**Software Architecture**

A software architect's duties include designing, documenting, evaluating, and evolving a software architecture that can be used to deploy and implement functionality while still meeting system quality standards. Many significant design decisions are known to be contained in the software architecture definition, and it is accepted that these decisions are difficult to modify at a later stage. Despite the fact that software architecture is not a new discipline, there is still disagreement on what it truly is. There are several definitions available, and they differ in several key aspects. The majority of definitions leave the distinction between application design and architecture up for interpretation. Here, "application design" is defined as the process of creating components that meet the functional needs of the user. The resulting application components are then implemented in a software architecture that satisfies the system's quality standards as well as the architectural requirements. The many approaches to software architecture are expressed in a broad variety of Architecture Description Languages (ADLs), each of which has a different set of capabilities for describing a software architecture. The definition of software architecture that is chosen also affects the architectural requirements specification's content, the architectural design process and its outputs, and how software architectures are evaluated. Therefore, the choice of software architectural definition has a direct influence on the duties of and skill requirements for software architects. Thus, there would be enormous advantages if the software development community could agree on the idea of software architecture The meticulous design, description, and evaluation of software architecture have been the main topics of study in the field of software architecture, which has included workshops, such as the ISAW series, conferences, such as the WICSA series, special issues of journals, and publications. Although the evolution of software architecture has received some attention, the main issue facing the software architecture community has been that software architectures must be carefully designed because changing a system's software architecture after its initial design is frequently very expensive. Changes that have an architectural influence are frequently cited as the key consequence to avoid. It's interesting that research on software architecture nearly solely concentrates on this element of the issue. so Effective continuous system evolution is made possible by an effective software architecture backed by agile architectural methods. Practices like these include analyzing the deployed system for architecture conformance and documenting the architectural elements and relationships intended to achieve key qualities. These evaluations are repeated to determine whether the architecture is fit for an organization's business and mission goals. These procedures, when carried out correctly, provide for predictable product quality, fewer difficulties later on, time and money savings during integration and testing, and cost-effective system evolution.

الاسم: لينا نوار المطيري عدد الكلمات: 450 كلمة

الرقم الجامعي: نسبة الاقتباس: 5%

**Software Design**

Software design is the process by which an agent develops a specification of a software artifact designed to achieve goals using a collection of simple building blocks and subject to limitations. Software design may be defined as either "the activity following requirements definition and before to programming, as... [in] a stylized software engineering process," or "the activity involved in envisioning, framing, developing, commissioning, and eventually altering large systems." Planning a software solution and problem-solving are often components of software design. This covers both high-level architectural design and low-level component and algorithm design.Software design is a model as well as a method. A series of processes known as the design process provide the designer the ability to fully explain every component of the program before it is built. Critical success criteria for a competent design include creativity, expertise, knowledge of what constitutes "excellent" software, and a general dedication to quality. However, it's crucial to keep in mind that the design process is not always a simple process; the design model might be likened to a home's architectural designs. It starts by depicting the entire item that has to be created (for example, a three-dimensional depiction of the home), then gradually refines it to offer instructions for building each detail (e.g., the plumbing lay).Similar to how multiple viewpoints of the computer program are provided by the design model designed for software, The software engineer may traverse the design process using fundamental design concepts. offers a list of guidelines for software design, which the following list has expanded and modified: There shouldn't be any "tunnel vision" in the design process. A smart designer should think about different strategies, evaluating each one according to the needs of the issue and the tools at hand, The analytical model must be able to be followed from the design. A method for monitoring how requirements have been completed by the design model is required since one design model component might frequently be linked to several requirements, The design shouldn't create something entirely new. Systems are built using a collection of design patterns, many of which you've already seen before. It is usually preferable to use these patterns as opposed to innovation. Since time is limited and resources are scarce, design attention should be spent on illustrating (really novel) concepts by using existing patterns (when applicable) , The program and the issue as it exists in the actual world should be as close to one another intellectually as possible. In other words, if possible, the software design's structure should resemble that of the issue domain, The design ought to be unified and integrated. If a design seems completely coherent, it is uniform. Before beginning design work, a design team should determine the guidelines of style and format in order to attain this result. If the interactions between design elements are well defined, a design is integrated.

