Design of Multi-Platform & SCA-compliant Software Components

Daniele Olmisani – SELEX Communications Spa

daniele.olmisani@selex-comms.com

WInnForum Webinar Series:
The Software Communications Architecture
26/05/2011

SELEX Communications
A Finmeccanica Company

Tutorial Overview

- The SDR-concept introduces challenging design issues in the definition of the portable SW Components
- The SCA specification provides the framework needed in order to enable the portability of Waveform applications among several Platform implementations.
- This tutorial provides an overview on how the WF Portability concepts may be extended in order to enable a better Reusability design and implementation of SW Components.
- The industrial experience of the last years in this field has identified some design key-points that may be applied in order to reduce the effort in adapting Waveforms and Platforms on different environments

Tutorial Workflow

- SW Components as Reusable Products
- Introduction of a specific Case Study
  - Description of the SCA FileSystem Framework Model
  - Identification of Platforms Profiles
  - Analysis of the Components Features
  - Allocation of Component Features to PTF Profiles
  - Examples of Design & Implementation issues
    - Read-only FileSystem implementation
    - FileSystem Backend Support
    - FileManager Connectivity Support
    - FileSystem Security Extension
- Domain Anatomy of a SW Component
- Conclusions

SW Components as Reusable Products

- In terms of Reusability, a SW Component may be considered as a standalone product.

- SW Components may be implemented in different forms, such as:
  - Libraries
  - WF Resources or PTF Devices/Services
  - Distributed Services
  - Stand-alone Applications

- Each SW Component implements a set of features that needs to match specific Platform requirements

- The design of the Component needs to be oriented to the Reusability in order to reduce the effort in adapting the component on different Platform Profiles

Waveform Portability

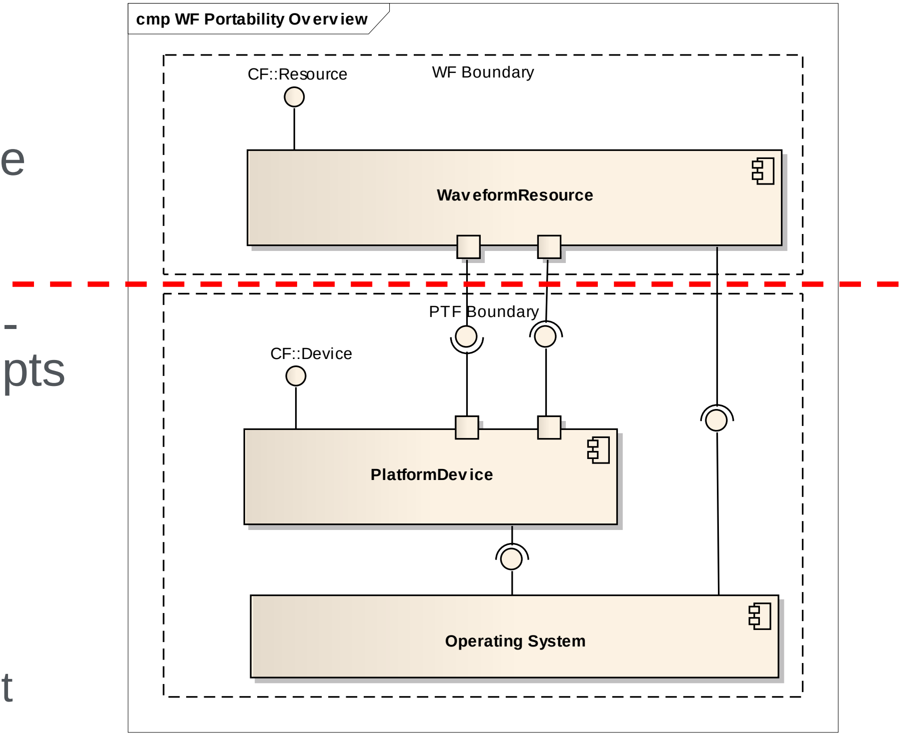
The Domain Scope is specified by the SDR paradigm, as identified in the JTRS SCA specification.

The driver-concept of SCA specification is the **Waveform Portability**.

WF Portability regards the porting at design-time of WF applications (but the same concepts also applies to PTF components);

This goal is achieved by:

- the definition of an Application Framework (i.e. CoreFramework specification)
- the identification of an Application Environment Profile
- the definition of Connectivity Middleware (i.e. CORBA)
- the definition of APIs (e.g. Device I/O API, …)



Component Reusability

The component boundary identified for the **WF Portability** is too wide to enabling an efficient **Component Reusability**.

