

SWE 363: Web Engineering & Development

Module 6-1

Introduction to **Web Engineering**



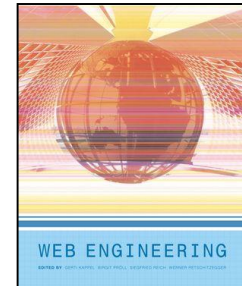
Objectives

- ❑ Understand the role of web engineering
- ❑ Learn a systematic process for web applications development

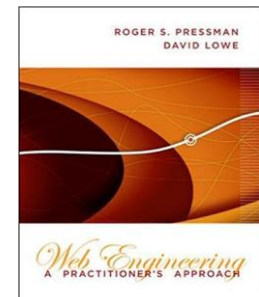
Used References

❑ Books

- “Web Engineering: The Discipline of Systematic Development of Web Applications” by Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006), Wiley & Sons.



- “Web Engineering: A Practitioner's Approach” by Roger S. Pressman and David Lowe, 2008, McGraw-Hill Education



❑ Others:

- “The expressive power of UML-based web engineering.”
Koch, Nora, and Andreas Kraus. *Second International Workshop on Web-oriented Software Technology (IWWOST02)*. Vol. 16. CYTED, 2002.

- ❑ What is Web Engineering?
- ❑ Web applications
 - Characteristics
 - Categories
- ❑ Web Engineering Process
- ❑ Requirement Analysis

What is Web Engineering?

- ❑ “The application of **systematic and quantifiable** approaches to cost-effective analysis, design, implementation, testing, operation, and maintenance of **high-quality web applications**” – Kappel et al.
- ❑ **Web engineering** extends **Software Engineering** to **Web applications**, but with **Web-centric** approaches.
- ❑ >>The **gap** between native apps and web apps is narrowing.
 - Most projects are now **Web-based** and More “**mission-critical**” applications **moving to the Web**
 - Application development on the Web shouldn’t be **ad-hoc** approach
 - Unplanned, one-time events, Individual experience and little or no documentation for code/design

Web applications (WebApp)

- ❑ The Web originally was designed as a purely **informational medium**, it is now increasingly evolving into an **application medium**
 - Web applications today are full-fledged, **complex software systems**
 - Interactive, data intensive, and customizable services accessible through different devices
- ❑ **WebApp** - is an application that was **designed** from the beginning to be **executed in a Web-based environment**. Two very important aspects of such an application:
 - >> the hypermedia aspect in terms of hypertext and multimedia in combination with traditional application logic must be taken into account throughout the application lifecycle, which makes it **different with respect to a conventional application**.
 - >> It is not just a set of Web pages. In particular, it enforces the notion of a **session**, which differentiates it from the ordinary **request-response Web paradigm**
- ❑ **Web App** is a **client-server software** application in which the client (or user interface) runs in a web browser.
 - Common web applications include webmail, online retail sales, online auctions, e-commerce websites, etc.

WebApps vs. Desktop Apps

□ Some of the advantages of web applications include:

- **Accessible** from any Internet-enabled computer.
- Usable with **different operating systems** and browser applications.
- Easier to **roll out program updates** since only software on the server needs to be updated and not on every desktop in the organization.
- **Centralized storage** on the server means **fewer security concerns** about local storage (which is important for sensitive information such as health care data).

□ Some of these disadvantages include:

- Requirement to have an **active Internet connection** (the Internet is not always available everywhere at all times).
- **Security concerns** about sensitive private data being transmitted over the Internet.
- Concerns over the **storage, licensing**, and use of uploaded data.
- Problems with certain websites on certain **browsers** not looking quite right.
- Restrictions on access to the operating system can prevent software and hardware from being installed or accessed (like Adobe Flash on iOS)

