Enterprise System Integration
(MTAT.03.229)

LECTURE 2: FUNDAMENTAL PRINCIPLES FOR ENTERPRISE SYSTEMS ENGINEERING

MOHAMAD GHRARIB
UNIVERSITY OF TARTU
Outline

Part 1 - [Enterprise] System requirements
Part 2 - [Enterprise] System Architecture
Part 3 - Domain modeling
Part 1

1. [Enterprise] System requirements
What Are Requirements?

Requirements constitute a specification for the new system. A problem can be defined as the difference between things as they are now and things as they are desired.

*It is more important to understand the problem than the solution* - Albert Einstein

---

What Are Requirements?

Requirements ...must identify:

- **why** a system is needed, based on current and foreseen conditions;
- **what** system features will serve and satisfy this context;
- **how** the system is to be constructed.

*A problem unstated is a problem unsolved*

- Douglas T. Ross
What Are Requirements?

According to Jackson\(^2\), the term requirements might refer to many things:

- The system must tolerate user errors.
- Passenger should not be allowed to open a door when the vehicle. Requirement should be satisfiable.
- Passenger should not be allowed to open a door when the vehicle is moving.
- Can be satisfied by implementing a mechanism to lock the doors while car is moving.

---

What Are Requirements?

There are two main types of requirements:

- Functional requirements: describe what the system is supposed to do;
- Non-functional requirements: describe how the system will do it.

**Functional requirements:**
Describe the functionality, inputs/outputs, etc. of the new system.

**Non-functional requirements:**
Describe safety/reliability/security needs, performance/usability criteria, etc.
What is Requirements Engineering?

Requirements Engineering (RE) refers to the process of “defining”, documenting and “maintaining” requirements\(^3\).

Requirements Engineering is about determining\(^4\)
- problems with the current status (As-Is);
- objectives to achieve;
- changes to bring about for a better future (To-Be).


Why Requirements Engineering?

Bad requirements is a main reason for a project failures. Error s in requirements propagate to other activities.

cost (correcting implementation errors) = 100 X cost (correcting req. errors)\(^5\).

Errors should be detected and addressed at requirements level rather than after implementing the system.

---

How to do Requirements Engineering?

Requirements Engineering Process:

1. **Requirements elicitation**: requirements discovered through consultation with stakeholders;

2. **Requirements analysis and negotiation**: requirements are analyzed and conflicts resolved through negotiation;

3. **Requirements documentation**: a requirements document is produced;

4. **Requirements validation**: The requirements document is checked for consistency and completeness.

---

1. Requirements elicitation

Where do Requirements come from?

- **Stakeholders (actors) of the system-to-be**: end-users; designers; developers; user support staff; management staff, etc.
- **Application domain**: should be explored together with its political, organizational, social aspects, constraints that may influence the system.

How do we “elicit” Requirements?

Among existing elicitation techniques: Interviews; Questionnaires; Task analysis; Scenarios (use cases); Prototyping; etc.

Each of these techniques have **Strengths and Weaknesses**.

A compination of **two or more techniques** is common for **complex projects**.
2. Requirements analysis

How do we “analyze” Requirements?

We must perform: requirements validity check; requirements “completeness” check; requirements consistency analysis.

How do we “negotiate” Requirements?

We must perform: Conflict resolution (relaxing and/or modifying some requirements);

In addition, Requirements must be refined to a level of detail sufficient for system design, i.e., can be operationalized (derive specifications).
2. Requirements analysis

Requirements vs Specifications

A *specification* is a restricted form of requirement, providing *enough information* for the implementer to build the machine without further environment knowledge [2].

Requirements should be *clear* but not very *specific*. *Why?*

Example: a main Requirement for the lecture to success is having *adequate time*. 
3. Requirements documentation

How do we “document” Requirements?

Basically, we write them down in structured, clear and unambiguous way\(^7\).

The requirements document serve as a “contract” between stockholders and developers.

\(^7\) Actually, it is a bit more complex than that, but it is out of the scope of this course.
4. Requirements validation

How do we “validate” Requirements?

We must perform:

- **Requirements “completeness” check** aims to verify that the requirements capture all the functions, features, constraints, etc. expected by the stakeholders.
- **Requirements consistency check** aims to verify that no inconsistency exist among the requirements.
- **Requirements realism check** aims to verify that the requirements can actually be implemented.
Part 1

1.2 [Enterprise] System requirements – practical example
This scenario will help you to understand how you can design your own case study and, later, implement it.

This will help you to understand:

- why the integration is required;
- what modifications are required to achieve the required integration;
- how such modifications can be implemented.

Based on an example used in the ESI’19 course
Plant hire scenario - practical example

• **Buildit** is a construction company specialized in public works (roads, bridges, pipelines, tunnels, railroads, etc.).

