

# Software technology

## 02 - Software Development Process and Life Cycle

Dr. Attila Gludovátz

# Online catalog - every week

- <https://catalog.inf.elte.hu/>
- Log in
- Username: yourUsername ( @inf.elte.hu )
- Password: your email password
- Captcha: I generate a number for you...
- Lecture attendance is **not** optional! Max 3 misses and you are out

# What is a Software?

- The goal is to make a Software, but is it a ...

- Product?



- Service?



- Infrastructure?



# What is a Software?

- Historically Software is most often handled as a **Product** despite having very different features
  - No value loss in copying, different – potentially longer life cycle, distributed nature, sometimes data is more valuable...
- But business model desires product (probably the easiest way to make money), so we have to productize whatever it is
  - Planned life cycle, planned obsolescence, put it in a box, write features on it

# What is a Software?

- **Software As A Service (SaaS)** is a more modern approach but is still usually serves mainly as a business model
  - Changing in billing and place of deployment
- Open Source community handles software more as an Infrastructure where multiple parties finance the development and maintenance of a common asset



# History of Product Handling and Optimization

- We force software to be a Product →
  - Division of Labor
  - Mass Production
  - Interchangeable Parts
- Specialization →
  - **The Assembly Line / Production Line**
- Predictability



# History of Product Handling and Optimization

- Software is handled as a product traditionally → All different assembly line techniques are used (or forced)
- But these techniques were not 100% success:
  - Repetitive
  - Stress
  - Injuries
  - Boredom
  - Alienation (Entfremdung)

Products should be „mirrors in which workers see their reflected essential nature.“ Karl Marx



# Assembly Line Features

- Consists of Phases
  - Phase uses Results of Previous Phases
- High Quality (involves separate QA phases)
  - QA checks, but quality comes from ...?
- Based on Customer Requirements
- Predictably meets
  - Time Schedules
  - Cost Estimates



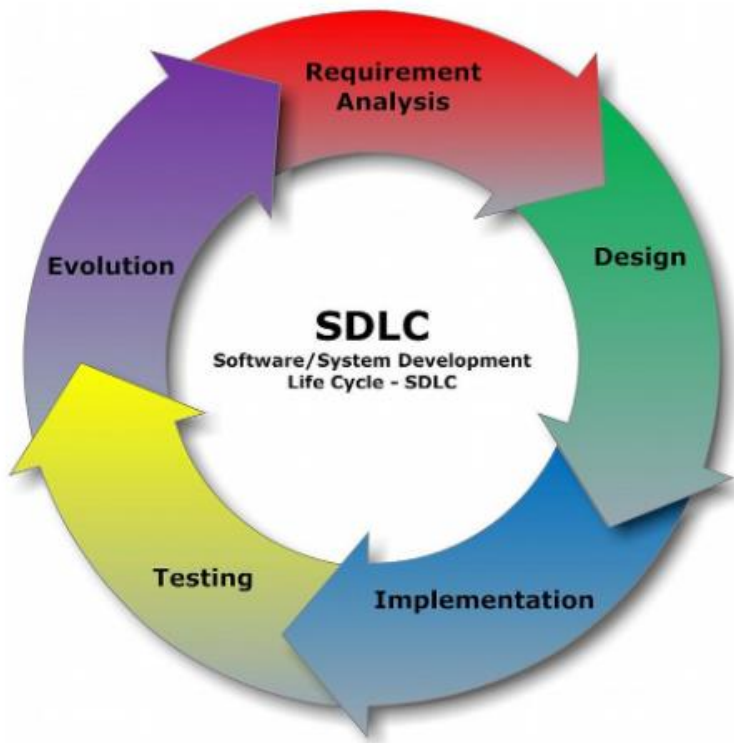
# Assembly Line Features

- Key features are what make the process manageable:
  - Predictability
  - QA
- Customer requirements would be important but unfortunately in practice can also be invoked via marketing...



# Software/System Development Life Cycle (SDLC)

# Software/System Development Life Cycle (SDLC)



- Software Development Life Cycle (SDLC) is a process used by the software industry to **design, develop and test high quality software**
- The SDLC aims to produce a high-quality software that *meets customer expectations*, reaches completion within times and cost estimates
  - It is also called as Software Development Process
  - SDLC is a framework defining tasks performed at each step in the software development process
  - ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software



# Software/System Development Life Cycle

## **1. Initiation**

- Begins when a sponsor identifies a need or an opportunity
- Concept Proposal is created

## **2. System Concept Development**

- Defines the scope or boundary of the concepts
- Includes Systems Boundary Document
- Cost Benefit Analysis
- Risk Management Plan and Feasibility Study



# Software/System Development Life Cycle

## **3. Planning**

- Develops a Project Management Plan and other planning documents
- Provides the basis for acquiring the resources needed to achieve a solution

## **4. Requirements Analysis**

- Analyses user needs and develops user requirements
- Create a detailed Functional Requirements Document





# Software/System Development Life Cycle

## **5. Design**

- Transforms detailed requirements into complete, detailed Systems Design Document focuses on how to deliver the required functionality

## **6. Development**

- Converts a design into a complete information system, includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities

# Software/System Development Life Cycle

## **7. Integration and Test**

- Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document
- Conducted by Quality Assurance staff and users
- Produces Test Analysis Reports

## **8. Implementation**

- Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

# Software/System Development Life Cycle

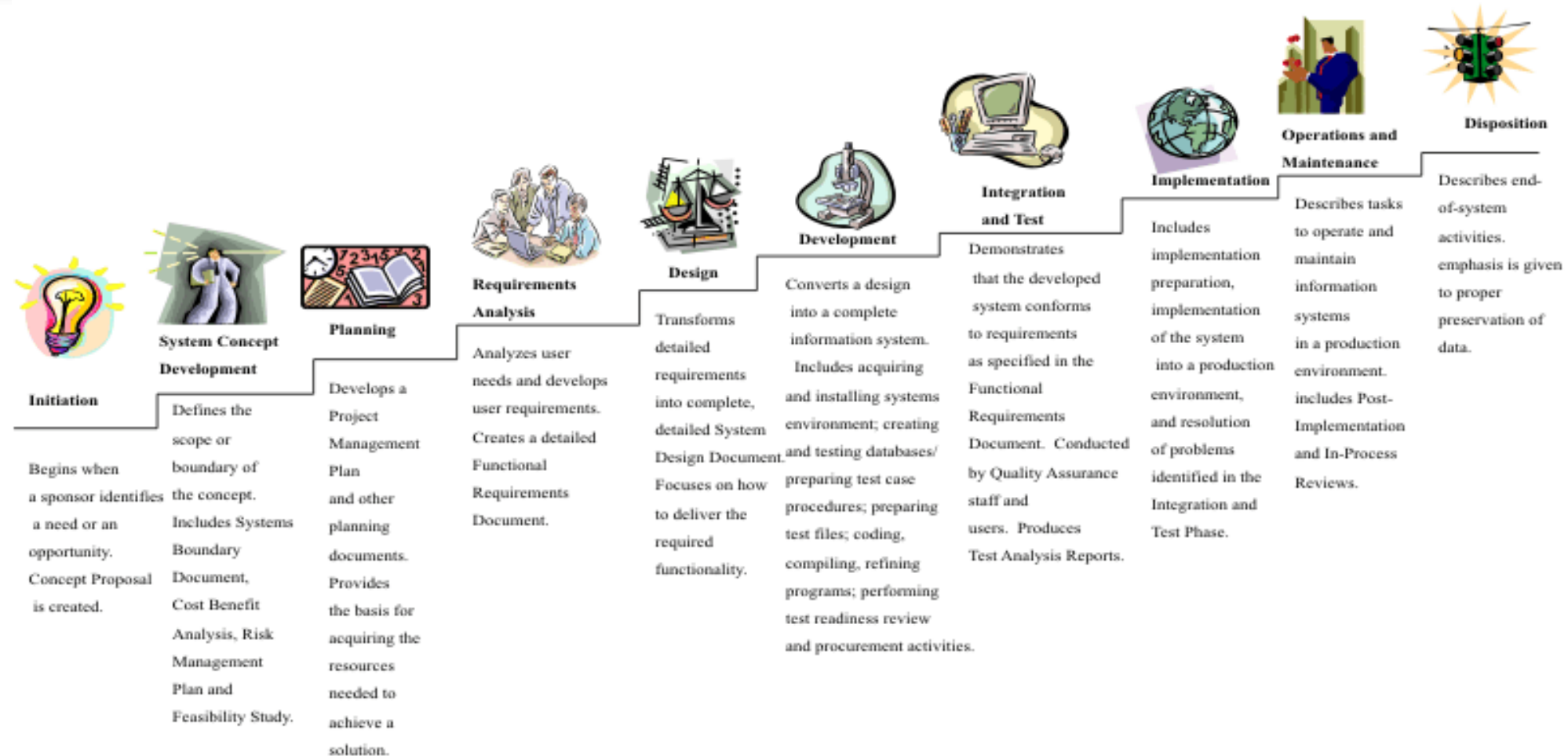
## **9. Operations & Maintenance**

- Describes tasks to operate and maintain information systems in a production environment includes
- Post-Implementation and In-Process Reviews

## **10. Disposition**

- Describes end-of-system activities, emphasis is given to proper preparation of data

# Software/System Development Life Cycle Summary






# Testing (Quality Assurance - QA)



# Testing (Quality Assurance - QA)

- When a software or an application is created, it is vital to make several types of tests, to make sure the product is complete, secure and efficient






# Testing (QA)

## Unit testing

- **Unit testing** is a software testing method by which individual units of source code – sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures – are tested to determine whether they are fit for use
- Unit testing is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing and even Beta Testing after application is built
- If proper unit testing is done in early development, then it saves time and money in the end
- ISTQB definition: The testing of individual software components



# Testing (QA)

## Unit testing

- Unit testing, also known as *component testing*, is a level of software testing where individual units / components of a software are tested
- The purpose is to validate that each unit of the software performs as designed
- A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output
- In object-oriented programming, the smallest unit is a method, which may belong to a base / super class, abstract class or derived / child class
  - Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module
- Unit testing frameworks, drivers, stubs, and mock / fake objects are used to assist in unit testing

