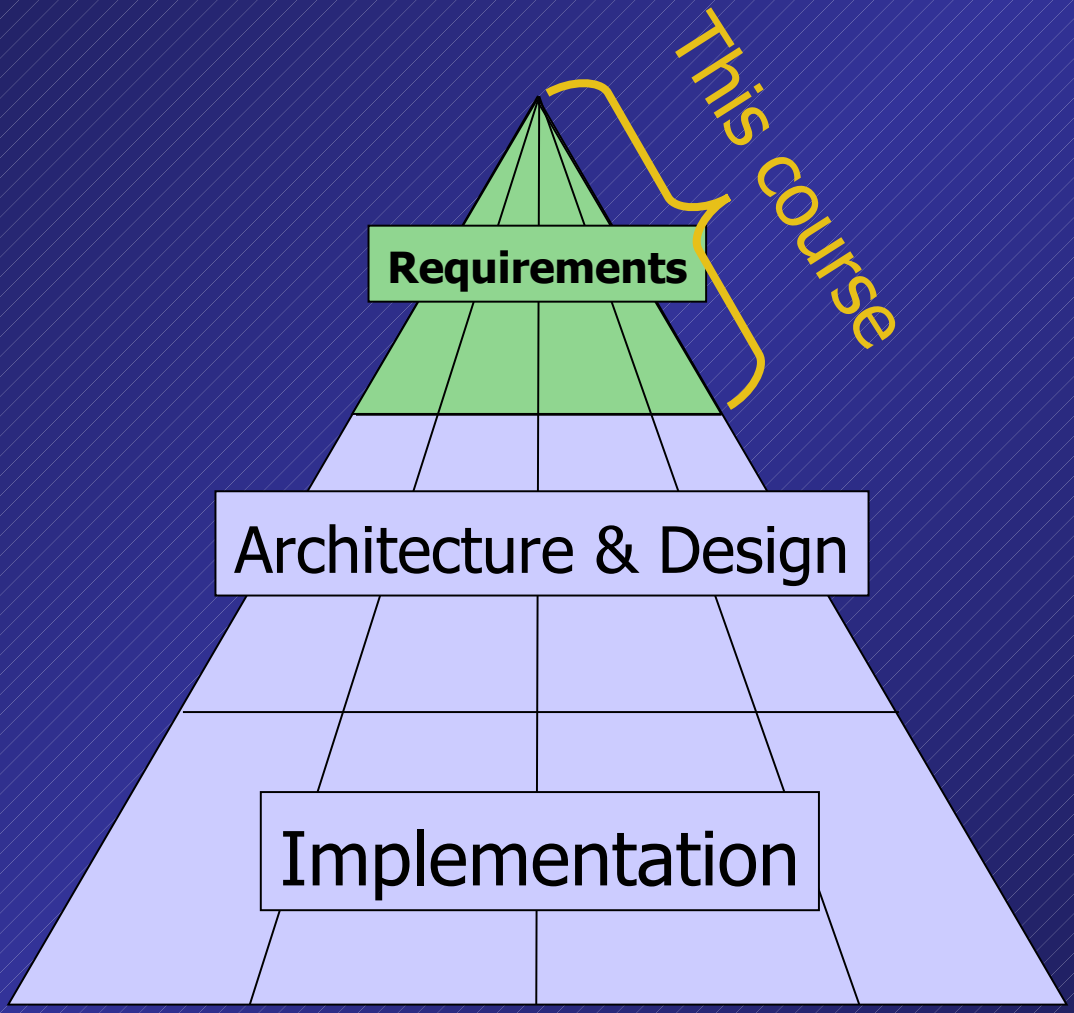
Lecture 2: Specifying user requirements

- System vision
- Business process description
- Determining the scope of a software system
- Modelling of user requirements
- User requirements documentation

Requirements – where are we?

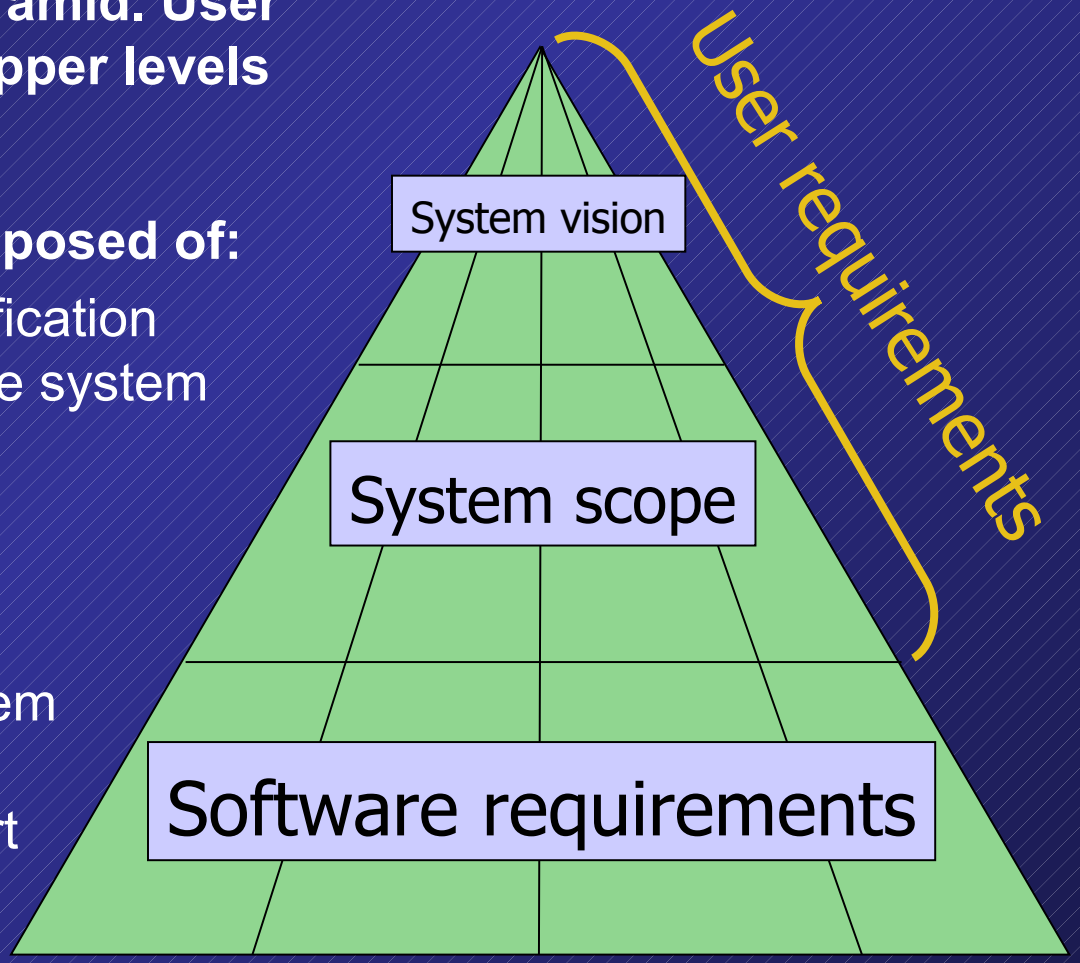


User requirements – where are we?

All requirements form a pyramid. User requirements occupy the upper levels of the pyramid.

User requirements are composed of:

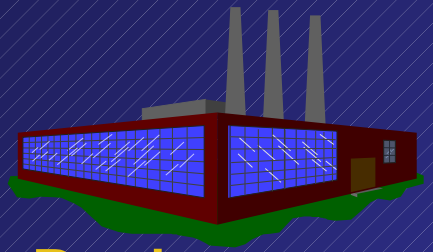
- System vision – a specification of general features of the system in close connection with the business needs of the client
- Scope requirements – a specification of the system needed to determine its size and amount of effort needed to build it



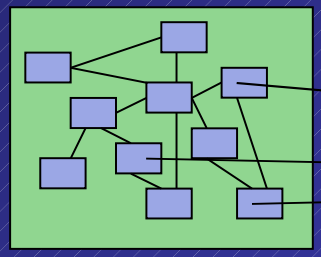
User requirements – where do they come from?

User requirements come from the needs of the business.

Before we define specific requirements, we need a "vision".

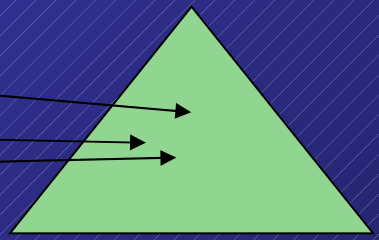


Business



Business description

Problems



System vision

"Trawling for requirements"

System scope

- The system vision is a response to problems.
- The system scope is a result of seeking for ways to automate the business according to the vision.

Purpose of user requirements – how to reach it?

