

# DEVELOPMENT OF INFORMATION SYSTEMS

## **Lecture 11**

### **Software project management**

# Review of the last lecture

## Domain Specific Languages

Domain-specific language is a computer programming language of limited expressiveness focused on a particular domain.


1. **Computer programming language:** A DSL is used by humans to instruct a computer to do something.
2. **Language nature:** A DSL is a programming language, and as such should have a sense of fluency where the expressiveness comes not just from individual expressions but also from the way they can be composed together.
3. **Limited expressiveness:** A DSL supports a bare minimum of features needed to support its domain.
4. **Domain focus:** A limited language is only useful if it has a clear focus on a small domain. The domain focus is what makes a limited language worthwhile.

# Review of the last lecture

## Domain Specific Languages


In general a DSL construct a shared language between the domain people and programmers.

### Why DSL?

- Improving productivity for developers.
  - Improving communication with domain experts.
  - DSL is a language facade over a library or framework
- 
- Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# Review of the last lecture

## Domain Specific Languages - examples

- SQL
  - HTML
  - A programming language "library"
    - E.g. **DateTime** and support for date and time tasks.
  - UML diagram of a State machine.
  - Regular expression
  - Configuration file, XAML
- 
- Several white lines of varying lengths and orientations are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

# Review of the last lecture

## Categories of Domain Specific Languages

- **External DSL** - Language separate from the main language of the application it works with. Usually, has a custom syntax, but using another language's syntax is also common (e.g. XML config files, SQL, regular expressions).
- **Internal DSL** - A particular way of using a general-purpose language. A script in an internal DSL is valid code in its general-purpose language, but only uses a subset of the language's features in a particular style to handle one small aspect of the overall system. (Ruby, Lisp)
- **Language workbench** - A specialized IDE for defining and building DSLs. Is used not just to determine the structure of a DSL but also as a custom editing environment for people to write DSL scripts.

# Review of the last lecture

## Benefits of using DSLs

- **Productivity** - You can replace a lot of GPL code with a few lines of DSL code.
- **Quality** - fewer bugs, better architectural conformance, increased maintainability.
- **Validation and Verification** - programs are more semantically rich than GPL programs. Analyses are much easier to implement,
- **Data Longevity** - models are independent of specific implementation techniques.
- **Productive Tooling** – external DSLs can come with tools, i.e. IDEs that are aware of the language.

# Review of the last lecture

## Summarizing - why use Domain-Specific Languages?

- DSLs are very good at taking certain narrow parts of programming and making them easier to understand and therefore quicker to write, quicker to modify, and less likely to breed bugs.
- Since DSLs are smaller and easier to understand, they allow nonprogrammers to see the code that drives important parts of their business.
- Communication between programmers and their customers is the biggest bottleneck in software development, so any technique that can address it is worth its weight in single malts.



# Review of the last lecture

## Problems of DSL

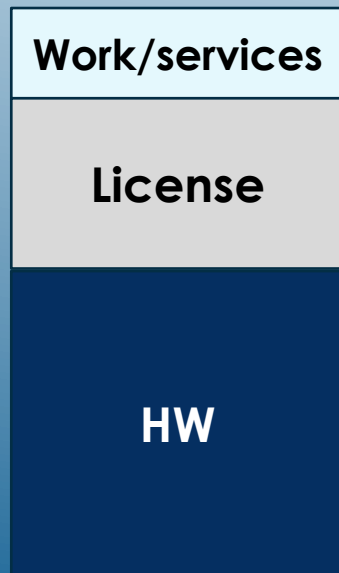
- **Language Cacophony**
  - Using many languages will be much more complicated than using a single one.
- **Cost of Building**
  - The cost of a DSL is the cost over the cost of building the model.
- **Ghetto Language**
  - Company that's built a lot of its systems on an in-house language which is not used anywhere else.
- **Blinkered Abstraction (Tunnel vision)**
  - DSL It allows you to express the behavior of a domain much more easily than if you think in terms of lower-level constructs.



# Enterprise Information Systems

Consequences of technological development and growth

**Before**



**Now**



# ERP - Enterprise Resource Planning

Type of software that organizations use to manage day-to-day business activities such as:

- accounting,
- procurement,
- project management,
- risk management and compliance,
- and supply chain operations.

A complete ERP suite also includes enterprise performance management, software that helps plan, budget, predict, and report on an organization's financial results.

It is based on the idea that every piece of information is entered into the system only once and is stored in only one place.


SAP, Navision, Axapta, Oracle EBS, JD Edwards.

