



# *Joint Program Executive Office Joint Tactical Radio System*

---

## Conditional Inheritance Overview



Statement A: Approved for public release, distribution is unlimited (04 May 2011).



# Introduction

---

- **SCA Next includes a techniques referred to as Conditional Inheritance (CI)**
- **This brief attempts to accomplish the following:**
  - Define conditional inheritance
  - Discuss why the SCA Next development team felt it was an appropriate solution
  - Compare and contrast CI with some of the other alternative approaches that may have been incorporated within the spec



# SCA Next Design Objective

---

- **Goal**

- To have a flexible architecture that can accommodate various platforms requirements instead of one size fits all solution
  - mobile versus static,
  - single channel versus multiple channels,
  - single waveform versus multiple waveforms,
  - small form factor,
  - ...

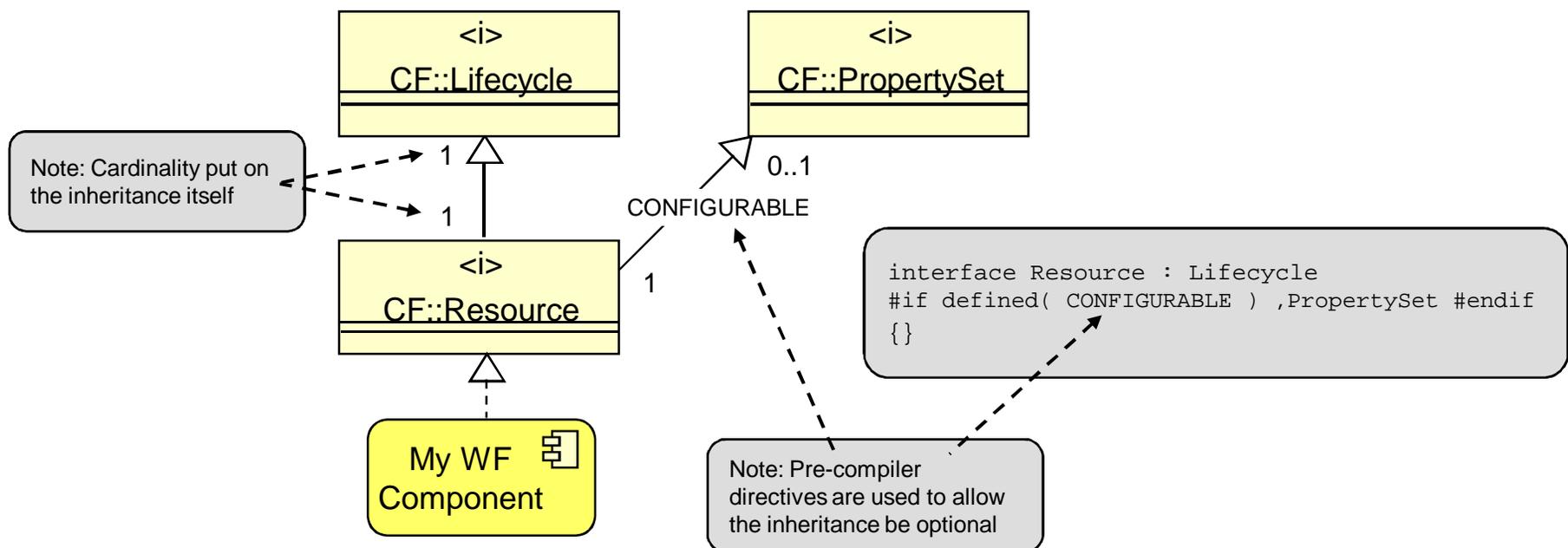
- **Benefits**

- The elimination of interfaces that are not needed by a component results in:
  - Increased Information Assurance - increased by removing unused operations from deployed code.
  - Footprint Size - reduced executable software's size.
  - Performance - smaller software executable will consume less overhead
  - Development Time - fewer requirements reduces time spent implementing, testing, and integrating.



# What is Conditional Inheritance?

- Components always realize a single interface for framework management, but what that interface inherits from is optional
- Optional inheritance is expressed in pre-compiler directives that resolve at Interface Definition Language (IDL) compile time





# CI Pros and Cons

---

- **Benefits**

- Framework only requires the component provide one management object reference
- Components are clearly provided this management interface by the architecture and don't have to invent their own
- Backwards Compatible with SCA v2.2.2
- Coupled with IDL decomposition it provides a means to reduce implementation footprint and tailor components to an end product