# Testing (QA)

## Integration (+interface) testing

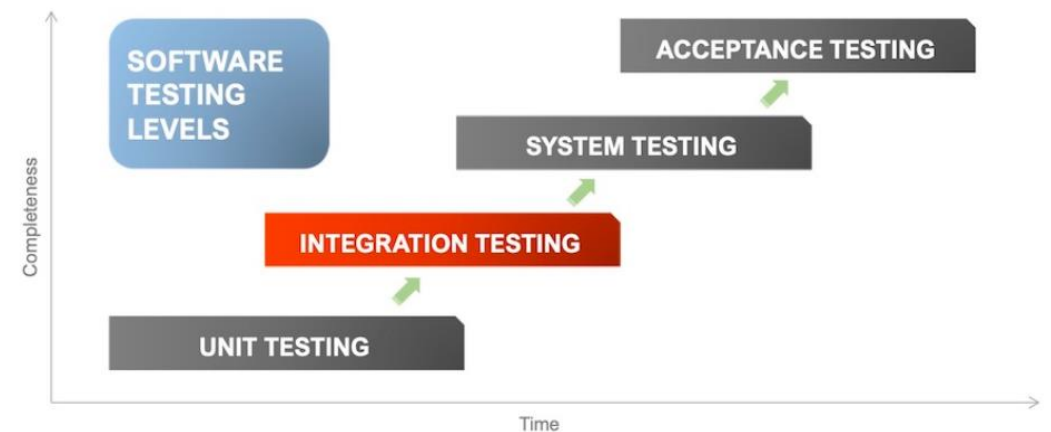
- Integration test is a level of software testing where individual units are combined and tested as a *group*
- The purpose of this level of testing is to expose faults in the interaction between integrated units
- Test drivers and test stubs are used to assist in Integration Testing



# Testing (QA)

## Integration (+interface) testing

- ISTQB definitions:
  - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
  - Component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components
  - System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet)







## Testing (QA)

### Integration (+interface) testing

- Any of **Black Box Testing**, **White Box Testing** and **Gray Box Testing** methods can be used (integration testing)
- Normally, the method depends on your definition of 'unit' and what exactly you are integrating
- The test can be either *manual* or *automated*
- Developers themselves or independent testers perform Integration Testing

# Testing (QA)

## White-box / Black-box / Gray-box testing



- **"White box"** tests consist in reviewing the functioning of an application and its internal structure, its processes, rather than its functionalities. Here, all of the internal components of the software or application are tested through the source code, main work base of the tester.
- **"Black box"** tests consist in reviewing *only* the functionalities of an application, i.e. if it does what it is supposed to, no matter how it does it. Its internal structure and functioning are not studied. The tester thus needs to know what the role of the system is, and its functionalities, but does not know its internal mechanisms. He has a "user" profile.
- **"Grey box"** testing compiles the two previous approaches: they test both the functionalities and functioning of a software. That means that a tester gives an input to the system, checks that if result is what is expected, and checks through which process this result was obtained

# Testing (QA)

## System testing

- System testing is a level of software testing where a complete and integrated software is tested
- The purpose of this test is to evaluate the system's compliance with the specified requirements
- ISTQB definition: The process of testing an integrated system to verify that it meets specified requirements



# Testing (QA)

## System testing

- Usually, Black Box Testing method is used
- Tests are normally done manually but the trend of test automation, specially for Regression Testing, is picking up
- System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing
- Normally, independent software testers perform System Testing



# Testing (QA)

## System testing - Types

- System Testing is the most comprehensive level of testing and many types of tests are performed. Some of them are mentioned below:
  - Smoke Testing to ensure that the most important functions work and to decide whether the build is fit for further testing
  - Functional Testing to ensure that features work as per the functional requirements
  - Regression Testing to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it
  - Usability Testing to determine if the system is easily usable from an end-user's perspective
  - Performance Testing to determine how the system performs in terms of responsiveness and stability under a certain load
  - Security Testing to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders
  - Compliance Testing to determine the compliance of the system with internal or external standards



# Testing (QA)

## Acceptance testing

- Acceptance testing is a level of software testing where a system is tested for acceptability
- The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery (or writing that big check)
- ISTQB Definition: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system



# Testing (QA)

## Acceptance testing

- **Method:**

- Acceptance Testing normally uses the Black Box Testing method and is executed manually. Mostly, the testing does not follow a strict procedure and is not scripted but is rather ad-hoc.

- **When is it performed?**

- Acceptance Testing is the fourth and last level of software testing performed after System Testing and before making the system available in production for actual use.



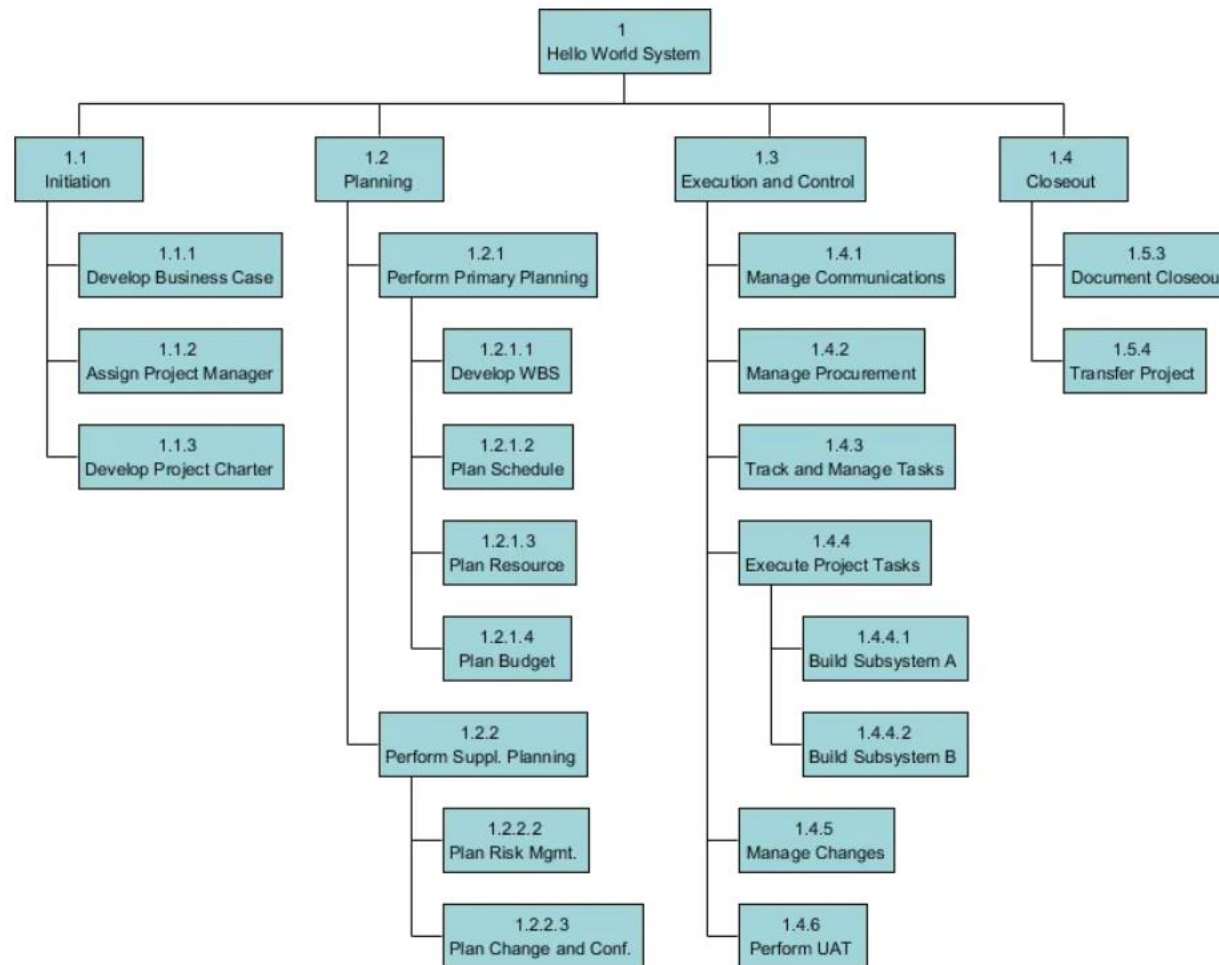
# Testing (QA)

## Acceptance testing - Who performs it?

### Alpha / Beta Testing

- **Internal Acceptance Testing** (also known as *Alpha Testing*) is performed by members of the organization that developed the software but who are not directly involved in the project (Development or Testing)
  - Usually, it is the members of Product Management, Sales and/or Customer Support
- **External Acceptance Testing** is performed by people who are not employees of the organization that developed the software
  - Customer Acceptance Testing is performed by the customers of the organization that developed the software. They are the ones who asked the organization to develop the software. (This is in the case of the software not being owned by the organization that developed it.)
  - User Acceptance Testing (also known as *Beta Testing*) is performed by the end users of the software. They can be the customers themselves or the customers' customers

# Work Breakdown Structure (WBS)



# Work Breakdown Structure (WBS)

- Measurement and control are important, but can only be done on manageable parts
- **WBS**
  - Hierarchical (**Tree**) structure
  - Contains the whole **Scope** of the Project
  - Only tells scope, not schedule or detailed execution plan (resource allocations)
  - Breakdown is **Complete** (parent element equals all leaves added)

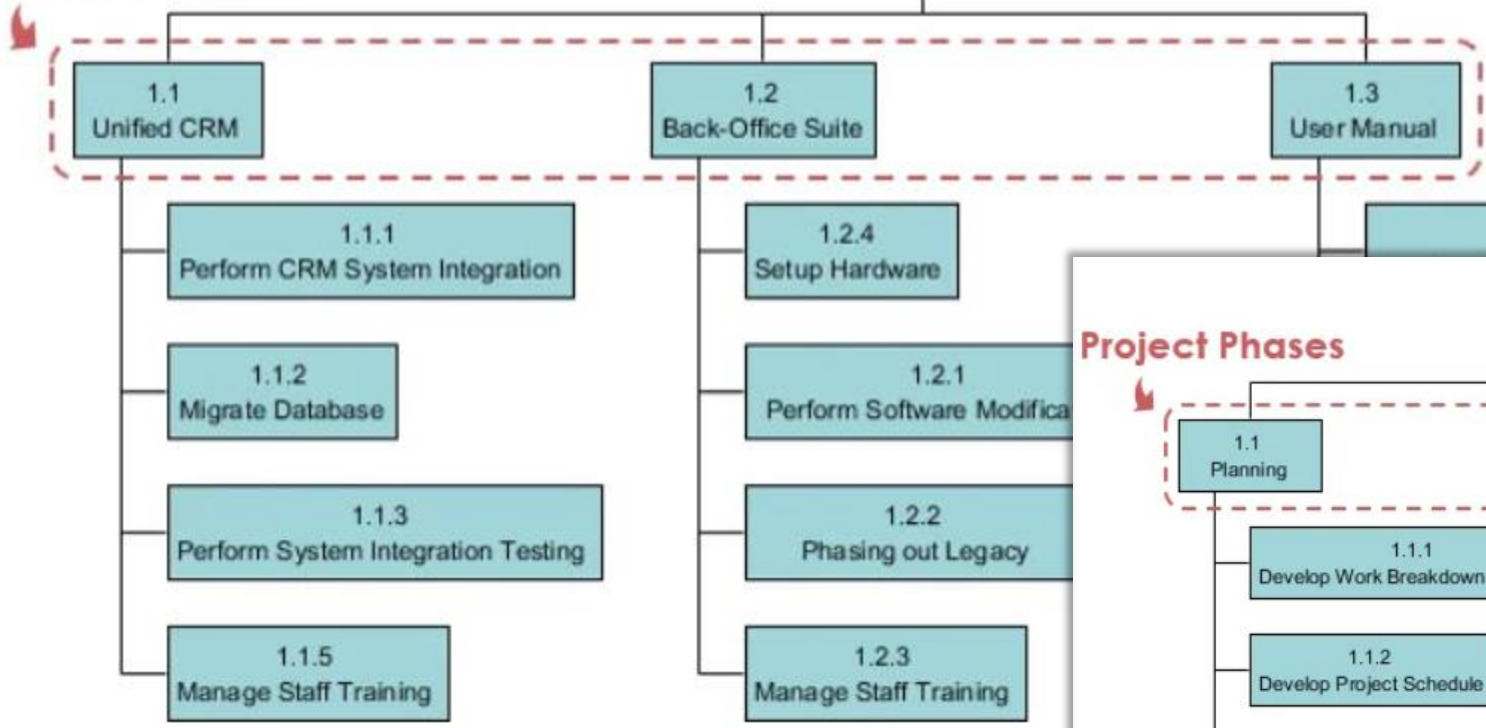
# Work Breakdown Structure (WBS)