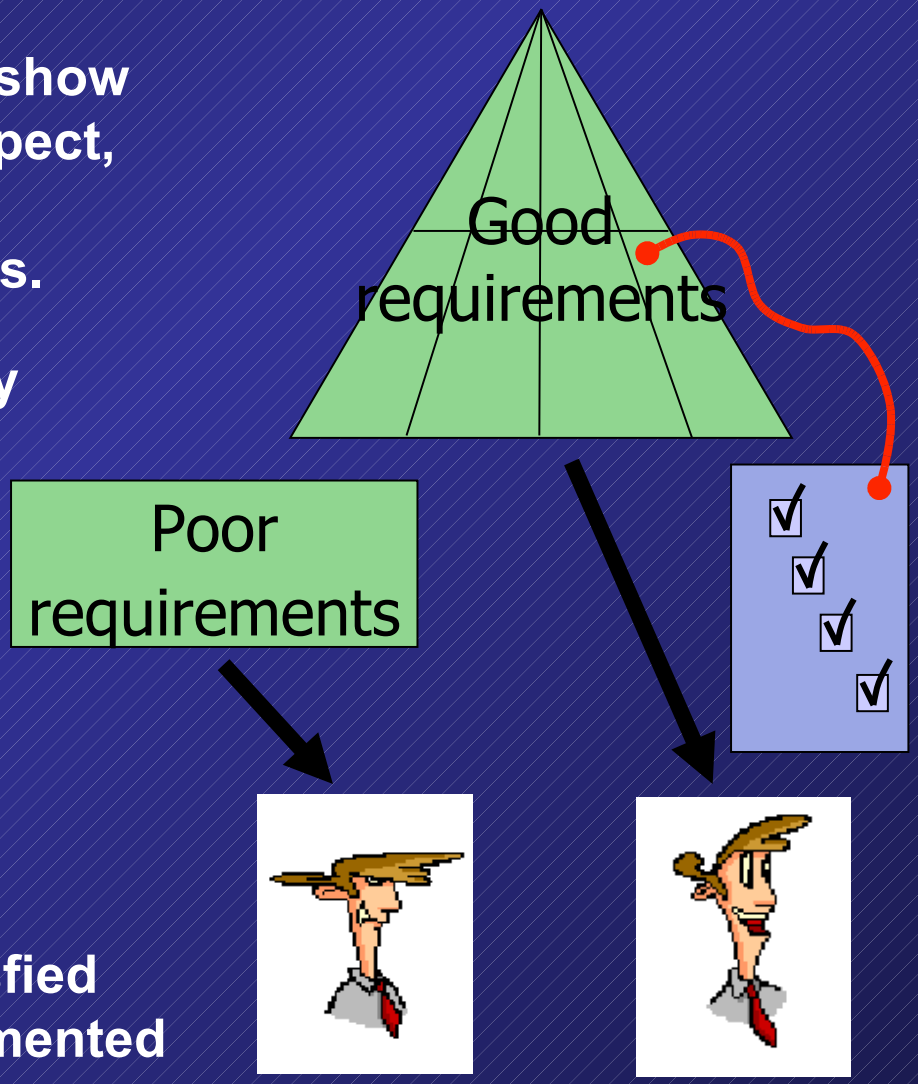
User requirements are there to show the client what he/she might expect, and to allow for verifying the realisation of these expectations.

Requirements are there to verify system's quality.

User requirements should be:

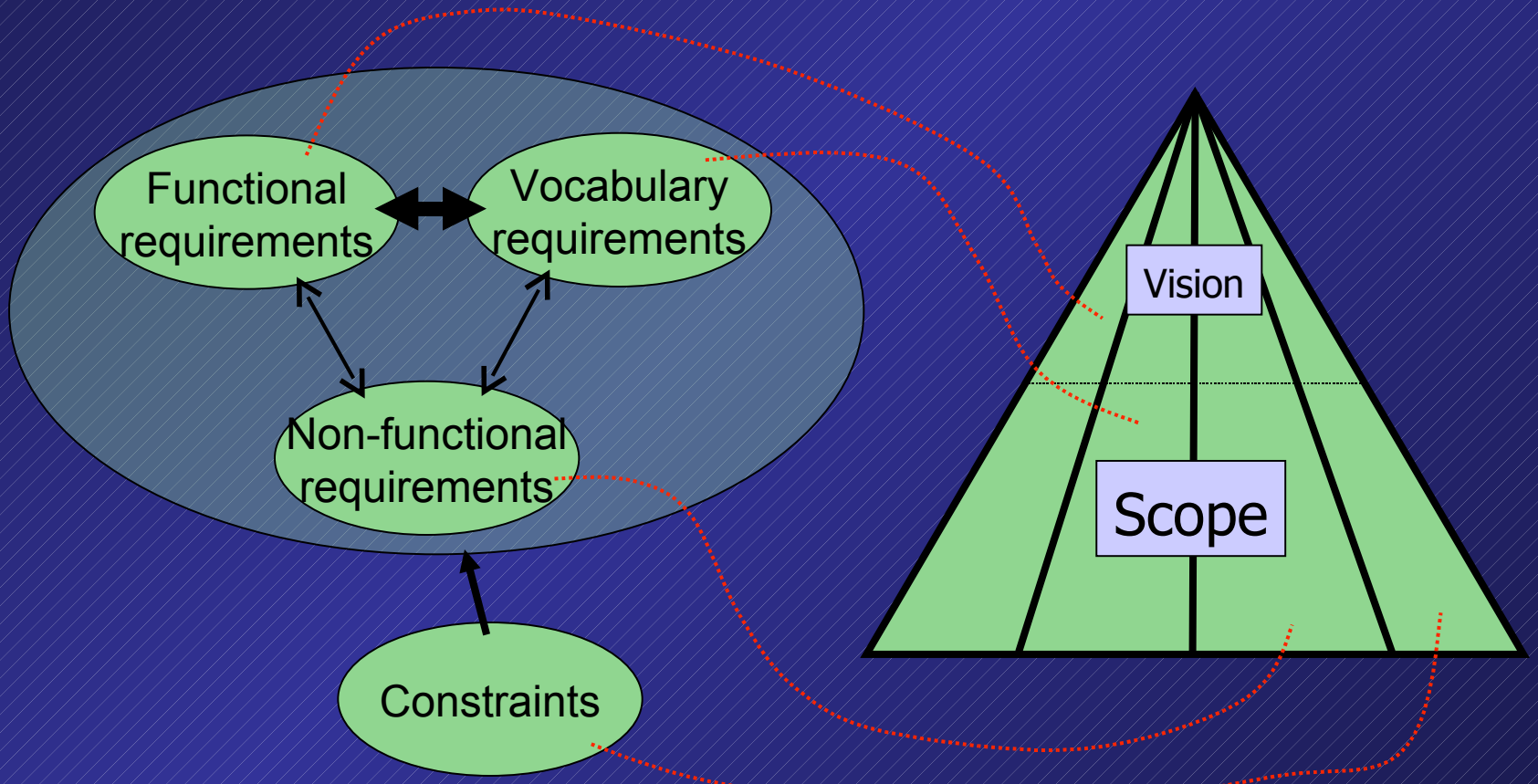
- complete,
- consistent,
- unambiguous
- measurable (testable)

Good user requirements = satisfied client after the system is implemented



Structure of user requirements

How to make user requirements complete, consistent, unambiguous? Structure them well!



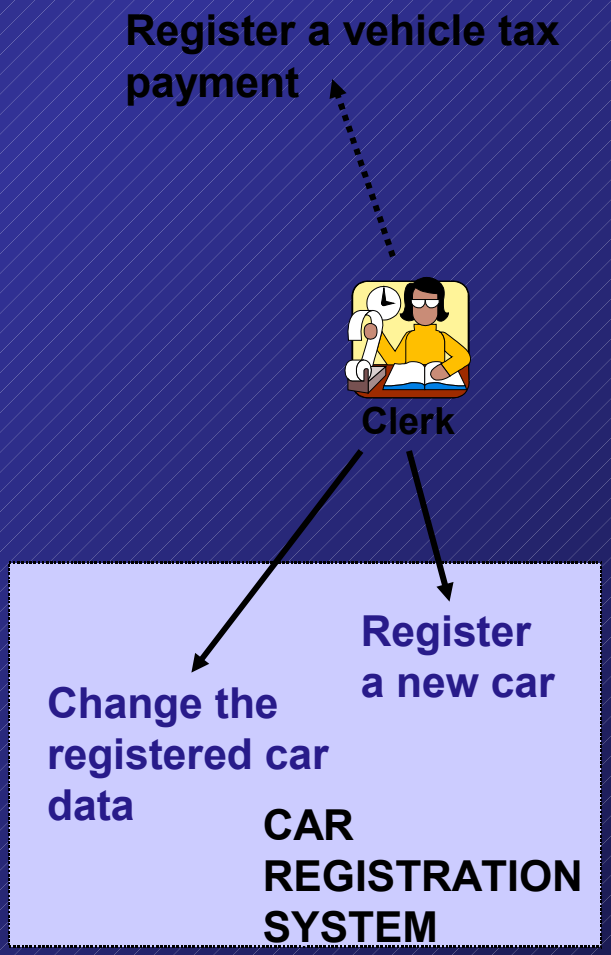
Functional requirements

Functional requirements determine the system's behaviour while it is interacting with the user (or other system).

- What services should the system offer?
- How should it react to specific input messages?
- How to behave in specific situations?

