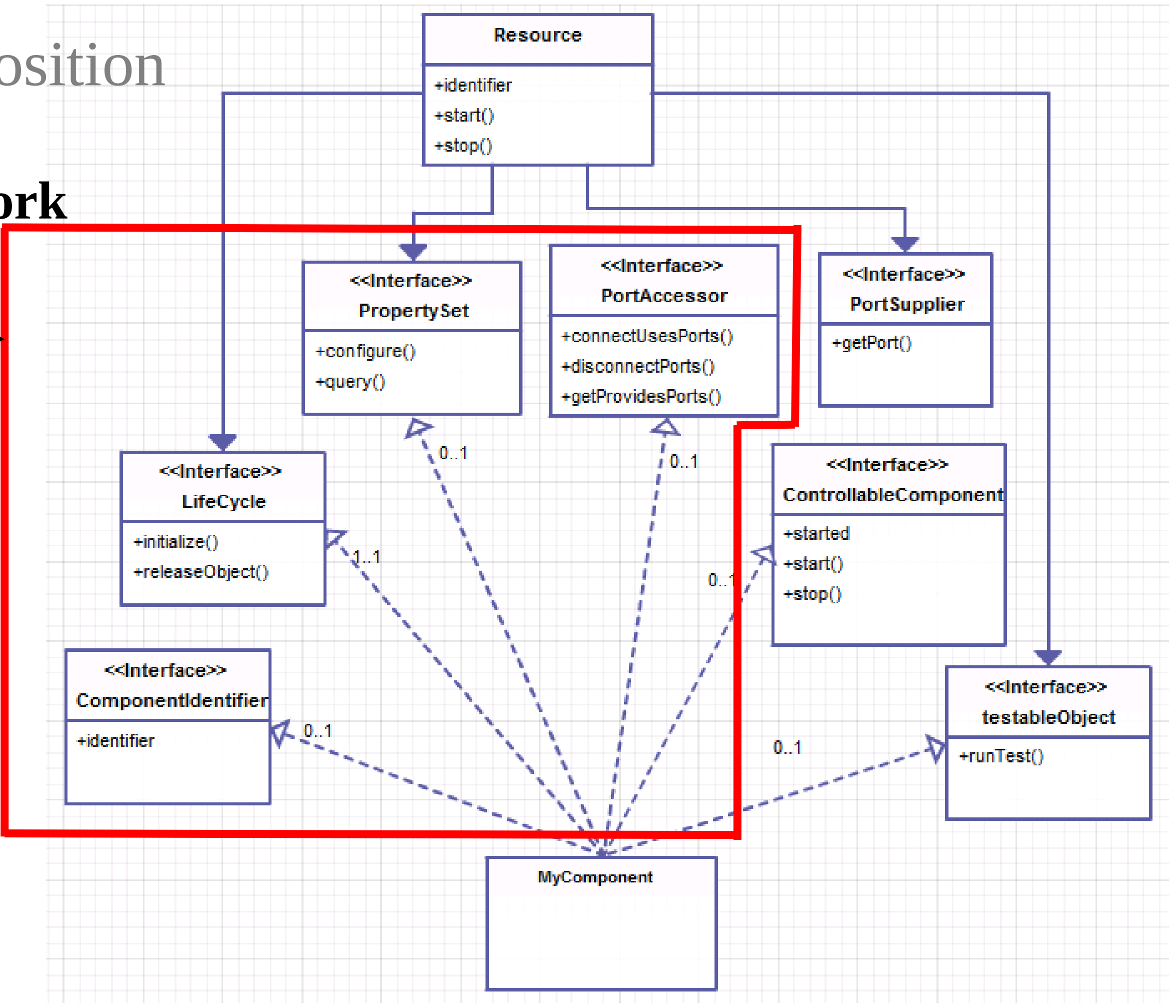
The Need for Backwards Compatibility

- **3. Scalable Components**
  - Change Proposal from NordiaSoft to eliminate Conditional Inheritance in favor of Selective Composition (aggregation)
  - With this proposal, an SCAv4.1 Core Framework does not deal with an all-encompassing interface (i.e. Resource in SCAv2.2.2). Instead, it deals directly with the sub-component interfaces of the components
  - Side benefit: the interface Resource can be eliminated from SCAv4.1 and only kept as defined in SCAv2.2.2
  - Huge benefit: components of varying size can be mixed in a same address space. This supports the evolution of requirements for future radios that need to run earlier waveforms

