Software Engineering for Business Applications Lecture Notes

Efe Kamasoglu

February 7, 2023

1 IT Support for Business Applications

1.1 Classification of Business Applications

- **Definition "Business Application":**

  - <u>in narrower sense:</u> totality of all programs, i.e. **application software**, and associated **data** for a concrete business use case

  - <u>in broader sense:</u> additionally **hardware**, **system software** and necessary **communication** facilities required for the use of application software

- **Two roles of Business Applications:**

  - **supporting**, **improving** or **automating** existing operational processes in bookeeping, accounting, etc. (size, speed, correctness...)

  - **enabling** new products and services (e.g. online shopping and banking)

- **Classification of Business Applications by Business Purpose:**

Examples of

  - **administrative systems:** financial accounting, payroll accounting, administration of stocks

  - **disposition systems:** calculation and cost accounting, material procurement, field service control

  - **management information systems (MIS):** use of internal company data, use of external data, combination of multiple data sources in a flexible form

  - **planning systems:** planning of individual functional areas, integrated planning of several functional areas, corporate planning

- **Cross-Cutting Applications:**

  - independent of compant hierarchy and fuctional domains

  - used either directly via user interface or programmatically via administration and disposition systems

  - *Examples:* office suites, groupware, workflow management systems

- **Enterprise Resource Planning (ERP): ERP system** is an integrated business application (suite, collection of programs), which supports all essential functions of administration, disposition and management with a common interface and a shared and integrated data management.

  – consists of platform and function-oriented application components that exchange info and events
  – is realized as (customizable) standard software
  – *Examples:* external accounting, controlling, procurement
  – Today's ERP systems support an **extended value chain**[1].

## 1.2   Standard and Custom Software

- **Standard Software vs. Custom Software:**

  – **Standard software***(e.g. SAP)*
    * developed for specific **market**
    * distributed by a software house
    * can be used by **several companies**
    * implements "standard business processes" at its core
    * maintained by **manufacturer**, adapted to changes
    * must or can be **customized** to company (e.g. authorizations and roles, currencies)

  – **Custom software**
    * specifically developed for **one company**
    * tailored to specific business processes/requirements
    * result of a project for a known client
    * **individually** maintained and adapted to changes

---

[1]**Value chain** is a business model that describes the full range of activities needed to create a product or service.

- **Adaptation Techniques for Standard Business Software:**

  - Adaptation of operational standard software can be divided into **Configuration**, **Extension** and **Coupling** (= **Customizing**).
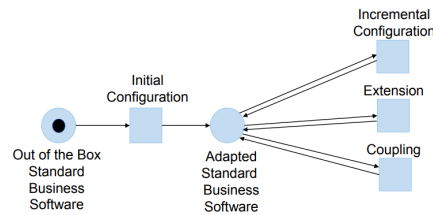
  - **Configuration** describes functionalities and techniques
    * that are obligatory on first deployment
    * that allow to define predefined settings
    * that lead to an individual variation of standard software
  - **Extension** describes functionalities and techniques
    * that are optional for productive use
    * that allow to map requirements not foreseen by manufacturer
    * implemented by manufacturer to expand the range of services
  - **Coupling** refers to functionalities and techniques
    * to connect external systems of other manufacturers
    * to connect external systems of the same type
    * that are predefined in the form of data file formats, APIs, or communication protocols
  - *Example:* mapping the structure of a company to SAP applications via organizational units (can be assigned to single or multiple apps)

- **Configuration: Challenges**

  - A **standard software** must
    * provide all relevant configuration options
    * support a wide range of different corporate structures and processes
    * check dependencies between these many variants
    * provide appropriate documentation about the effects of individual configurations
  - **Consequences:**
    * need for experts who are familier with configuration options of each release and componant
    * scarcity of such experts

4

* expensive training
* expensive consultancy services

- **Examples for Extensions:**

  - automation of **multi-step business workflows**
  - integration of company-specific calculations/rules/checks
  - connecting customers

- **Coupling Options:**

  - different coupling options depending on the scenario
  - programming language used for coupling
  - available mechanisms to couple

- **Multi Tenancy:** Software multitenancy is a software architecture in which a single instance of software runs on a server and serves multiple tenants (e.g. companies).

