Tutorial 2: Six Questions (Web Engineering)

1. Describe the common Content Architectures of WebApps. Highlight the differences between Content Architecture and WebApp architecture.
2. Discuss the various possible organisational paradigms for SE teams.
3. What are the design goals of WebApps?
4. List the “best practices” that should be applied to build quality WebApps.
5. What are the six key elements in creating a complete design model of WebApps?
6. Describe the interface testing strategy for WebApps.

T2_Q1: Four different content architectures

T2_Q1_ans

Linear structure – Such structures are encountered when a predictable sequence of interactions is common. The sequence of content presentation is predefined and generally linear. E.g. product order entry sequence in which specific information must be specified in a specific order.

Grid structure – This structure is applicable when WebApp content can be organized categorically in two (or more) dimensions. Such WebApp architecture is useful only when highly regular content is encountered. E.g. Consider E-commerce site selling golf clubs - the horizontal dimension of the grid represents the type of club to be sold (woods, irons, wedges, putters), and the vertical dimension represents the offerings provided by various gold club manufacturers.

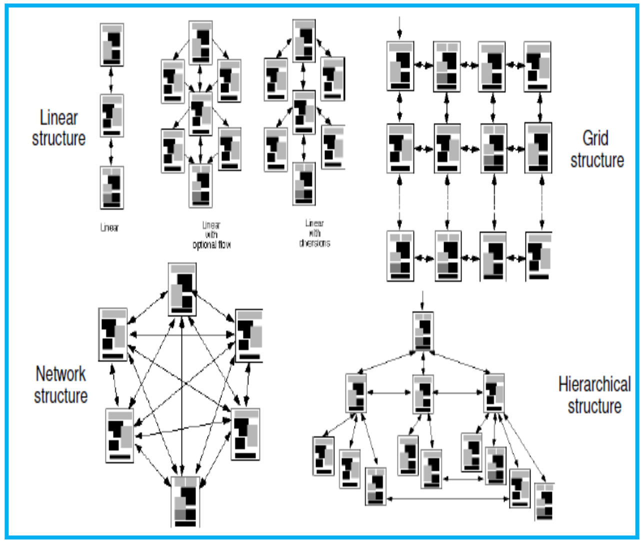
Hierarchical structure – This is the most common WebApp architecture. While the branches of WebApp having hierarchical structure is usually structured top-down, a WebApp hierarchical structure can also be designed in a manner that enables (via hypertext branching) flow of control horizontally or across vertical branches of the structure.

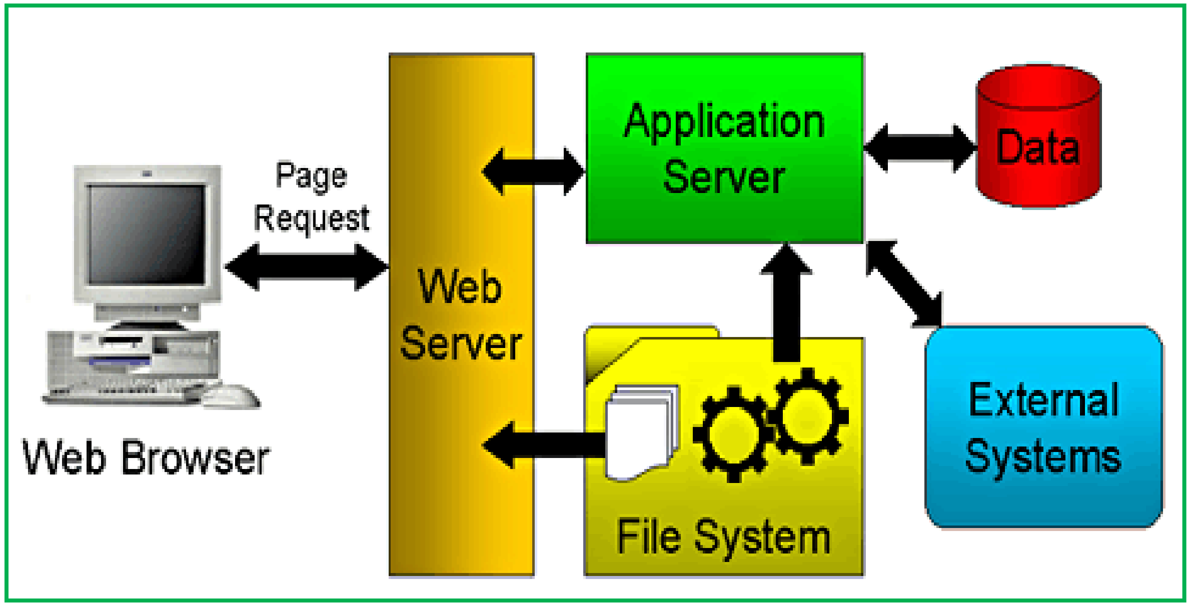
Networked structure – Such type of structures can be confusing to the user as they may pass control (via hypertext links) to virtually every other component in the system, thus allowing considerable navigation flexibility.

T2_Q1ans:

Difference between WebApp architecture and Content architecture:

The architectural designer must identify content architecture and WebApp architecture. Content architecture focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation. WebApp architecture addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.





