Arquitecturas Software

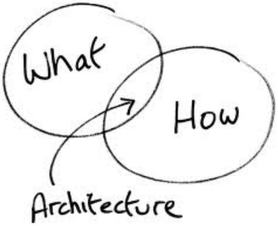
Desarrollo de Software Basado en Componentes y Servicios

M.I. Capel

ETS Ingenierías Informática y
Telecomunicación
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

October, 9th 2017
Máster Universitario en Ingeniería Informática

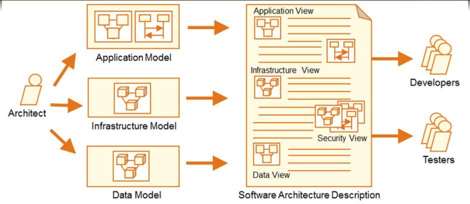
Software Architecture



Definition

"High-level representation of a software system/application's own structure, which *defines* its parts, the interactions between these parts and the (architectural) patterms that supervise composition of parts and the constraints to abide when the patterns are applied".

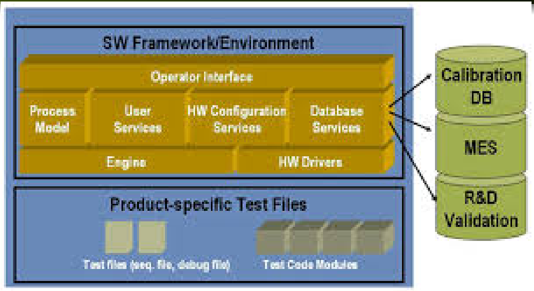
Software Architecture - II



Objectives

- To understand and better manage the internal structure of complex software
- To ease the reuse of that structure as a whole or parts of it
- To plan software applications evolution by identifying mutable and immutable parts and future changes cost

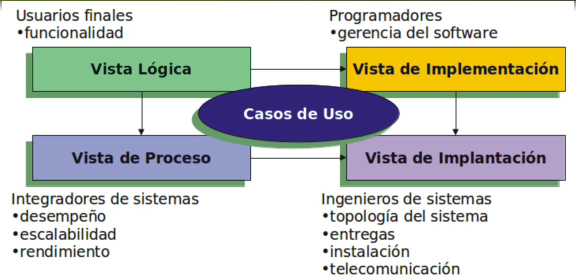
Frameworks



Definition

“Reusable design of a software system as a whole or its parts, which is represented by a set of abstract classes and defines the way in which they interact with others in their environment”.

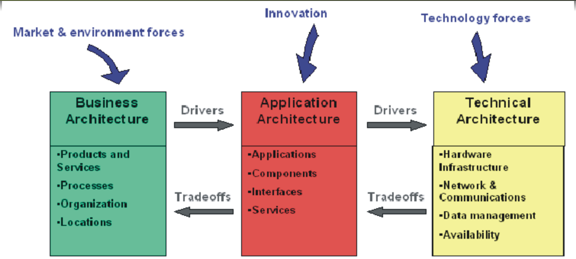
Abstract Models of Software Architectures



4+1 View Model of Krutchen

This the RUP model of software architecture. Currently, software systems development is centered on the Software Architecture (SA), which is understood by using the following 4+1 views model above.

Business Architectures Cycle (ABC)



- According to [Bass et al., 2012], the mechanisms in figure above make up what is known as “ABC”
- In [Kruchten, 1995] is also disclosed the feedback that exists between the SA and the business. Three architectural fields are mentioned, which show a close connection among them during software development.

What is a Software Architectural Style

Wikipedia

“A software architectural style is characterized by a set of features that make the software architecture uniquely identifiable. Software architectural styles generally provide a high level direction for solutions unlike software patterns which are focused on solving one or more specific problems”.

- A SAS defines a family of software systems in terms of their structural organisation.
- Defines a vocabulary of components and connector types

SAS examples

Pipes and filters, ADTs and OO, Repositories, layer-based architectures, event-driven architectures, Intérpreters, etc.

Architectural Styles - II

- To adhere to a specific architectural style improves or worsens the chances of satisfaction of software quality attributes [Jansen and Bosch, 2005]
- Each architectural style propitiates quality attributes and to make the decision of implementing one of them depends on the system's quality requirements
- An important success criteria for selecting a style to use is how these reach the SE objectives in a satisfactory way [Buschmann and et al., 2004]

Differences between architectural styles and patterns

Architectural Style

-Only describes the structural and general skeleton of software applications
-Independent of the application context
-Each one is independent of the others
-Express design techniques from a independent perspective regarding the current design target
-Can also be understood as a classification of software systems

Architectural Pattern

-Several different scale ranges, beginning with the definition of the application basic structure
-Need the definition of the problem context
-Depend on smaller patterns, with which they interact
-Depend on bigger patterns in which they are included
-Recurrent problem solution expresion, which is too specific to a particular context
-General solution to a common problem

Architectural Pattern

Buschmann (1996) Definition [Buschmann and et al., 2004]

A rule of three parts that express a realtionship among a context, a problem and it solution.