  - sevearal companies can be represented in one system
  - distinction between tenant-dependent and -independent data
  - supporting tenant-dependent authorization (e.g. A may only perform transactions in client 002)
  - individual adaptations of tenants (e.g. currency, couplings)

- **Multilingualism:**

  - **Multilingualism of a business information system** makes it possible to
    * store and display texts in different languages in the system
    * assing graphics and symbols specific to different languages
  - Multilingualism requires
    * that one system can process all relevant character sets at once
    * storage and recognition of words, numbers etc.
    * that a system can assign users to languages or user can choose their own
    * that texts (graphics, symbols) can be assigned to a language

- **Localization (l10n):** Adaptation of a software product to meet the language, culture, and other requirements of each locale (e.g. adaptation of graphics, currencies, date and time)

- **Internationalization (i18n):** Process of preparing a software-based product for localization (to support global markets)

1.3 Characteristics of Business Applications

- **Multiple Stakeholders and changing requirements:**

  - **Requirements Elicitation and Requirements Management**
    * many stakeholders, different views and concerns
    * Waterfall: upfront requirements document and/or technical specification => Req. Documentation
    * Issue: changing requirements once IT support is implemented
    * Agile: incremental and iterative => Agile Req. Engineering
    * typically, very large number of requirements
    * need for formalization and early consistency checking => Conceptual Modeling
    * need for cost and time prediction => Software Estimation

  - **Programming Challenges**
    * design, implement and test changes in an existing complex system => Change Mgmt.
    * deliver incremental changes without invalidating existing data => Release Mgmt.
    * parallel development at manufacturer and at customer site => Version Mgmt.
    * automated and quality-controlled assembly of application software => Build Mgmt.

- **Persistent Data and Concurrent Data Modification:**

  - **Data consistency** is a must:
    * many users perform **transactions** simultaneously on central databases
    * data must not be lost even in case of system failures.

  - **Programming challenges:**
    * database is managed by an independent application, on a different server / hardware
    * object orientation is not supported by common data bases
    * database concepts must be transferred to the application logic (transactions, rights, primary keys)

- **Distributed Actors and Data Repositories:**

  - **Many users access central data concurrently:**
    * users need data in different locations at different times
    * Client-Server architecture => Layered Architectures
    * web clients => REST protocol

  - **Programming challenges:**

* software components must be able to found in network => Naming services
* communication always via a network => Serialization[2] & failed execution
* authentication and authorization => Security
* concurrent accesses => Transactions

- **Integeration of Data and Application from (Semi-)Autonomous Sources:**

  - **Separation of applications and data repositories:**
    * multiple apps work on independent or shared data resources
    * multiple apps communicate with each other => RPC, Message Passing
    * business processes involve multiple apps => Workflow Mgmt. Systems
    * application landscapes with lots of interacting applications => Enterprise Architecture Mgmt.

  - **Programming challenges:**
    * integration of multiple languages and databases
    * loose coupling through interfaces to avoid code change propagationi
    * error recovery to avoid runtime failure propagation

- **Scalability:**

  - **Growing number of users and data volume**
    * business apps are used by thousands of employees world-wide around the clock
    * customers and business partners interact directly with business apps and expect real-time sub-second response times
    * volatile load (e.g. online shop in christmas season vs. summer season)

  - **Programming challenges:**
    * delayed execution of resource-intesive operations => Batch processing[3]
    * dynamically increasing/decreasing number of users => Instance pools
    * single server cannot handle the load => Load balancing, Caching

---

[2] **Serialization** is the process of translating a data structure into a format that can be stored or transmitted and reconstructed later.