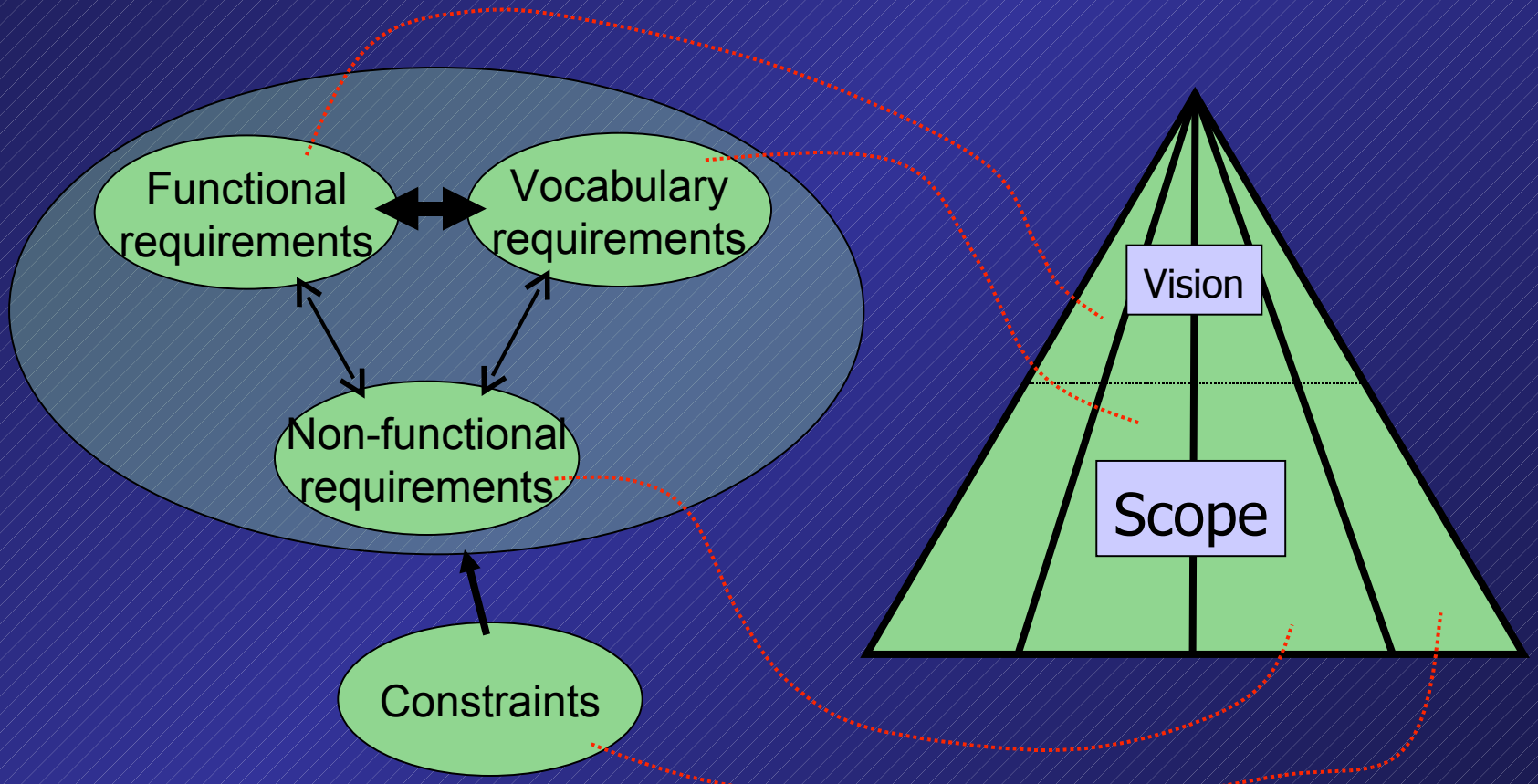
The set of functional requirements determines the scope of the system to be built.

Often it is wise to state also what functionality is out of scope (out of the system's functionality).



Structure of user requirements

How to make user requirements complete, consistent, unambiguous? Structure them well!

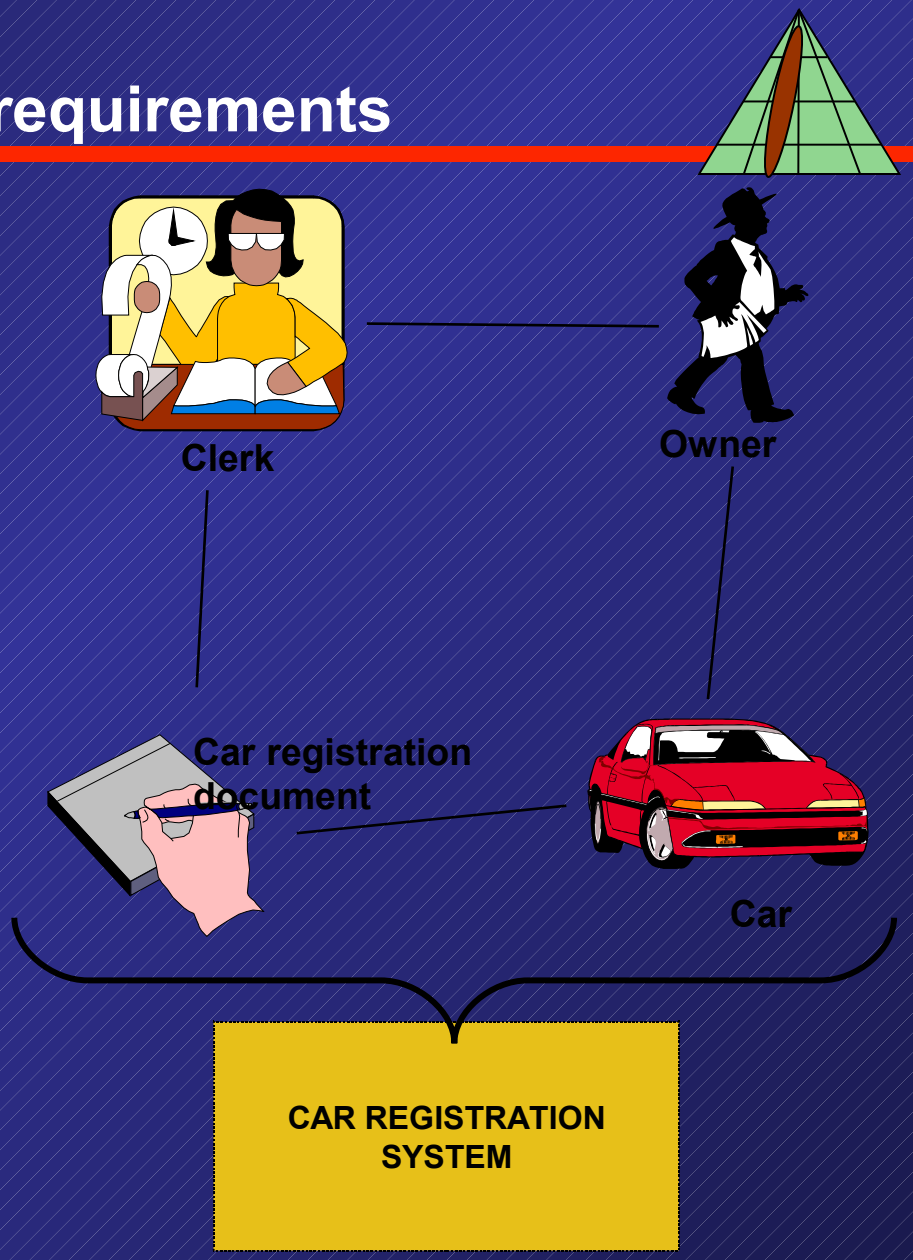


Vocabulary requirements

Vocabulary requirements define the scope of notions and associated data to be handled by the software system.

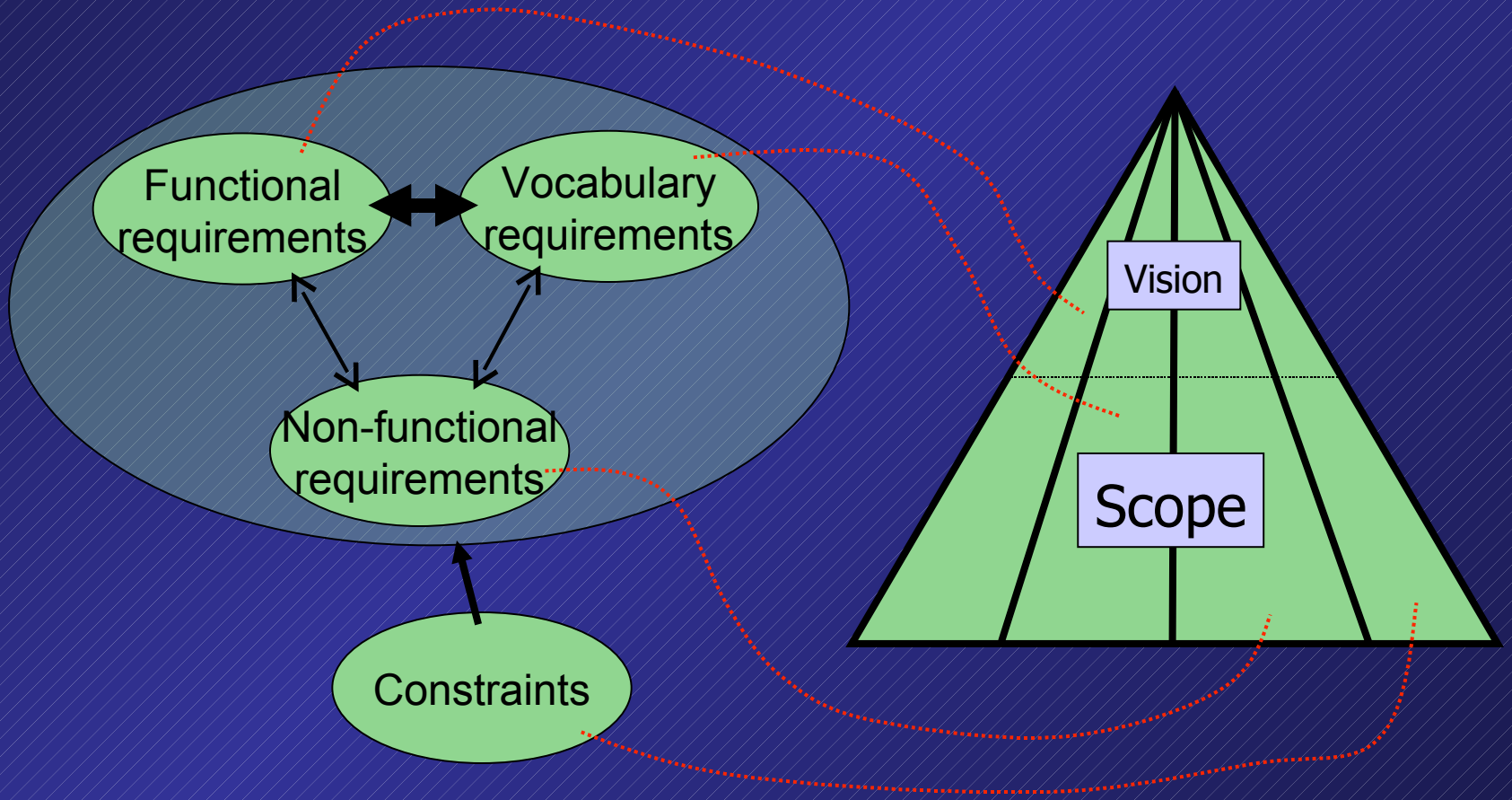
The vocabulary contains definitions of notions used inside the functional requirements (and non-functional too…).

When defining notions we also describe relationships between them. We can show these relationships graphically.



Structure of user requirements

How to make user requirements complete, consistent, unambiguous? Structure them well!