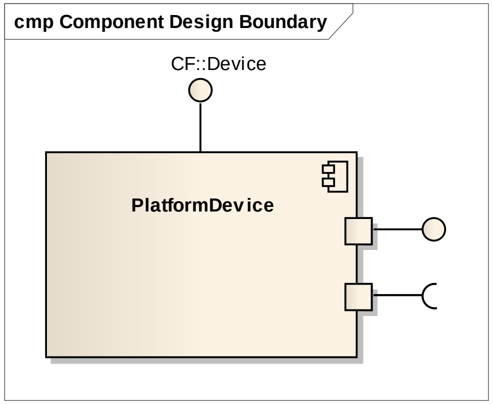
A more detailed design dissection is needed in order to improve the efficiency of design/implementation reusability.

Component Reusability deals with the decomposition in functional modules in order to:

- provide a partitioning in coherent set of capabilities
- support set of specific functional requirements (i.e. Platform Profiles)
- support specific interfaces and connectivity mechanisms

The **Component Reusability** is supported by the following design concept:

- **Modularity**: decomposition in autonomous entities
- **Connectivity**: interaction and middlewares
- **Scalability**: resources consumption
- **Extensibility**: improvement of basic functionalities



Case Study

In this Tutorial, the concept of **Components Reusability** is detailed by the means of a specific Case Study based on the FileSystem Framework specified in the JTRS SCA.

The Case Study covers the following topics:

- Description of the SCA FileSystem Framework Model
- Identification of Platforms Profiles
- Analysis of the Components Features
- Examples of Design & Implementation issues
  - Read-only FileSystem implementation
  - FileSystem Backend Support
  - FileManager Connectivity Support
  - FileSystem Security Extension

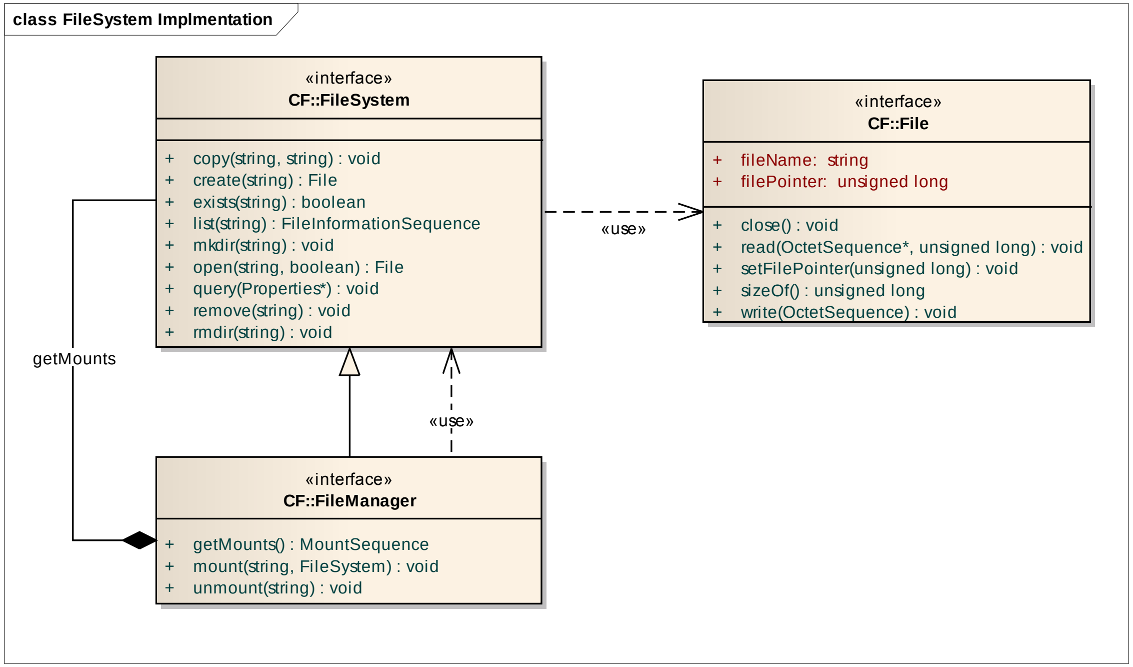
The main goal is to identify common approaches for component design/implementation in order to improve the **Component Reusability**

SCA FileSystem Framework Model

- **File** represents a single file or folder; provides the operations needed in order to read and write the content of a file

- **FileSystem** represents a portion of the file system (folders and files); provides operations needed in order to list, create, delete and open files and folders

- **FileManager** represents a specialization of the FileSystem in order to manage a federation of FileSystem as a single entity



class FileSystem Implmentation

«interface»
CF::FileSystem

+ copy(string, string) : void
+ create(string) : File
+ exists(string) : boolean
+ list(string) : FileInformationSequence
+ mkdir(string) : void
+ open(string, boolean) : File
+ query(Properties*) : void
+ remove(string) : void
+ rmdir(string) : void