• In **Buildit**, **site engineers** may need a special type of equipment (e.g., truck, a bulldozer, a water pump, etc., called a “plant”). However, **Buildit** owns very little plants and it rents the needed plants from another company, called **Rentit**.

• **Rentit** lists on its website its available plants along with their IDs, availability periods, cost, etc., and potential customers can make **Plant Hire Request** from the listed plants.

**Buildit** decided to buy **Rentit**, yet **Rentit** will keep providing its services to other construction companies.
Plant hire scenario - practical example

The process for renting a plant:

• When a site engineer needs to hire a plant, he/she consults the catalogue of a Rentit to identify a plant that fulfils the requirements, check whether it is available for the required period, and whether the cost is appropriate.

• If all criteria are met, the site engineer can creates a plant hire request.

• Each plant hire request has to be approved or rejected by a works engineer at Built.

• Works engineer can also modify the plant hire request before approving it.

• Once the a works engineer has approved the plant hire request, Builtit’s information system automatically generates a Purchase Order (PO) for hiring the plant and sends this PO to the plant supplier.
Plant hire scenario - practical example

The process for renting a plant:

• *Rentit* may accept or reject a *PO*.
• If a *PO* is rejected, the requesting party should be notified.
• If a *PO* is accepted, *Rentit* delivers the requested plant to the construction site at the required date.
• Upon delivery, the *site engineer* inspects the plant and if everything is in order, he/she accepts the engagement. Otherwise, he/she rejects it, and send the plant back.
• The *site engineer* may request an extension of the period of engagement. If the plant is available, the *site engineer* can modify the *plant hire request*. *Buildit*’s information system, then, sends a modified *PO* to the *Rentit*.
Plant hire scenario - practical example

The process for renting a plant:

• When the period of engagement is concluded, Rentit picks up the plant, then, sends an invoice to Buildit.
• The invoice is automatically matched with a PO by Buildit’s information system. The invoice should then be approved/rejected by the site engineer.
• If the invoice is approved, payment is scheduled and, then, paid in full to Rentit. If the invoice is rejected, the payment is not scheduled and the site engineer is responsible for communicating with Rentit to resolve the issue.
• Rentit may send reminders of unpaid invoices. When Buildit’s system receives a payment reminder, it checks if the payment has been scheduled or not or made, Buildit re-sends related information.
• If the payment has not been made yet because the site engineer still needs to approve it, the site engineer is notified that there is an unapproved invoice.
Buildit - initial functional requirements

BR1. The system should allow site engineers to create a plant hire request.

BR2. The system should allow site engineers to modify a plant hire request prior to its approval by the works engineer.

BR3. The system should allow site engineers to cancel a plant hire request. If a cancellation is requested after the PO has been sent, a request for cancellation should be sent to the supplier.

BR4. The system should allow site engineers to check the plant hire request status.

BR5. The system should allow works engineers to approve, reject or modify a plant hire request.

BR6. The system should produce a PO for every approved plant hire request and forward it to the corresponding supplier. The supplier may respond that the plant being requested is no longer available (which means the PO is rejected), or it may respond with a confirmation of the PO.
BR7. The system should allow site engineers to request an extension in order to keep a plant longer than its initial period of engagement. When an extension is requested, the system should produce a modified PO and forward it to the supplier. The supplier may accept/reject the modified PO.

BR8. The system should allow a supplier to submit invoices for a given plant hire.

BR9. When an invoice is received, the system must check that the PO number in the invoice corresponds to an existing and unpaid Purchase Order. If the PO does not exist, an error message is returned to the supplier.

BR10. The system must allow site engineers to approve an invoice and to retrieve the PO associated to an invoice.

BR12. The system must submit a notice to the supplier after the invoice is approved and paid.
Rentit - initial functional requirements

RR1. The system should show available plants, availability status, price per day, etc.

RR2. The system should allow a customer to list the available plants and their prices.

RR3. The system should allow a customer to check the price for a given plant.

RR4. The system should allow a customer to check the availability of a given plant for a given time period.

RR5. The system should allow a customer to submit a PO for hiring a plant. The PO may be accepted or rejected depending on the plant’s availability.

RR6. The system should allow employees at Rentit to determine which plants need to be delivered on a given date.

RR7. The system should allow a customer to request modifying a PO request and accept/reject a modified PO depending on the plant’s availability.
Rentit - initial functional requirements

RR8. The system should allow a customer to submit a request to cancel a PO. A cancellation request is normally accepted if the request arrives prior to the plant being delivered. If the plant has already been delivered, the cancellation request is rejected.

RR9. The system should allow employees at the plant depot to mark the plant as “rejected by customer”.