- Declares acceptance criteria / **outcomes**, not actions
- Breaks down until work unit
  - Can be reliably **estimated**
  - Is **measurable**
  - Fits in a reporting period

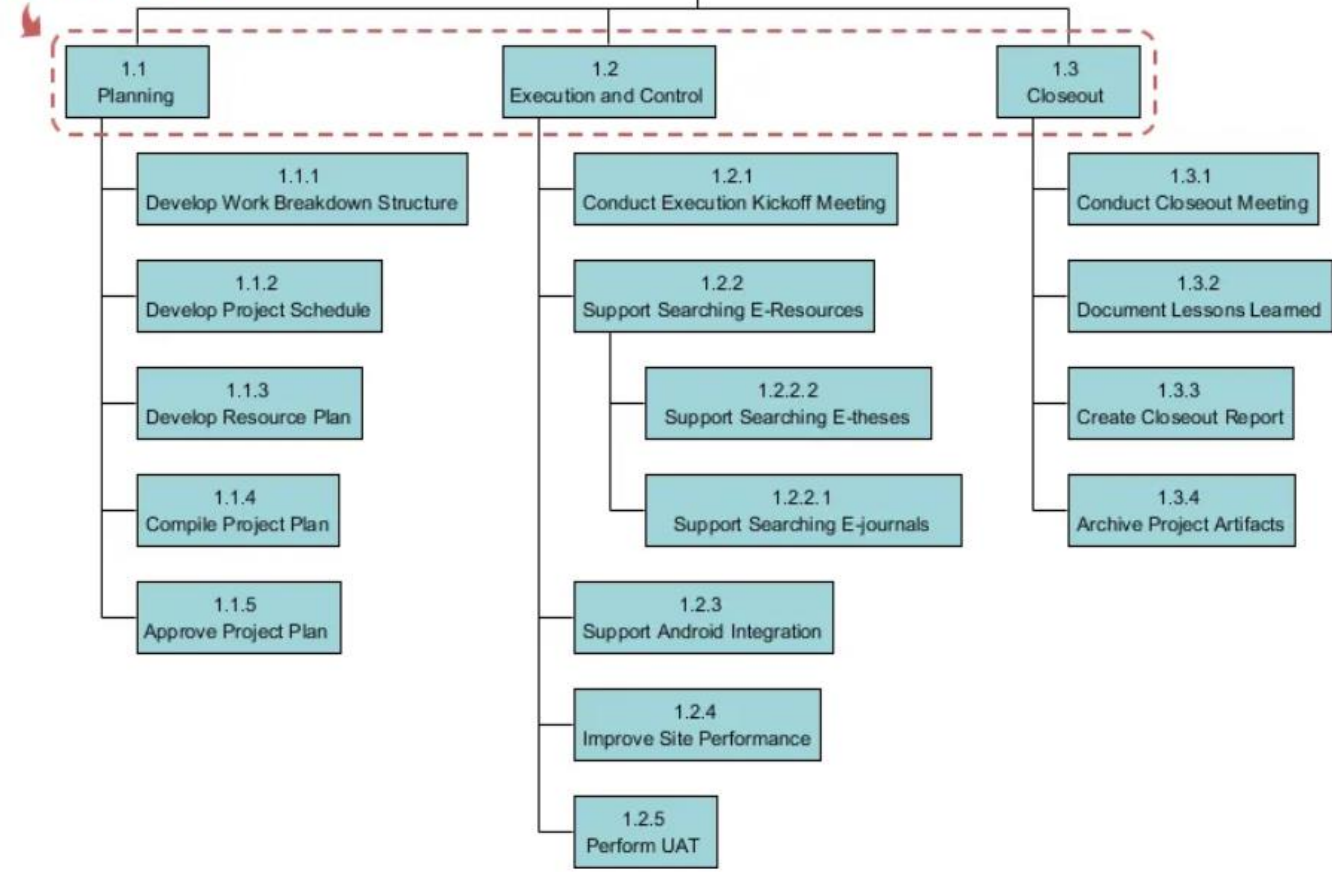
# Work Breakdown Structure (WBS)

- There are two types of WBS:
  - 1) Deliverable-Based (most common and preferred) and
  - 2) Phase-Based
- The main difference between the two approaches are the **Elements** identified in the first Level of the WBS

Deliverables



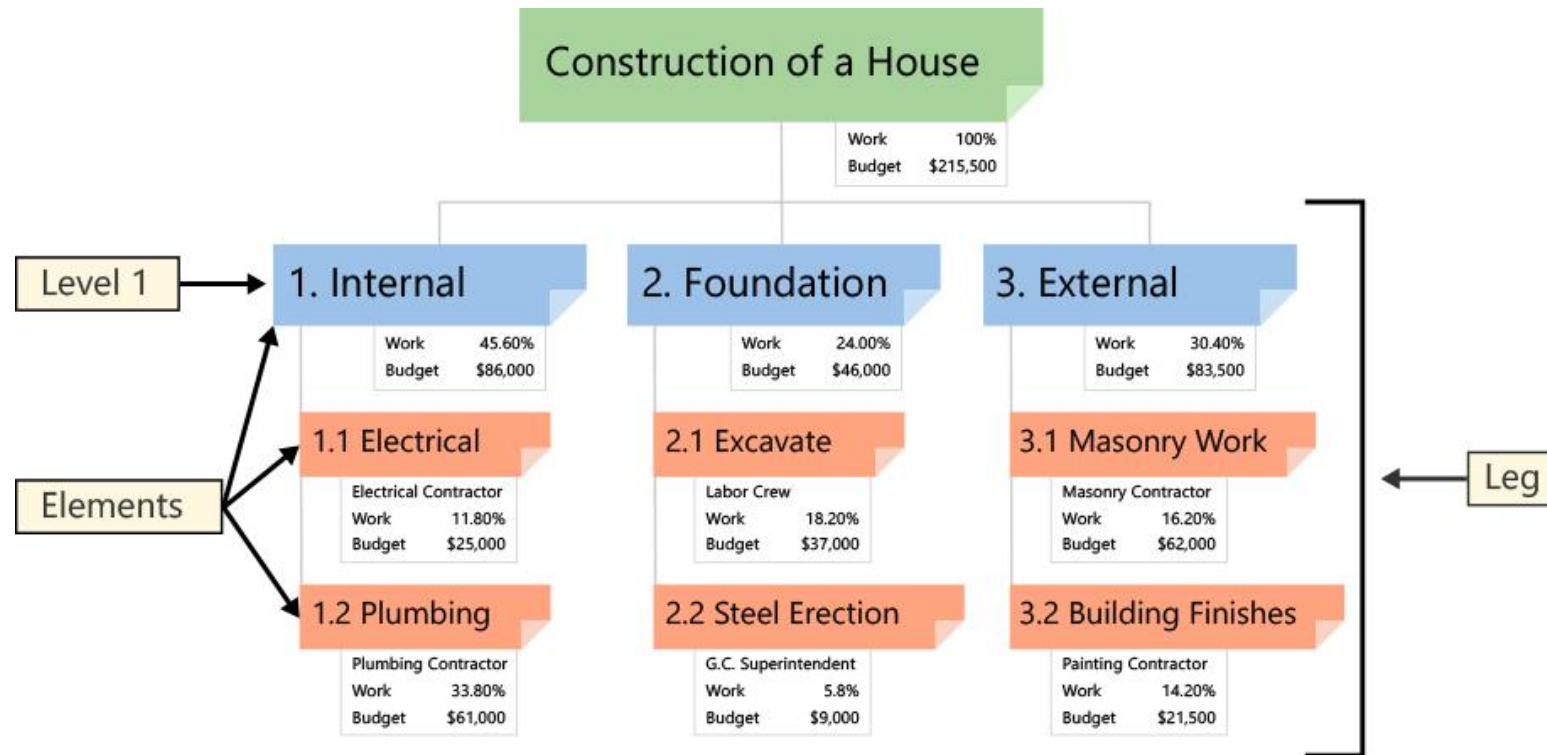
Project Phases



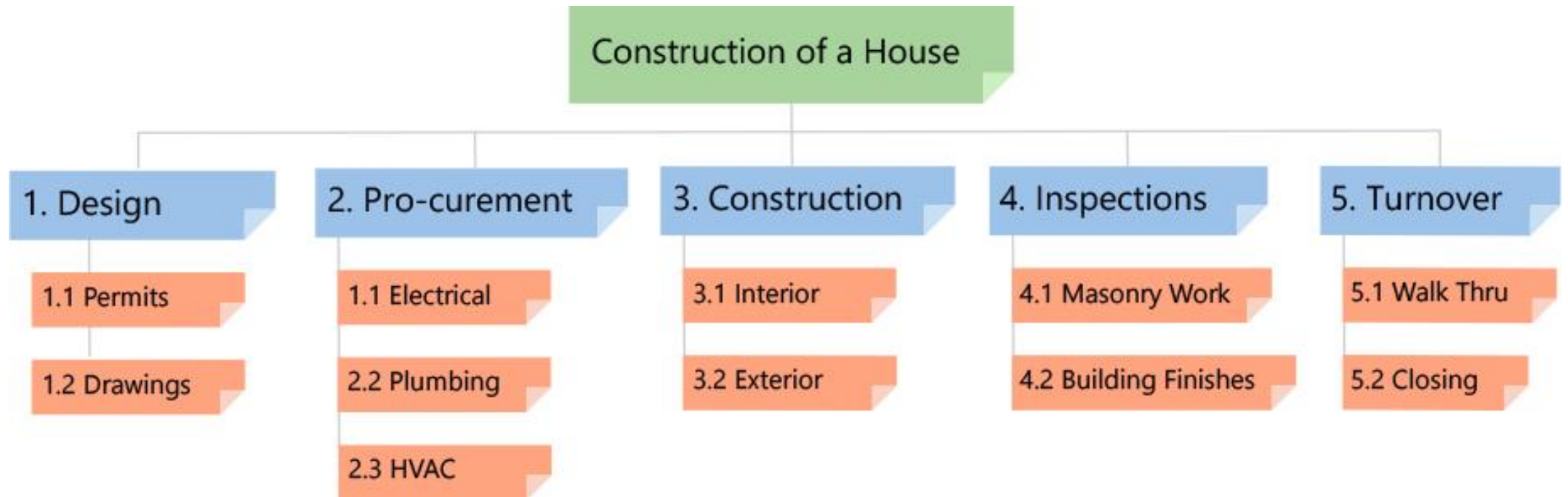
WBS Types  
Deliverable-based &  
Phase-based  
Examples