T2_Q2: Possible Organizational Paradigms for SE teams: **T2_Q2_Ans**

1. A **closed paradigm** structures a team along a traditional hierarchy of authority. Such teams can work well when producing software that is quite similar to past efforts, but they will be less likely to be innovative when working within the closed paradigm.

2. A **random paradigm** structures a team loosely and depends on individual initiative of the team members. When innovation or technological breakthrough is required, teams following the random paradigm will excel. But such teams may struggle when "orderly performance" is required.

3. An **open paradigm** attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm. Work is performed collaboratively. Heavy communication and consensus-based decision making are the trademarks of open paradigm teams. Open paradigm team structures are well suited to the solution of complex problems but may not perform as efficiently as other teams.

4. A **synchronous paradigm** relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves.

T2_Q3: The design goals of WebApp

T2_Q3_ans

Simplicity – Although it may seem old-fashioned, the aphorism "all things in moderation" applies to WebApps. There is a tendency among some designers to provide the end-user with "too much" – exhaustive content, extreme visuals, intrusive animation, enormous Web pages, the list is long. Better to strive for moderation and simplicity.

Consistency – The design goal applies to virtually every element of the design model. Content should be constructed consistently (e.g., text formatting and font styles should be the same across all text documents; graphic art should have a consistent look, color scheme, and style). Graphic design (aesthetics) should present a consistent look across all parts of the WebApp. Architectural design should establish templates that lead to a consistent hypermedia structure. Interface design should define consistent modes of interaction, navigation, and content display. Navigation mechanisms should be used consistently across all WebApp elements.

Identity – The aesthetic, interface, and navigational design of a WebApp must be consistent with the application domain for which it is to be built. A Web site for a hip-hop group will undoubtedly have a different look and feel than a WebApp designed for a financial services company. The WebApp architecture will be entirely different, interfaces will be constructed to accommodate different categories of users, navigation paths will be organized to accomplish different objectives. A Web engineer (and other design contributors) should work to establish an identity for the WebApp through the design.

T2_Q3: The design goals of WebApp

T2_Q3_ans. Cont

Robustness – Based on the identity that has been established, a WebApp often makes an implicit "promise" to a user. The user expects robust content and functions that are relevant to the user's needs. If these elements are missing or insufficient, it is likely that the WebApp will fail.

Navigability – We have already noted that navigation should be simple and consistent. It should also be designed in a manner that is intuitive and predictable. That is, the user should understand how to move about the WebApp without having to search for navigation links or instructions.

Visual appeal – Of all software categories, Web applications are unquestionably the most visual, the most dynamic, and the most unapologetically aesthetic. Beauty (visual appeal) is undoubtedly in the eye of the beholder, but many design characteristics (e.g., the look and feel of content, interface layout, color coordination, the balance of text, graphics and other media, navigation mechanisms) do contribute to visual appeal.

Compatibility – A WebApp will be used in a variety of environments (e.g., different hardware, Internet connection types, operating systems, browsers) and must be designed to be compatible with each.

T2_Q4_ans: some “best practices”

1. Take time to understand the business and product needs of the WebApp.
2. Describe how users will interact with the WebApp using a scenario-based approach.
3. Develop a brief project plan
4. Spend some time modeling what you plan to build
5. Review models for consistency and quality
6. Use tools and technologies that system construction with reusable components
7. Do not rely on users to debug the WebApp. Design tests and execute them before releasing the system

T2_Q5_ans: The 6 key elements

Interface design – Describes the structure and organization of the user interface. This includes a representation of screen layout, a definition of the modes of interaction, and a description of navigation mechanisms.

Aesthetic design – Also call graphic design, this describes the "look and feel" of the WebApp. Includes color schemes, geometric layout, text size, font and placement, the user of graphics, and related aesthetic decisions.

Content design – Defines the layout, structure, and outline for all content that is presented as part of the WebApp. Content design establishes the relationships between content objects.

design rept; relationships; 2types of attributes

Navigation design – Represents the navigational flow between content objects and for all WebApp functions.

flow control &mgt

Architecture design – This identifies the overall hypermedia structure for the WebApp.

Component design – Develops the detailed processing logic required to implement functional components.

make use of local processor

T2_Q6: WebApp Interface Testing Strategy

T2_Q6_ans

- Interface features are tested to ensure that design rules, aesthetics, and related visual content are available for the user without error. Features include type fonts, the use of color, frames, images, borders, tables, and related elements that are generated as WebApp execution proceeds.
- Individual interface mechanisms are tested in a manner that is analogous to unit testing. For example, tests are designed to exercise all forms, client-side scripting, dynamic HTML, CGI scripts, streaming content, and application specific interface mechanisms (e.g., a shopping cart for an e-commerce application). In many cases, testing can focus exclusively on one of these mechanisms (the "unit") to the exclusion of other interface features and functions.
- Each interface mechanism is tested within the context of a use-case or NSU for a specific user category. This testing approach is analogous to integration testing in that tests are conducted as interface mechanisms are integrated to allow a use-case or NSU to be executed.
- The complete interface is tested against selected use-cases and NSUs to uncover errors in the semantics of the interface. This testing approach is analogous to validation testing because the purpose is to demonstrate conformance to specific use-case or NSU semantics. It is at this stage that a series of usability tests are conducted.
- The interface is tested within a variety of environments (e.g., browsers) to ensure that it will be compatible. In actuality, this series of tests can also be considered to be part of configuration testing.