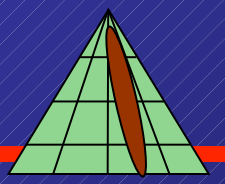
Non-functional requirements

Non-functional requirements describe the quality features of the prospective system.

- How fast should be the system?
- How reliable should be the system?
- How safe should be the system?
- How user-friendly should be the system?
- What norms should the system comply to?

Non-functional requirements can be global (pertain to the whole system) or local (directly pertain only to specific functional requirements).

Non-functional reqs – example taxonomy



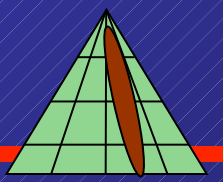
Functionality: A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs **(suitability, accuracy, interoperability, compliance, security)**

Reliability: A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time **(maturity, recoverability, fault tolerance)**

Usability: A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users **(learnability, efficiency of use, operability)**

Efficiency: A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions **(time behaviour, resource behaviour)**

Non-functional reqs – example taxonomy (2)

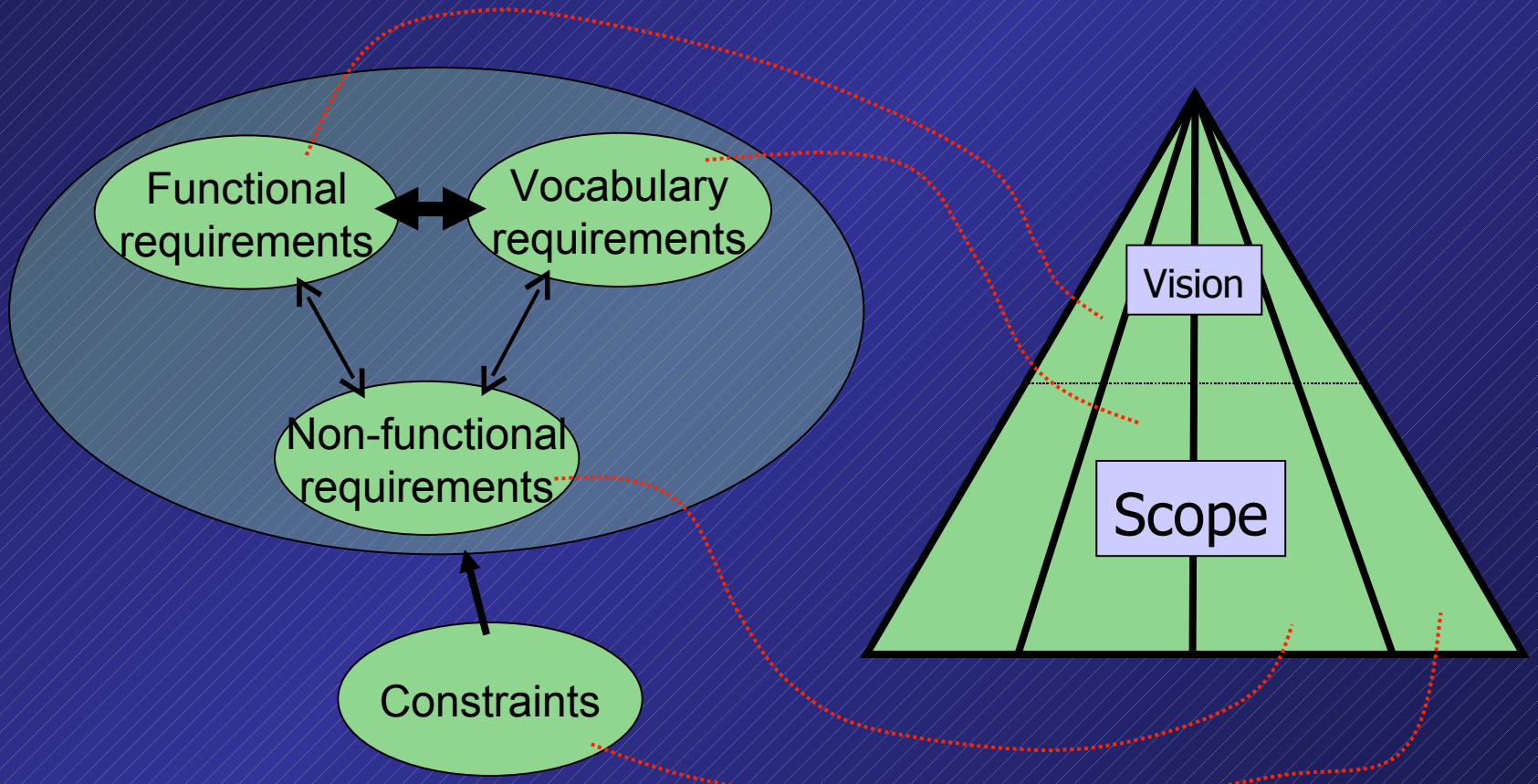


Maintainability: A set of attributes that bear on the effort needed to make specified modifications **(stability, analyzability, changeability, testability)**

Portability: A set of attributes that bear on the ability of software to be transferred from one environment to another **(installability, replaceability, adaptability)**

Structure of user requirements

How to make user requirements complete, consistent, unambiguous? Structure them well!



Constraints

Constraints are determined by the business and technical environment that surrounds the system. They describe the external conditions set for the system (software, hardware, work conditions).

Examples of constraints:

- Local area network configuration
- Performance of client and server machines
- Database engine system owned
- Working environment (in the field, in a factory)
- Administrative personnel (how many people)
- Legal conditions

CAR REGISTRATION SYSTEM

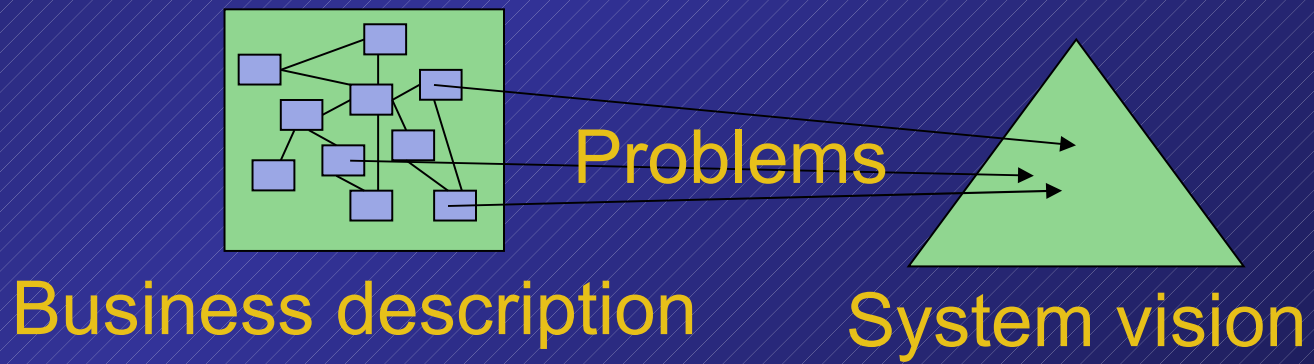
Building the system's vision

Before we start finding more detailed requirements we need to formulate the vision of the prospective system. This vision should be accepted by the interested groups (sponsors) of the project.

We need to answer the following questions:

- Why do we build this system? What business problems will the system solve? Who is this system for? What features should the system have?

We seek answers to these questions directly within the business.



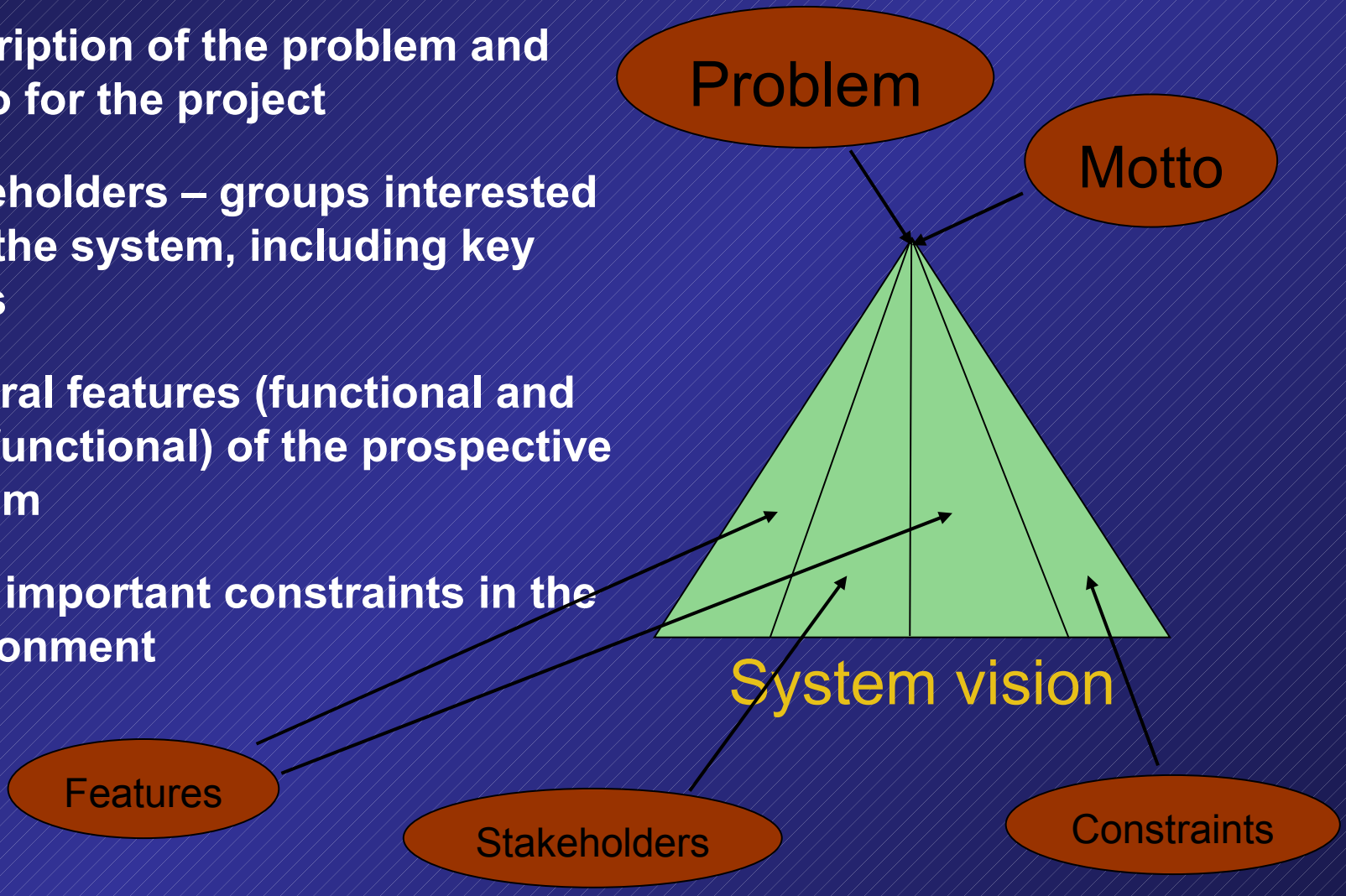
Elements of the system vision

Description of the problem and motto for the project

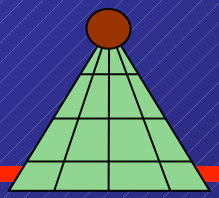
Stakeholders – groups interested with the system, including key users