Characteristics of WebApps

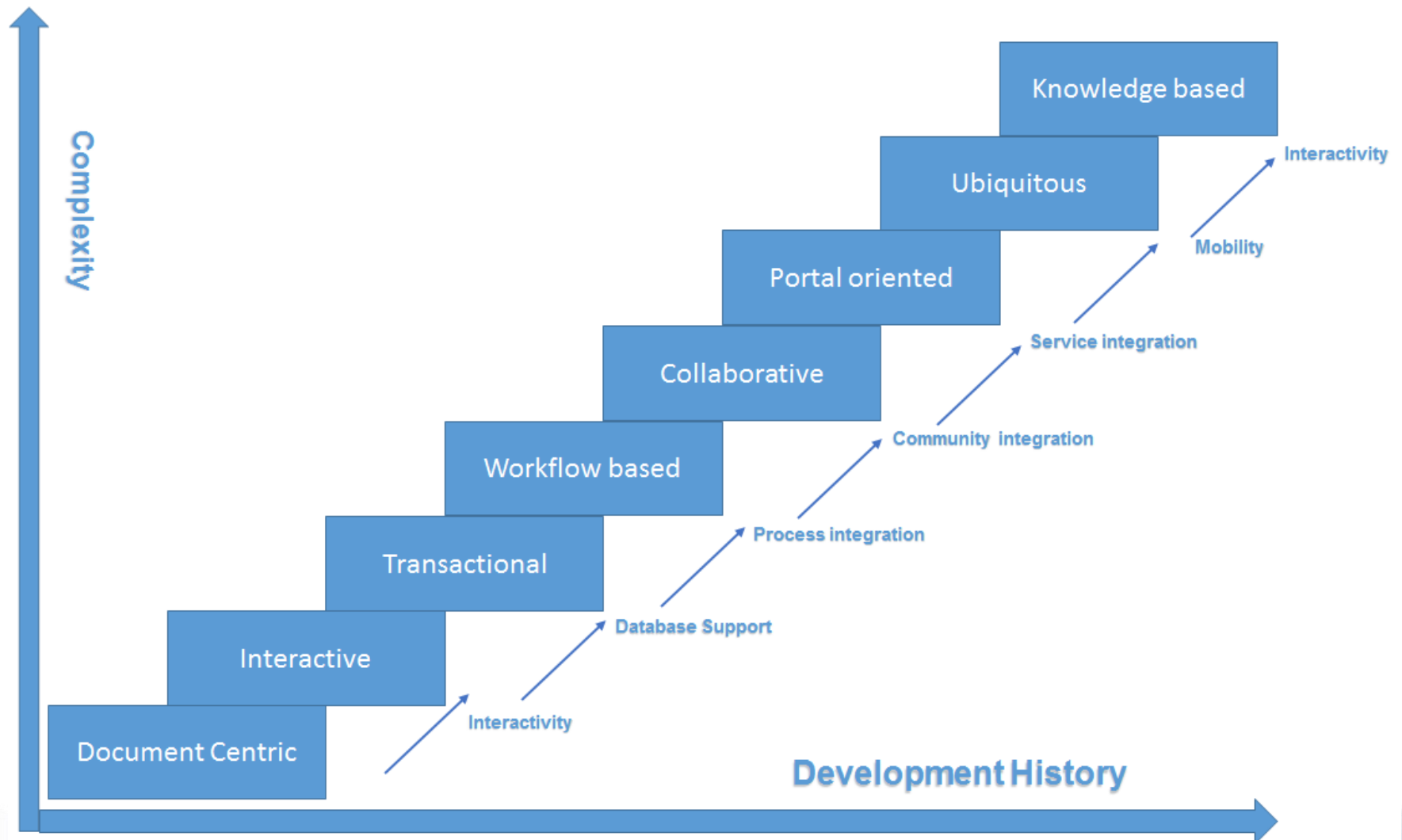
How do Web applications differ from traditional applications?

- ❑ **Network intensiveness** - Resides on a network and must serve the needs of a diverse community of clients
- ❑ **Availability** - Users of popular WebApps often demand access on a “24/7/365” basis.
- ❑ **Performance** - If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), >> alternatives !
- ❑ **Data driven** - The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user
- ❑ **Continuous evolution** - unlike conventional application software that evolves over a series of planned chronologically-spaced releases, **Web applications evolve continuously**.

Characteristics of WebApps..