# *Deliverable-Based Work Breakdown Structure*



# *Phase-Based Work Breakdown Structure*



# How to **Make** a Work Breakdown Structure?

## 1) Gather Critical Project Documents

- Identify content containing project deliverables, such as the Project Charter, Scope Statement and Project Management Plan (PMP) subsidiary plans

## 2) Identify Key Team Members

- Identify the appropriate project team members
- Analyze the documents and identify the deliverables

# How to **Make** a Work Breakdown Structure?

## 3) Define Level 1 Elements

- Level 1 Elements are summary deliverable descriptions that must capture 100% of the project scope
- Verify 100% of scope is captured. This requirement is commonly referred to as the 100% Rule

# How to **Make** a Work Breakdown Structure?

## 4) Decompose (Breakdown) Elements

- Begin the process of breaking the Level 1 deliverables into unique lower Level deliverables. This “breaking down” technique is called *Decomposition*
- Continue breaking down the work until the work covered in each Element is managed by a single individual or organization. Ensure that all Elements are mutually exclusive
- Ask the question, would any additional decomposition make the project more manageable? If the answer is “no”, the WBS is done

# How to **Make** a Work Breakdown Structure?

## 5) Create WBS Dictionary

- Define the content of the WBS Dictionary. The WBS Dictionary is a narrative description of the work covered in each Element in the WBS
- The lowest Level Elements in the WBS are called Work Packages
- Create the WBS Dictionary descriptions at the Work Package Level with detail enough to ensure that 100% of the project scope is covered
- The descriptions should include information such as, boundaries, milestones, risks, owner, costs, etc.

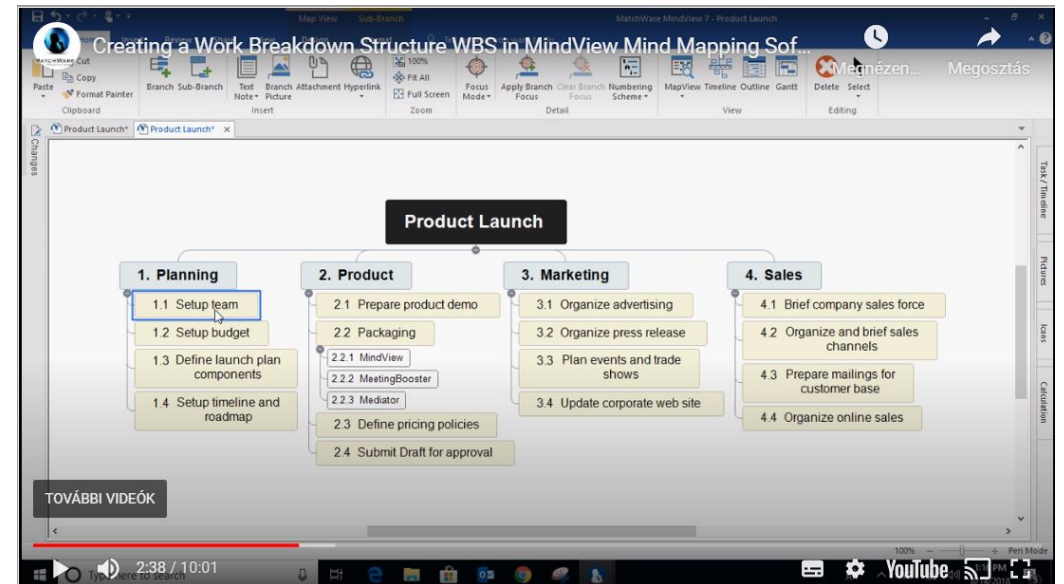
# How to **Make** a Work Breakdown Structure?

## 6) Create Gantt Chart Schedule

- Decompose the Work Packages to activities as appropriate
- Export or enter the Work Breakdown Structure into a Gantt chart for further scheduling and project tracking

# Creating a Work Breakdown Structure WBS in MindView Mind Mapping Software

- <https://youtu.be/cULMKqVh3QE>



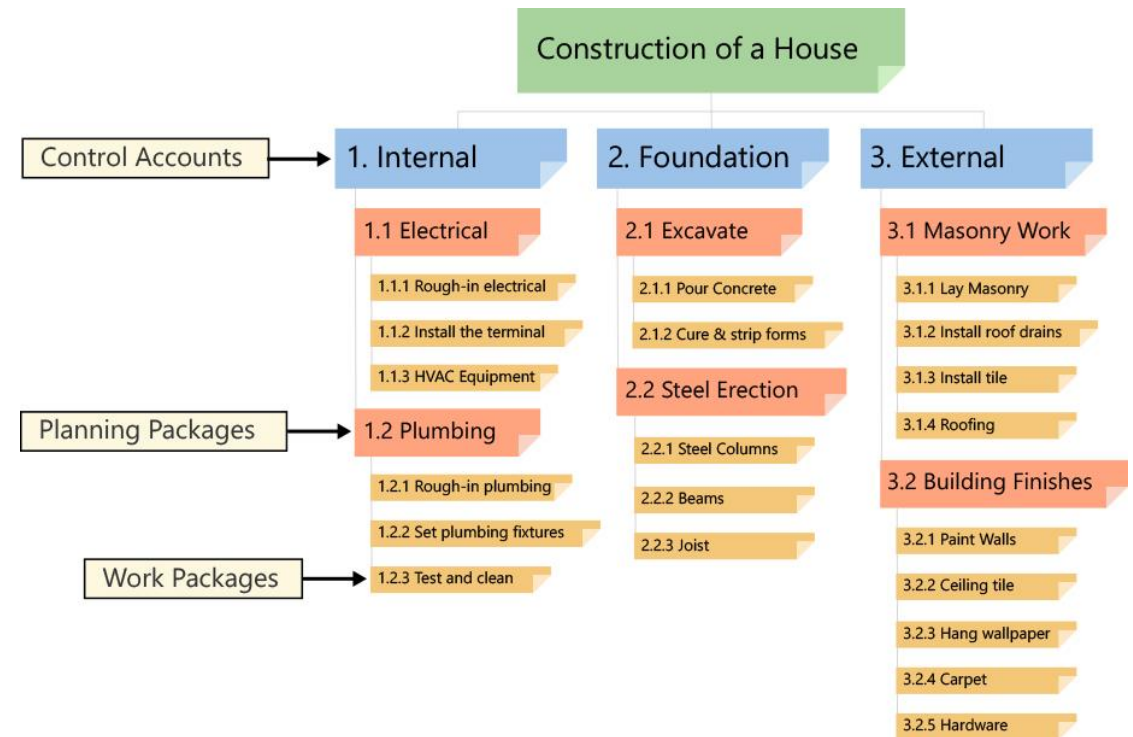


# How to **Use** a Work Breakdown Structure?

- The Work Breakdown Structure is used for many different things
  1. Initially, it serves as a planning tool to help the project team plan, define and organize scope with deliverables
  2. The WBS is also used as the primary source of schedule and cost estimate activities
  3. But, its biggest contributions to a project are its use as a description of all of the work and as a monitoring and controlling tool

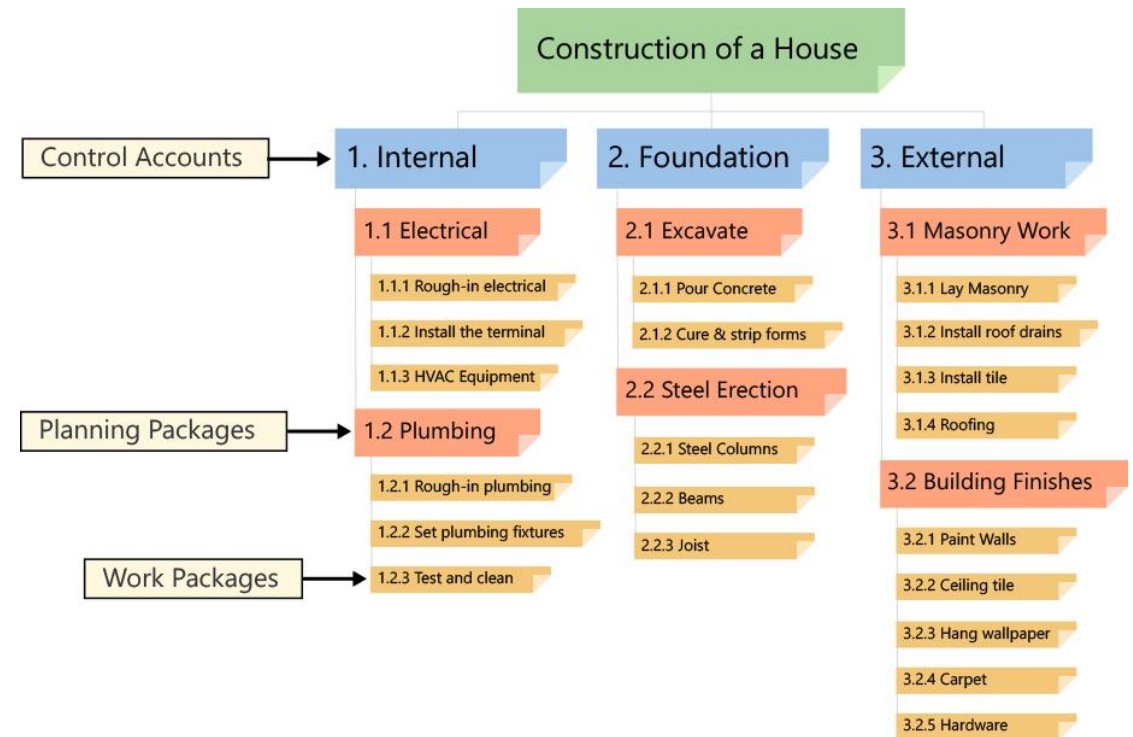
# WBS – Work packages

- This figure shows the House Project Work Breakdown Structure expanded to Level 1, 2, and 3 Elements
- The lowest Levels of each Leg and Branch of the WBS are called **Work Packages**
- Work Packages cover information related to the deliverable, such as owner, milestones, durations, resources, risks, etc.
- This information is described in the WBS Dictionary