The Need for Backwards Compatibility

- 3. Scalable Components
  - Selective Composition

SCAv4.1 Core Framework only deals with those interfaces



The Need for Backwards Compatibility

- **3. Scalable Components**
  - Change proposals were approved by TG as part of Application Backwards Compatibility and as part of Scalable Components
    - Elimination of conditional inheritance; Use of selective composition
    - Elimination of the Resource interface in SCAv4.0.1
    - Support for components of varying sizes in a same address space
    - Compliance with UML
  - Change proposal approved in June 2014

The Need for Backwards Compatibility

- **4. Scalable Managers (Core Framework manager components)**
  - Allow Managers to implement a subset of the standard interfaces
    - Also relies on selective composition which is more flexible and UML compliant
  - Part of an optional unit of functionality (UOF)
  - Change proposal approved by TG in June 2014

The Need for Backwards Compatibility

- **5. Naming Convention**
  - The SCAv4.x specification document has been reformatted to separate the component concepts from the definition and descriptions of the interfaces components might realize
  - In doing so, many new names were introduced in the specification and the general feeling was that many of the names lead to confusion
  - Several names inherited from SCAv222 were also not very descriptive (ex: Resource)
  - A long list of new names have been proposed for SCAv4.1 applications and platforms

The Need for Backwards Compatibility

NordiaSoft

- ❖ **5. Naming Convention**
  - ➢ The new names follow a proposed naming convention whenever possible
  - ➢ Over 27 name changes in total
  - ➢ Example:

SCAv4.0.1	SCAv4.1
ComponentIdentifier	IdentifiableInterface
PortAccessor	ConnectableInterface
LifeCycle	InitializableInterface
TestableObject	TestableInterface
PropertySet	ConfigurableInterface
ControllableComponent	StartableInterface
ManagerRelease	ReleasableManagerInterface
ManageableComponent	AdministrableInterface
CapacityManagement	AllocatableInterface
LoadableObject	LoadableInterface
ExecutableDevice	ExecutableInterface

The Need for Backwards Compatibility

- **5. Naming Convention**
  - Change proposal approved by TG in June 2014

The Need for Backwards Compatibility

- **6. Device Registration**
  - Changes how Devices can perform push registration
  - Requires less XML parsing for the registration of Devices which can translate in faster boot times with a Core Framework which does not already support a form of push registration
  - Reintroduces support for late device registration that was eliminated in SCAv4.0.1. Synchronization of all Devices being launched often leads to slower boot times.
  - Change proposal to be balloted by end of June

The Landscape for SCAv4.1

❖ ***Status as of June 13, 2014:***

- **Application Backwards Compatibility WINNF-14-R-0004**
  - **consensus ballot 3 weeks ago**
- **Scalable Components WINNF-14-R-0010**
  - **consensus ballot today**
- **Scalable Manager Components WINNF-14-R-0011**
  - **consensus ballot today**
- ***Device Registration WINNF-14-R-0012***
  - ***Balloting before end of June***
- **Naming Convention WINNF-14-R-0013**
  - **consensus ballot today**
- ***Application Mixture WINNF-14-R-0014***
  - ***Balloting before end of June***
- *Self-launching Platform Components*
  - ***Integrated in Device Registration Proposal***

The Landscape for SCAv4.1

- **SCAv4.1 will also contain a number of change proposals coming from two other WinnF Task Groups**
  - Changes regarding CORBA profiles and IDL support
  - Changes regarding new Application Environment Profiles (AEPs)
- **The SCAv4.1 specification documents will require a fair amount of integration**
  - Several change proposals require changes of the same documents
  - Goal of JTNC is to produce a draft release of SCAv4.1 by September 2014
  - Will subsequently need to be implemented which will probably lead to additional changes and minor specification releases

Summary

- **Effort to make SCA Next Backwards compatible with SCAv2.2.2**
  - SCAv4.0.1 is the latest draft of the specification and is not backwards compatible with SCAv2.2.2
  - SCAv4.1 is expected to offer backwards compatibility with SCAv2.2.2

- **SCAv4.1 offers more flexibility**
  - SCAv4.1 scalability features offer potential for footprint improvements
  - SCAv4.1 will need more maturing before it is ready for being used in acquisition programs

NordiaSoft would like to thank all the Contributors to the SCAv4.1 Backwards Compatibility Task Group!

- ❖ **Questions?**
  - ➢ **Point of Contact Information**
    - ▪ **Email:** info@nordiasoft.com
    - ▪ **Phone:** +1 819 307 0333
    - ▪ **Web:** www.NordiaSoft.com