# CRM - Customer Relationship Management

CRM systems allow you to collect, sort and process customer data, especially their contacts, ongoing business processes (marketing, sales, service...) and the achieved sales.

Allows customers to be not only maintained, but also find out their wishes, needs and buying habits

New tools are most needed has become more evident with the emergence of call centres and the the development of e-business.

Three white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, extending from the right edge towards the center.

# SCM - Supply Chain Management

Management strategies to optimise all activities and systems to ensure delivery products and services from raw material suppliers through their production or development, through distribution channels to the end consumer.

Replaces separate production planning, distribution and sales by optimizing the entire chain.

The manufacturer of the end product is vitally dependent on an error-free and time-accurate flow of materials and components.

Three parallel white lines of varying lengths are positioned in the bottom right corner of the slide, angled upwards from left to right.

# APS - Advanced Planning and Scheduling

APS combines forward and reverse planning and scheduling combined with capacity-constrained planning and designation bottleneck

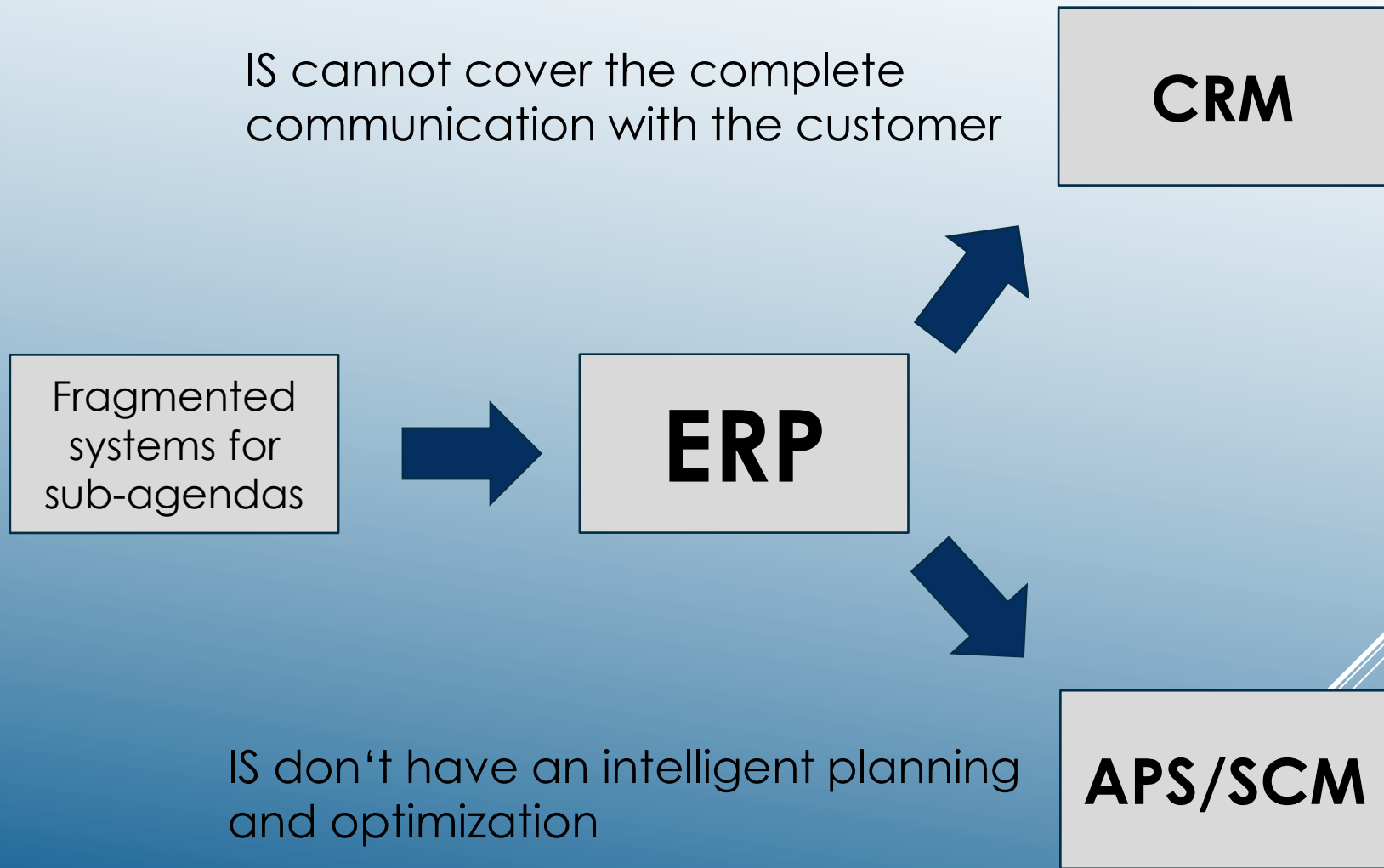
Planning is based on bottleneck management - bottlenecks are identified and managed

E.M.Goldratt - Theory of Constraints:

Manage your constraints and don't allow constraints manage us

"We can't do it because...." will change to "We can do it if....."