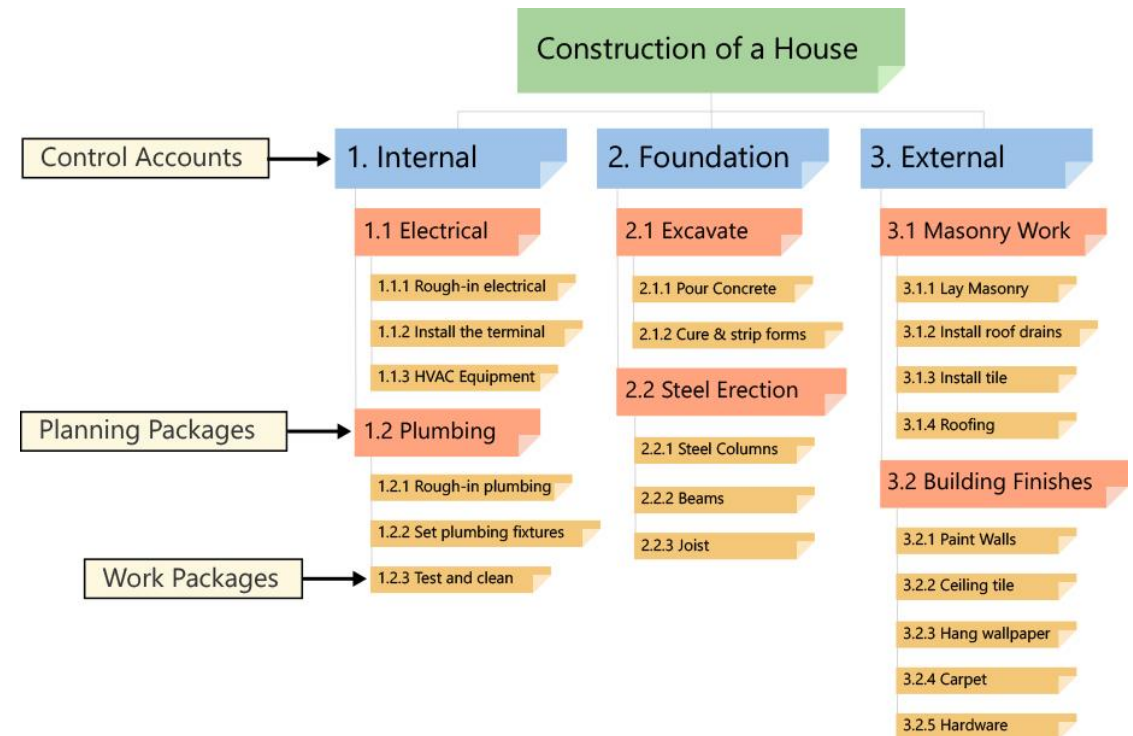
# WBS - Planning Packages

- When the project management plan is approved, scope is known, but not necessarily all of the details
- In order to apply the 100% Rule and capture all of the scope, Planning Packages are created
- It is understood that as details are defined, the Planning Packages eventually evolve to Work Packages
- In the House Project, the project manager knows that the house will have fixtures, but at the time construction begins, there is only a fixture allowance and no fixtures identified
- Once the fixtures are determined, the associated Planning Package becomes a Work Package
- This planning process is called Rolling Wave Planning and is a form of Progressive Elaboration



# WBS - Control Accounts

- Control Accounts are WBS Elements at which the project plans to monitor and report performance
- The Control Accounts can be any Element in the WBS
- In the House Project, the project manager decides that the project risks associated with using subcontractors can be better managed if the project reports performance for each subcontractor. To monitor their performance, Elements 1, 2 and 3 have been identified as Control Accounts
- However, the remaining work in Elements 1.0 and 2.0 will be performed by company resources with less risk and the project does not feel like monitoring and controlling is needed at lower Levels
- To assist with the monitoring and reporting, project management information tools are used to collect, analyze and report information at any Element within the WBS



# Top-down vs. Bottom-up Design

- WBS is a Top-down approach
  - Focuses on high-level features
  - Delivers value (can give the highest value possible to the client / user)
  - Keeps focus on solution requirements
- While Bottom-up can be better in
  - Find simplest / cheapest solutions
  - Reuse existing technology and solutions
  - Find most efficient composition of already made stuff

# Summary, conclusion

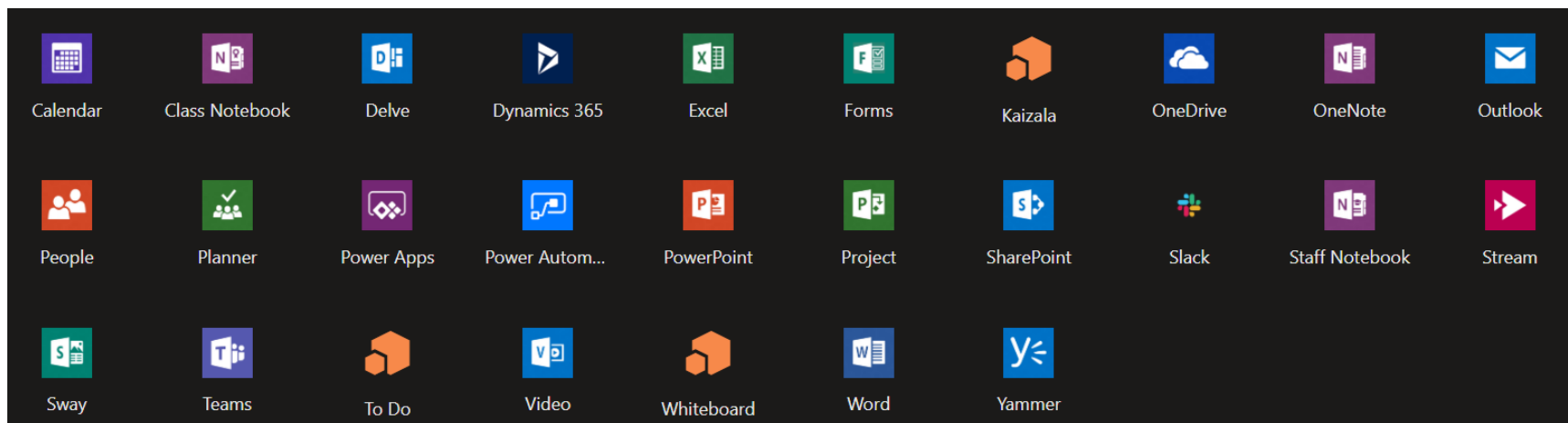
- Not the concrete software are important but the principle
- The goal is to divide the resources into parts (time, cost – roles, machines, materials)
- There are software for calculating critical path, resource allocation...
- I will recommend you (usually) free, open source, multiplatform solutions (for everyone or ELTE students)



Toolbox for software development  
from the viewpoint of project  
management

# Software development Project management tools

- ELTE: Office 365 apps
- Teamwork: Teams, Slack
- *Do you know these ones?*





# Version Control Systems

- ELTE:
  - GitLab <https://szofttech.inf.elte.hu/> (want to access → ask)
  - Azure DevOps <https://dev.azure.com/> (with your Office 365 account)
- Github <https://github.com/>
  - Guide <https://guides.github.com/activities/hello-world/>

# GitLab



## Welcome to GitLab

Faster releases. Better code. Less pain.



### Create a project

If you are added to a project, it will be displayed here.



### Explore public projects

Public projects are an easy way to allow everyone to have read-only access.



### Learn more about GitLab

Take a look at the documentation to discover all of GitLab's capabilities.

# Azure DevOps

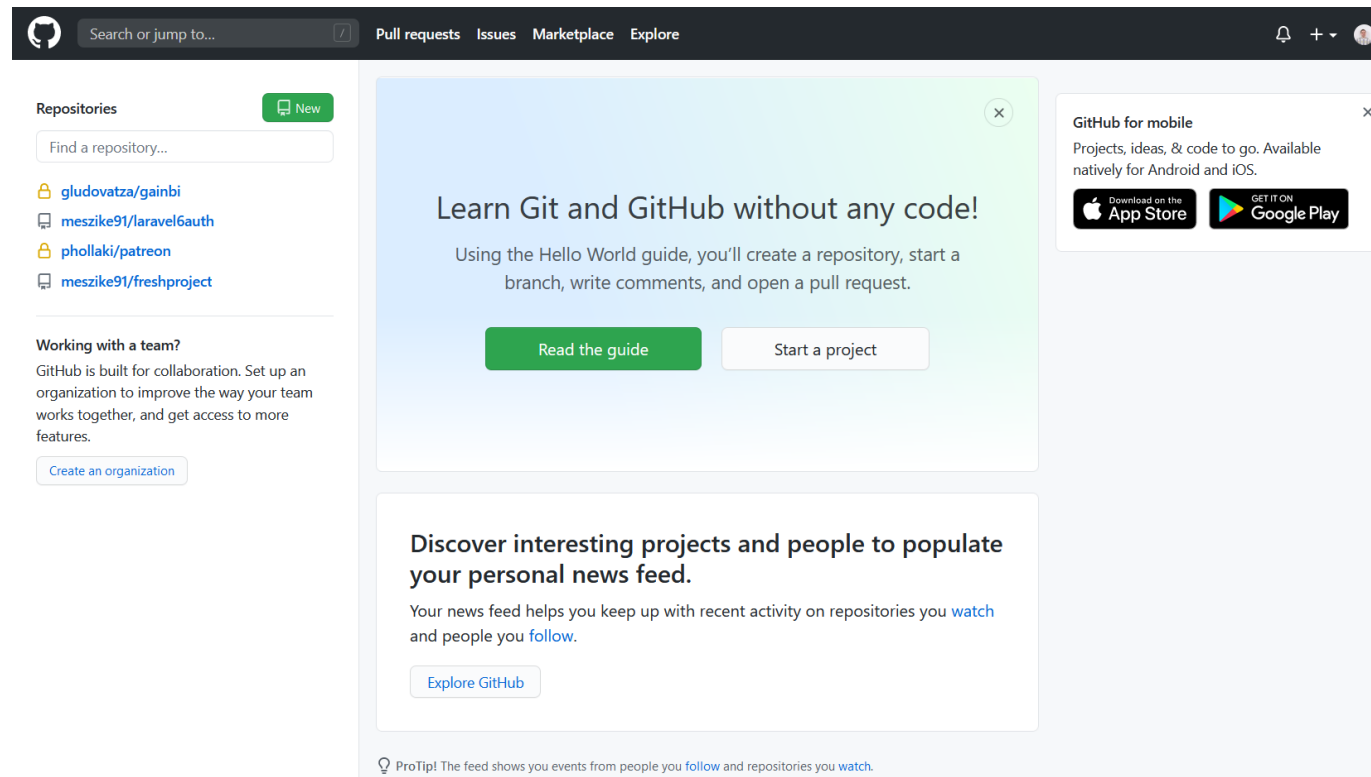
The screenshot displays the Azure DevOps interface for a project named 'Test project'. The left sidebar contains a navigation menu with the following items: 'Test project' (selected), 'Overview', 'Summary', 'Dashboards', 'Wiki', 'Boards', 'Repos', 'Pipelines', 'Test Plans', and 'Artifacts'. At the bottom of the sidebar is a 'Project settings' link with a double-left arrow icon.