General features (functional and non-functional) of the prospective system

Most important constraints in the environment



Problem statement



Problem associated with [here a description of a problem found in the business description]

affecting [here a list of stakeholders affected by the problem]

which results in [here a descriptions of problem implications]

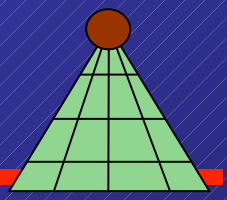
can be solved by building a system [here a descriptive name of the system]

thanks to which [here a list of benefits from building the system]

IMPORTANT: the problem statement is too easily omitted by the analysts!

Specifying the problem statement forces us to find the real causes of problems in the business that necessitate the change.

Motto for the project

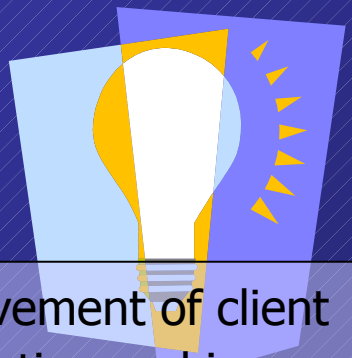


It is quite advantageous for the project to have it's "motto".

One or two sentences placed on the title page of all the project documents. These sentences prevent from forgetting the real goal for building the system.

The motto "gives light" (unifies) all the members of the project.

This way we don't forget about the requirements!



Improvement of client satisfaction and increase in sales through an on-line order handling system.

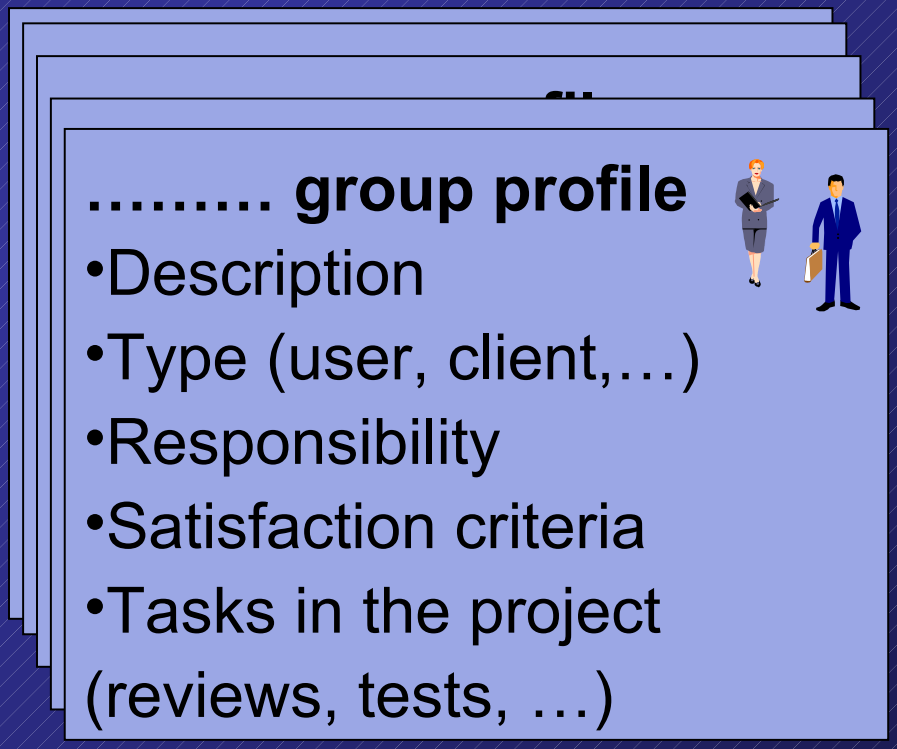
Sales system

Stakeholders

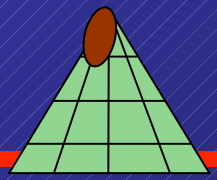
The system will affect different group in the business organization. These groups might have different expectations…

Examples of such groups:

- Direct stakeholders (“fund keepers”)
- Users and their representatives
- Clients and their representatives
- Business investors
- Business owners
- Managers of the users
- IT service people



System features



A short statement (paragraph) about what the system shall do (functional) or what quality the system shall have (non-functional).

Often the statement starts with "The system shall…".

System features reflect the needs of different groups in the business (stakeholders).

Usually, the system features are formulated directly by the stakeholders.

F: The system shall allow for checking for stolen car data from the police system (SCS) in real time.

F: The system shall allow the managers to allocate workload evenly between clerks

NF: The system shall be fast enough to shorten processing time of car owners by 20%.

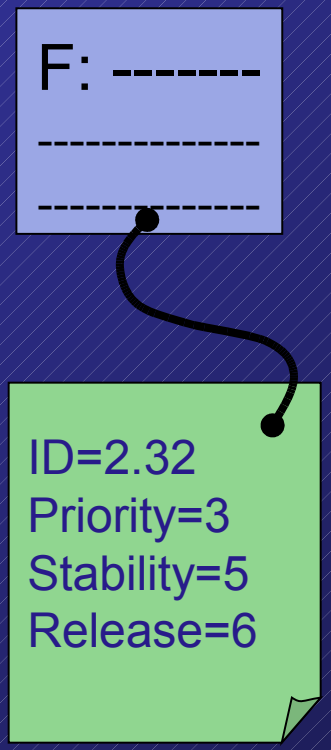
Requirements' attributes

System features (and other requirements too!) should be assigned with attributes.

Attributes allow for efficient requirements management.

Requirements might have assigned attributes as:

- Identifier – an official symbol assigned to this specific requirements for tracking purposes
- User priority – describes how critical it would be for the client to have the requirement realized in the system (critical, important, useful)
- Technical priority – describes how critical (risky) the requirement is for the system's architecture
- Stability – describes the probability of the requirement to change
- Target release – a number denoting system version (iteration) at which the requirement should be realized



Requirements' attributes (2)

Other attribute types include:

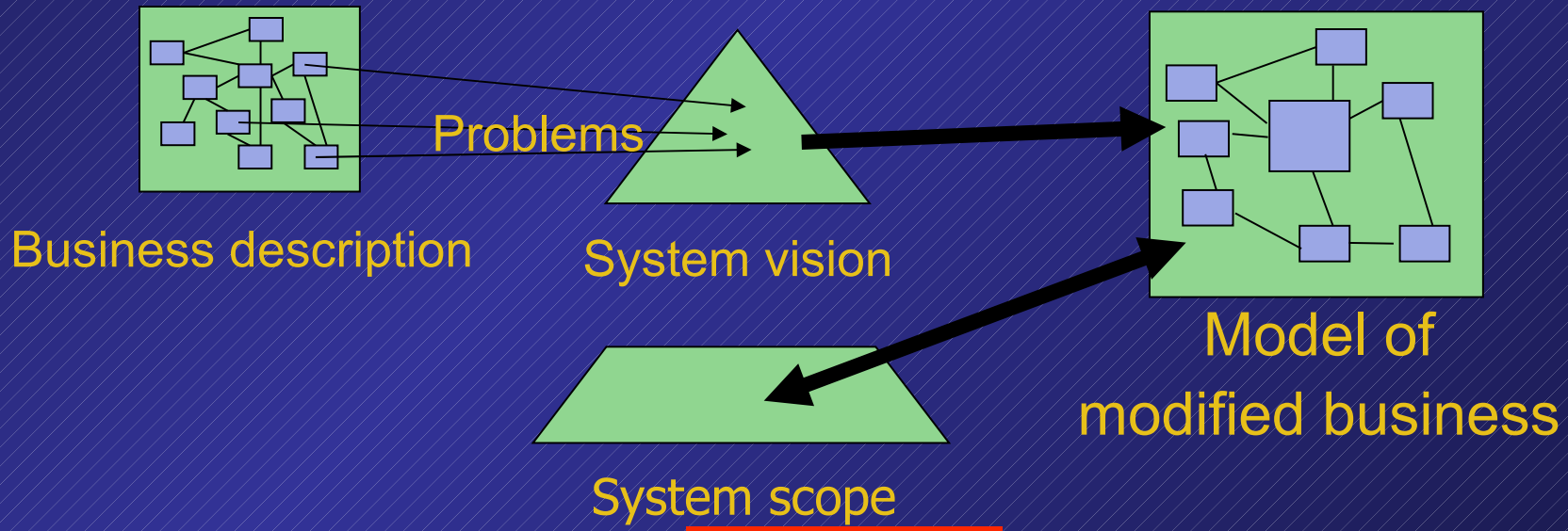
- Type classification – assignment of the requirement into one of the requirement classes (functional, non-functional, constraint)
- Status – determines how far is the requirement in the acceptance and realization process (proposed, accepted, realized, …)
- Level of effort – determines how hard it would be to realize the requirement (large, medium, small)
- Risk – describes the level of (technical, business) risk associated with realizing the requirement
- Responsible person – determines who is responsible for maintaining (formulating, changing) the requirement
- Revision number – a number denoting the current version of the requirement definition

Vision influences the business

While defining the system's vision we should analyze its influence on the business itself!