- ❑ **Self-explanation** -User can easily access, register, navigate, download, etc..
- ❑ **Concurrency** - Large number of users may access the WebApp at the same time
- ❑ **Diversity of users and Unpredictable load** –
 - Number of users may vary by orders of magnitude from day to day
 - Patterns of usage among end-users will vary greatly
- ❑ **Diversity of Development Team** -marketing & computing, art & technology etc.
- ❑ **Integration:**
 - Internal – with existing legacy systems
 - External – with Web services

WebApp Categories



WebApp Categories

Document-centric web sites

- ❑ Example: static home pages, company website
- ❑ Originator of Web applications
- ❑ **Web pages** are stored on a **Web server** as **ready-made**, i.e. static, HTML documents and sent to the Web client in response to a request.
- ❑ **Manual updates**- Especially for Web sites requiring frequent changes or for sites with huge numbers of pages this is a significant cost factor and often results in outdated information
- ❑ **Pros**: Simple, stable, short response times.
- ❑ **Cons**:
 - High management costs for frequent updates & large collections
 - More prone to inconsistent/redundant info
 - Outdated information

WebApp Categories

Interactive & transactional

- ❑ Examples: news sites, travel planning, booking systems, online banking etc.
- ❑ Not only read-only content but also **allow content modification**
- ❑ Come with the introduction of **HTML forms**
- ❑ *Simple interactivity*- the user can perform updates on the underlying content
- ❑ **Dynamic page creation** - Web pages and links to other pages generated dynamically based on user input
- ❑ Content updates -> Transactions
 - Decentralized
 - Database connectivity
 - Increased complexity

WebApp Categories

Workflow-based applications

- ❑ Examples: Business-to-Business (B2B) solutions in e-commerce, e-government applications, a **purchase order** that moves through various departments for authorization, etc.
- ❑ The applications automate, to at least some degree, a process or processes.
 - The processes are usually business-related but can be any process that requires **a series of steps** to be automated via software.
- ❑ Expected Challenges:
 - The **complexity** of the services,
 - the **autonomy** of the participating companies and
 - the necessity for the workflows to be **robust** and **flexible**.
- ❑ The role of Web services:
 - Interoperability, Loosely-coupled and Standards-based
- ❑ High complexity; autonomous entities

WebApp Categories

Collaborative & Social Web

- ❑ Collaborative Web applications **support shared information and workspaces** in order to generate, edit, and manage shared information
 - Classic example: **Wikis**
- ❑ **Social networking** means socializing for personal, or professional purposes
 - for example, **LinkedIn** and **Facebook**.
- ❑ **Social collaboration** means working socially to achieve a common goal
 - for example, **GitHub** and **Quora**.
- ❑ Unstructured, cooperative environments
 - Support shared information workspaces to create, edit and manage shared information
- ❑ An increasing trend towards the *Social Web*
 - Moving towards communities of interest
 - Integration with other forms of web applications

WebApp Categories

Portal-Oriented

- ❑ Portal-oriented Web applications provide a single point of access to separate, potentially heterogeneous sources of information and services.
 - Example, ERP
- ❑ Designed to give partners focused access to different sources of information and services in a uniform way
- ❑ Each information source gets its dedicated area
- ❑ Specialized portals
 - Business portals (e.g., employee intranet)
 - Marketplace portals (horizontal & vertical)
 - Community portals (targeted groups)

WebApp Categories

Ubiquitous WebApps

- ❑ **Examples:** Geolocation, weather forecast
- ❑ Ubiquitous computing is a concept in SWE and CS where computing is made to appear **anytime** and **everywhere**.
 - WebApps are becoming **ubiquitous systems** that are available **anytime**, **anywhere**, and **with any media**.
- ❑ UWA have to take into account, **individually** for each user, **time** and **location of access**, together with the different capabilities of devices comprising display resolution, local storage size, method of input and computing speed as well as network capacity.
- ❑ A new class of applications that:
 - make (a large amount of) multimedia information accessible to the users
 - provide “operational” services
 - provide a variety of interaction paradigms (e.g. navigation, query, search, operation invocation, etc.)
 - are multi-channel, in the sense that they are available on a variety of different devices
 - be accessible anywhere at anytime.
 - support different categories of users (each one with different characteristics and needs)

Knowledge-based Semantic Web

- ❑ Most of the web's content today is designed for **humans to read** , and not for computer programs to process meaningfully
- ❑ **Typical uses** of the Web today are information **seeking, publishing, searching**, etc.
- ❑ Dynamic pages generated **based on information from databases** but without original information structure found in databases.
 - e.g. With HTML and a tool to render it, one can create and present a page that lists items for sale, but can't manipulate data.
- ❑ Limitations of the Web Search today:
 - The Web search results are **high recall, low precision**.
 - Results are **highly sensitive** to vocabulary.
 - Most of the publishing contents are not structured to allow logical reasoning and query answering.