.الاسم: لينا نوار المطيري عدد الكلمات: 480 كلمة

الرقم الجامعي: نسبة الاقتباس: 0%

**Architectural patterns**

We must select an architecture that will provide us the needed quality qualities before beginning serious work. As a result, before architectural alternatives are used in a design, a mechanism must be found to debate them and the quality implications in advance. At this point, the only basis for our judgments is our prior knowledge of the architectural options available in various systems. Architectural patterns serve as a record of such architectural experience, distilled to its core and freed from the complexities of the processes that generated it. A lot of systems have a comparable structure. For instance, client-server architectures are a common feature of distributed systems. Clients submit requests, which the server processes and responds to. When we see a pattern, we want to discover what the solutions have in common and what alternative variants there are. We enquire as to the conditions under which a method may be applied as well as the best way to apply it and customize it for a certain system. Such concerns are addressed by an architectural pattern.,,Patterns : A collection of specified subsystems and their roles are described by a pattern. For example, in a system designed using the Client-Server pattern, there are two distinct subsystems: the client (which may have several instances) and the server (which is unique). The client may be in charge of providing the user with a user interface, while the server may be in charge of handling numerous queries and protecting client-relevant data.The interactions between the subsystems are organized according to rules and standards that are described in a pattern. The client and server have a dialogue in which the client asks inquiries and the server responds.9.. People with a lot of experience write patterns. In the form of patterns, knowledge that may have remained implicit in the minds of these seasoned individuals is made plain. By doing this, others can benefit from that experience. The experience of many developers is reflected in patterns, which are not created by a single individual. They record current, tried-and-true software development solutions and support the advancement of excellent design techniques.Architectural patterns are often known as architectural styles or standard architectures, however the term "architectural style" is more frequently used to refer to a less precise idea than a pattern; as a result, many patterns may share the same architectural style.so, WHY ARE PATTERNS HELPFUL? It becomes a pattern when several developers tackle a particular type of problem in a consistent manner, and it is widely acknowledged that this approach effectively resolves the problem. Therefore, a pattern is anything that tackles a common design issue for which an all-encompassing answer is well-known among seasoned practitioners. A pattern records current design solutions that have been successfully implemented, Reusing the solution is made simpler by documenting the pattern. Patterns provide design solutions a common language and comprehension.

..الاسم: لينا نوار المطيري عدد الكلمات: 476 كلمة

الرقم الجامعي: نسبة الاقتباس: 0%

**Architectural patterns**

Patterns assist in building software that has specific qualities. For example, the server shouldn't be constructed in a client-server application so that it begins contact with the clients.Numerous designs specifically address the needs of software systems that are nonfunctional. The MVC (Model-View-Controller) design, for instance, facilitates the flexibility of user interfaces. Thus, patterns can be thought of as the foundation of a more intricate design.DESIGN PATTERNS AND ARCHITECTURAL PATTERNS : What distinguishes design patterns from architectural patterns? Through the cooperation of classes, design patterns provide a universal remedy for a universal issue. As a result, they are more compact than architectural patterns where classes are the components instead of subsystems.Design patterns have no effect on a software system's underlying structure. They only have an impact on one subsystem. An architectural pattern may be implemented with the use of design patterns. For instance, the Observer pattern (a design pattern) is useful for putting an MVC architectural pattern-based system into place. Layers of different abstraction levels : Applications that can be broken down into groups of subtasks, each of which is at a different level of abstraction, can be structured more effectively using the Layers pattern. Each layer offers services to the layer above it. Services from the layer below are used to implement services in that layer. Synchronous procedure calls are widely used to process service requests.This design, in brief, ensures that conceptually distinct problems are handled independently and that layers of higher abstraction employ services of lower abstraction, rather than the reverse. Client-Server pattern: A server component offers services to several client components in the client-server paradigm. A client component asks the server component for services. The servers are always listening for clients. Requests are transmitted across machine and process boundaries. Clients and servers may be located on separate computers, and as a result, they may be in distinct processes, which necessitates the need for some sort of inter-process communication method. The Client-Server pattern really straddles process and machine boundaries, with clients forming the higher level and the server forming the lower one. Master-Slave paterns : Parallel computing and fault tolerance are supported by the Master-Slave paterns. A final result is calculated from the results the slave components return after the master component divides the work among identical slave components.The Master-Slave pattern is used, for example, in fault-tolerant systems, embedded systems, large-scale parallel computations, and process control. Pipe-Filter pattern : The Pipe-Filter design gives systems that generate streams of data structure. Each stage of the processing is contained in a filter component. Piping is used to transfer data. The pipes may be used for synchronization or for buffering. Broker pattern: Distributed systems with separated components that communicate via distant service invocations are structured using the broker pattern. When components need to be aware of one other's position and other information, such systems are exceedingly rigid.