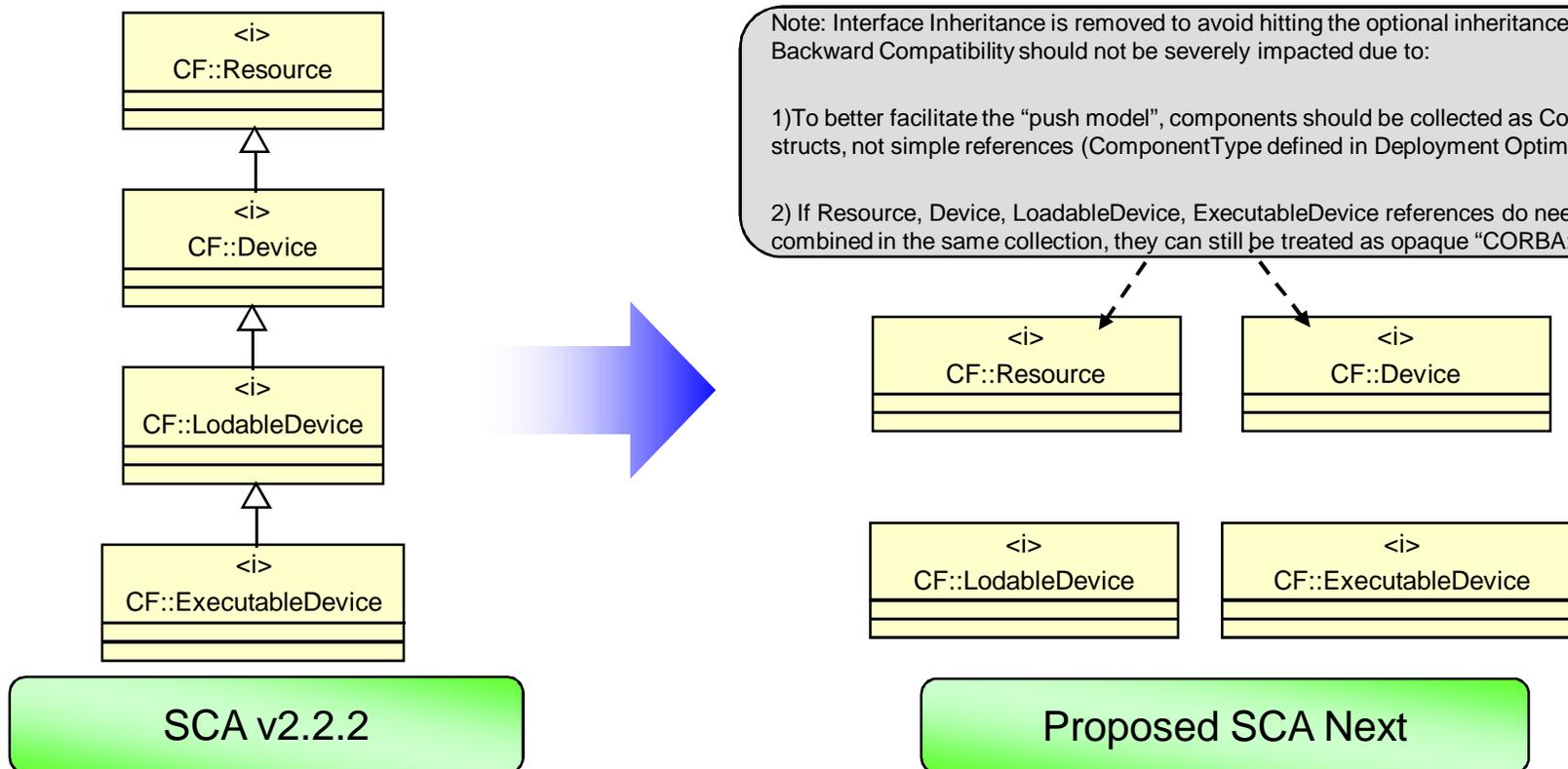
- **Problems**

- Violates Unified Modeling Language (UML) inheritance rule
  - May impact out of the box tool support
    - IDL generation
    - Reverse engineering
- Single Operating System (OS) Address Space Restriction
  - Means same Resource/Device IDL translation needs to be used, which is the same result as now for SCA v2.2.2



# Interface Refactoring

- The incorporation of CI was integrated with other SCA Next changes
- Several SCA Next interfaces were restructured to account for the address space limitation





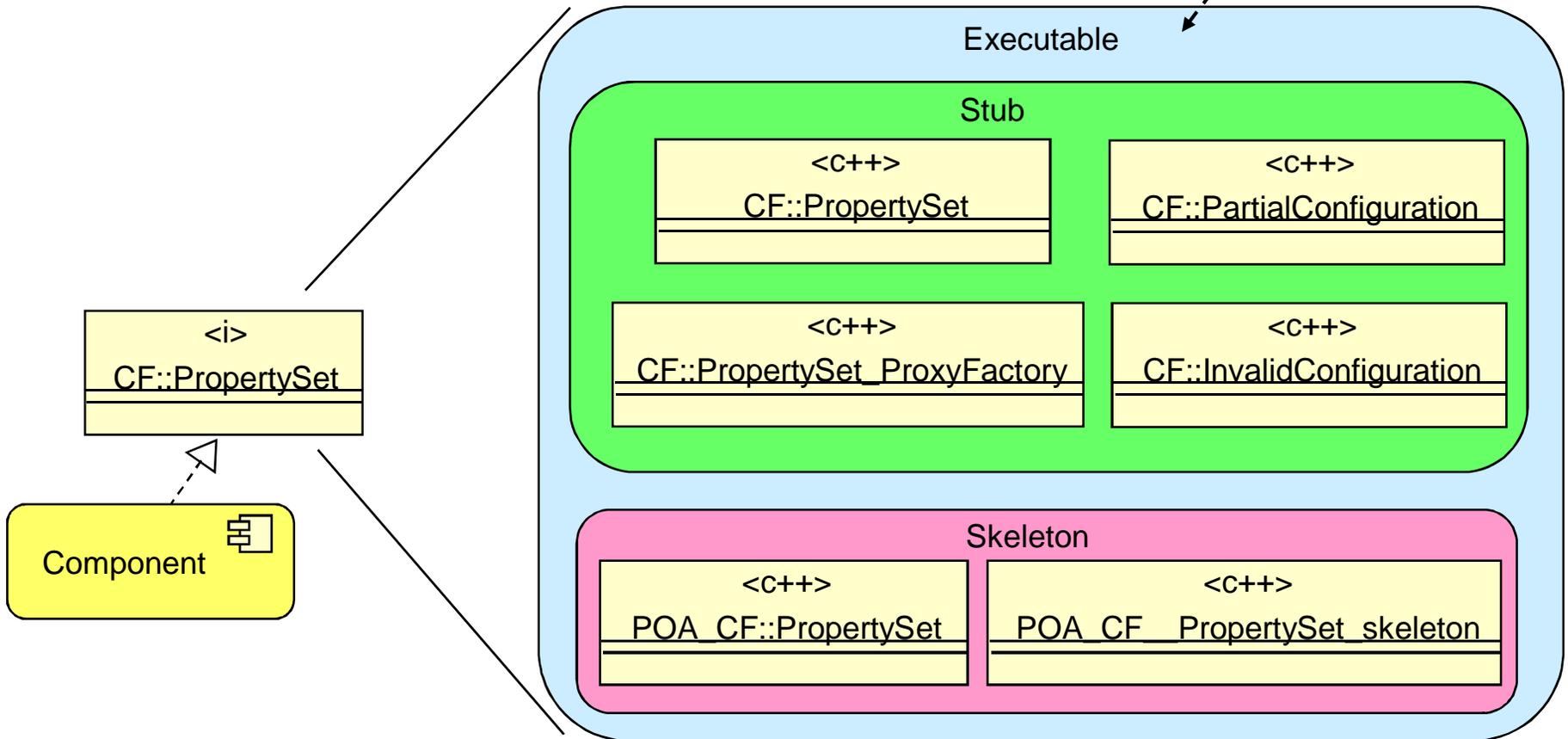
# CI Performance Optimizations - Pro

- **Savings in Runtime Costs**

- Executable Footprint
- Load (i.e. startup) time

**Executable Footprint Savings**

For CF::PropertySet, there's roughly a 19 – 40 KB savings per executable (estimate) in Stub and Skeleton code alone. Doesn't include any impl code (e.g. to throw required exceptions, etc.)





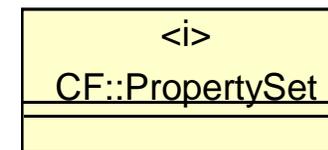
# CI Cost Savings - Pro

## • Savings in Software Lifecycle Costs

- Where base classes are not utilized there are savings that can be realized in Requirements Analysis, Design, Implementation, Test, and Security Verification