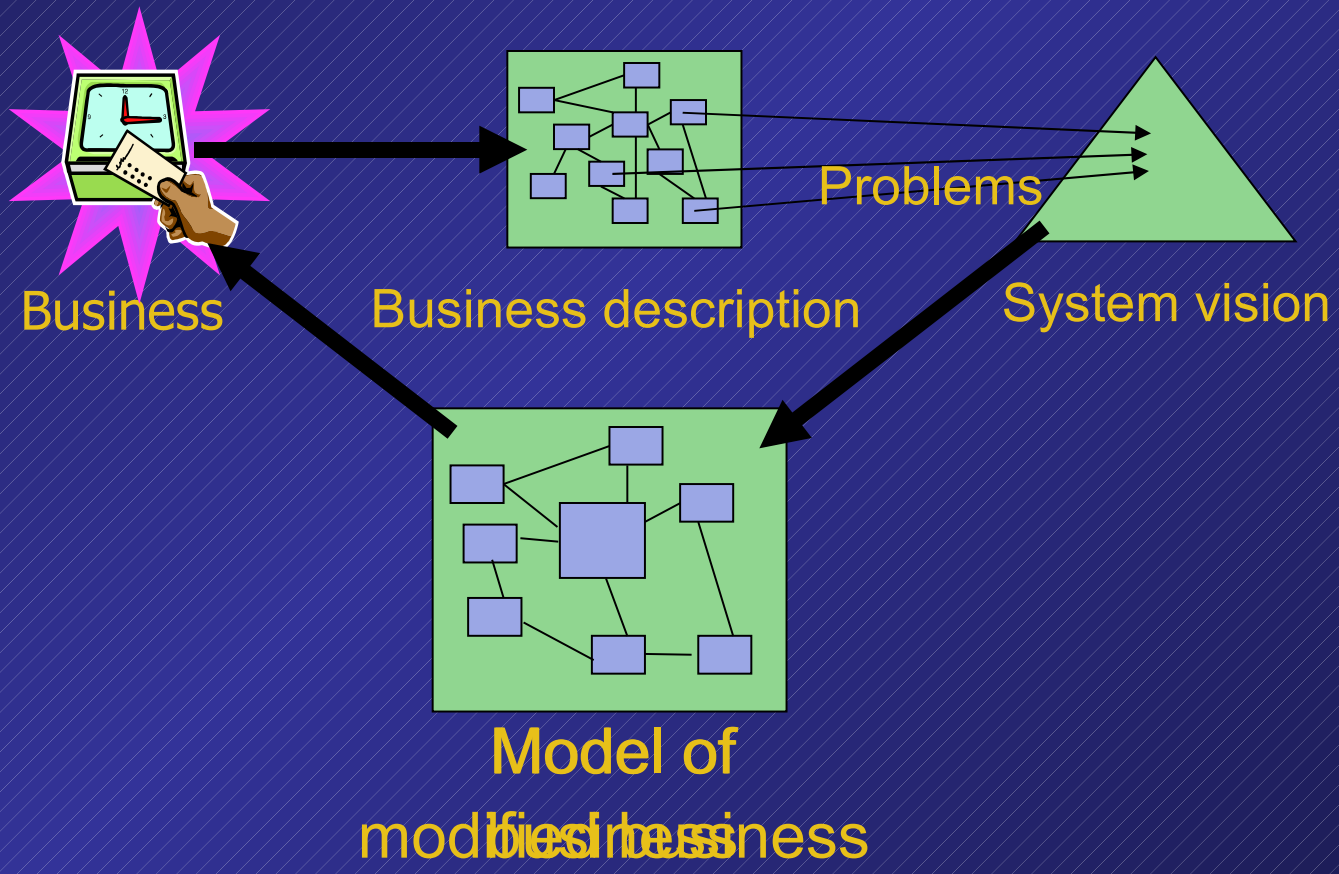
User requirements specification should also specify how business processes will be shaped after delivering the system.

Business process description depends on the vision of the system but also reflects its scope.



Requirements analysis influences the business

While analysing the system we may also find out about non-software solutions to the business problems.

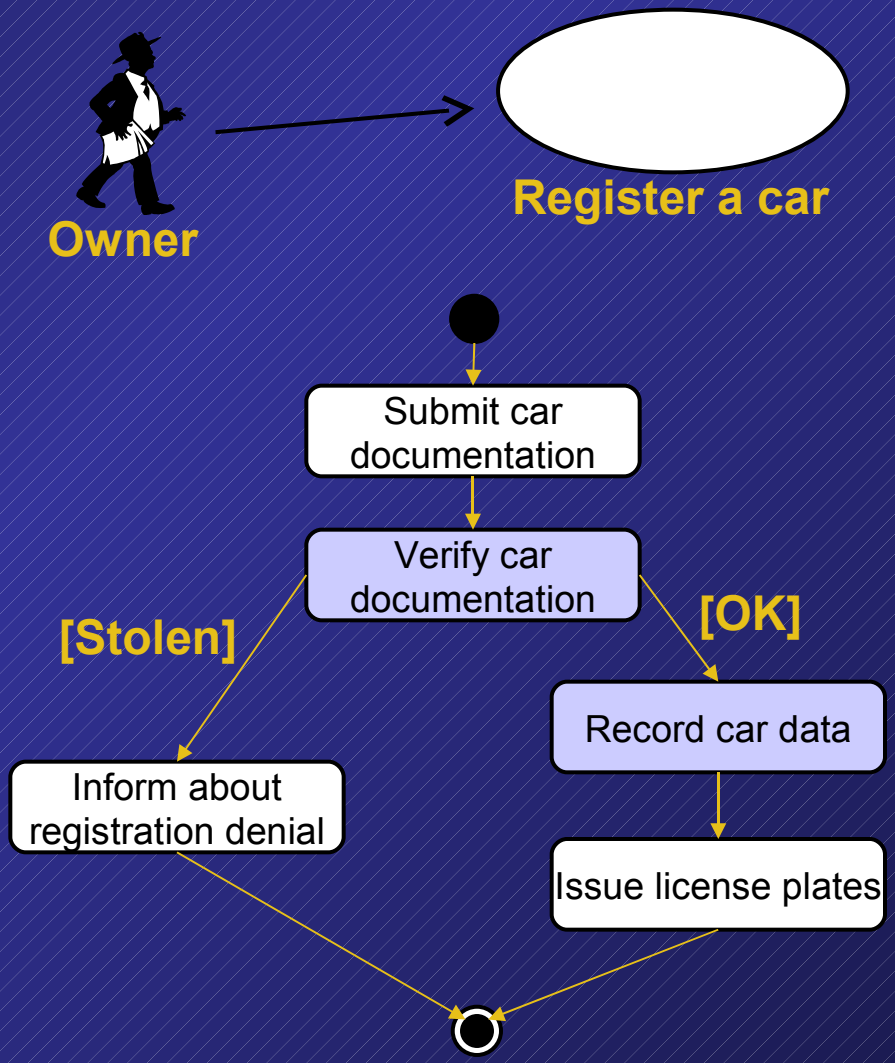


How to model the influenced business?

When verifying the user requirements we mainly want to model the business dynamics – business processes.

Business vocabulary is reflected in the system scope vocabulary (no need to have a separate "business vocabulary").

Business processes can be modelled with UML use case diagrams and activity diagrams.

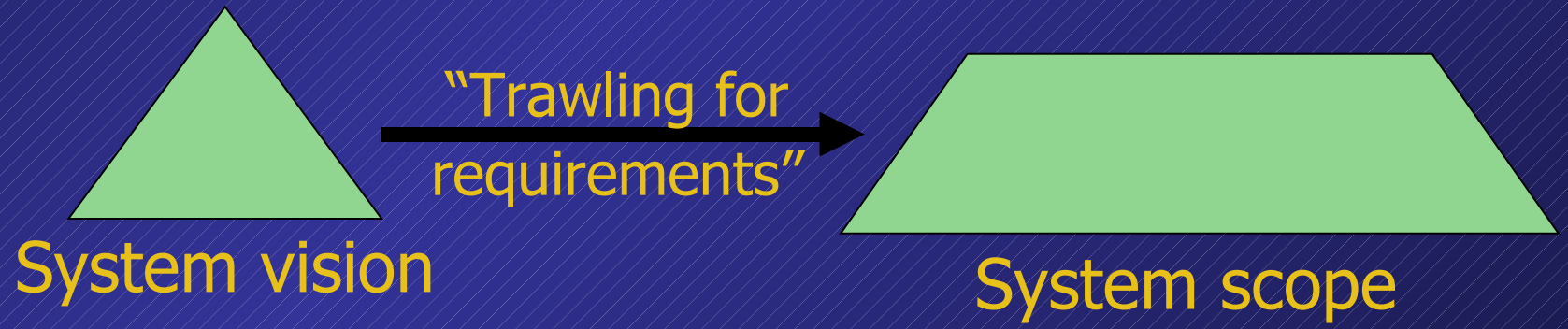


From vision to scope

Vision is not a good basis for formulating a contract between the client and the developers. This is due to the fact that system features give rather vague understanding of the scope of work.

To formulate a contract we need a precise definition of the scope. This prevents possible future misunderstandings.

At the same time, the scope requirements should be general enough not to suggest technical solutions (data base structure, middleware platform, component structure, …).