# Evolution of basic types of Enterprise IS



# Parameterizable information system

The reason for the origin:

- Development cost savings
- Reduction of project duration
- Transfer of process know-how

Possibility of local modifications of the system (IS are usually delivered with source code)

Dilemma:

- Trying to cover as many variants and possibilities as possible
- The system becomes more complex and therefore more error-prone, more difficult to maintain and especially more difficult to understand/operate for end customers.



# What does an Enterprise IS consist of?

## IT perspective:


- HW, OS, DB,
- Communications
- Applications
- Tables
- Programs
- Procedures
- Scripts
- Interface

## Business perspective:

Processes

# Type of Enterprise processes

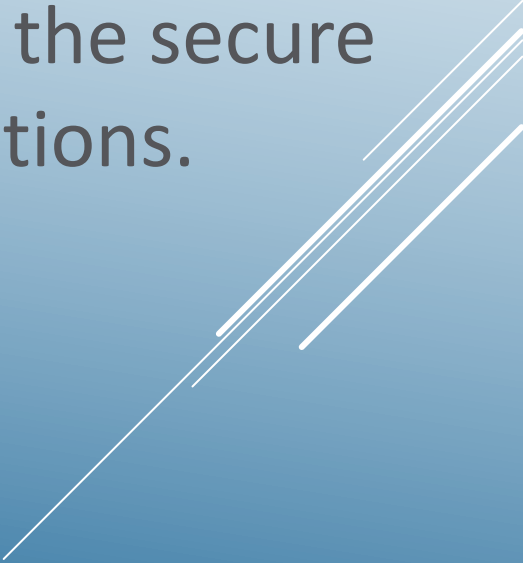
A process is an ordered set of steps that transform defined inputs into the desired outputs

- Core processes - "core business" or "business critical" - create added value
  - Supporting/servicing processes - processes that a company needs to operate
  - The same process may be core in one company and only support/services in another
  - Outsourcing
- 

# Integration process

Business Process Integration (BPI) is essential for businesses looking to connect systems and information efficiently.

BPI allows for automation of business processes, integration of systems and services, and the secure sharing of data across numerous applications.

Several white lines of varying lengths and angles are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# Integration by the File transfer

The oldest method of integration

One application puts a file in a directory, the other reads it and archives the file at the same time

- Controlled by a process or batch
- Mainly used between solutions on different platforms
- Can use a variety of files - flatfile, XML, Dasta, HL7....
- Includes EDI

# Integration by the Copy of data

- A copy of the table, part of the table or of a database view
- It is controlled by a process or batch
- A subset or narrower variant is direct replication of the database
- It runs online at the database level, it is its native process
- Both solutions to be linked must be on the same platform

# Integration by the Web service

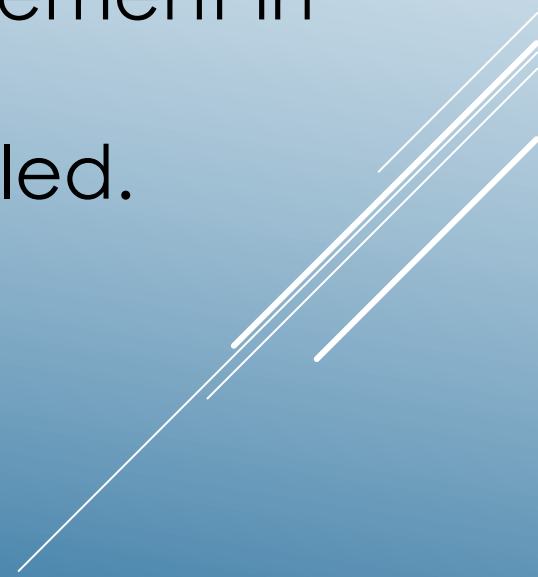
The latest variant of the integration methods

- Allows you to call a function on a remote system
- Web service can have multiple methods
- Not dependent on the platforms of the systems being linked
- A more sophisticated solution can be developed based on this principle - the bus :
  - Multiple source IS
  - Multiple target IS

# Software project management

Software project management is an art and science of planning and leading software projects.