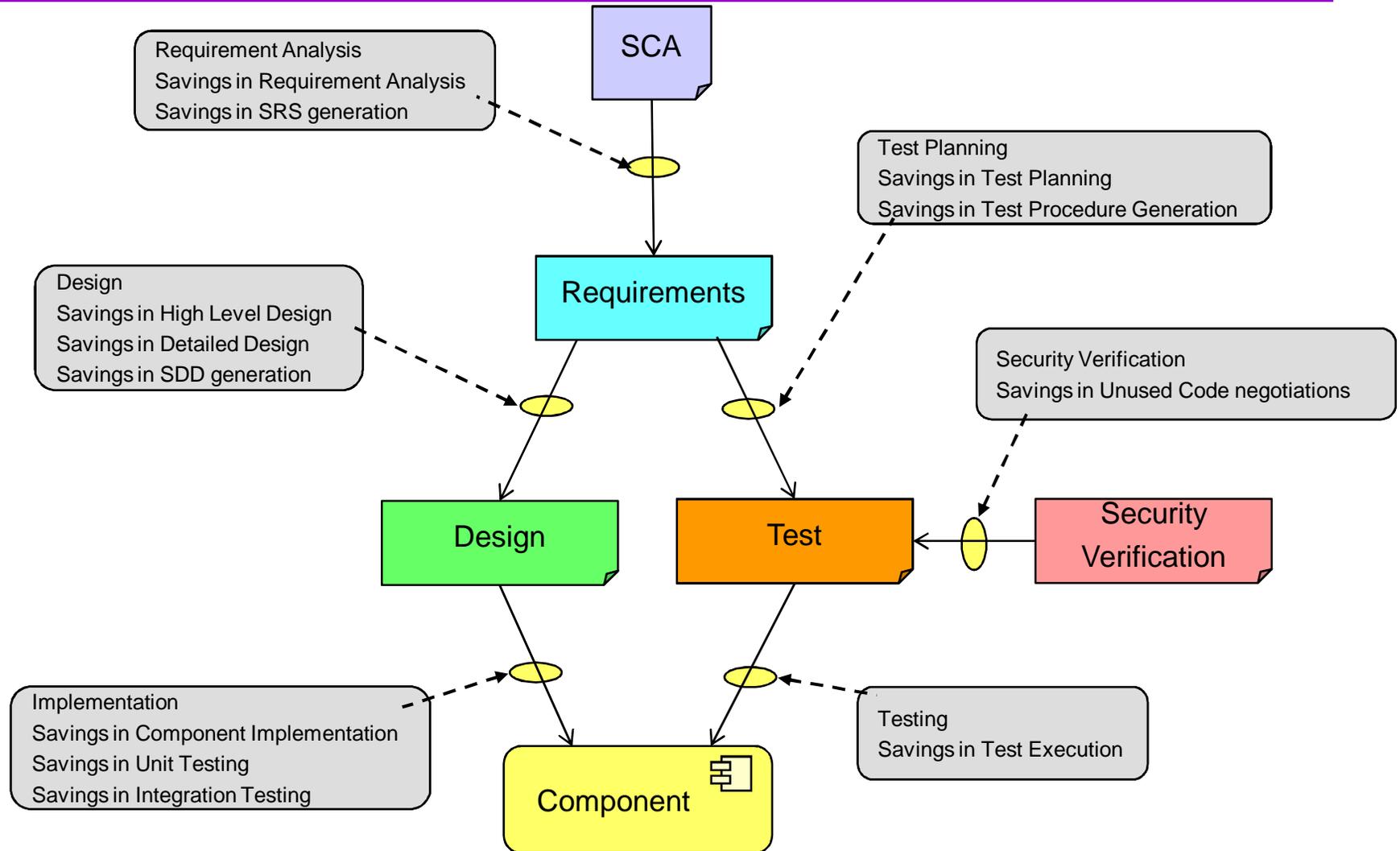
### •CF::PropertySet Requirements (SCA v2.2.2)

- The configure operation shall assign values to the properties as indicated in the input configProperties parameter.
- Valid properties for the configure operation shall at a minimum be the configure readwrite and writeonly properties referenced in the component's SPD.
- The configure operation shall raise a PartialConfiguration exception when some configuration properties were successfully set and some configuration properties were not successfully set.
- The configure operation shall raise an InvalidConfiguration exception when a configuration error occurs and no configuration properties were successfully set.
- The query operation shall return all component properties when the inout parameter configProperties is zero size.
- The query operation shall return only those id/value pairs specified in the configProperties parameter if the parameter is not zero size.
- Valid properties for the query operation shall be all configure properties (simple properties whose kind element's kindtype attribute is "configure") whose mode attribute is "readwrite" or "readonly" and any allocation properties with an action value of "external" as referenced in the component's SPD.
- The query operation shall raise the CF UnknownProperties exception when one or more properties being requested are not known by the component.





