Homework #3

Exercise 1: Architecture-Significant Requirements (Catalogue of Criteria)

Consider the following list of the the software non-functional requirements or quality factors. They are classified as external (black) and internal (red).

Correctness	The extent to which the software satisfies its specifications and fulfils the user's objectives. Also the extent to which the software is fault free (i.e., free from design defects and from coding defects).
Efficiency	The extent to which the software performs its intended functions with a minimum consumption of computing resources (resource consumption for given load).
Effectiveness	Effectiveness (resulting performance in relation to effort)
Extensibility	The ease with which the software can be modified to add functionality.
Fault Tolerance	The property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components.
Integrity	The extent to which unauthorized access to or modification of the software or data can be controlled in a computer system (either manually or with automated tools).
Interoperability	The ability of two or more systems to exchange information and to mutually use the information that has been exchanged.
Maintainability	The ease with which the software components can be maintained over their expected useful life.
Modularity	The extent to which the software is structured in high cohesive and low coupled units (modules)
Portability	The ease with which the software can be transferred to new operating environments, hardware platforms, and operating systems.
Privacy	The ability of software to keep away its users and information about they from other people and thereby express itself selectively.
Reusability	The extent to which the modules can be used in multiple applications.
Reliability	The ability of the software to perform a required function under stated conditions for a stated period of time.
Robustness	The ability of the system to handle abnormal situations .
Security	The ability of the software to ensure confidentiality, integrity, and availability of computers.
Scalability	The capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth
Testability	The extent to which the software is easy to test to ensure that it performs its intended function.
Understandability	The extent to which the software is easy to be understood.
Usability	The extent to which the software is easy to learn and operate.

Please, distinguish architecture-significant requirements from such that do not drive architecture decisions or have significant architectural impact if they are adjusted in future.

This exercise is about developing a catalogue or list as the following one that architects can use in order to determine for a given requirement, whether it is architecture-significant or not, and why. The criteria catalogue must be generic, thus it can be used and applied in different architectural contexts and engagements. Order, if possible, requirements depending on the weight of the impact they have on architectures.

Nr.	Non-functional requirement name	Yes /Not architecture-significant	If Yes, how may the requirement influence the architecture, according you?
1.			
2.			

2. What is the difference between the architecture of a system and its representation?
3. What are frameworks? How are design patterns different from frameworks?
4. Is the code generated by a "wizard" in an Integrated Development Environment (IDE) a framework?

5. Ruby on Rails, Struts, and other frameworks for delivering dynamic web pages follow the model-view-control (MVC) architecture pattern. Identify the dependencies and nature of the dependencies between the model, view and controller components when the MVC architecture pattern is used to deliver dynamic web pages. Express your answer using the following diagram by crossing out the dependencies that don't exist and describing those that do.