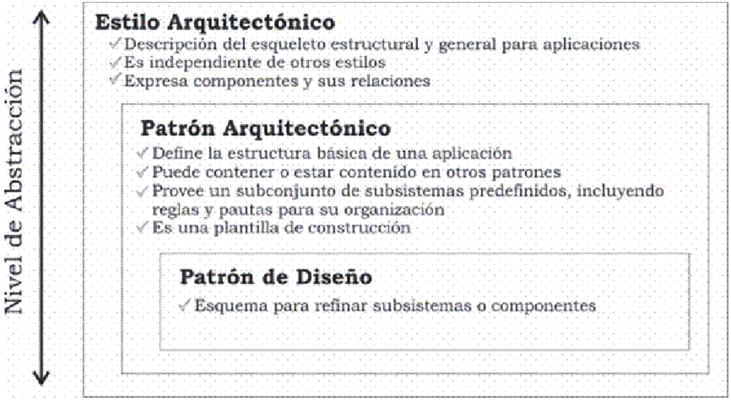
1. *Context*: a modeling situation of a system's part in which appears a specific design problem
2. *Problem*: a set of forces that repetitively appear in a given context
3. *Solution*: a configuration of objects that balances the forces.

Architectural Patterns - II

According to Buschmann, architectural patterns are specific software architectures, which describe the structural properties of a software system and have impact of the architecture of its subsystems

The deployment of specific mechanisms, such as architectural patterns and styles, allow us to improve the quality characteristics of software [Jansen and Bosch, 2005], whether they can be observed or not at execution time [Bass et al., 2012].

Differences according to the *abstraction level*


Estilo Arquitectónico
✓ Descripción del esqueleto estructural y general para aplicaciones
✓ Es independiente de otros estilos
✓ Expresa componentes y sus relaciones

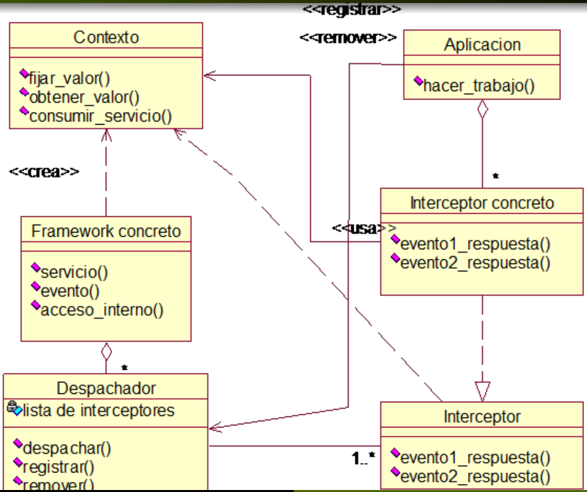
Patrón Arquitectónico
✓ Define la estructura básica de una aplicación
✓ Puede contener o estar contenido en otros patrones
✓ Provee un subconjunto de subsistemas predefinidos, incluyendo reglas y pautas para su organización
✓ Es una plantilla de construcción

Patrón de Diseño
✓ Esquema para refinar subsistemas o componentes

Interceptor Pattern

- It allows us to include specific services to a framework, in a way that is entirely *transparent* for us, which will be automatically performed when certain event occur in a *context*
- Pattern-Context: Development of frameworks susceptible of being extended transparently to their users
- Problem: Frameworks, software architectures, etc. that have the capability of anticipating changes or requests of specific services on user demand
- Dynamic integration of new components without affecting the SA or to other software components already included in the framework
- Solución: Register of services *offline*, through a predefined interface of the framework, then these services will be

Interceptor



Interceptor Classes Structure

- A *FrameworkConcreto* that instantiates a generic and extensible architecture in order to give users services implemented by a particular system
- *Interceptores*, each one is associated to a particular event
- *InterceptoresConcretos*, each one specializes the interceptor interfaces and implement the interceptor's binding methods
- *Despachadores* (dispatchers) for configuring and firing specific interceptors
- An *Aplicación* that executes on a *specific framework* and uses the services provided by it