# Development Lifecycle Optimizations - Pro

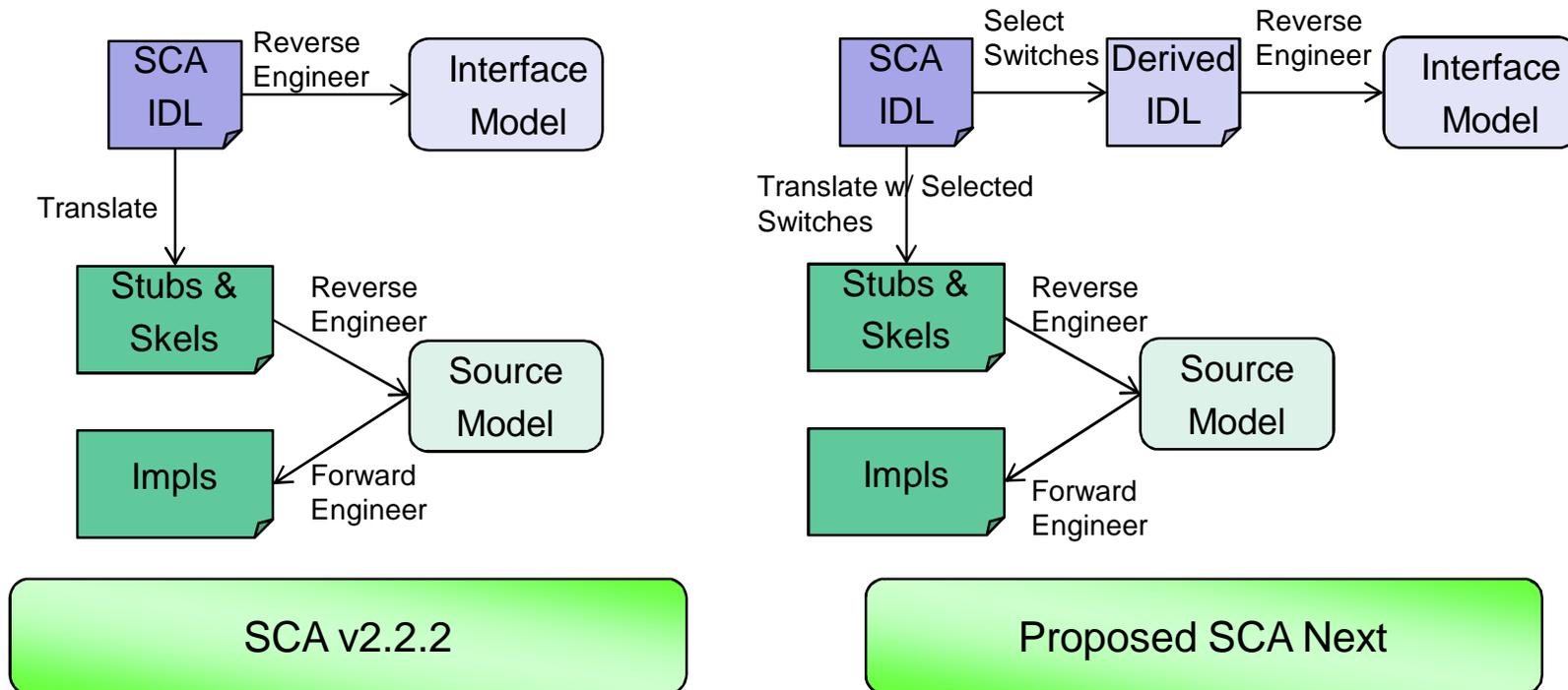




# CI Tool Support - Mitigation

## Effect on modeling can be mitigated

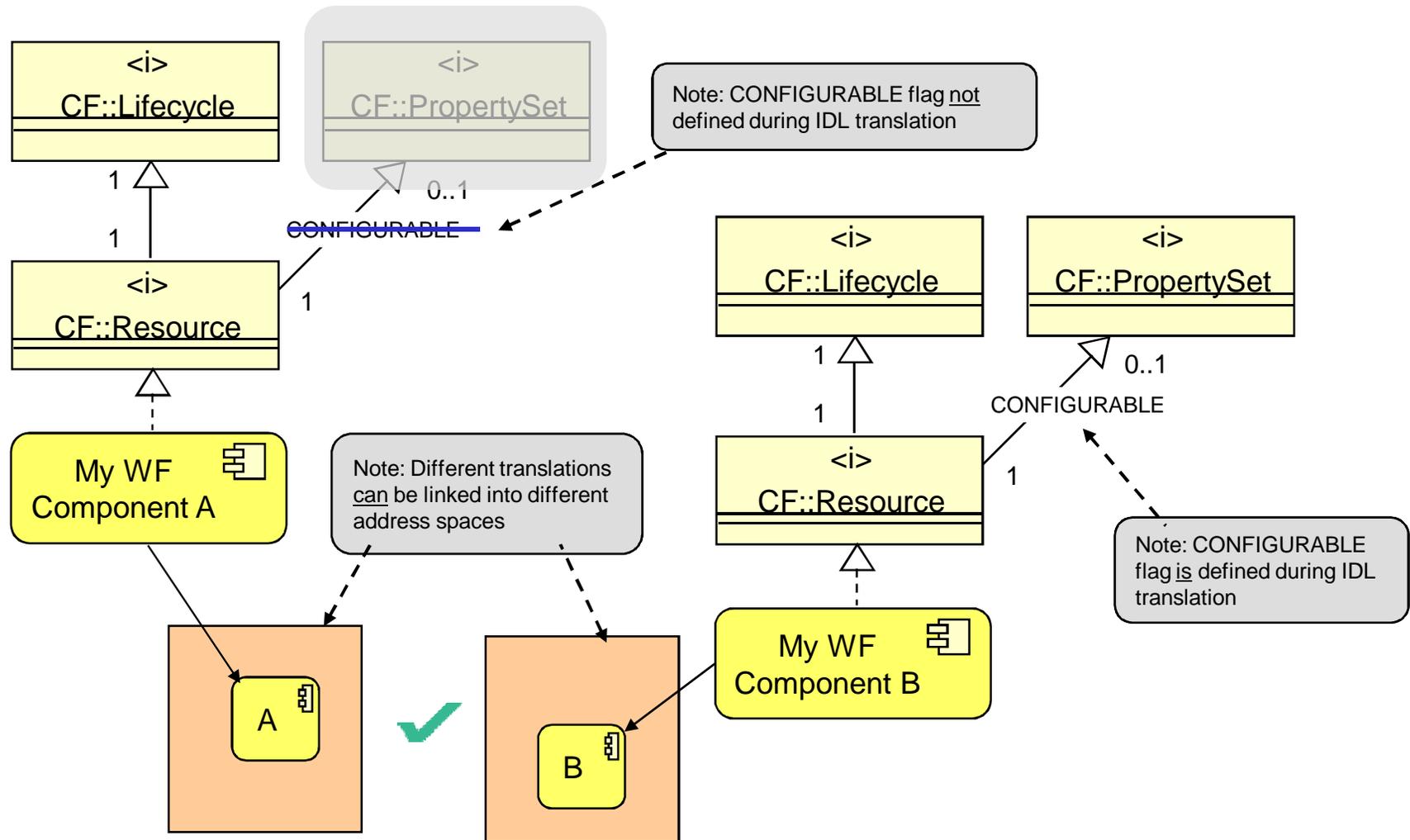
- Decision on customization can be made early in the design process
  - Derived IDL could be created with desired inheritance made statically, then imported into the model
  - SCA IDL could be imported and desired inheritance entered into the model directly
- Most of the modeling process unaffected
  - No change to modeling after IDL translation





# Conditional Inheritance Address Space Restriction - Con

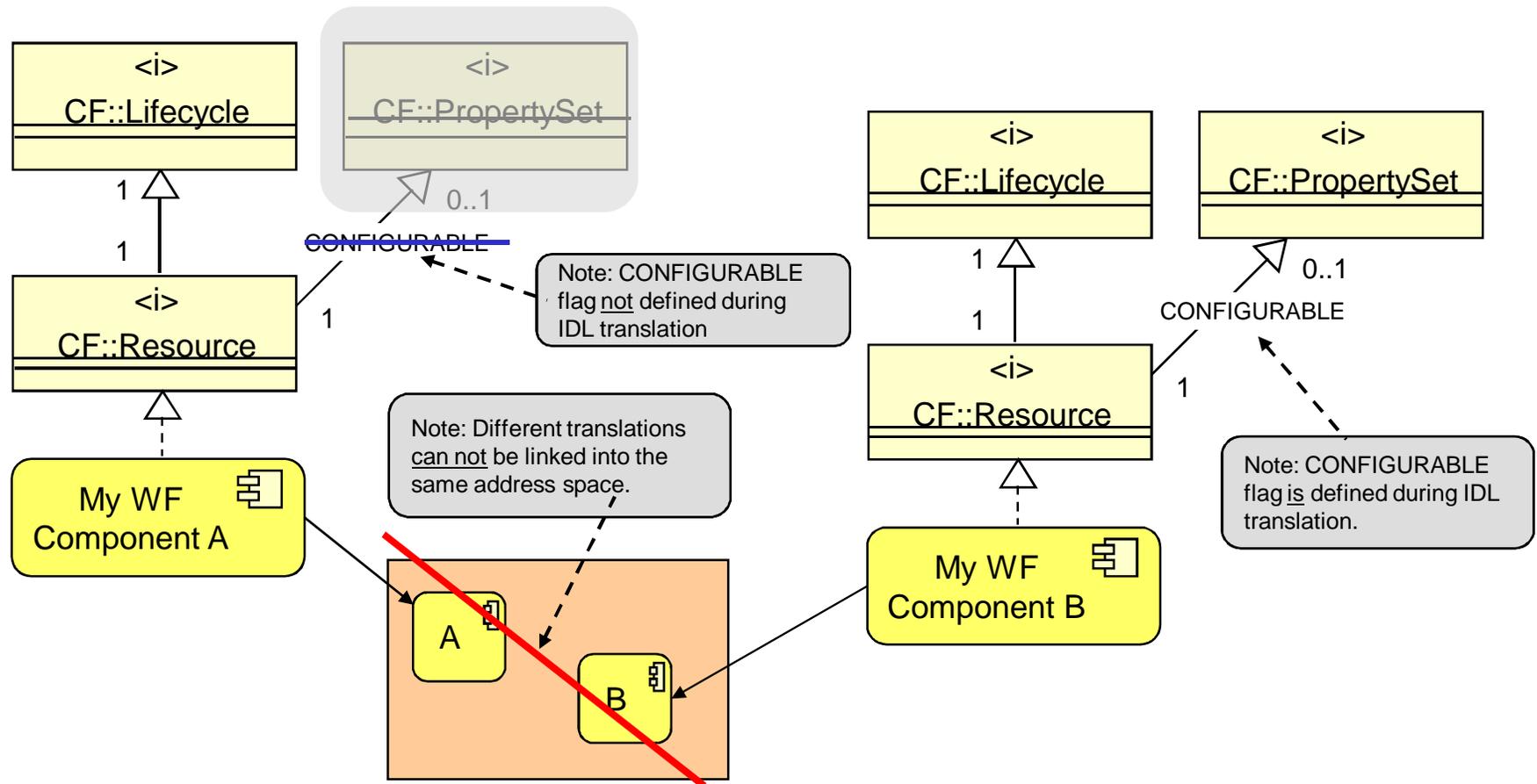
- Two Address Spaces with different IDL translations