Semantic Web

“The next generation of the Web”

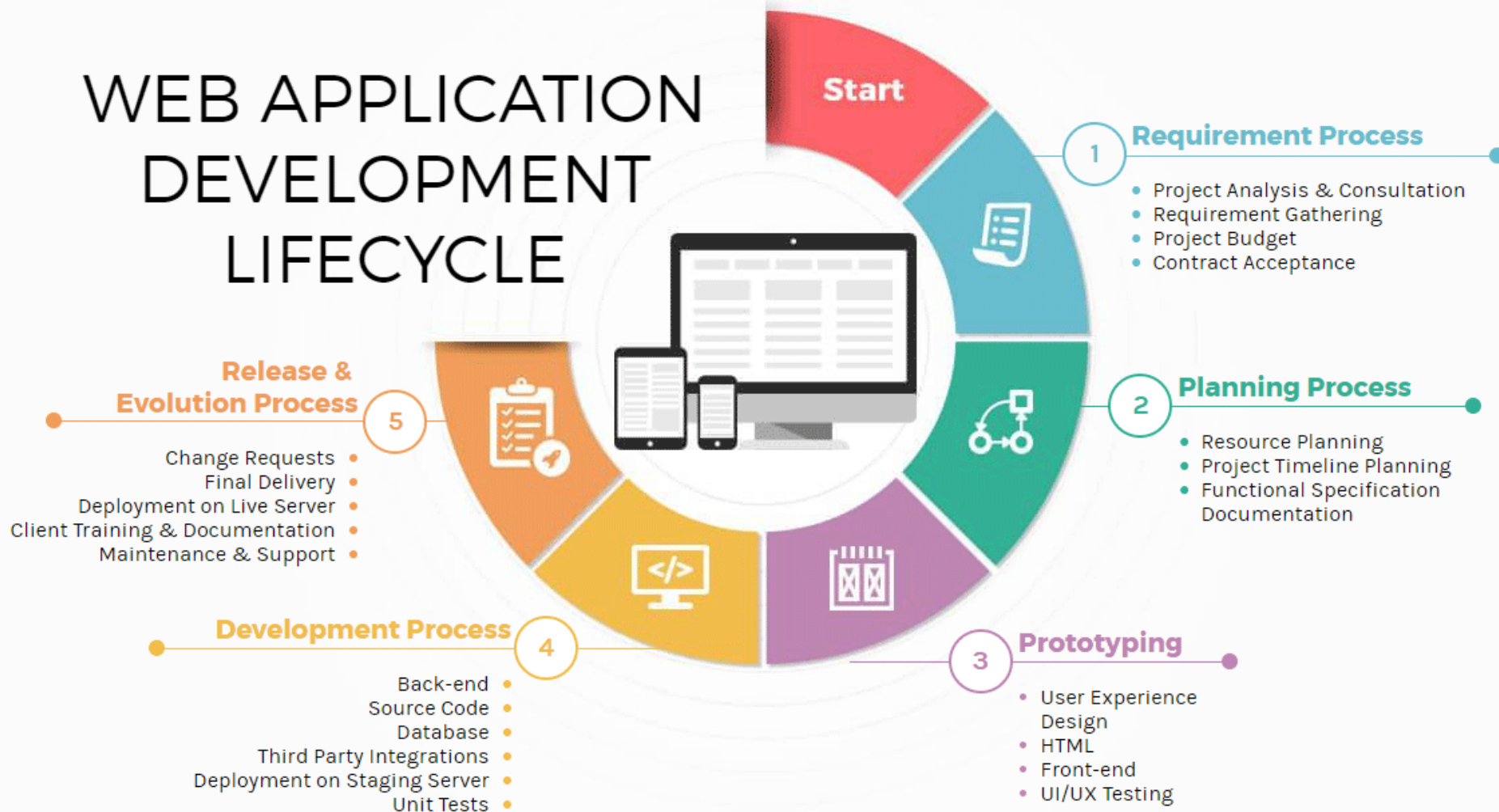
- ❑ The **Semantic Web** is a web that is able to describe things in a way that computers can understand.
 - It is much more **expressive**, **comprehensive** and **powerful** form of data modeling.
- ❑ **Semantic Web** is an extension of the Web through standards by W3C. The standards *promote common data formats and exchange protocols* on the Web, most fundamentally the **Resource Description Framework** (RDF).
- ❑ **Goal:** Information on the Web should be readable to machines, as well as humans.
- ❑ The Semantic Web describes the **relationships between things** (like A is a part of B and Y is a member of Z) and the **properties of things** (like size, weight, age, and price)