..الاسم: لينا نوار المطيري عدد الكلمات: 477 كلمة

الرقم الجامعي: نسبة الاقتباس: 4%

**Architectural patterns**

Peer-to-Peer pattern : The peer-to-peer design can be viewed as a symmetric client-server pattern since peers can act as both a client and a server, seeking services from and giving to other peers, respectively. A peer can switch its role dynamically between acting as a client, a server, or both. In the Peer-to-Peer paradigm, clients and servers are frequently multithreaded. The services may be requested implicitly rather than explicitly (for example, through a connected stream). Event-Bus pattern : A pattern that deals with events is called Event-Bus. The way it works is that specific channels on an event bus receive messages from event sources. Individual channels are subscribed to by event listeners. When messages are posted to a channel to which they have subscribed, listeners are alerted,Asynchronous message generation and notification means that an event source can create a message without waiting for all event listeners to hear it. It can then go on to other tasks, Android development uses it, as do notification services. Model-View-Controller : An interactive application is separated into three components according to the Model-View-Controller (MVC) pattern: the model holds the essential functionality and data, the view shows the information to the user (more than one view may be created), and the controller manages user input. Multiple graphical user interfaces are especially suited to the MVC paradigm (GUIs). The pattern allows for simple modifications to the "look and feel," as the model is independent of the quantity and kind of GUIs, It is used in the architecture of web applications written in popular programming languages and using frameworks like Django and Rails. Blackboard pattern : For issues for which there are no deterministic solution techniques, the Blackboard pattern is helpful. Multiple specialized subsystems pool their expertise to provide a potentially incomplete or approximative solution. Each component has access to the blackboard, a common data repository. New data items that are added to the blackboard may be produced by components. Components may use pattern matching to identify certain types of data on the blackboard that they are looking for, Application to Sonar signal interpretation, protein structure identification, and vehicle identification and tracking. Interpreter pattern : When creating a component that interprets programs written in a certain language, the Interpreter pattern is employed. Replacement of the interpreted software is simple,utilized when creating a part that translates programs written in a certain language. It primarily lays forth the criteria for evaluating program lines, sometimes referred to as phrases or expressions written in a specific language. The fundamental concept is to establish a class for each language sign. microservices pattern : combines design patterns to provide a number of interconnected services that together make up a bigger application. Each application is modest, making it simpler to update them when necessary, but because of their complexity, everything must function properly, which requires more advanced architectural knowledge,etc.

الاسم: لينا نوار المطيري عدد الكلمات: 477 كلمة

الرقم الجامعي: نسبة الاقتباس: 0%

**Architectural patterns**

Architectural styles : All patterns have been created from the ground up; for a given problem, a certain type of solution is employed repeatedly, and this answer is then recorded as a pattern.Contrarily, architectural styles have all been developed top-down. When you see a software system as a configuration of components and connectors, you may categorize it based on the characteristics of those parts. Patterns often fall under one of these categories.A taxonomy of styles based on the component parts (components and connectors), control concerns, and data issues has been presented by Mary Shaw and Paul Clements. These categories are as follows: - Each process that interacts has an own control thread. Asynchronous message transmission, implicit invocation through events, and remote method calls are all examples of forms of communication. This pattern is an illustration of this fashion when the Event Bus pattern is used with independent processes or with objects that have their own thread of control. Examples include the Client-Server pattern and the Peer-to-Peer pattern, - Data flows from one component to another in a stream when using the Dataflow style. This pattern is an example of this kind. This style also includes some applications of the Client-Server pattern, such as when a client is used to receive and show streaming audio or video delivered by a server, -The data-centered approach calls for central data storage. Blackboard is a pattern that exemplifies this design aesthetic. Again, this style may include instances of the ClientServer pattern. For instance, if the server's primary duty is to administer a database, and users access the database through clients, - The system is divided into subsystems with little interaction in the hierarchical approach. The Layers pattern and the Interpreter pattern are patterns that fit this category,etc . CHOOSING A STYLE OR A PATTERN : Which architectural pattern is ideal in a certain circumstance relies on the needs that are most important, such as: Reusability: Can separate parts be utilized in different systems? The adoption of a common data format in this instance makes the Pipe-and-Filter technique more reusable.Has the response time been fast enough? Is the general use of resources (in this case, memory use) acceptable? Performance will be enhanced for patterns that employ parallelism, including the Pipe-Filter and EventBus patterns. However, beginning a complicated system, such as the Event Bus system, or changing data in every filter using a different data structure may result in decreased performance.- Explicitness: Is it feasible to offer the user feedback? each stage? For example, the Pipe-Filter design prohibits this. - Fault tolerance: There is no difference between the various methods for the KWIC example, although fault tolerance would have been improved if a Master-slave pattern had been used.and there are many other things.

الاسم: لينا نوار المطيري عدد الكلمات: 459 كلمة

الرقم الجامعي: نسبة الاقتباس: 0%