# Conditional Inheritance Address Space Restriction cont.

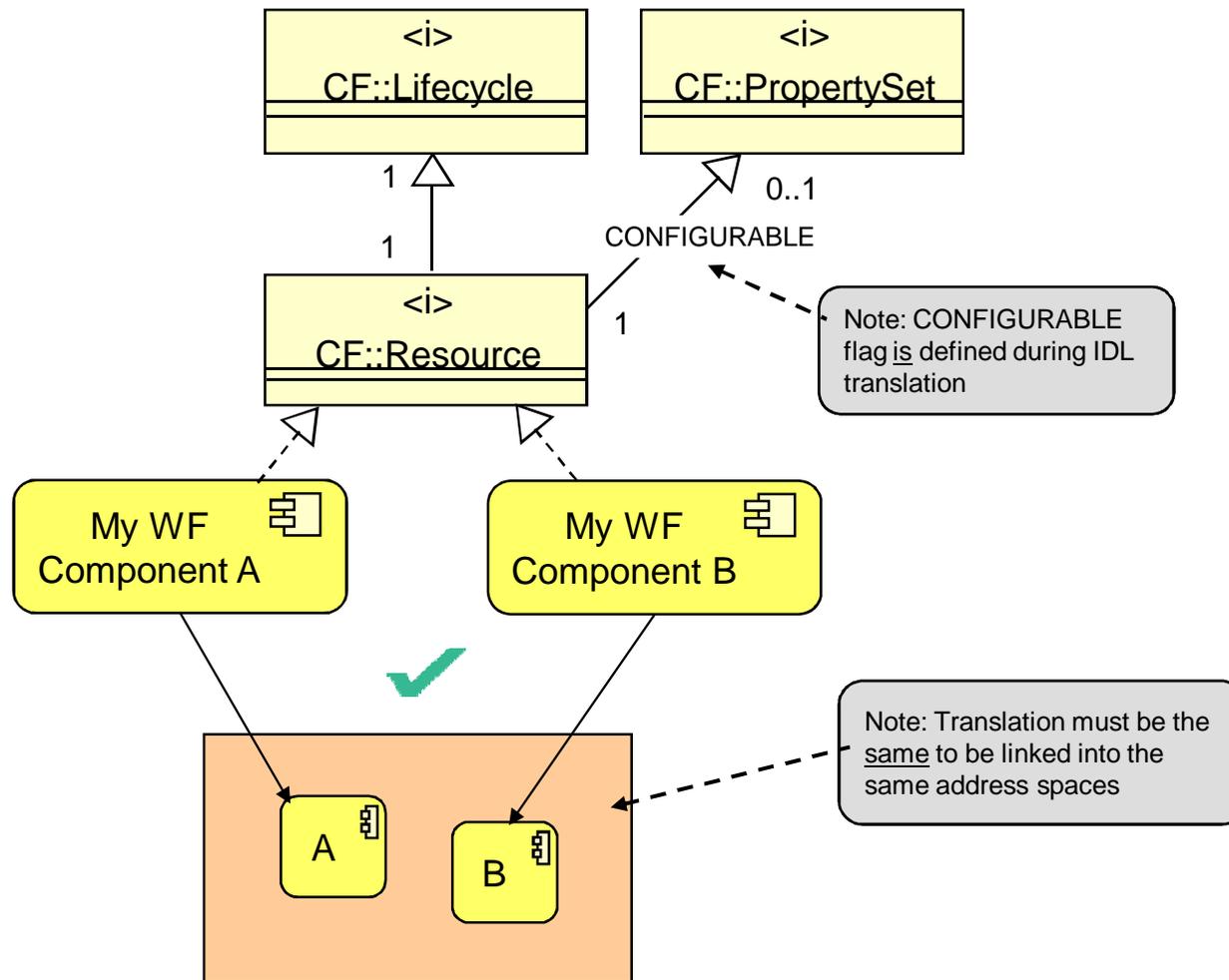
- Link problems ensue when trying to pull IDL that has been translated differently into the same address space





# Conditional Inheritance Address Space Restriction cont.

- One Address Space with same IDL translation





# Address Space Limitations – Mitigation & Pro

---

- **Scope may be larger than “per address space”**
  - While “per address space” is the lowest level of granularity, the technique does not need to be applied at that level
    - Switches can be restricted to variations at processor, OE, WF or system level
  - The technique allows an OE on one platform to be customized differently from an OE on a different platform
    - One platform may have fault management done through the Resource/Device interfaces on all components while another may only perform it on certain components through direct port connections
- **Number of address spaces (i.e. potential benefit) could be large**
  - If “per address space” customization is desired, on certain platforms the technique could be well utilized at that level
  - Example larger-scale program deployment
    - 4 processor types / 15 address spaces per processor type (average)
    - 30 opportunities for OE component customization (average)
    - 5 opportunities for WF component customization (average)



# CI Complexity Management - Mitigation

---

- **Complexity and Differences can be mitigated**
  - If deployment flexibility is a primary concern, throwing all switches can provide for that
  - If all switches are thrown, development effort and resource requirements are no worse off than if the technique were not made available at all
    - Without this technique, all interfaces would always be required anyways



# CI Alternatives

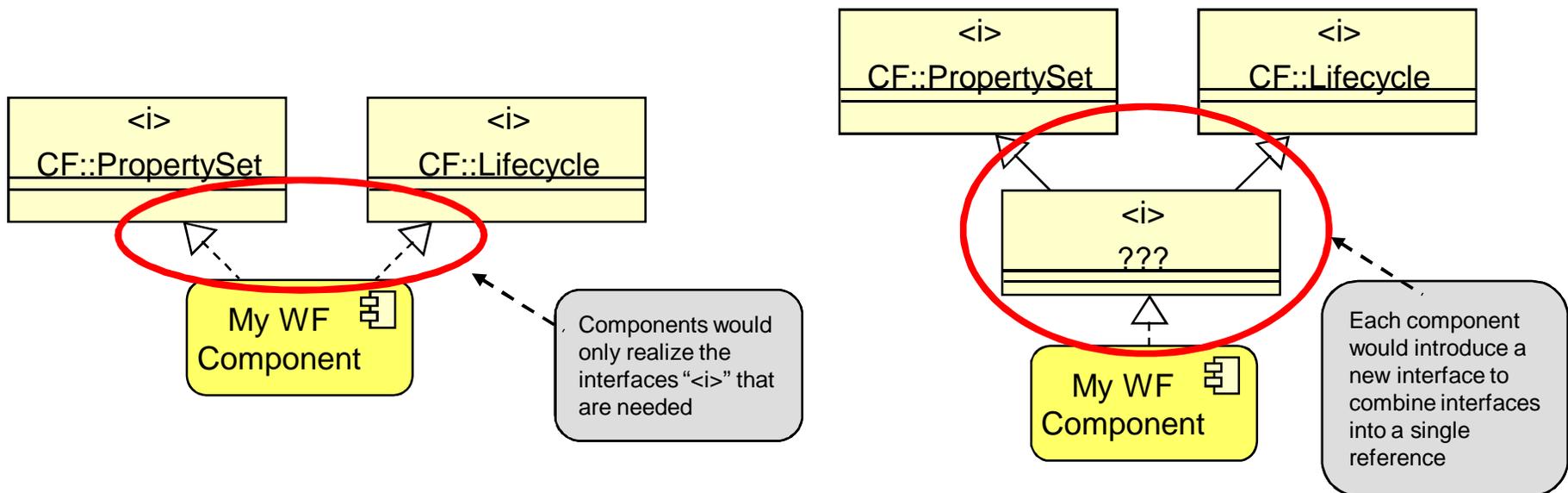
---

- **Four alternative approaches were evaluated**
  - Optional Realization
  - Interfaces modeled as provided ports
  - Default Implementation
  - Maintaining the Status Quo
- **The remainder of the brief describes each approach and provides a comparative assessment**