Simple Semantic Web Example:

http://w3schools.sinsixx.com/semweb/semantic_example.asp.htm

WebApp Development Process

WEB APPLICATION DEVELOPMENT LIFECYCLE



WebApp Development Process

❑ DISCOVERY – Meet, talk, define, explore

- Exchange of knowledge. Ask questions, and most of all, listen. Define expectations, goals, and capabilities for your website and gather all the information we need to define milestones, create a schedule, and supply an estimate.

❑ STRATEGY & PLANNING – Measure twice, cut once

- Focus on **features** and **functionality** of the site and offer search engine optimization (SEO) recommendations. Create a **sitemap** and **wireframes** that will help us plan every type of page on the site and its contents for an optimal user experience.

❑ DESIGN – Designs that engage

- Take all of our upfront planning and bring it to life with simple, intuitive design. Apply brand characteristics, colors, fonts, and imagery to the approved wireframes to enhance and streamline your site, while visually engaging your users.

Website Development Process

❑ DEVELOPMENT AND TESTING –

- Begin building and programming the website (open source content management system (CMS) software is usually used), **but not in our course**. Perform **browser and functionality testing** for a consistent experience across all major browsers including Chrome, Firefox, Safari, Opera, and Internet Explorer. In addition, **test the website on multiple devices and screen sizes** to make sure the mobile-friendly design delivers the optimum experience across mobile devices, tablets, and desktop computers. After browser and functionality testing is complete, you'll have the opportunity to review the site.

❑ LAUNCH – Welcome to your new site

- Begin the file transfer necessary to push the site live. At each point in this process, we'll do comprehensive final testing to **make sure all links and functionality respond as expected**.

❑ MAINTENANCE & SUPPORT – A smooth transition to the future

Requirement Engineering

Introduction

A **requirement** describes a **property** to be met or a **service** to be provided by a system.