It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled.

Several thin, white, parallel diagonal lines are positioned in the bottom right corner of the slide, extending from the right edge towards the bottom.



# Why do software projects fail?

**People begin programming before they understand the problem:**

- Everyone likes to feel that they're making progress
- When the team starts to code as soon as the project begins, they see immediate gains
- When problems become more complex (as they always do!), the work gets bogged down
- In the best case, a team that begins programming too soon will end up writing good software that solves the wrong problem


# Why do software projects fail?

**The team has an unrealistic idea about how much work is involved.**

- From far away, most complex problems seem simple to solve
- Teams can commit to impossible deadlines by being overly optimistic and not thinking through the work
- Few people realize the deadline is optimistic until it's blown

# Why do software projects fail?

**Defects are injected early but discovered late.**

- Projects can address the wrong needs
  - Requirements can specify incorrect behavior
  - Design, architecture and code can be technically flawed
  - Test plans can miss functionality
  - The later these problems are found, the more likely they are to cause the project to fail
- 

# Why do software projects fail?

Programmers have poor habits – and they don't feel accountable for their work.

- Programmers don't have good control of their source code
- Code written by one person is often difficult for another person to understand
- Programmers don't test their code, which makes diagnosing and fixing bugs more expensive
- The team does not have a good sense of the overall health of the project.

# Why do software projects fail?

Managers try to test quality into the software.

- Everyone assumes that the testers will catch all of the defects that were injected throughout the project.
- When testers look for defects, managers tell them they are wasting time.
- When testers find defects, programmers are antagonized because they feel that they are being personally criticized.
- When testers miss defects, everyone blames them for not being perfect.

# How can we make sure that our projects succeed?

Make sure all decisions are based on openly shared information

- It's important to create a culture of transparency, where everyone who needs information knows where to find it and is comfortable looking at it.
- All project documents, schedules, estimates, plans and other work products should be shared with the entire team, managers, stakeholders, users and anyone else in the organization who wants them.
- Major decisions that are made about the project should be well-supported and explained

# How can we make sure that our projects succeed?

Don't second-guess your team members' expertise

- Managers need to trust team members.
- Just because a manager has responsibility for a project's success, it doesn't mean that he's more qualified to make decisions than the team members.
- If you don't have a good reason to veto an idea, don't



# How can we make sure that our projects succeed?

Introduce software quality from the very beginning of the project

- Review everything, test everything.
- Use reviews to find defects – but don't expect the review to be perfect.
- Use reviews to gain a real commitment from the team.
- It's always faster in the long run to hold a review than it is to skip it.

# How can we make sure that our projects succeed?

Don't impose an artificial hierarchy on the project team

- All software engineers were created equal.
- A manager should not assume that programming is more difficult or technical than design, testing or requirements engineering.
- Managers should definitely not assume that the programmer is always right, or the tester is always raising false alarms

# How can we make sure that our projects succeed?

Remember that the fastest way through the project is to use good engineering practices

- Managers and teams often want to cut important tasks – especially estimation, reviews, requirements gathering and testing.
- If it were faster to build the software without these practices, we would never use them.
- Every one of these practices is about saving time and increasing quality by planning well and finding defects early. Cutting them out will cost time and reduce quality.

# MANAGEMENT AND LEADERSHIP



# RESPONSIBILITY, AUTHORITY AND ACCOUNTABILITY


A person has *responsibility* for a task if:

- He is given sufficient *authority* to perform it
- He is *accountable* for its completion
- A person has *authority* to perform a task only if he has adequate control over the resources necessary to complete the task.
- A person is *accountable* for a task if failure to adequately perform that task carries professional consequences.

# Delegation


Delegation is assigning responsibility of a task to a team member.

When delegating a task, the project manager must ensure that the team member has the authority to perform it and is accountable for the results

Several thin, white, parallel diagonal lines are located in the bottom right corner of the slide, extending from the right edge towards the bottom.

# Transparency


When the project manager creates a document, holds a meeting of interest to others, or makes an important project decision, all of the information produced should be shared and used with everyone involved in the project

Three parallel white lines of varying lengths are positioned in the bottom right corner of the slide, slanted diagonally upwards from left to right.



# Transparency

All work products should be public

- All team members, senior managers and stakeholders should have access to every work product produced for the project.
  - Project managers and team members benefit because they make more informed decisions.
  - Senior managers and stakeholders are always kept informed
- 


# Transparency

Decisions should be made based on known guidelines

- Published *standards documents* help others understand the way certain roles must be filled.
- Documents should be based on *templates* when possible.
- *Process documents* ensure that each project is done using a repeatable process.
- Use *performance plans* to set expectations for individual team members.