«interface»
CF::File

+ fileName: string
+ filePointer: unsigned long

+ close() : void
+ read(OctetSequence*, unsigned long) : void
+ setFilePointer(unsigned long) : void
+ sizeOf() : unsigned long
+ write(OctetSequence) : void

«interface»
CF::FileManager

+ getMounts() : MountSequence
+ mount(string, FileSystem) : void
+ unmount(string) : void

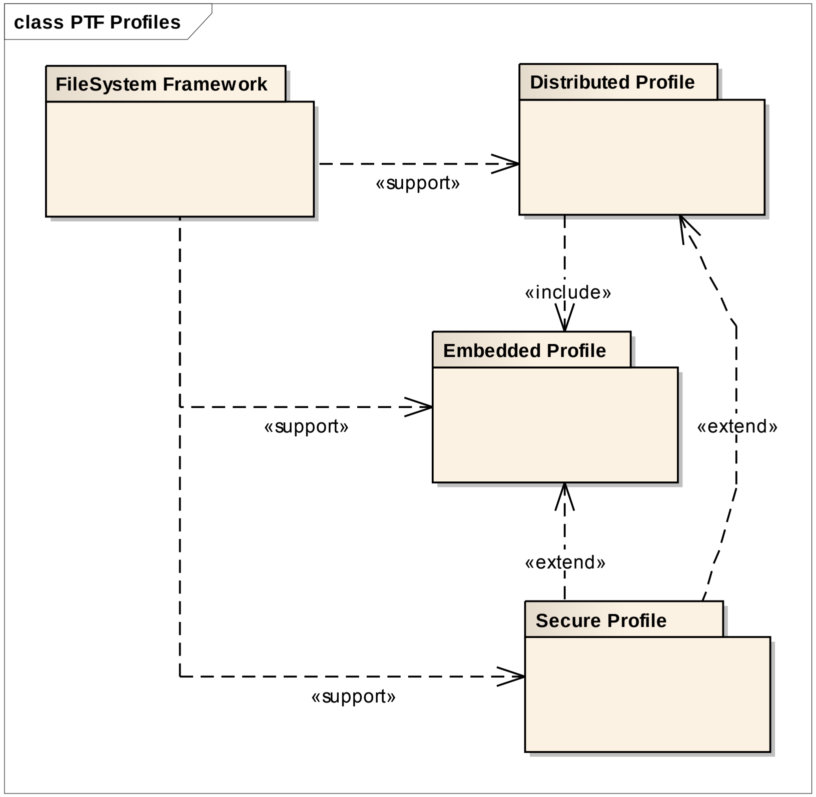
Definition of Platform Profiles

The first step in component modularization is the identification of the families of Platform requirements.

Three different PTF Profiles are identified:

- **Embedded Profile**: PTF with limited HW resources and limited set of functions (e.g. a read-only file system)
- **Distributed Profile**: PTF distributed among heterogeneous processing environments, with a larger set of functions (e.g. federation of file system implemented using different technologies)
- **Secure Profile**: PTF where security-related functions are used in order to extend the basic file system functions

The FileSystem Framework should support the requirements of the identified PTF Profiles