Implications on quality of software when the Interceptor pattern is used

Benefits	**Quality Attributes**	**ISO 9126 Characteristics**
Change/include services of a framework without being necessary to change it	Extensibility, Flexibility, Dynamicity	Mantenibility Ease changes
To included interceptors without affecting to application code	Ciupling	Maintainability Ease changes
Dynamic information of the framework with interceptors and context-objects	Monitoring Control	Fault tolerance Resource deployment
Stratified service infraestructure with symmetric correspondent interceptors	Encapsulation	Mantenibility Ease changes, analysis
Interceptor resuse in different applications	Reusability	Maintainability Ease changes

Implications on quality of software - II

Drawbacks	Quality attribute	ISO 9126 Characterstics
Difficult setting of the number of dispatchers and interceptors	Complexity Flexibility, extensibility	Ease changes Eases analysis
Application locking by interceptor failure	Disponibility Modifiability	Maintainability Maturity, Fault tolerance
Performance deterioration by chain of interceptors	Performance Locking	Efficiency,maturity Fault tolerance

- insufficient number of interceptors and dispatchers lessens the flexibility and extensibility of a given framework
- A big and ineficient system, a deep learning curve, difficult to implement, complex to use and optimize can result if we use too many interceptors
- To avoid the complete locking of the software application, we can implement *time-out* strategies but it might cause a complex design of the entire framework
- Chains of interceptors can imply a deterioration of software performance or lead to application locking up

Summary of software quality characteristics

Característica	Sub-característica	Impacto	Atributo
Mantenibilidad	Facilidad de cambio Facilidad de análisis	+	Reusabilidad Modificabilidad Encapsulamiento Extensibilidad Flexibilidad Acoplamiento Dinamismo
	Facilidad de cambio Facilidad de análisis	-	Extensibilidad Flexibilidad Complejidad Modificabilidad
Eficiencia	Tiempo de respuesta	-	Desempeño
	Uso de recursos	+	Monitoreo Control
Fiabilidad	Tolerancia a fallas	-	Disponibilidad Bloqueo
		+	Monitoreo Control
	Madurez	-	Disponibilidad Bloqueo

Summary of software quality characteristics propitiated by the pattern -II

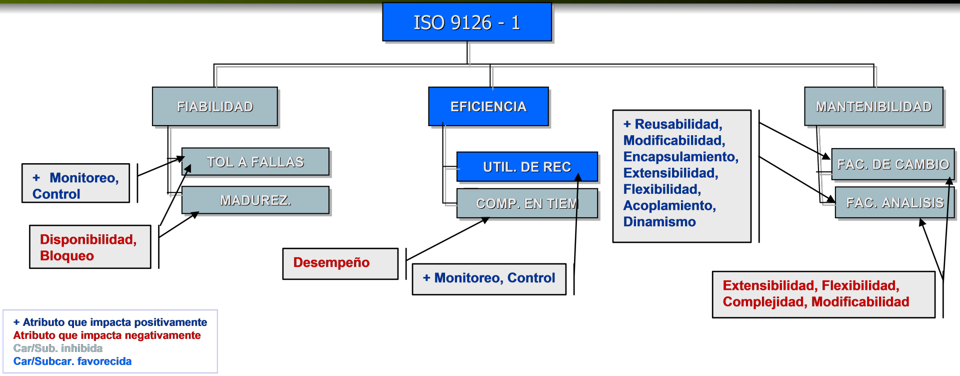

ISO 9126 - 1
FIABILIDAD
EFICIENCIA
MANTENIBILIDAD
TOL A FALLAS
MADUREZ.
UTIL. DE REC
COMP. EN TIEM
FAC. DE CAMBIO
FAC. ANALISIS
+ Monitoreo, Control
Disponibilidad, Bloqueo
Desempeño
+ Monitoreo, Control
+ Reusabilidad, Modificabilidad, Encapsulamiento, Extensibilidad, Flexibilidad, Acoplamiento, Dinamismo
Extensibilidad, Flexibilidad, Complejidad, Modificabilidad
+ Atributo que impacta positivamente
Atributo que impacta negativamente
Car/Sub. inhibida
Car/Subcar. favorecida

Figure: ISO 9126 characteristics of “Interceptor" pattern

Bibliografía Fundamental

Bass, L., Clements, P., and Kazman, R. (2012).
Software Architecture In Practice.
Addison-Wesley, Boston, Masssachussets, third edition.

Booch, G. (2008).
Handbook of Software Architecture.
http://www.booch.com/systems.jsp.

Buschmann, F. and et al. (2004).
Pattern-oriented software architecture, volume I.
Wiley, Chichester, England.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2010).