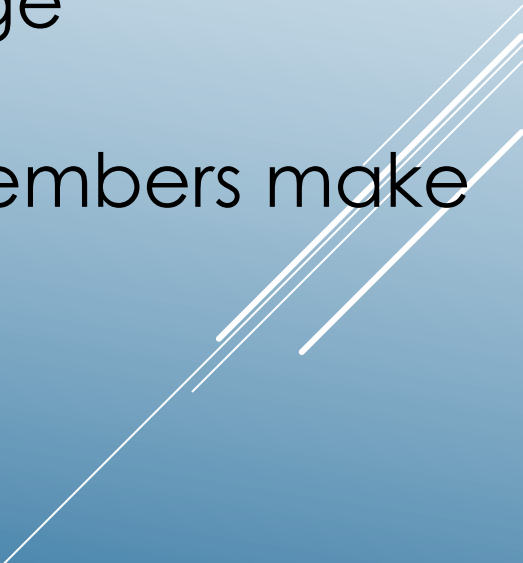
# Manage Your Team

Avoid common management pitfalls

- Don't manage from your gut.
  - Don't second-guess estimates.
    - Remember Brooks' Law: "Adding manpower to a late software project makes it later."
  - Don't expect consensus all of the time.
  - Make your mistakes public.
  - Accept criticism
- 

# Manage Your Team

Avoid micromanagement

- Don't expect to review everything
  - Don't fall into the “hands-on manager” trap
  - Use transparency to your advantage
  - Don't be afraid to let your team members make mistakes
- 
- Three parallel white lines of varying lengths are positioned in the bottom right corner of the slide, angled upwards from left to right.


# Manage Your Team

Address performance problems early

- Work with each team member to develop a performance plan.
- Set standards that are fair and attainable.
- Measure each team member's progress against known and agreed-upon goals.
- Correct performance problems as early as possible

# Requirements vs. Design

Many people have difficulty understanding the difference between scope, requirements and design.

- **Scope** demonstrates the needs of the organization, and is documented in a vision and scope document
  - **Requirements** document the behavior of the software that will satisfy those needs
  - **Design** shows how those requirements will be implemented technically
- 

# Change control

Change control is a method for implementing only those changes that are worth pursuing, and for preventing unnecessary or overly costly changes from derailing the project.

- Change control is an agreement between the project team and the managers that are responsible for decision-making on the project to evaluate the impact of a change before implementing it.
- Many changes that initially sound like good ideas will get thrown out once the true cost of the change is known

# Change control

A change control board (CCB) is made up of the decision-makers, project manager, stakeholder or user representatives, and selected team members.

- The CCB analyzes the impact of all requested changes to the software and has the authority to approve or deny any change requests once development is underway.
- Before the project begins, the list of CCB members should be written down and agreed upon, and each CCB member should understand why the change control process is needed and what their role will be in it



# Change control

Whenever a change is needed, the CCB follows the change control process to evaluate the change:

- The potential benefit of the change is written down, and the project manager works with the team to estimate the potential impact that the change will have on the project.
- If the benefit of the change is worth the cost, the project manager updates the plan to reflect the new estimates.
- Otherwise, the change is thrown out and the team continues with the original plan.
- The CCB either accepts or rejects the change

# Software Requirements Specification

The software requirements specification (SRS) represents a complete description of the behavior of the software to be developed.

The SRS includes:

- A set of use cases that describe all of the interactions that the users will have with the software.
- All of the functional requirements necessary to define the internal workings of the software: calculations, technical details, data manipulation and processing, and other specific functionality that shows how the use cases are to be satisfied
- Nonfunctional requirements, which impose constraints on the design or implementation (such as performance requirements, quality standards or design constraints).

# Semestral project -Artifact 7

## Presentation your IS

Present your developed IS, which consists of:

- three implemented use cases, include alternative flows.
- two different graphical user interfaces
- two different data repositories
- complete documentation (one pdf A1-A6)

Presentation will be at the last exercise on 15.12.2021

# Final test information

The final test will take place in January 2022

The exact date of the final test will be published in the school information system within the next 14 days.

The final test is a written exam with 6 open questions.

The test questions cover the areas of IS development presented in the lectures and questions from the lectures.

The maximum score for the final test is 55 points. A score of at least 50% is required to pass the final test.

# Exercise tasks

- Present your current state of semestral work code.
- Continue the implementation the semester project.