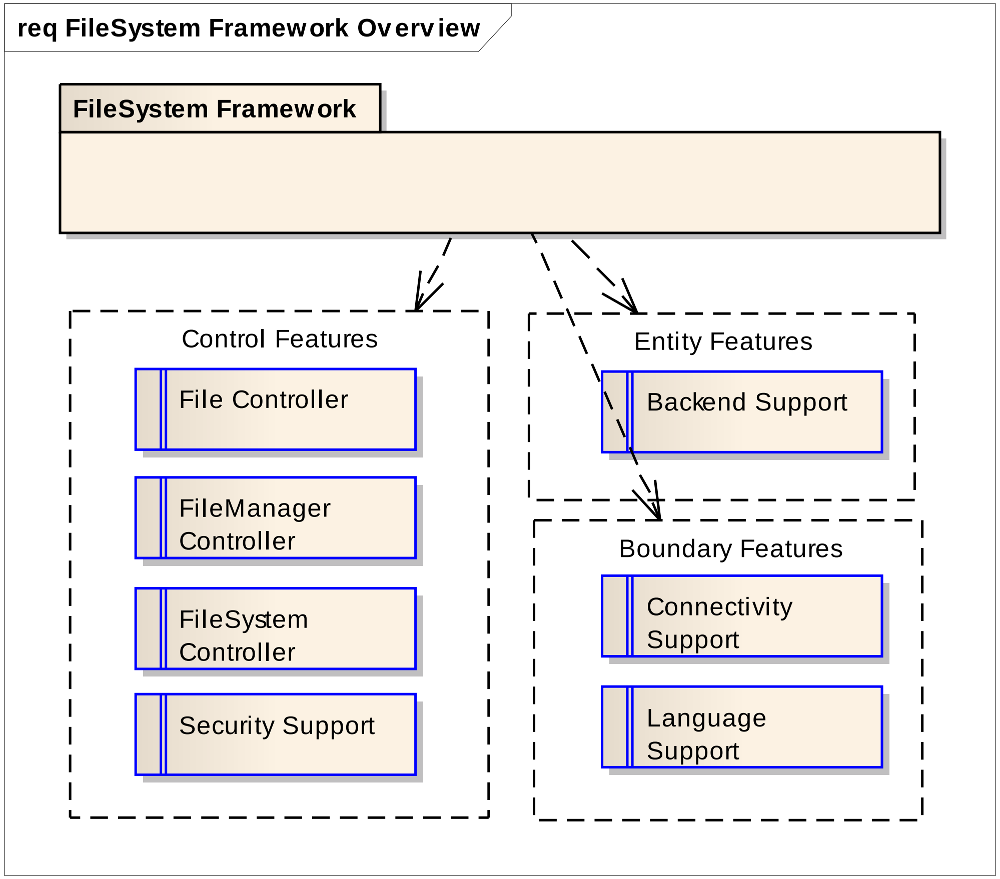
Feature Analysis 1/2

The second step in component modularization is related to the analysis of the requested features.

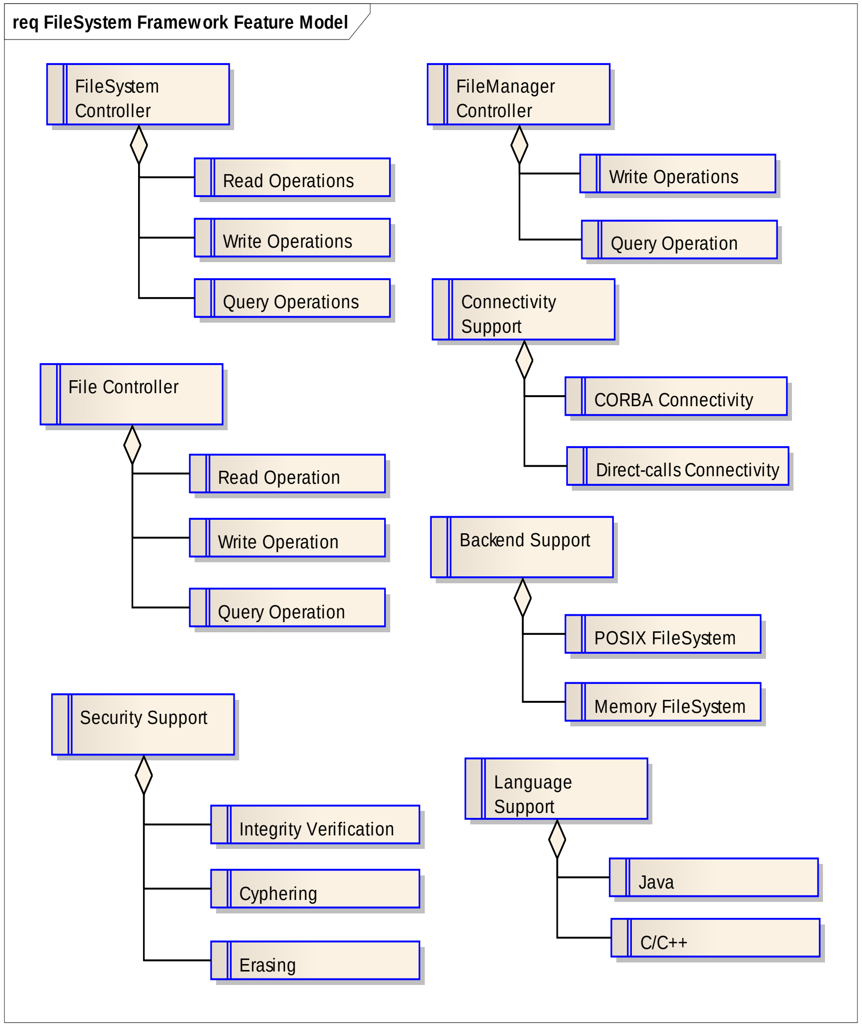
Entity Features are related to the storage of the information

Control Features are related to the processing of information (the business logic)

Boundary Features are related to exchange of information (interfaces and middlewares).