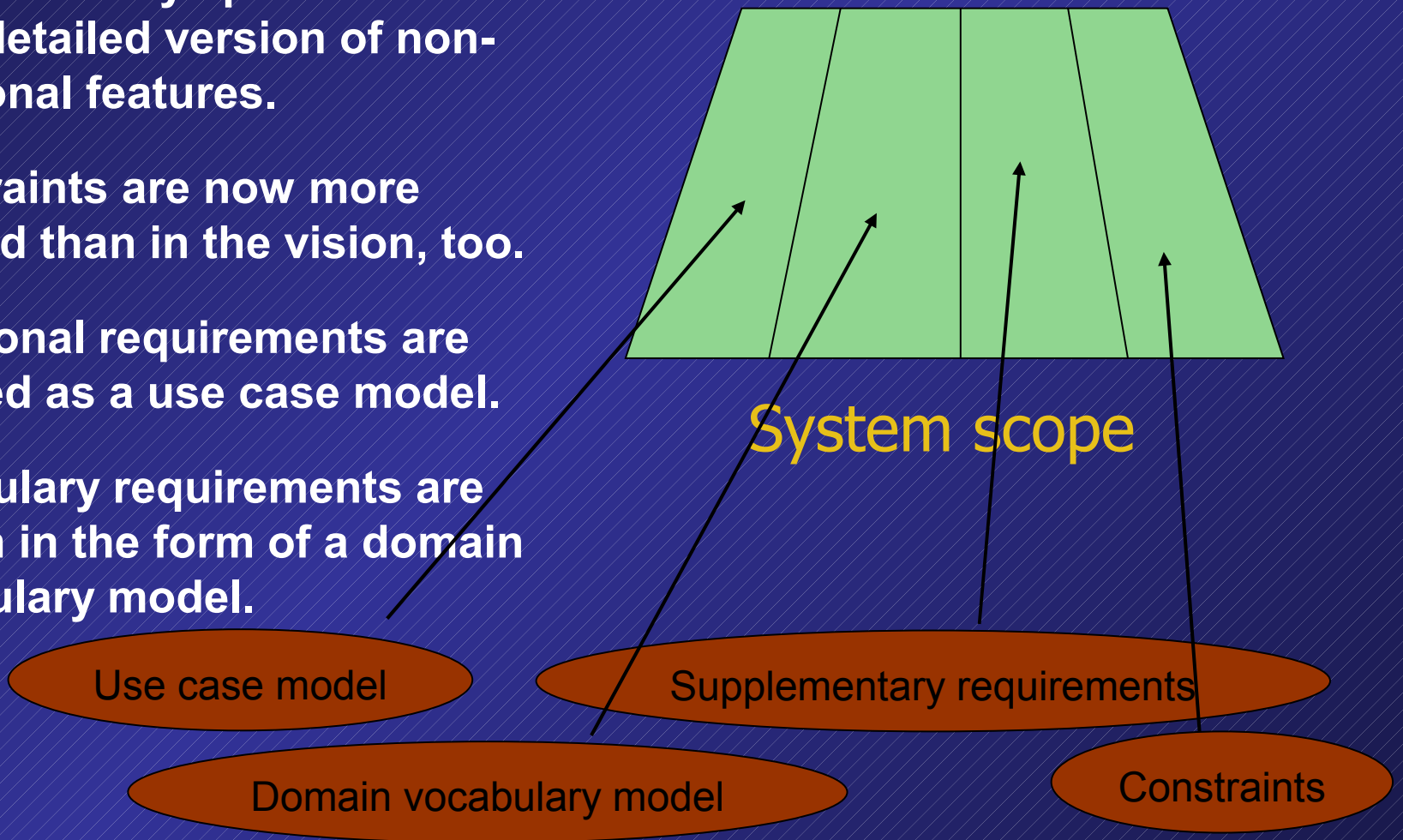
Elements of the system's scope

Supplementary specification is a more detailed version of non-functional features.

Constraints are now more detailed than in the vision, too.

Functional requirements are denoted as a use case model.

Vocabulary requirements are written in the form of a domain vocabulary model.



Use case model

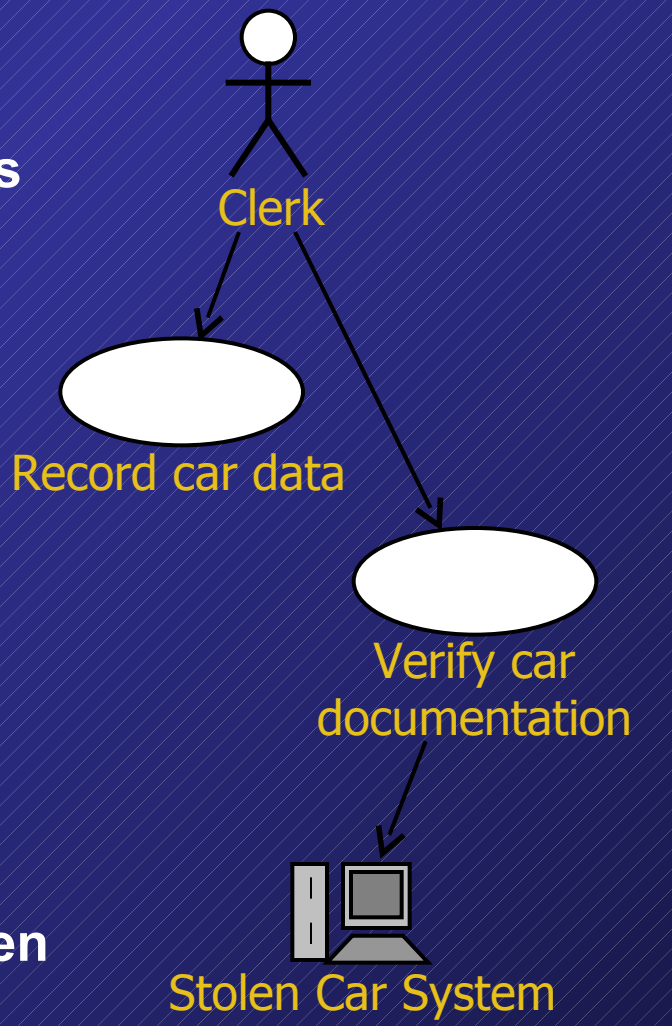
Use case model is composed of use cases and actors.

Actors denote "roles" of people or machines outside the described system.

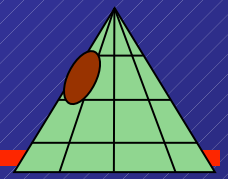
Use case is a small piece of the system's functionality that:

- begins with actor's interaction with the system,
- describes the system's "dialogue" with the actor, and
- leads to a specific goal that has a value to the actor.

Use case model shows relationships between actors and use cases.



Use case – a formal definition



Use Case

A collection of possible scenarios between the system under discussion and external actors, characterized by the goal the primary actor has toward the system's declared responsibilities, showing how the primary actor's goal might be delivered or might fail. (*A. Cockburn)*

Services provided by the system to its actors. Use cases capture the system functional requirements and may have associated performance requirements. (*Rational Unified Process, 2001*)

A kind of classifier representing a coherent unit of functionality provided by a system, a subsystem, or a class as manifested by sequences of messages exchanged among the system (subsystem, class) and one or more outside interactors (called actors) together with actions performed by the system (subsystem, class) (*UML 1.4*)

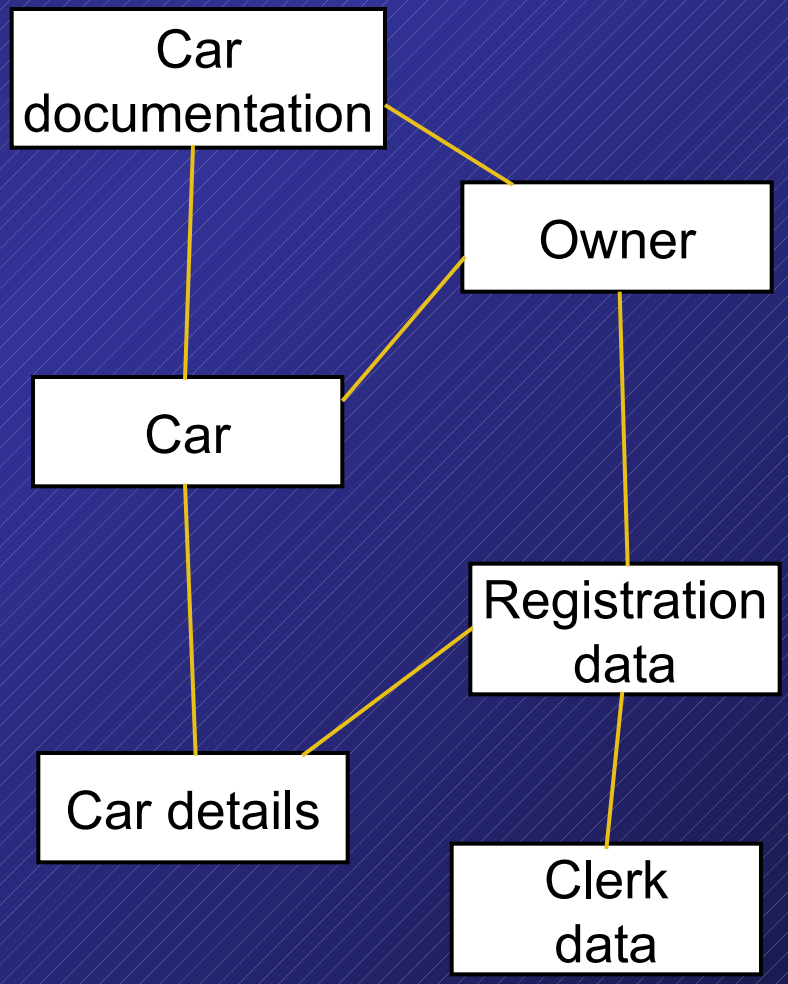
Domain vocabulary model

Domain vocabulary model is composed of notions (classes) and relationships between them.

Notions denote data that shall be handled by the system. Every notion can define a packet of several data elements (attributes).

Notion descriptions include also ways to process this data.