# Optional Realization (OR)

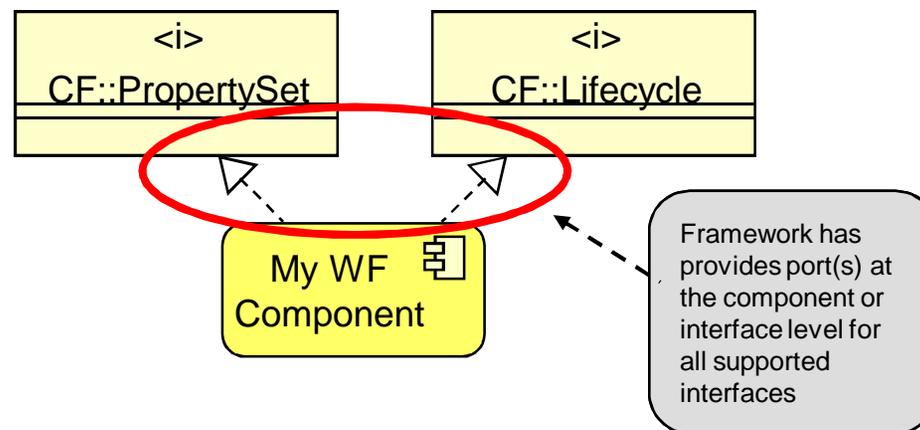
- Allows component developer to selectively include the required interfaces that will be realized in the final implementation
- Similar to the approach promoted within the Object Management Group Software Radio specification
- Does not provide a standardized, well known interface





# All Interfaces Provided as Ports

- Ports are identified and introduced for all interfaces realized within the implementation.
- Each supported interface will have a generalization relationship with the PortAccessor interface
- Introduces considerable changes upon existing SCA implementations





# Status Quo & Default Implementation

---

- **Status Quo maintains an interface model equivalent to that of SCA v 2.2.2**
  - SCA Next may change the underlying interface definitions but the overarching model preserves the 1..1 relationship between the composing and contained interfaces
- **The Status Quo provides designers and developers with the option of stubbing, omitting or partially implementing unused interfaces**
- **The Default Implementation differs from the Status Quo in that the spec would provide a trusted, default implementation that would be used within a product if no product specific additions are needed**
  - Provides uniformity across products
- **Providing code as part of or as a companion to the SCA would introduce a plethora of procedural challenges for a DoD program**



# SCA Next Analysis Results

Approach / Feature	assurance	foot print	performance	requirements	lifecycle cost	testability	portability	Standards support	tool support	dev support	Backward compat	total
Conditional Inheritance	5	5	5	5	5	3	3	2	2	3	3	41
Optional Realization	5	5	4	5	4	3	3	3	3	3	3	41
All Interfaces provided as ports	5	5	1	5	2	2	3	3	2	3	1	32
Default Implementation	4	3	3	3	4	3	3	3	3	3	3	35
Status Quo	3	3	3	3	3	3	3	3	3	3	3	33

**5 = significant improvement**

**3 = roughly equivalent to current state**

**1 = significant impediment**



# Enhanced Assurance

Approach / Feature	How does SCA Next augment the security profile of compliant products?
Conditional Inheritance	A large positive for this approach is that only implemented interfaces are present within the code base. A chain of trust can be developed between the design, the interfaces that are included within the product, the build time flags or options that are incorporated, and the properties that exist within the profile. There are additional positives that result from not having the typical collection of interfaces that exist beneath the top level defined interface.
Optional Realization	Optional Realization mirrors the benefits provided by conditional inheritance
All Interfaces provided as ports	The most security-aware implementation is likely to be the delegation approach used by an implementation in accordance with this alternative. The implementation receives the conditional inheritance and realization benefits. Furthermore, delegation also provides the mechanism to create and enforce a distinct separation between the core interface and each interface provided on a port.
Default Implementation	This alternative provides a less security-aware product, however it is better than the Status Quo. The default implementation guarantees the presence of unessential code within the product. The risk or presence of that code can be mitigated somewhat if a pedigree is built into the code and there is a level of assurance associated with its existence and functionality.
Status Quo	This permits a vulnerability to exist within the products as it forces a product to either provide a full implementation for something that will never be exercised, not provide an implementation and risk a non-deterministic result if the operations are inadvertently called or have a stubbed implementation of the operation which can vary from product to product.



# Footprint

Approach / Feature	How can SCA Next reduce the footprint required for compliant products?
Conditional Inheritance	Savings of 10-50K LOC viewed per executable by only including a subset of the possible interfaces, of course no savings would be realized if the full configuration is needed and the amount can be greater if additional capabilities are omitted.
Optional Realization	Savings would be consistent with those achieved by the conditional interface, with a small amount of additional overhead incurred if the approach was taken where each component was required to define its own intermediate interface. Potential improvement vis-a-vis the distinct interfaces provide an opportunity to co-locate similar but distinct components within an address space.
All Interfaces provided as ports	Savings would be consistent with Optional Realization except that the potential savings are due to only implementing the required interface. Note a small overhead is required (implementation dependent) based on the number of ports implemented.
Default Implementation	Somewhat analogous to the Status Quo approach in that all of the interfaces are always present and have an implementation. The size of the final product could actually be larger than the Status Quo if the default implementation had a richer, read larger, implementation than those that have been provided by the current set of implementers.
Status Quo	No savings



# Performance

Approach / Feature	How is the performance of SCA Next compliant products enhanced?
Conditional Inheritance	The options that rely on multiple inheritance incur more expense than would be required if a single level interface definition were utilized and the overhead increases with the complexity of the inheritance. The optimized implementations can enhance performance in cases where the end products are smaller and targeted towards a specific target.
Optional Realization	Framework had to be prepared to deal with multiple object references or each component implementation was left defining it's own child interface to combine the various interfaces chosen into a single reference. For the CORBA PSM this may result in "is_a" calls, which impact performance.
All Interfaces provided as ports	The port approach, even with the push model registration likely results in increased overhead as the connections are established but that expense is likely mitigated once the products are running. A substantial impact would be that the framework would need to be prepared to interrogate the base component in order to retrieve its implementation details.
Default Implementation	
Status Quo	



# Requirements

Approach / Feature	How much does SCA Next reduce the mandated requirements for compliant products?
Conditional Inheritance	This approach can significantly reduce then number of requirements for products that do not need to include the full complement of flags. A test was run that resulted in a ~30% reduction in the number of requirements that were allocated to a candidate devices. That number can increase or decrease based on the number of selected flags.
Optional Realization	The extent of requirements used by this approach is equivalent to those used by conditional inheritance as only those interfaces, and their corresponding requirements, are incorporated on an as needed basis.
All Interfaces provided as ports	The number of requirements will be reduced, they are introduced on an as needed basis, however this approach results in a slightly higher number than the conditional approaches as it requires a thin veneer or requirements for the port(s).
Default Implementation	The number of requirements will increase for each product however the cost of that increase will be incurred by the individual product developers as they will be able to integrate a preapproved implementation that satisfies those requirements within their products.
Status Quo	At a high level there will be no requirements savings as a result of using this approach because each product will be responsible for fulfilling the full complement of requirements.