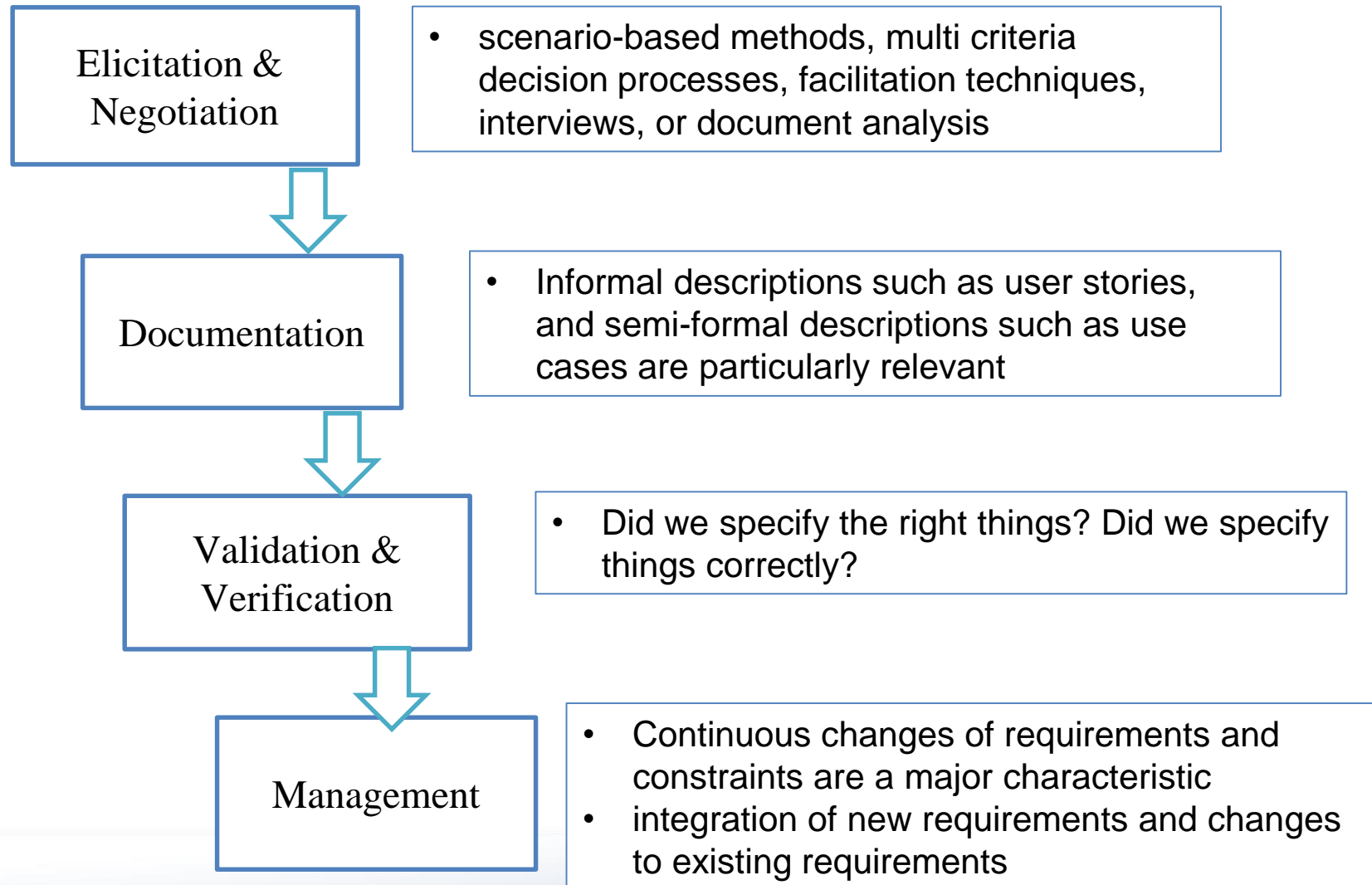
- ❑ IEEE 610.12 defines a **requirement** as
 1. Condition needed to solve a user's problem
 2. Condition to be met or possessed by the system to satisfy a formal agreement
 3. Documented representation of conditions as in 1 and 2
- ❑ **Requirements Engineering (RE)** – the principles, methods and tools for eliciting, describing, validating and managing **project goals and needs**.
- ❑ Given the complexity of Web apps, **RE is a critical initial stage**, but often poorly executed.
- ❑ What are the **consequences**?
 - Unclear objectives, unrealistic schedules & expectations, poor user participation
 - “Unforeseen” problems: Budget overruns, Production delays, “That’s not what I asked for”
 - Low user acceptance

Removal of mistakes post hoc is up to 200 times more costly (Boehm 1981)

Types of Requirements

- ❑ **Functional requirements** describe a system's capabilities and services
 - e.g., **the ability to transfer money between user accounts**
- ❑ **Non-functional requirements** describe the properties of capabilities and the desired level of services
 - e.g., **"The Web application shall support at least 2500 concurrent users."**
 - Other non-functional requirements refer to project constraints and system interfaces
 - **Quality**
 - Functionality, Usability, Portability, Scalability
 - Reliability, Efficiency, Security, Maintainability
 - **User Interface**
 - Self-explanatory

The Requirements Collection Process



Elicitation

- ❑ The intent is to gather detailed requirement collaboratively with all stakeholders
 - Define **user categories**, and develop descriptions for each category.
 - Define **content and functionality** using the lists each person prepared.
 - Consider **specific constraints** and performance issues.
 - Write **user scenarios** for each user class.