Feature Analysis 2/2



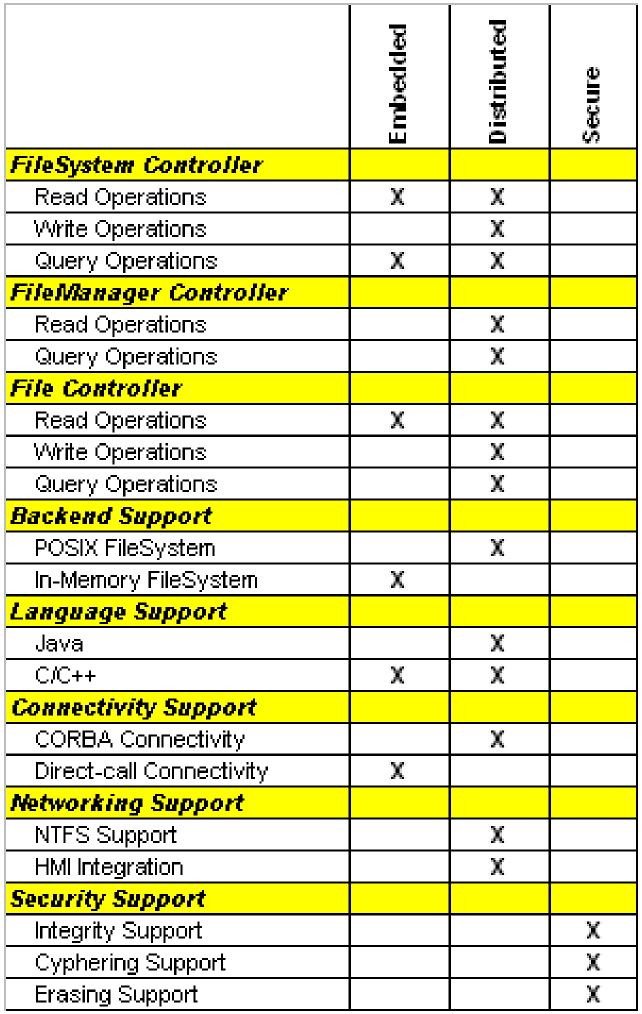
- **File Controller**: operations related to the management of files and folders
- **FileSystem Controller**: operations related to the management of sets of files and folders
- **FileManager Controller**: operations related to the federation of file systems
- **Security Support**: operations related to the protection of the stored information
- **Connectivity Support**: support of different interaction mechanisms
- **Language Support**: support to different application environments
- **Backend Support**: support to the implementations of the storage capabilities

Features Allocation to PTF Profiles

As a final step, the component features are allocated following the definition of each PTF Profile.

Several design/implementation issues may be identified:

- Separation of concerns is needed in order to split the implementation of read-only and read-write file systems
- Heterogeneous environments are to be supported (e.g. supported programming environment and connectivity mechanisms)
- New functions have to be plugged on existing ones (e.g. security functions)
- Different backend implementations are to be supported



	Embedded	Distributed	Secure
FileSystem Controller			
Read Operations	X	X	
Write Operations		X	
Query Operations	X	X	
FileManager Controller			
Read Operations		X	
Query Operations		X	
File Controller			
Read Operations	X	X	
Write Operations		X	
Query Operations		X	
Backend Support			
POSIX FileSystem		X	
In-Memory FileSystem	X		
Language Support			
Java		X	
C/C++	X	X	
Connectivity Support			
CORBA Connectivity		X	
Direct-call Connectivity	X		
Networking Support			
NTFS Support		X	
HMI Integration		X	
Security Support			
Integrity Support			X
Cyphering Support			X
Erasing Support			X

Design and Implementation Issues

The design and implementation of the FileSystem Framework should be optimized following the Feature allocation.

The following example of component design are proposed:

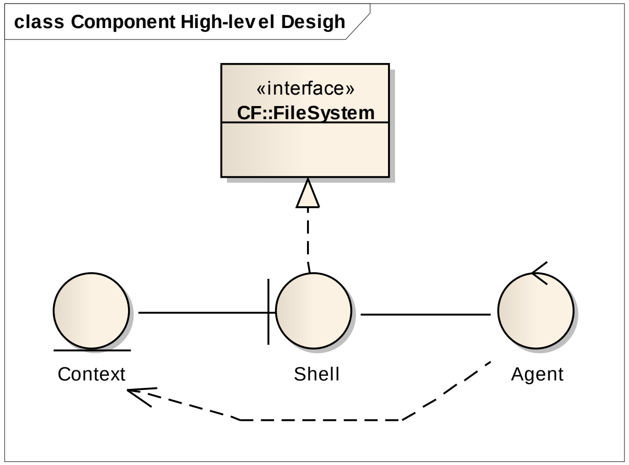
- Implementation of the Read-only FileSystem
  related to the separation of the operation provided by the same interface