# Lifecycle Cost

Approach / Feature	How much can SCA Next reduce the lifecycle cost attributed to compliant products?
Conditional Inheritance	This approach maximizes the potential savings for each product. The reduction or requirements, and resultant code eliminates the need these products to worry about the unit design, test of these elements. When those items are not part of the base units they also do not need to be tested, documented or maintained at the system level. As the number of requirements saved per unit and the number of components increases , the extent of this savings is multiplied.
Optional Realization	Conditional realization duplicates the lifecycle savings of conditional inheritance. Depending on the implementation approach there may be a slightly higher cost incurred if additional interfaces are required for the implementation.
All Interfaces provided as ports	This approach saves costs, in accordance with the requirements savings, however the presence of the port mechanism and the connection establishment that is introduced as a result requires a slightly higher cost to maintain. The degree to which the end to end costs are impacted is very implementation dependent.
Default Implementation	Better than the Status Quo, this approach requires minimal validation to insure that the correct implementation is included within the product. There is an initial validation cost associated with providing assurance that the default implementation behaves and performs correctly but once that is proven it should not need to be repeated.
Status Quo	No change, the approach designed and realized will dictate the eventual cost and how it is incurred. Providing more or less functionality within the product will incur the cost one way or the other but it will dictate if they are allocated more towards testing, development, waivers, etc.



# Testability

Approach / Feature	How does SCA Next impact the testability of a compliant products?
Conditional Inheritance	Improves and complicates testability at the same time. Testability is improved as the number of requirements are reduced, so there is less to validate, however it complicates matters because the composition of the individual components is not known a priori. There are ways that some of the uncertainty can be mitigated, such as the composition of the elements can be discerned by information contained within the profiles (supported interfaces, properties that exist, ...) or the flags that exist within the interface definition file. An item that allows the approach to be more deterministic is that there is always a guaranteed initial location for the interface.
Optional Realization	Similar to conditional inheritance, there is an additional variability because the implementation will dictate the base point for each of the interfaces. The interface information could be retrieved from the interface definition file or profile similar to some of the other information but using this approach would require this additional step and knowledge.
All Interfaces provided as ports	Similar to conditional inheritance, the variable aspect of this approach would likely require significant interaction with the interface definitions or profiles. The port mechanism likely requires the test tools to have more incorporation with the base components in order to access the delegated interfaces.
Default Implementation	Similar to the Status Quo however this approach requires the additional step of validating the proper integration of the approved default implementation.
Status Quo	Probably the easiest of the alternatives to use because, to the extent that the product designs are unaffected by SCA Next, this maximizes the potential reuse of the existing test methodologies and tools.



# Portability

<b>Approach / Feature</b>	<b>How do the SCA Next approaches impact portability of compliant products?</b>
Conditional Inheritance	There is a potentially significant approach in portability, especially if there are situations where products of differing profiles need to be integrated with one another. Conditional inheritance is designed using a least common denominator principle where if items of differing configurations are integrated, in order to assure functioning without incurring additional porting cost, a decision needs to be made for the integrated unit to function at the level of the least capable element. The different profiles are constructed in a fashion such that there is no need to "port" a more capable element being integrated with a less capable one, however it can be ported if one wants to benefit from the additional savings.
Optional Realization	Suffers from the same portability challenges as those encountered by conditional inheritance.
All Interfaces provided as ports	Approach should be relatively portable since all expectations about the implementation are encapsulated within the implementation. There are limited decisions or assumptions that the platform can make on the final products.
Default Implementation	This approach should not have an impact on portability, however a small amount of thought and effort needs to be focused towards this implementation to insure that this is the case. The number of default implementations will need to be expanded as the number of target platforms expands.
Status Quo	No impact on the complexity of portability.



# Standards Support

Approach / Feature	How well does SCA Next allow compliant products to utilize and benefit from commercial standards?
Conditional Inheritance	The strategy violates a formal inheritance constraint that exists within the specification, that being said, out of the box modeling and design tools will not work for forward and reverse engineering types of activities. Once the design activities transition from system to more solution oriented engineering , then the approach maps into platform specific representations that are consistent with the language provided Standards. Inquires to OMG have been made to OMG and are under consideration, we have considered proposing an additional UML profile or changes to the UML spec to accommodate this concept.
Optional Realization	Consistent with and utilizes industry Standards.
All Interfaces provided as ports	Consistent with and utilizes industry Standards.
Default Implementation	Consistent with and utilizes industry Standards.
Status Quo	Consistent with and utilizes industry Standards.



# Tool Support

Approach / Feature	Does SCA Next facilitate the incorporation of commercial tools in the production of compliant products?
Conditional Inheritance	<p>The Standards deviation makes it difficult to use out of the box Model Driven Development and software / systems engineering tools. This probably is not a universal deficiency as it is fairly likely that some customization could occur at the MOF layer that could be integrated within the products. If that customization is possible then it would mitigate some of the impacts but it would also present a lingering constraint as it would likely be the case that any other products or abilities combined with these product would need to allow for integration of custom MOF layouts. The impact of the lack of out of the box tool support would be most acutely felt in organizations using some aspect of Model /Tool Driven Engineering. If modeling is not used within a development process or it is, but no code is auto generated or no underlying architectures are extracted then the impacts should be negligible.</p>
Optional Realization	<p>Design approach is consistent with Standards so out of the box commercial tools should work</p>
All Interfaces provided as ports	<p>Design approach is consistent with Standards so out of the box commercial tools should work. Care should be taken in forward engineering to ensure that an optimal model to code transformation is performed . This approach would have more of a pronounced on existing SCA centric tools because it represents a major shift in the associations between the interfaces.</p>
Default Implementation	<p>Design approach is consistent with Standards so out of the box commercial tools should work. Minor customizations would need to be incorporated within the forward and reverse engineering processes . In the forward direction a special indicator would need to be applied to ensure that the default mapping was included. In the reverse direction an indicator would need to map elements to highlight that the default implementation was utilized in the code.</p>
Status Quo	<p>Design approach is consistent with Standards so out of the box commercial tools should work</p>



# Development Support

Approach / Feature	Can SCA Next effectively and efficiently used development environments and tool capabilities to build compliant products?
Conditional Inheritance	<p>Approach is built utilizing a philosophy that the top level interface is the only reliable one for a component and additional ones can be present if those extended capabilities are important to the component being developed. All products in this environment should be built with an alignment to that idea. With the shared understanding in place, it should serve to constrain the OE calls that are produced by generic tools or capabilities. General tools providers can use an iterative approach to identify the composition of constituent elements or those products can take advantage of the same well know switches and flags utilized by the products. A design consideration that needs to be taken into consideration is the address space restriction. Components of differing configurations can not reside within the same address space and it is likely that if they do all of the items that share attributes will have to use the richer definition. This can be mitigated as a weakness because it can be viewed as a design constraint which provides designers with the architectural freedom to create a solution that maximizes the savings.</p>
Optional Realization	<p>Approach complicates the application of environmental tools and capabilities because the supporting environment “doesn’t know” the composition of each of its constituent elements and it doesn’t know which of those aspects is important to the implementation. The framework tools and supporting capabilities can use an approach similar to that used with conditional inheritance to access the top level element, but it could lead to inconsistent results if the underlying products are built with different philosophies. This approach may also share the address space constraint.</p>
All Interfaces provided as ports	<p>Approach flattens the inheritance tree. It preserves the fact that a single well known interface. This shift mandates that identifying or accessing any of the additional aspects or characteristics of a component will be accomplished in an implementation specific manner. Having the ability to delegate removes the potential of being able to consistently utilize polymorphic approaches to access capabilities across implementations. In the long term that might hinder the development of additional capabilities to expand the availability of supporting products.</p>
Default Implementation	<p>Design approach utilizes currently available operating environment capabilities.</p>
Status Quo	<p>Design approach utilizes currently available operating environment capabilities.</p>