The main content area is titled 'Test project' and includes a breadcrumb trail: 'gla0799 / Test project / Overview / Summary'. A search bar is located in the top right corner. Below the title, there are buttons for 'Private' (with a lock icon), 'Invite' (with a plus and person icon), and a star icon.

The central section, 'About this project', features the text 'Help others to get on board!' and 'Describe your project and make it easier for other people to understand it.' Below this is a button labeled '+ Add Project Description'. To the right of the text is an illustration of a person running on clouds.

The right sidebar contains two panels. The 'Project stats' panel, with a 'Last 7 days' filter, shows 'Repos' with '0 Pull requests opened' and '0 Commits by 0 authors'. The 'Pipelines' panel shows a '0%' progress indicator and the text 'Builds succeeded'. The 'Members' panel shows '1' member, represented by a red circular profile icon with the initials 'AG'.

# Github



# Tools for WBS and beyond (MindView)

- Create a WBS with the MindView software free trial
- Check it out!
- <https://www.matchware.com/wbs-software>

# Tools for WBS and beyond (Trac)

- Trac is an enhanced wiki and issue tracking system for software development projects (bug tickets)
- Trac uses a minimalistic approach to web-based software project management
- It provides an interface to Subversion and Git (or other version control systems), an integrated Wiki and convenient reporting facilities

# Tools for WBS and beyond (Trac)

The screenshot displays the Trac web interface. At the top, the 'trac' logo is visible with the tagline 'Integrated SCM & Project Management'. A search bar and navigation links (Login, Preferences, Help/Guide, About Trac) are in the top right. A horizontal menu below the logo includes 'Home', 'Trac' (selected), 'Trac Hacks', 'Genshi', 'Babel', and 'Bitten'. A secondary menu contains 'WIKI', 'TIMELINE', 'ROADMAP' (selected), 'BROWSE SOURCE', 'VIEW TICKETS', 'NEW TICKET', and 'SEARCH'. On the left, a vertical sidebar lists links: Home, Download, Documentation, Mailing Lists, License, and FAQ. The main content area is titled 'Roadmap' and features a section for 'Milestone: 1.5.2'. Below this, it indicates the milestone is '2 months late (Jul 5, 2020, 3:00:00 AM)' and shows a progress bar at 34%. Ticket statistics are listed: 'Total number of tickets: 6 - closed: 2 - in progress: 2 - new: 2'. A yellow box highlights the 'Next development version for the 1.5.x release line'. At the bottom, a note states 'Python 3.5+ is supported. Python 2.7 is no longer supported. Jinja2 is the template engine, Genshi is no longer supported.' and provides links to 'API changes' and 'release notes'. A settings panel on the right allows users to toggle 'Show completed milestones' and 'Hide milestones with no due date'.

Edgewall Software

**trac**  
Integrated SCM & Project Management

Search

Login | Preferences | Help/Guide | About Trac

Home **Trac** Trac Hacks Genshi Babel Bitten

WIKI TIMELINE **ROADMAP** BROWSE SOURCE VIEW TICKETS NEW TICKET SEARCH

## Roadmap

**Milestone: 1.5.2**

**2 months late** (Jul 5, 2020, 3:00:00 AM)

34%

Total number of tickets: 6 - closed: 2 - in progress: 2 - new: 2

**Next development version for the 1.5.x release line**

Python 3.5+ is supported. Python 2.7 is no longer supported. Jinja2 is the template engine, Genshi is no longer supported.

See also corresponding drafts for [API changes](#) and [release notes](#).

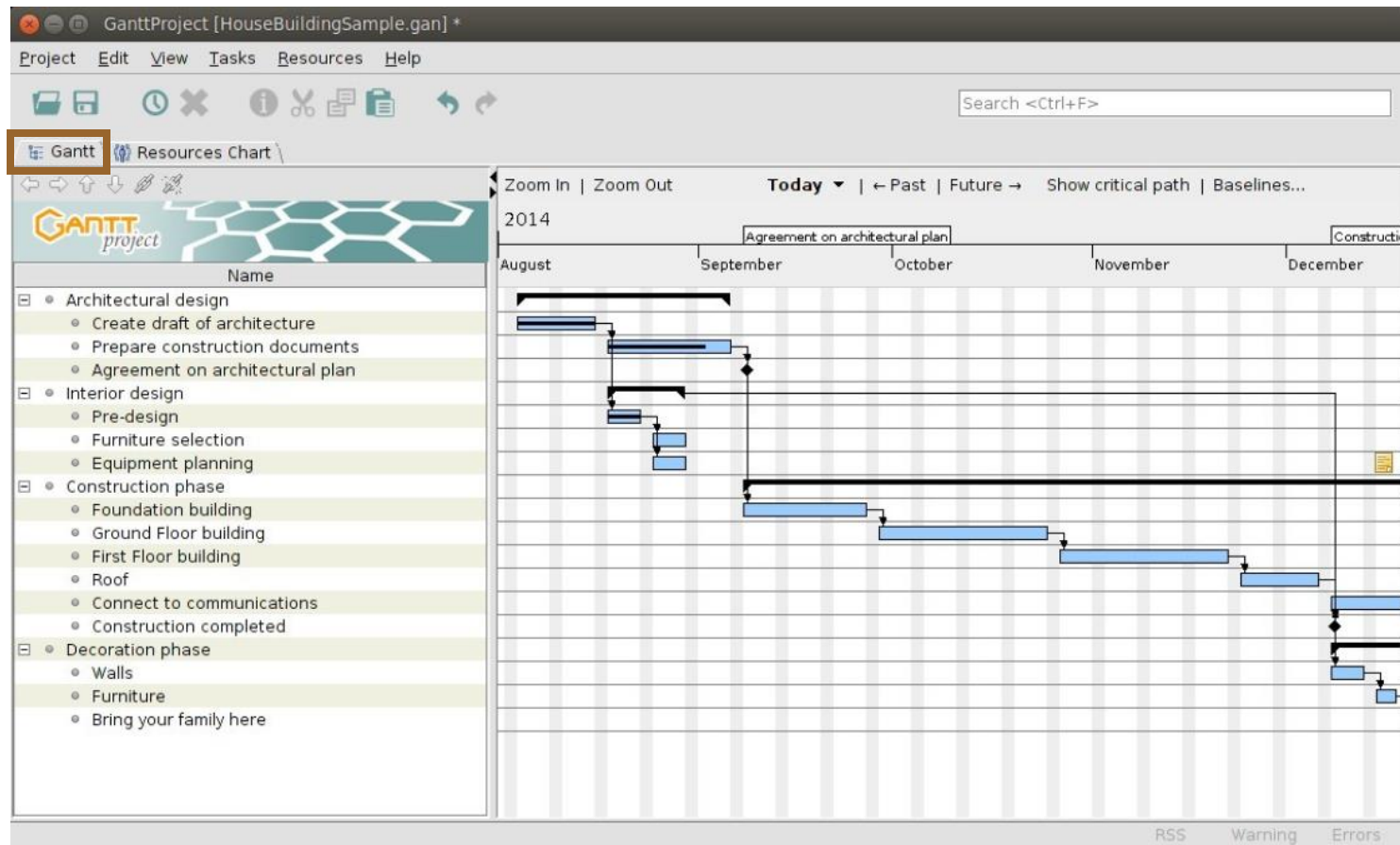
☐ Show completed milestones  
☐ Hide milestones with no due date  
Update

# Tools for WBS and beyond (GanttProject)

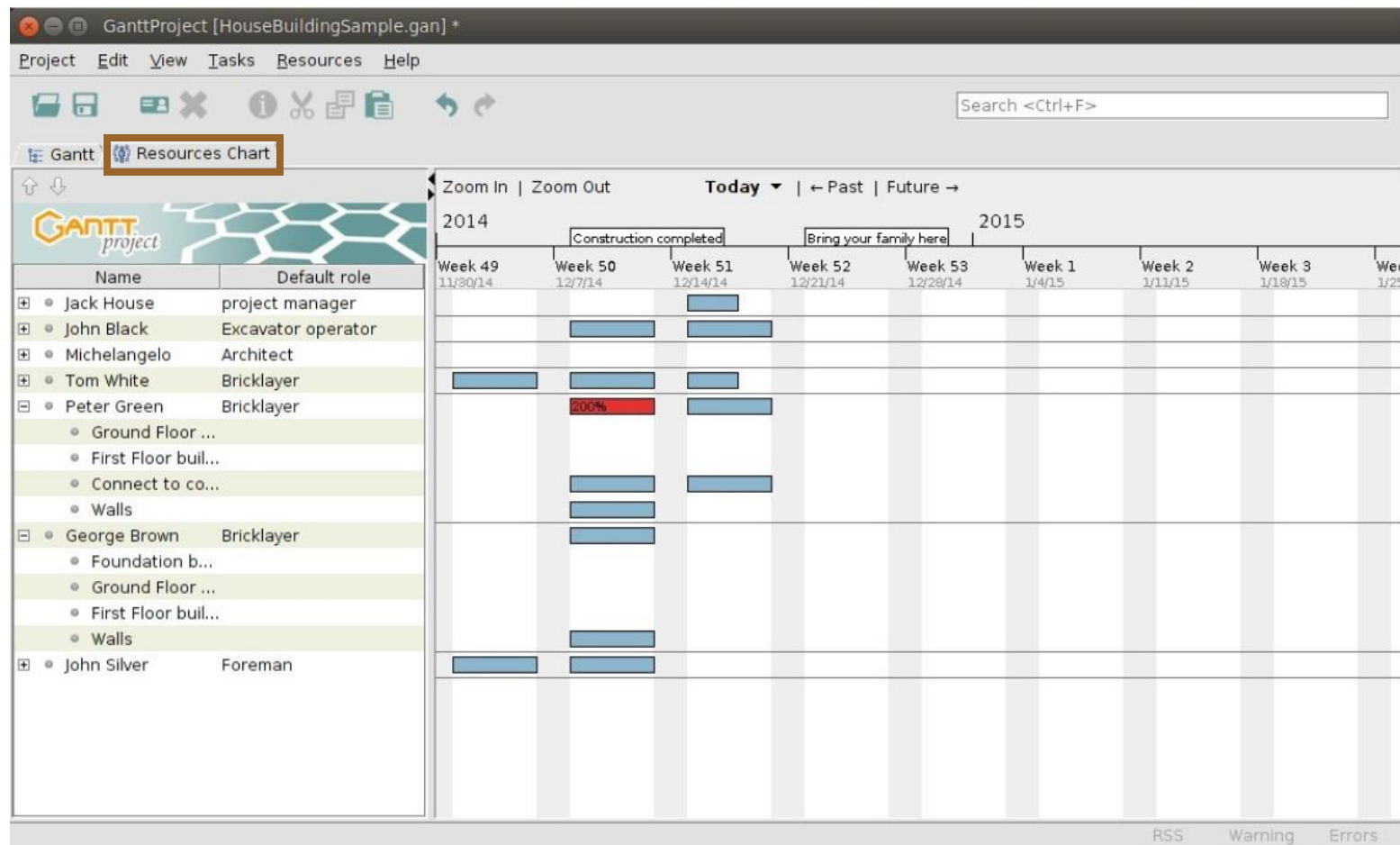
- <https://www.ganttproject.biz/>
- Easy-of-use multilingual solution
- Free, open source
- Export / import data, diagrams and projects
- Support: documentation, forum, tutorial videos