- FileSystem Backend Support
  related to the need of supporting different backend implementations

- FileManager Connectivity Support
  related to the support of heterogeneous connectivity mechanism

- FileSystem Security Extension
  related to the extensibility of basic functionalities

High-level Component Design



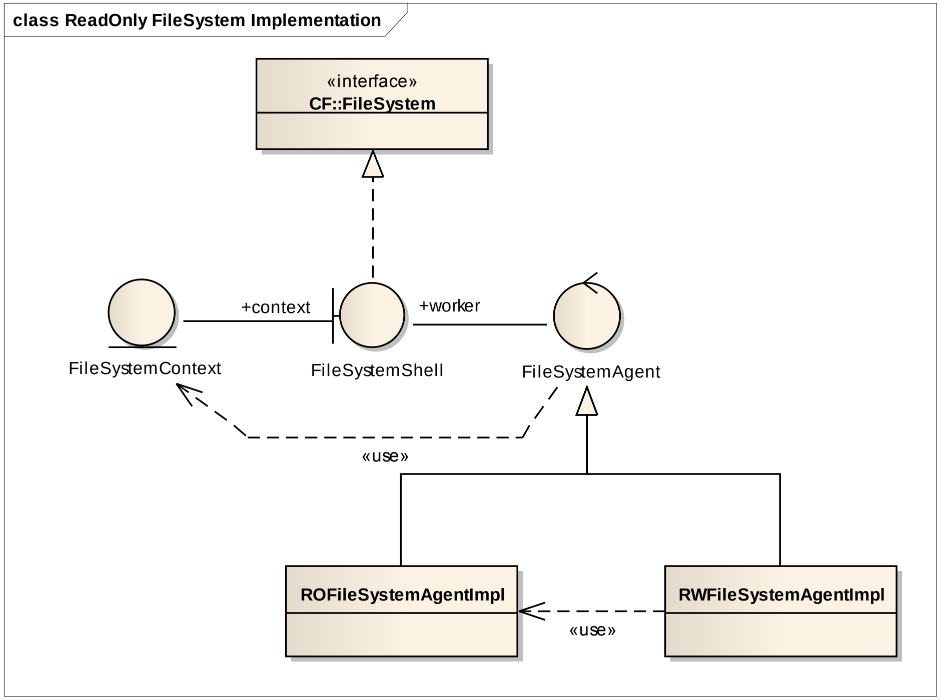
This high-level design model will be used in the following example:

- **Shell** implements the boundary framework; it supports the interaction and connectivity mechanisms and provides the external interfaces

- **Agent** implements the core logic of the component; it uses the Shell in order to communicate with other components

- **Context** implements the internal states and data structures of the component

Read-only FileSystem Implementation

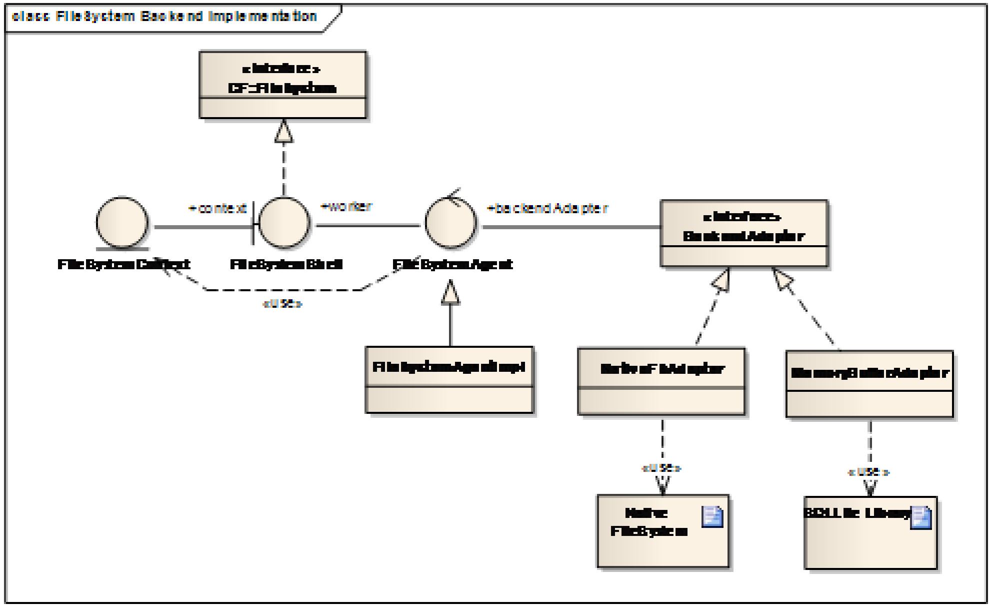


In this example the control logic implemented by the FileSystemAgent is partitioned in two classes:

- **ROFileSystemAgentImpl** implements only “read” and “query”operations; an exception is returned in case of “write” operations
- **RWFileSystemAgentImpl** implements the “write” operations and uses the other class implementation for the “read” and “query” operations

The CF::FileSystem interface is partitioned in two separated implementations

FileSystem Backend Support



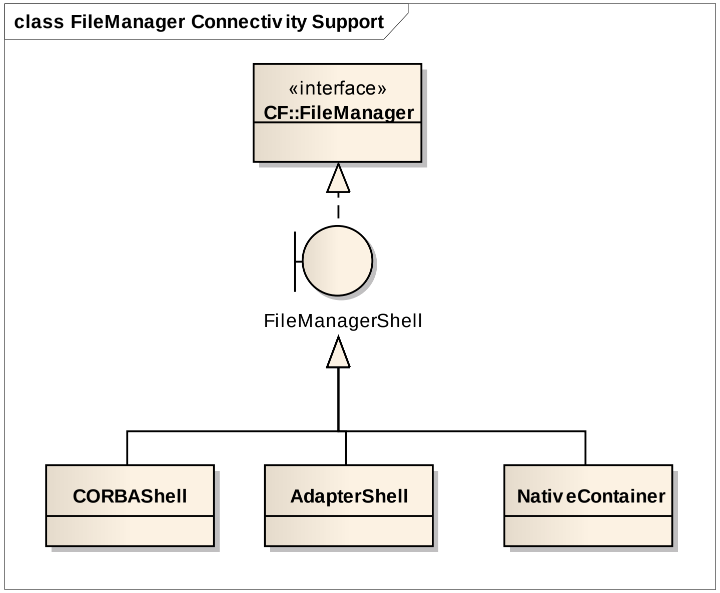
In this example the control logic implemented by the FileSystemAgent is designed in order to use a BackendAdapter.

This adapter is specialized in order to support different backend implementations:

- Native file system (typically accessible by a POSIX interface)
- A file system implemented in a memory buffer (e.g. by the support of a COTS library)

The FileSystem control logic is independent from the specific backend implementation

FileManager Connectivity Support

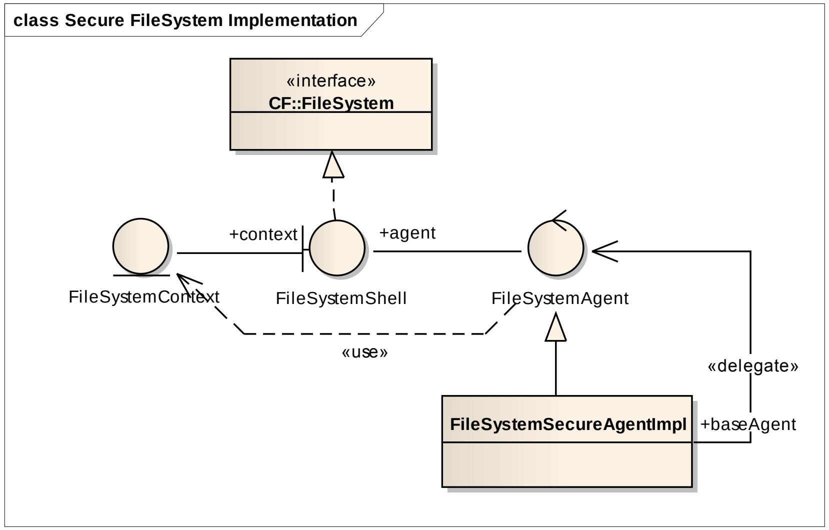


In this example the boundary framework implemented by the FileManager shell is extended in order to support different connectivity mechanisms.