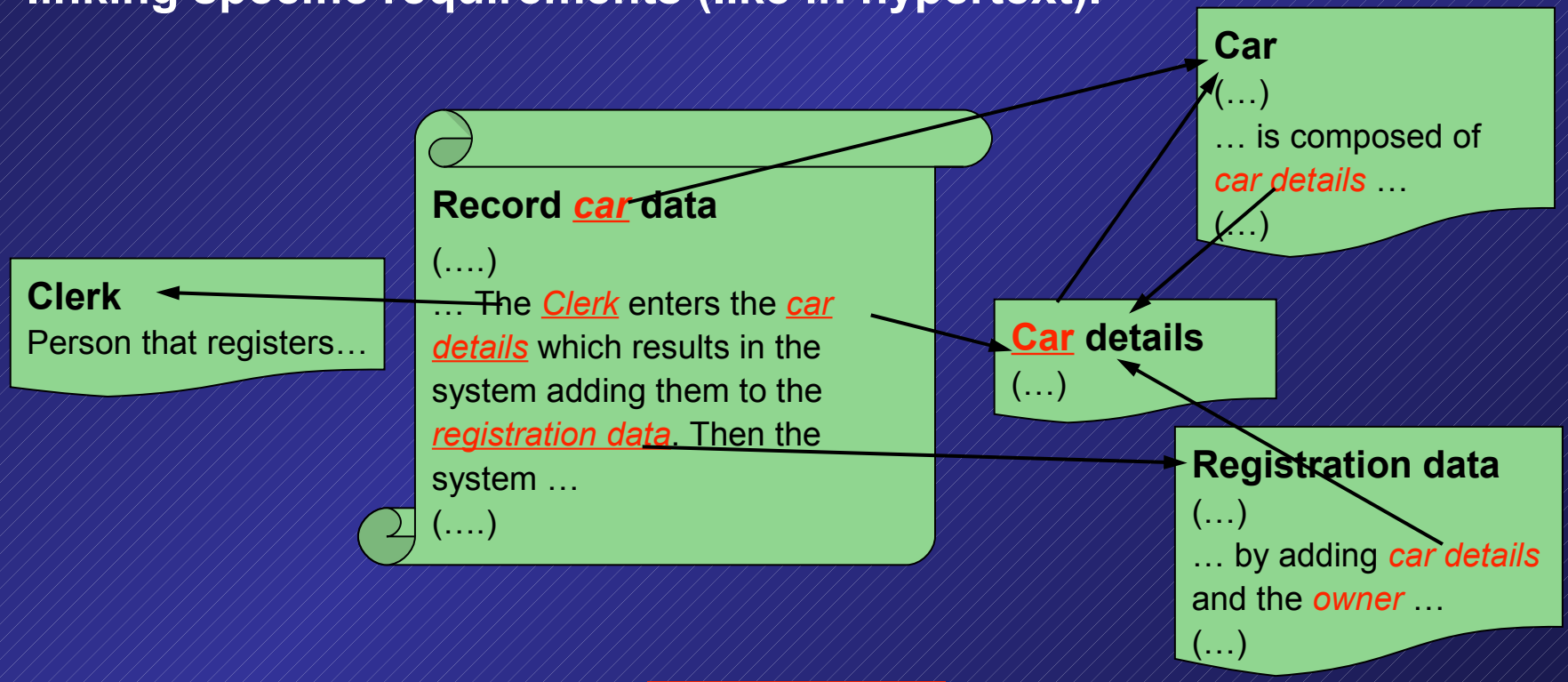
It is important to capture relationships between notions. These relationships denote to us that one notion can be described in terms of another notion.



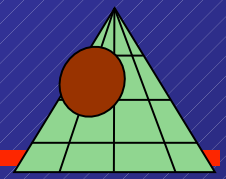
Describing actors, use cases and notions.

Every element of the system scope definition should be described with one or two paragraphs of text.

In this text we should use appropriate vocabulary entries thus linking specific requirements (like in hypertext).



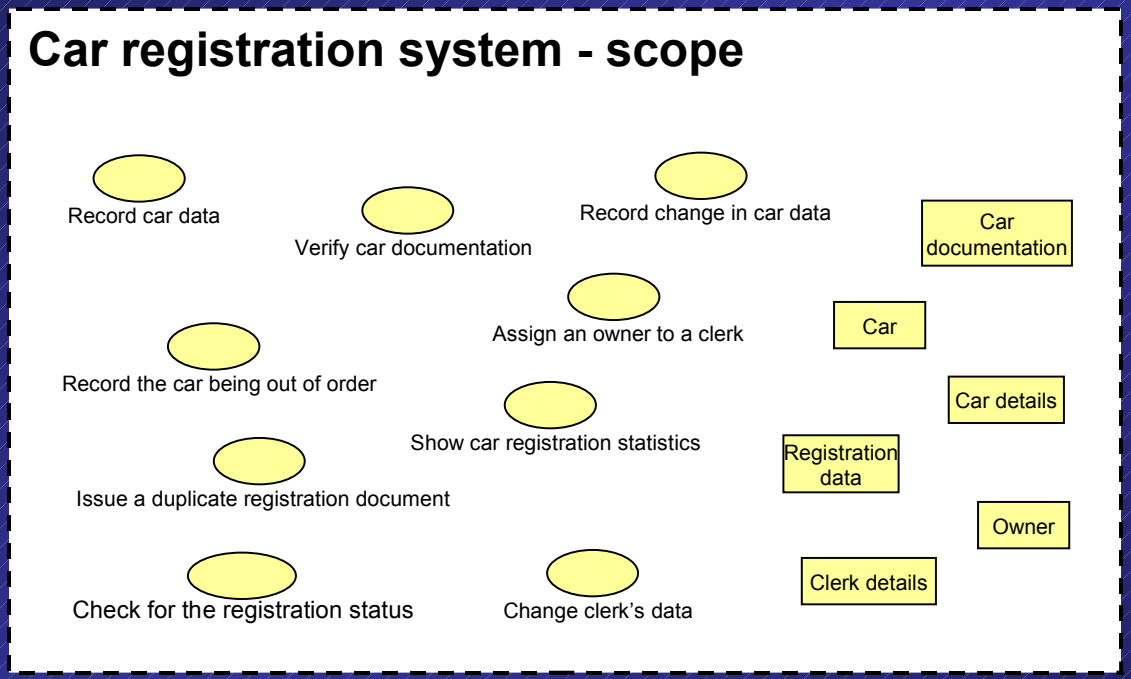
So, what is the system's scope?

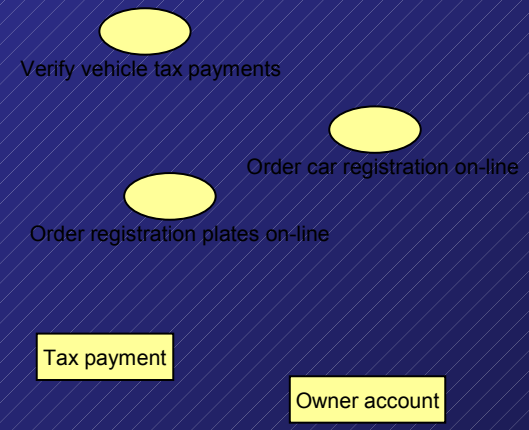


All the use cases – scope for the functionality.

All notions – scope for the handled data.

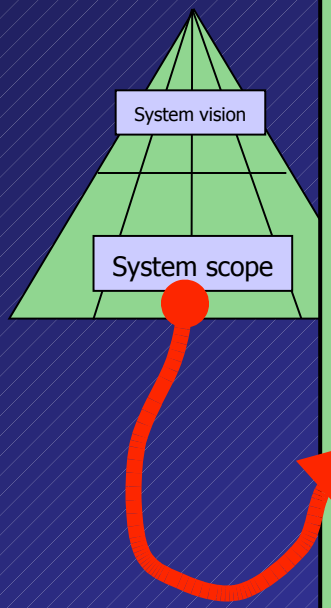
Good idea: define also use cases and notions that are out of scope (eg. will be in scope for the next version).





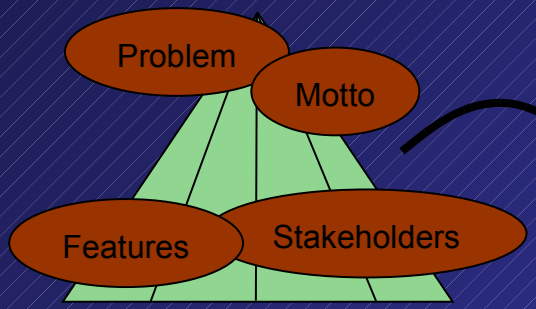
Summary – documenting it all…

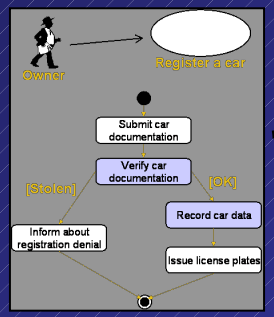
All the user requirements form a model that can be expressed in a single document: User Requirements Specification.



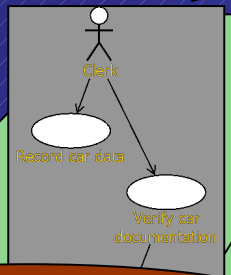
1 Introduction
(Purpose, Definitions, Overview)
2 General description
2.1 System vision
2.2 Business process description
3 Specific requirements
3.1 Functional requirements (use cases)
3.1.1 <Use case package name>
3.2 Users and external systems
3.3 Vocabulary
3.4 Important non-functional requirements
3.5 Environment constraints (technical and business)

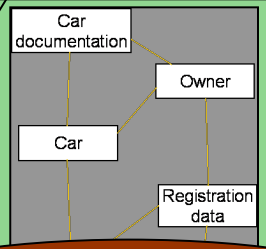
Summary – how will WE do it?





1 Introduction
(Purpose, Definitions, Overview)
2 General description
2.1 System vision
2.2 Business process description
3 Specific requirements
3.1 Functional requirements (use cases)
3.1.1 <Use case package name>
3.2 Users and external systems
3.3 Vocabulary
3.4 Important non-functional requirements
3.5 Environment constraints (technical and business)





Use case descriptions

Notion descriptions

Features

Constraints

System vision