# Tools for WBS and beyond (GanttProject)



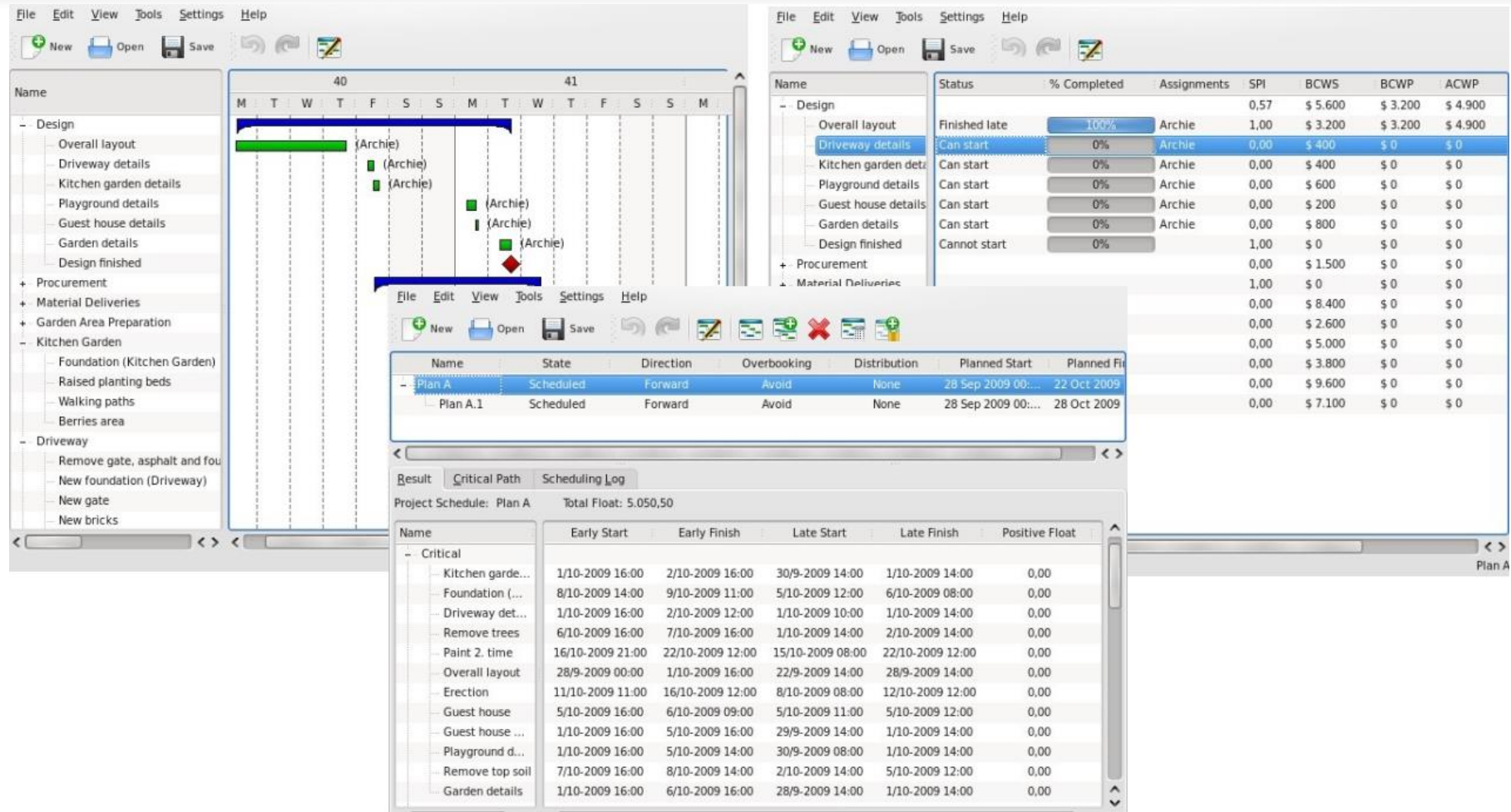
# Tools for WBS and beyond (GanttProject)



# Tools for WBS and beyond (Calligra Plan)

- Plan is a project management application. It is intended for managing moderately large projects with multiple resources
- To enable you to model your project adequately, Plan offers different types of task dependencies and timing constraints. The usual use case is to define your tasks, estimate the effort needed to perform each task, allocate resources and then let Plan schedule the tasks according to network and resource availability
- In case of changes, the project can be rescheduled, retaining original schedules for comparison. Especially useful is the ability to reschedule from the current state of the project. This can typically be used when the execution of the project does not go according to plan, resource availability changes or tasks have to be added to or removed from the project.

# Tools for WBS and beyond (Calligra Plan)



# Tools for WBS and beyond (LibrePlan)

- LibrePlan is a collaborative tool to **plan, monitor and control** projects and has a **rich web interface** which provides a desktop alike user experience. All the team members can take part in the planning and this makes possible to have a real-time planning . LibrePlan is open source and you can **download, install and customize it for free**.
- It is usual that you need to manage more than one project at a time, with resources that participate in several projects. LibrePlan was designed with these scenarios in mind, where multiple projects and resources interact to carry out the work inside your company.
- It is common that some of the data needed by planning tools are stored in other applications of the IT infrastructure. When this is the case, if you **reuse this shared data** you save both time and money by avoiding to insert them manually in two different points. LibrePlan **makes this sharing possible by providing** a wide set of web services to import and export data.

# Tools for WBS and beyond (LibrePlan)

**LIBREPLAN** OpenWebPlanning

Scheduling Resources Administration / Management Reports My account use

START > Scheduling > Project Details > LibrePlan customization

Task  with  options Filter

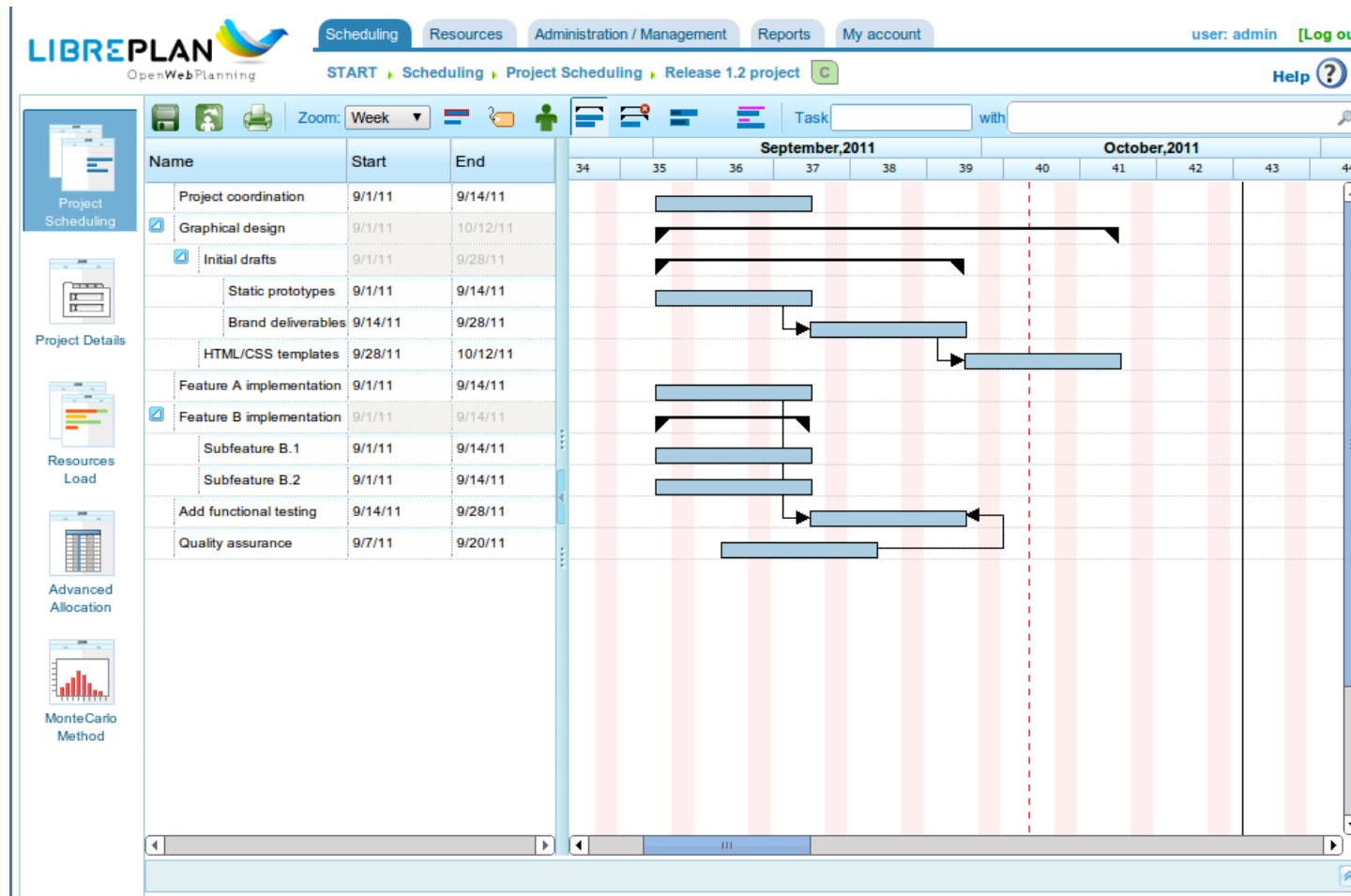
**WBS (tasks)** General data Imputed hours Progress Label Criterion Requirement Materials Task quality forms