- CORBA Middlewares: the component control logic is able to communicate in a transparent way
- Custom Adapter: specific communication adapters are implemented in order to support custom connectivity mechanisms
- Native calls: the interactions are performed by direct or adapted function calls (e.g. Calls to dynamic libraries or Java Native Interface adaptation)

The component control logic is not impacted by the specific connectivity mechanisms

FileSystem Security Extension



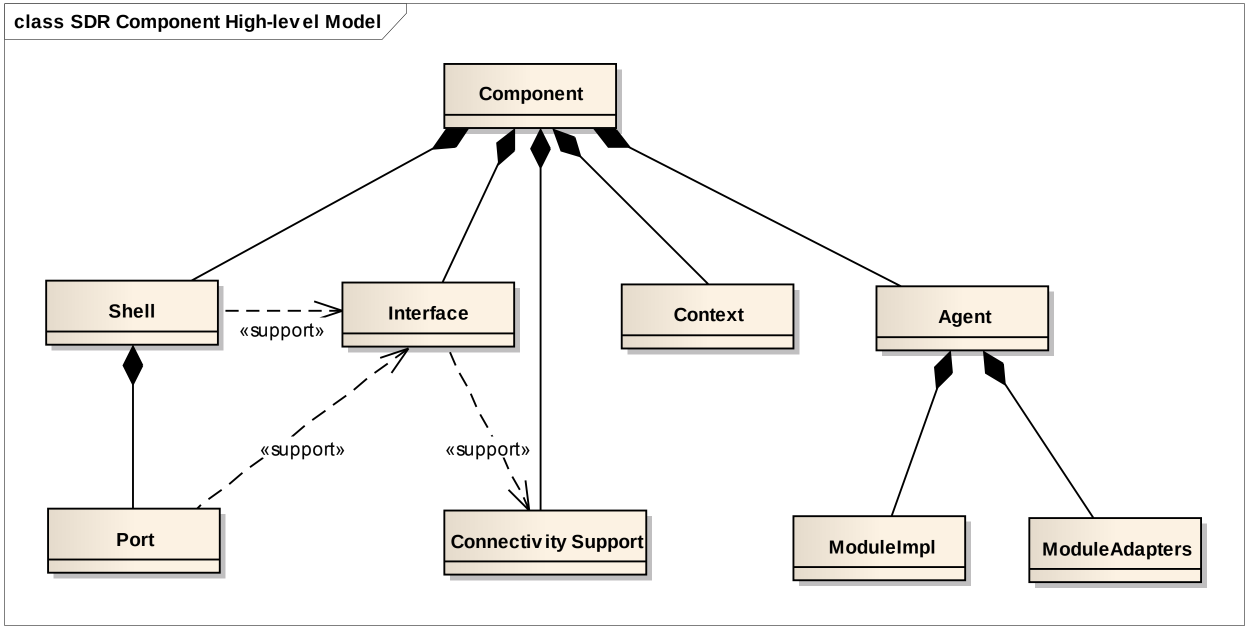
In this example the basic control logic implemented by the FileSystemAgent is extended in order to support security-related capabilities implemented by the FileSystemSecureAgentImpl.

- The basic FileSystemAgent logic is extended in order to be used as a standard control logic by the container logic

- A basic implementation (baseAgent) is used in order to delegate the execution of the standard “read”, “write” and “query” behavior.

The basic control logic is completely reused in order to extend the standard behavior.

Component Domain Analysis



A general component model is derived following the previous design approaches.

- The **Shell Modules** provide the needed connectivity by the implementation of ports and supported interfaces (connectivity)
- The **Agent Modules** provide the control logic and use:
  - **Impl Modules** in order to provide features composition (modularity and extensibility)
  - **Adapter Modules** in order to provide the abstraction from a specific feature implementation (scalability)

Conclusions

- The SW Components Portability is a Design achievement

- The main focus is related to the Domain Analysis:
  - **Feature Analysis Model**. related to the component needs and environment constraints
  - **Component Analysis Model**. related to the identification of architectural elements that resolve specific portability issues

- Implementation Challenges
  - **Shell Implementation**: related to the connectivity issues
  - **Module Implementation**: related to the modularity and extensibility of features implementation
  - **Module Adapters**: related to interactions and adaptations issues

References

People

- Daniele Olmisani, SELEX Communications Spa
  daniele.olmisani@selex-comms.com

Keywords

- SDR, Software Defined Radio, JTRS, SCA, Software Communications Architecture, Design Patterns, FOD, Feature Oriented Design, SOA, Service Oriented Architecture, MDA, Model Driven Architecture

Web & Documents

- JTRS SCA Specification
  http://www.public.navy.mil/jpeojtrs/sca/Pages/default.aspx
- Quick reference on Design Patterns
  http://en.wikipedia.org/wiki/Design_pattern_(computer_science)
- Tutorial on the CORBA Component Model Architecture
  https://svn.dre.vanderbilt.edu/viewvc/Middleware/trunk/CIAO/docs/OMG-CCM-Tutorial.pptx?view=co
- Introduction to the Feature Oriented Design
  http://www.jot.fm/issues/issue_2009_07/column5/
- Introduction to the Entity-Control-Boundary Design Pattern
  http://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/ecb/ecb.htm