[3] **Batch processing** is when a computer processes a number of tasks that it has collected in a group. It is designed to be a completely automated process, without human intervention.

2 Requirements Engineering

- **Software requirements** express the needs and constraints placed on a software product.

- **Requirements engineering** is concerned with **elicitation**, **analysis**, **specification** and **validation** of software requirements as well as the management of requirements.

- **Requirements Management** deals with the administration and maintenance of requirements documents, in particular:
    - change requirements (change management)
    - trace and link requirements (requirements tracing)
    - verify requirements

2.1 Traditional Requirements Engineering

- **Objectives of Requirements Management:**

    - **Efficient** preparation of **high quality** requirements and system specifications,
        * coordinated with all stakeholders (different objectives and interests)
        * coordinated with all specifications and constraints
        * evaluated according to profitability and feasibility
    - **Specification documents** are basis for:
        * contract negotiation and contractual agreements
        * coordination between the stakeholders (customers, developers)
        * design, realization, integration
        * software acceptance (test specification)
        * future developments, projects

- **Requirement Classification:** Distinction between <u>functional and non-functional requirements and constraints</u>:

    - **Functional requirements** describe <u>interactions</u> between the system and its environment independent of their realization.
    - **Non-functional requirements** describe <u>general properties</u> of the system.
    - **Restrictions (Constraints)** determine the <u>solution space</u> for the realization.

- **Stakeholder Management:** It includes

- processes required to identify people that could impact or be impacted by the project
- to analyze stakeholder expectations and their impact on the project
- to develop appropriate management strategies for effectively engaging stakeholders in project decisions and execution

- **Requirement Specification:**

  - technical result document of requirement identification phase
  - **contains** stakeholder identification, functional and non-functional requirements, constraints, evaluation plan and metrics
  - list of all deliverables and services to be fulfilled by contractor within contract as defined by customer
  - **what** is to expect from the solution (product)
  - formulation of requirements should be as general as possible and as restrictive as necessary
  - enables the contractor to develop optimal solutions

- **Requirements Validation: Validation**, **Consistency check** (no conflicts), **Completeness check**, **Reality check**, **Verifiability**

- **Functional Specification:**

  - defines the purpose of the system
  - solution proposal created by contractor based on the requirement specification provided by client
  - **contains** target determination, product usage, environment (e.g. hardware), functions, UI, global test cases
  - system description or solution specification, which describes **how** the solutions is to be realized (concrete solution approaches)
  - the **what** from **requirement specification** is detailed

2.2 Agile Requirements Engineering

- **Requirements Engineering and Agile Software Development:**

  - **Agile software development** focuses more on **continuous collabration** (workshops, interviews etc.) with stakeholders instead of relying on **specification documents** (*example: SCRUM*)
  - **Traditional requirements engineering**
    * focuses on customer collabration mainly at an early phase of the project (longer change cycles)
    * emphasizes a heavy-weight process with extensive, **static specification documents**

- **Agile requirements engineering**
  * fosters communication with the customer during the <u>whole development process</u> to **continuously update requirements**
  * focuses less on extensive documentation, but specification documents **might be necessary** because of legal or contracting reasons etc.
  * includes activities and artifacts that are similar to classical requirements engineering activities

- **Typical Requirement Artifacts in Agile Software Development:**

  - **user story**, **story card**, **use case**, **scenario**, **UML diagram**, **prototype**

- **User Stories:**

  - explanation of a software feature written from the perspective of the end user
  - most frequently used artifact in **agile software development**
  - mnemonic for writing good user stories: INVEST[4]

- **Typical Requirements Engineering Challenges:**

  - different interest groups can raise **conflicting requirements**
  - the people who **pay** for the system are rarely the ones who **use** it
  - the organization and the technical environment may **change** after the system rollout
  - requirements that change during implementation (Change Requests) can lead to additional costs -> project duration/milestones can be affected significantly

---

[4]independent, negotiable, valuable, estimable, small, testable