New task:  Hours: 0 Add Add From Template Selected node:

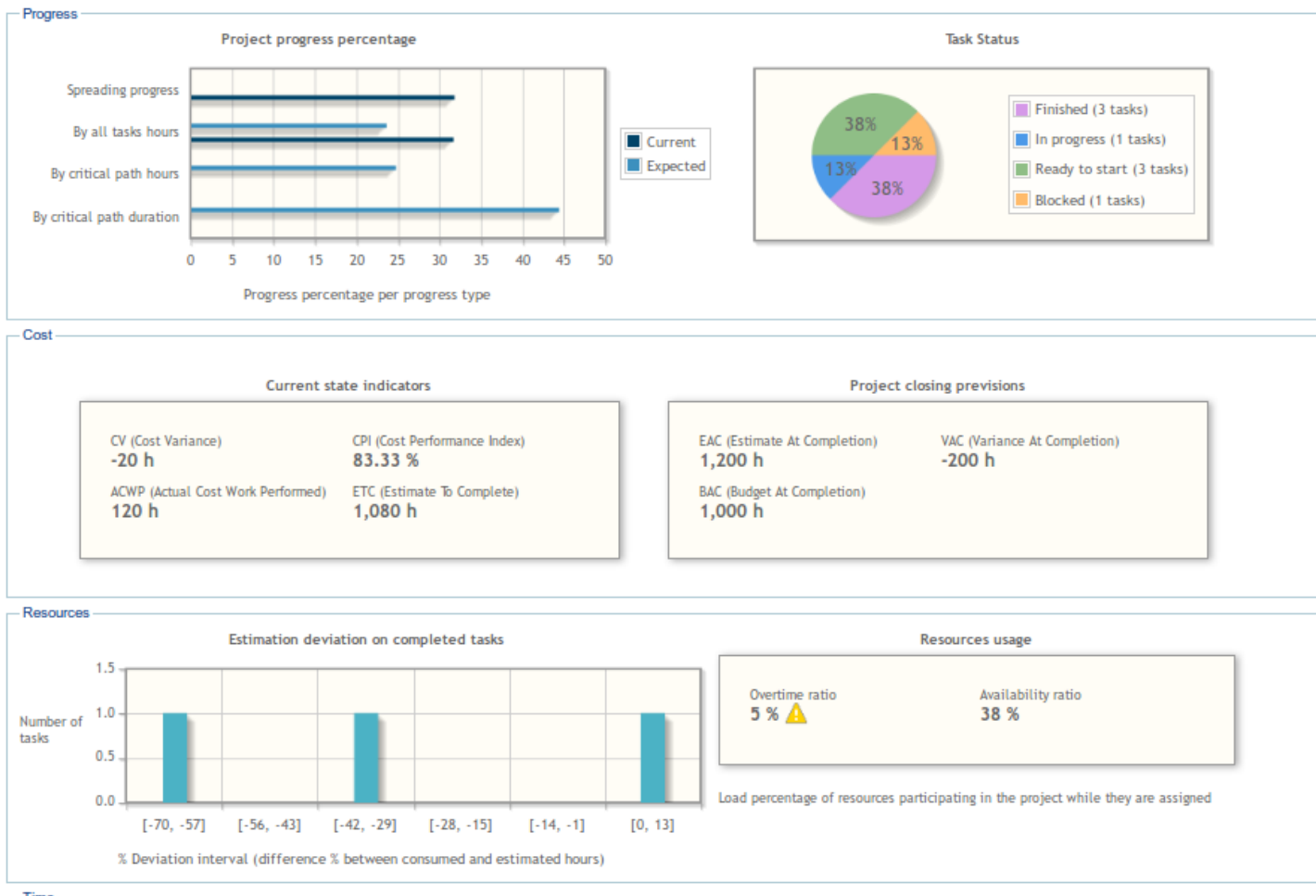
Scheduling state	Code	Name	Hours	Must start after	Deadline
		Project Coordination	200		
		Graphical design	140		
		Initial Drafts	60		
		Static prototypes	40		10/12/11
		Brand deliverables	20		
		HTML/CSS templates	80		
		Feature A implementation	70		
		Feature B implementation	150		
		Subfeature B.1	40		
		Subfeature B.2	110		
		Add functional Testing	60		
		Quality Assurance	20		

Project Scheduling  
Project Details  
Resources Load  
Advanced Allocation  
Monte Carlo Method

# Tools for WBS and beyond (LibrePlan)



# Tools for WBS and beyond (LibrePlan)






# Software development Project management tools

- G Suite


<https://gsuite.google.com/>


- Gantter Project Management





 Gmail  
Custom business email


 Meet  
Video and voice conferencing


 Chat  
Messaging for teams


 Calendar  
Shared calendars

 Drive  
Cloud storage


 Docs  
Word processing


 Sheets  
Spreadsheets


 Slides  
Presentation builder


 Forms  
Surveys builder

 Sites  
Website builder

 Currents  
Engage employees

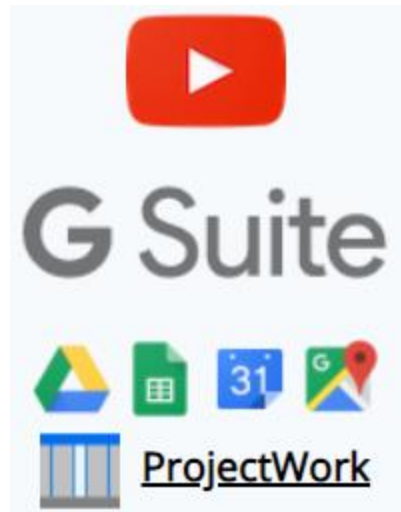
 Keep  
Notes and lists

 Apps Script  
Optimize how you work

 Cloud Search  
Smart search across G Suite

# ProjectWork

- <https://www.forscale.nl/en/products/projectwork/projectwork.html>




- Project Management planning software shows a project in one schedule with its scope, work, time, costs and documentation. ProjectWork is a web app to create and track schedules in the cloud by multiple users, even from various locations simultaneously.

# Pro


**Free Read Only.** Start your free one month trial of Project Plan 365 or Sign In to activate an existing subscription

[illegible]




# Get Started with Project Plan 365


Try **Project Plan 365 free for 30 days** to get full access to your mpp files. View, edit, share, export your projects and more. You can cancel the subscription anytime.



Install Project Plan 365 on your Mac or PC



Unlock full Project Plan 365 experience on your phones and tablets



Collaborate with your team in real-time using Project Plan 365

Start your free 30 Days Trial Now

By clicking Start, you agree to our [Terms of Use](#) and [Privacy Policy](#).

Continue with Free Read Only

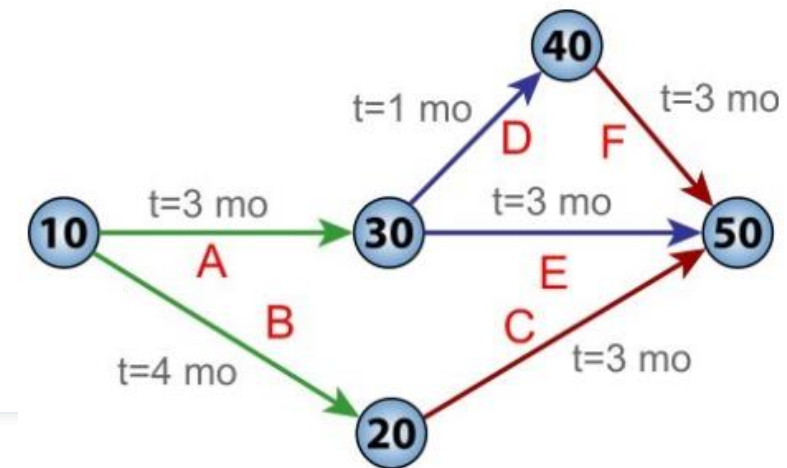
Read Only Mode does not allow you to save or export your projects.

[Restore Previous Purchase from Windows Store](#)

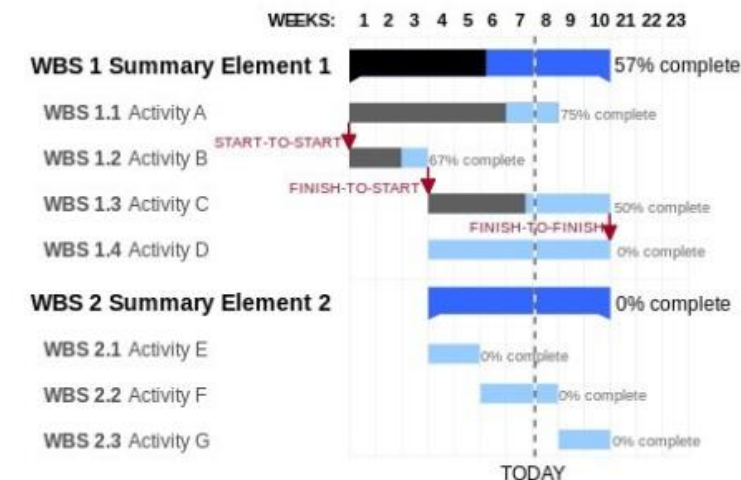
[Already have a Project Plan 365 Account? Sign in here](#)

☐ Don't show this dialog again

# Tools for WBS and beyond



- For more see:
  - [https://en.wikipedia.org/wiki/Comparison\\_of\\_project\\_management\\_software](https://en.wikipedia.org/wiki/Comparison_of_project_management_software)
- Also see:
  - Critical Chain Project Management and
  - Critical Path Method
  - PERT (Project Evaluation and Review Technique)
  - Gantt Chart



# Software Development Life Cycle (SDLC) Criticism

- **Good** in
  - Large projects, complex solutions
  - Detailed, well documented
  - Ease of maintenance
  - Can follow (and certify) standards
- **Bad** in
  - Time and cost is increased
  - Not flexible, cannot handle problems without full-design-first approach
- There are other life cycle approaches
  - Project Life Cycle
  - Application Life Cycle
- Example: SSADM (Structured Systems Analysis and Design Method)
  - Takeaway: Works well on similar problems / projects (in this case information systems)

# SDLC vs Assembly Line Analogy

- Assembly Line Features:
  - Every Product is exactly the same we make
  - All work phases are the same, we know
    - How long do they take
    - How much do they cost
  - Workers know everything after training, no further self-development is needed

→ Process risk is very low

**NOT TRUE for SW Development!!!**

# SDLC vs Assembly Line Analogy

- Obviously, the Assembly Line analogy is not true for multiple important parts of Software Development which causes significant risk when the model is used without groundbreaking changes



# Questions?

- ...
- Or write me an email to [gla@inf.elte.hu](mailto:gla@inf.elte.hu)



# Ideas - Your project/software plans? Your teams? Canvas