- ❑ To do this:
 - A **meeting** (either physical or virtual) is conducted and attended by all stakeholders.
 - **Rules** for preparation and participation are established.
 - An **agenda** is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
 - A **facilitator** (can be a customer, a Web engineer, or an outsider) controls the meeting.
 - A **definition mechanism** (can be worksheets, flip charts, or wall stickers or an electronic bulletin board, chat room, or virtual forum) is used.

Negotiation

- ❑ Ideally, requirements are defined in sufficient detail to proceed
 - BUT, in reality, **requirements are often contradictory or infeasible** (within the context of real-world constraints, such as cost or time).
- ❑ **Negotiation** involves working with the stakeholders to balance functionality, performance, and other product or system characteristics against cost and delivery time.
- ❑ Recognize that it's not a competition.
 - To be successful, both parties have to feel they've won or achieved something. Both will have to compromise.
- ❑ Decide what you'd like to achieve, what the other party wants to achieve, and how you'll go about making both happen.
 - Example: WinWin Approach

Techniques for Elicitation & Negotiation

- ☐ Interviewing
- ☐ Joint Application Design
- ☐ Brainstorming
- ☐ Concept Mapping
- ☐ Storyboard
- ☐ Use Case Modeling
- ☐ Questionnaires

Requirements Documentation

- ❑ Use **informal**, **semi-formal** and **formal** methods in the description
- ❑ 4 Categories of Notations
 - **Stories** – Plain-language scenarios; understandable to non-technical persons.
 - **Itemized Requirements** – Plain-language lists of requirements
 - **Formatted Requirements** – Accurately-defined, but allow for plain-language descriptions, e.g. **use cases**
 - **Formal Specification** – Expressed in formal syntax & semantics; **rarely used in Web applications**.
- ❑ So, what's best for a Web development project?
 - **Formatted requirements** (i.e. use cases) and **stories** are heavily used.
 - Scalability is (most likely) important.

- ❑ This step is essential to **verify that requirements specification** corresponds to user's needs and customer's requirements

- ❑ **Iterative feedback** from stakeholders is essential
 - Is the requirement feasible?
 - Do the results meet stakeholders' expectations?
 - e.g., Internet users can be **invited to participate in Web surveys** to communicate their satisfaction with a Web application

- ❑ Validation Techniques
 - **Review or walk-through**- Reading and correcting the requirements definition documentation and models
 - **Traceability Matrix**
 - **Comparison** of the application objectives with the requirements of the system
 - **Prototyping for Validation**- Implement a partial set of functional requirements but provide a global vision of the user interface

Management

- ❑ **Requirement management** is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders.
- ❑ It is a continuous process throughout a project.
- ❑ It is an essential to enable, Traceability, Modifiability, and Verifiability

- ❑ **Requirements traceability** is concerned with documenting the life of a requirement.
 - It should be possible to **trace back to the origin of each requirement** and every change made to the requirement should therefore be documented in order to achieve traceability.

- ❑ Several tools are available to support Requirements management such as Accompa, GatherSpace, and Visure Requirements.
- ❑ **Tool support is crucial for big project**

Challenges

❑ Challenges with Stakeholders (McConnell 996)

- Users don't know what they want.
- Lack of commitment.
- Ever-expanding requirements.
- Communication delays.
- Users don't take part in reviews.
- Users don't understand the technology.
- Users don't understand the process.

❑ Challenges with Developers

- Users and engineers/developers speak different “languages”.
- Engineers & developers are also asked to do RE, but sometimes lack negotiating skills and domain knowledge.

Good Requirements Specifications

- ❑ **Correct**- Correspond to actual need
- ❑ **Unambiguous**- Can be interpreted only in one way
- ❑ **Complete**- Any external imposed requirement should be included
- ❑ **Consistent**- Conflicting requirements should be avoided

- ❑ **Ranked** for importance and/or stability
 - Requirements are not equally important
 - Requirements are not equally stable
- ❑ **Verifiable**-It's possible to use a cost-effective process to check it

- ❑ **Modifiable**- Can be restructured quickly
 - Adopt cross reference
 - Requirements are clearly separated
- ❑ **Traceable**- Can be tracked from originating design documentation

RE Specifics in Web Engineering 1

Is RE for the Web really that different than RE for conventional software?