# Backward Compatibility

Approach / Feature	How much will SCA Next impact the current collection of compliant products?
Conditional Inheritance	The approaches other than provides ports all provide a mechanism to minimize the impact on existing implementations. The versions that allow optimizations all have an alternative that provides an interface structure similar to that which currently exists. Changes need to be applied to products if the designer chooses to include them in their design. Additional changes will need to occur within the frameworks to support multiple configurations in the configurations that allow optimizations, however those changes should not be overly challenging, in many implementations it should simply be a byproduct of introducing additional decision logic.
Optional Realization	
All Interfaces provided as ports	The approach requires a modification of the existing products and infrastructure to accommodate the new model. The problem of port identity also becomes a question if the spec does not enforce a uniform set of port definitions.
Default Implementation	
Status Quo	



# Summary

---

- **Optional Realization and Conditional Inheritance are the two highest rated alternatives**
  - Both approaches represent viable alternatives that should provide benefits to SCA Next
  - Results could be manipulated if weighting were applied to the categories
- **Conditional Inheritance was selected for the SCA Next draft**
  - Lifecycle cost and performance improvements are primary drivers
  - Approach prevents certain techniques and tools from being applied, however in many cases the lack of those capabilities are desirable for SCA implementations or there are mitigation strategies being explored
  - Prototyping will validate the effectiveness of the mitigation strategy and the extent of the realized improvements



# Backup Slides

---



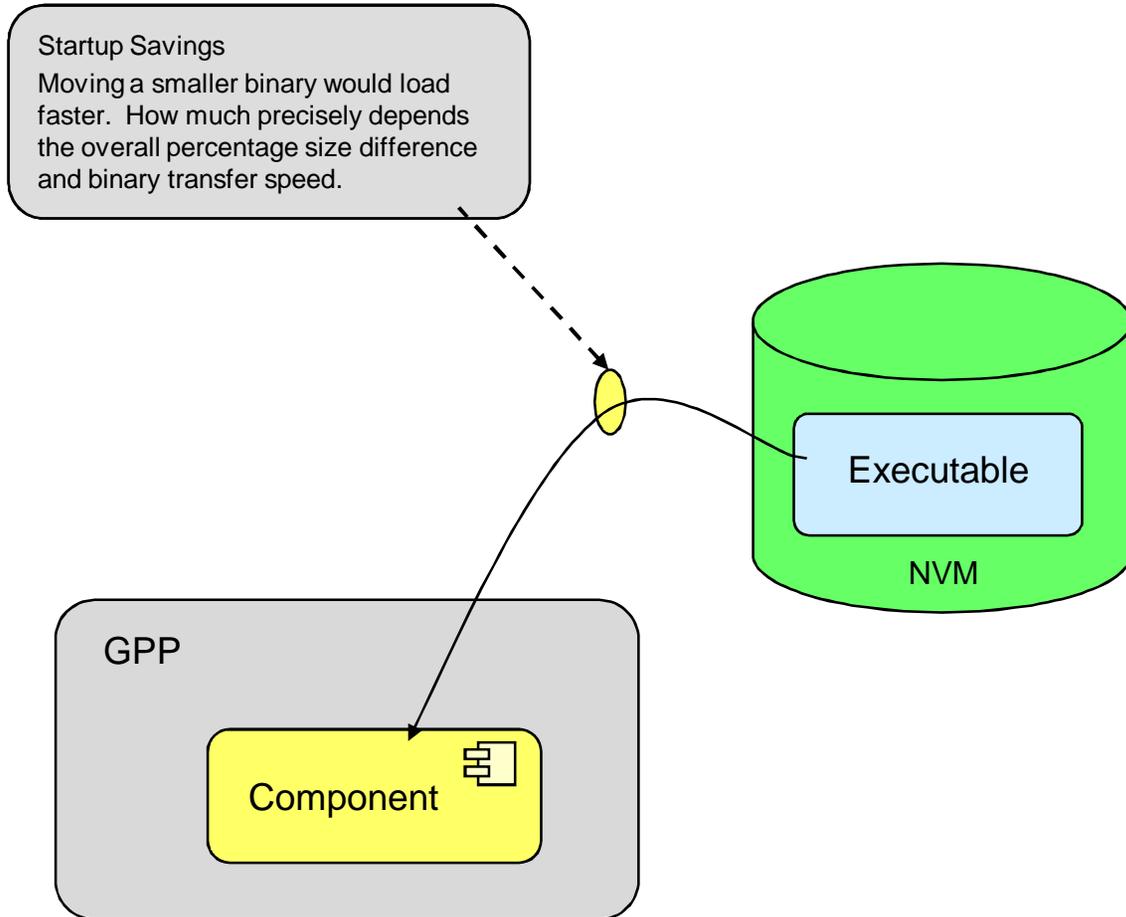
# Prioritized CI Benefits

---

- **Provides additional SCA architecture flexibility to fit wider range of environments**
  - E.g. NASA, commercial industry felt the SCA was too heavy
- **Development Lifecycle Savings**
- **Security improvement**
  - Removes rather than stubs unused interfaces/operations
- **Aligns with and complements other SCA Next Features**
  - Optional Application interfaces
  - Units of Functionality
  - CORBA Profiles task
    - Current micro profile disallows ANY
- **Smaller footprint uses fewer resources and potentially improves start time**



# CI Execution Savings





# Case Study: CI Sizing

- Savings Measurements

- Using *Resource.idl*
  - Compiler Options: ***no debug*** and ***optimize for space***
  - Compiled Stubs and Skeletons code only, ***no impl code***
    - Some Impl code would be required (e.g. to throw required exceptions, etc.)
- Roughly the minimum amount of savings per address space:

Switch	Executable Size (bytes)	delta from NONE (bytes)
NONE	757,060	-
TESTABLE	769,332	12,272
CONFIGURABLE	775,984	18,924
CONNECTABLE	768,244	11,184
CONTROLLABLE	773,392	16,332
INTERROGATABLE	766,824	9,764
V222_COMPAT	825,524	68,464



# Case Study: CI Sizing

- Savings Measurements

- Using *Resource.idl*
  - Compiler Options: ***debug*** and ***no optimization***
  - Compiled Stubs and Skeletons code and ***empty impl code***
    - Impl code still smaller than what may be required (e.g. to throw required exceptions, etc.)
- Roughly the maximum amount of savings per address space:

Switch	Executable Size (bytes)	delta from NONE (bytes)
NONE	146,816	-
TESTABLE	214,004	67,188
CONFIGURABLE	235,192	88,376
CONNECTABLE	208,796	61,980
CONTROLLABLE	226,348	79,532
INTERROGATABLE	204,960	58,144
V222_COMPAT	485,372	338,556