RR10. The system should allow employees at the plant depot to mark a plant as “returned”, meaning that the plant has been returned in due form and the rental period has expired.

RR11. The system should submit invoices for “returned” plants.

RR12. The system should submit payment reminders for unpaid invoices.
Part 2

2.1 [Enterprise] System Architecture
A system architecture is "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution."  

[Enterprise] System architecture is an abstract representation of the system components, their relationships, and the principles governing its design and evolution.

Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure — Melvin E. Conway

1 ANSI/IEEE 1471-2000, "Recommended Practice for Architecture Description of Software-Intensive Systems."


Do we need an [enterprise] system architecture?
How to define Enterprise Architecture

The process of constructing an enterprise architecture involves,

• identify key **components** *(building blocks)* that compose the enterprise system;
• identify the **relationships** among these components;
• identify how these components use each other’s functionality and data to achieve their own functionalities;
• identify how control is managed between the components.
Part 2

2.2 [Enterprise] System Architecture – practical example
[Enterprise] System Architecture
[Enterprise] System Architecture
[Enterprise] System Architecture
[Enterprise] System Architecture

Buildit

Rentit

Database

Database
[Enterprise] System Architecture

**Buildit**
- Plants rent service
- Plant management service
- Purchase Order (PO) service
- Financial service
- Database

**Rentit**
- Plants catalog
- Billing services
- Booking service
- Notification and collection services
- Warehouse services
- Delivery and pickup services
- Database
[Enterprise] System Architecture

**Buildit**
- Plants rent service
- Purchase Order (PO) service
- Plant management service
- Financial service
- Database

**Rentit**
- Plants catalog
- Booking service
- Billing services
- Warehouse services
- Delivery and pickup services
- Notification and collection services
- Database
[Enterprise] System Architecture

**Buildit**
- Plants rent service
- Plant management service
- Purchase Order (PO) service
- Financial service

**Rentit**
- Plants catalog
- Billing services
- Booking service
- Notification and collection services
- Warehouse services
- Delivery and pickup services

Database
[Enterprise] System Architecture

**Buildit**
- Plants rent service
- Purchase Order (PO) service
- Plant management service
- Financial service

**Rentit**
- Plants catalog
- Booking service
- Billing services
- Notification and collection services
- Warehouse services
- Delivery and pickup services

Database
Part 3
Domain modeling
Domain modeling

A *domain model* should include a set of *fundamental language constructs* that represent the key *concepts* of the domain along with the *relationships* among them as well as a set of *constraints* that govern how such constructs can be used to produce valid models\(^1\).

The definition of a *domain model* required several *iterations*, and came to its end when the model is considered complete with respect to the main properties of the domain that we aim to capture.

\(^1\) A systematic approach to domain-specific language design using UML. In: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2007, IEEE, pp 2–9
A **class diagrams** can be used to describe a **system** in terms of the **entities** "composing" it and their **inter-relationships**.
A **class diagrams** can be used to describe a **system** in terms of the **entities** “composing” it and their **inter-relationships**.

**Aggregation** implies a relationship where a class can exist independently of the other class. *E.g.*, If we delete the family, the pet still exist.

**Composition** implies a relationship where the class cannot exist independent of the other class. *E.g.*, If we delete the family, the person should not exist.
A class diagrams can be used to describe a system in terms of the entities "composing" it and their inter-relationships.

**Generalization** is a mechanism for combining similar classes of objects into a single, more general class. **Specialization** is the reverse process of Generalization that means creating new sub classes from an existing more general class.
A class diagrams can be used to describe a system in terms of the entities “composing” it and their inter-relationships.

An attribute (property): is a named slot within a classifier that describes a range of values that instances of the classifier may hold.

Attributes are always single-valued in UML, and their value might be derived.

An operation is a behavioral feature related to the class.
An operation is not a standard property in a domain//meta/conceptual model because such models are static. We only included in this course because we will use it in this way later. Usually, such behavioral attributes are added after the domain model is finalized and when you move to implementation. You should know that classes in modeling languages contain operations/functions. So we will allow using it in our model in order not to lose any important information but do not do that outside of this course.
A class diagrams can be used to describe a system in terms of the entities “composing” it and their inter-relationships.

Associations represent semantic relationships. Each association can have up to two roles for participating objects.

Associations represented with directed arrows are navigable only in one direction.
A **class diagrams** can be used to describe a **system** in terms of the **entities** “composing” it and their **inter-relationships**.

**Associations** represent semantic relationships. Each **association** can have up to two roles for participating objects.

Each **role** can also have an associated cardinality range (“**multiplicity**”).

**Multiplicity** allows to specify cardinality, i.e., number of elements.
Thank You for your attention

Mohamad Gharib
mohamad.gharib@ut.ee

unitartu

tartuuniversity