>> Many aspects of WebApps say that:

❑ Multidisciplinary

- multimedia experts, content authors, software architects, usability experts, database specialists, or domain experts

❑ Unavailability of stakeholders

- stakeholders (potential Web users) still unknown during RE activities
- project management needs to find suitable representatives that can provide **realistic requirements**

❑ Rapidly changing requirements & constraints

- properties of deployment platforms or communication more difficult in RE for Web Application
- new development platforms and standards, or new devices for end users

RE Specifics in Web Engineering 2

❑ Unpredictable operational environment

- e.g., changing bandwidths affect the response time of mobile applications

❑ Integration of legacy systems

- integration of existing software components
- Web developers have to be aware of the system architecture and architectural constraints

❑ Significance of Quality Aspects

- performance
- security
- availability, or usability

❑ Quality of the User Interface

- IKIWISI (I Know It When I See It) phenomenon
- adding prototypes of important application scenarios

RE Specifics in Web Engineering 3

❑ Quality of Content

- developers have to consider the content, particularly its creation and maintenance
- separating content from layout

❑ Developer Inexperience

- technologies development tools, standards, languages rapidly developed
- wrong estimates when assessing the feasibility and cost of implementing requirements

❑ Firm Delivery Dates

- all activities and decisions have to meet a fixed final project deadline
- negotiation and prioritization of requirements are particularly crucial

Principles for RE of WebApp 1

❑ Understand the system context

- Web apps are always a **component** of a larger entity
- Why do we need the system? **Purpose** and motivation !
- How will people use it? **Users** !

❑ Identify and involve the stakeholders

- Who **directly influence** the requirements
- Get all groups involved.
- What are their **expectations**?
 - May be misaligned or in conflict.
 - May be too narrowly focused or unrealistic.
- **Balance** – one group's gain should not come at the expense of another.
- **Repeat** the process of identifying, understanding and negotiating.

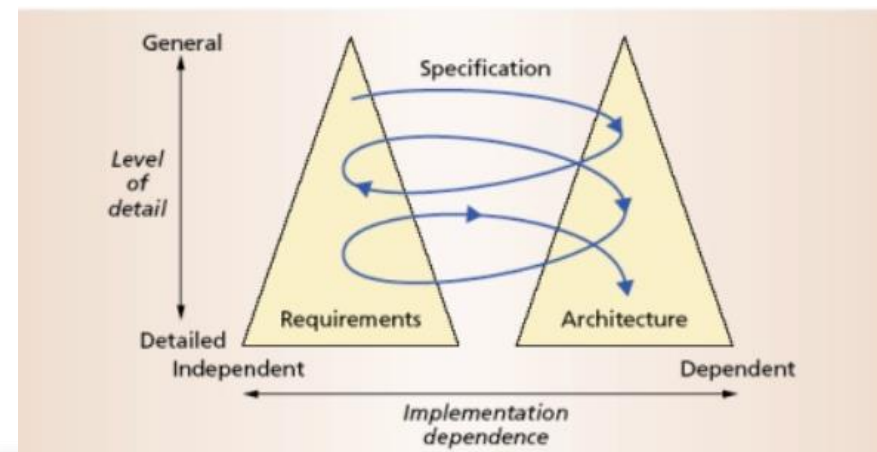
Principles for RE of WebApp 2

□ Iteratively define requirements

- Start with *key requirements at a high level*; these will serve as the basis for:
 - Feasible architectures
 - Key system use cases
 - Initial plans for the project
- As the project progresses, *requirements can become more concrete*.

□ Focusing on the System Architecture

- **Twin-Peaks** model suggests to **refine** both requirements and system architecture **iteratively** with increasing level of detail.



Principles for RE of WebApp 3

❑ Risk Orientation

- Risk management is at the **heart of the analysis process**.
- **Coming form:** Undetected problems, unsolved issues, and conflicts among requirements
- **Examples of risks and possible mitigation:**
 - **Risk:** IKIWISI problem>>
Mitigation: Show changes to customer iteratively to collect feedback, develop Prototyping or wireframes to clarify thing to the customer.
 - **Risk:** Integration issues of existing components/systems>>
Mitigation : early incorporation of external components to avoid late and severe integration to avoid

