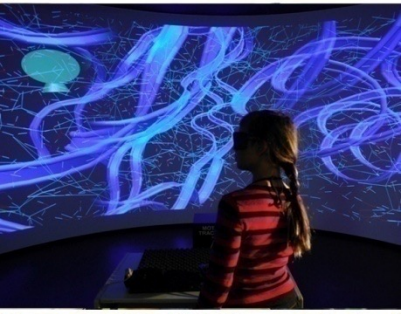
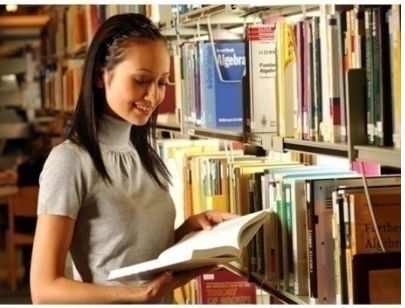
JOHANNES KEPLER
UNIVERSITY LINZ





Institute for
Software Systems Engineering
Technologies for building better software

Alexander Egyed **Paul Grünbacher**

alexander.egyed@jku.at *paul.gruenbacher@jku.at*

http://www.isse.jku.at

Univ.-Prof. Dr. **Alexander Egyed**

JOHANNES KEPLER UNIVERSITY LINZ

Ph.D. University of Southern California, USA under Dr. Barry Boehm
VR Research JKU (2015-2019)

Current Affiliations:

- Professor at Johannes Kepler University, Austria 2008
- Adjunct Professor at University of Southern California, USA 2005

Past Affiliations:

- Vice Rector at Johannes Kepler University, Austria, 2015-2019
- Research Fellow at University College London, UK 2007
- Research Scientist at Teknowledge Corporation, USA 2000
- Research Assistant at University of Southern California, USA 1996

http://www.alexander-egyed.com/

ISSE

a. Univ.-Prof. Dr. **Paul Grünbacher**

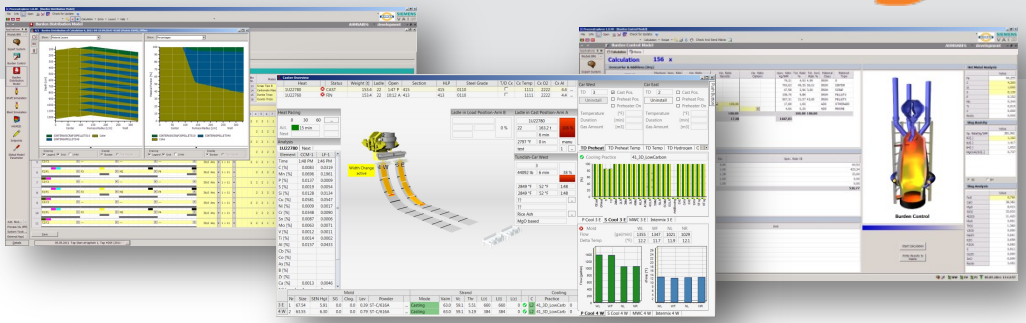
JOHANNES KEPLER UNIVERSITY LINZ

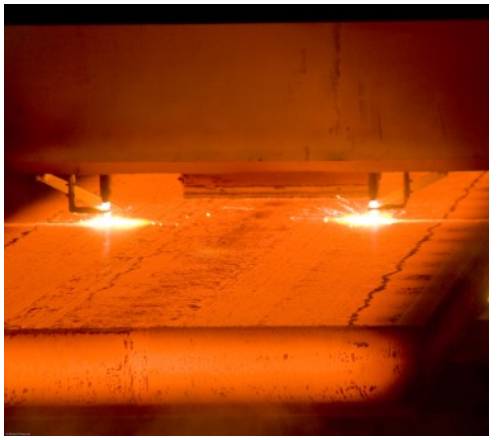
Deputy Head Institute of Software Systems Engineering

Head of Christian Doppler Lab MEVSS (2013-2020)

- Requirements Engineering
  - Requirements Elicitation and Negotiation (with University of Southern California)
  - Scenario-based Requirements Engineering (with City University London and Simula Research Oslo)
  - Requirements and Reference Architectures (with Primetals Technologies)
- Software Product Lines
  - Variability modeling and product configuration (with Siemens VAI)
  - Feature-oriented engineering (with KEBA AG)
- Software Monitoring
  - Event-based Monitoring (with Primetals Technologies)

ISSE





Level 3 (production planning systems)

HMI
- I-HMI
- S-HMI
- C-HMI

Level 2 (process optimization)
- Iron
- Steel
- Casting

Database
- I-DB
- S-DB
- C-DB

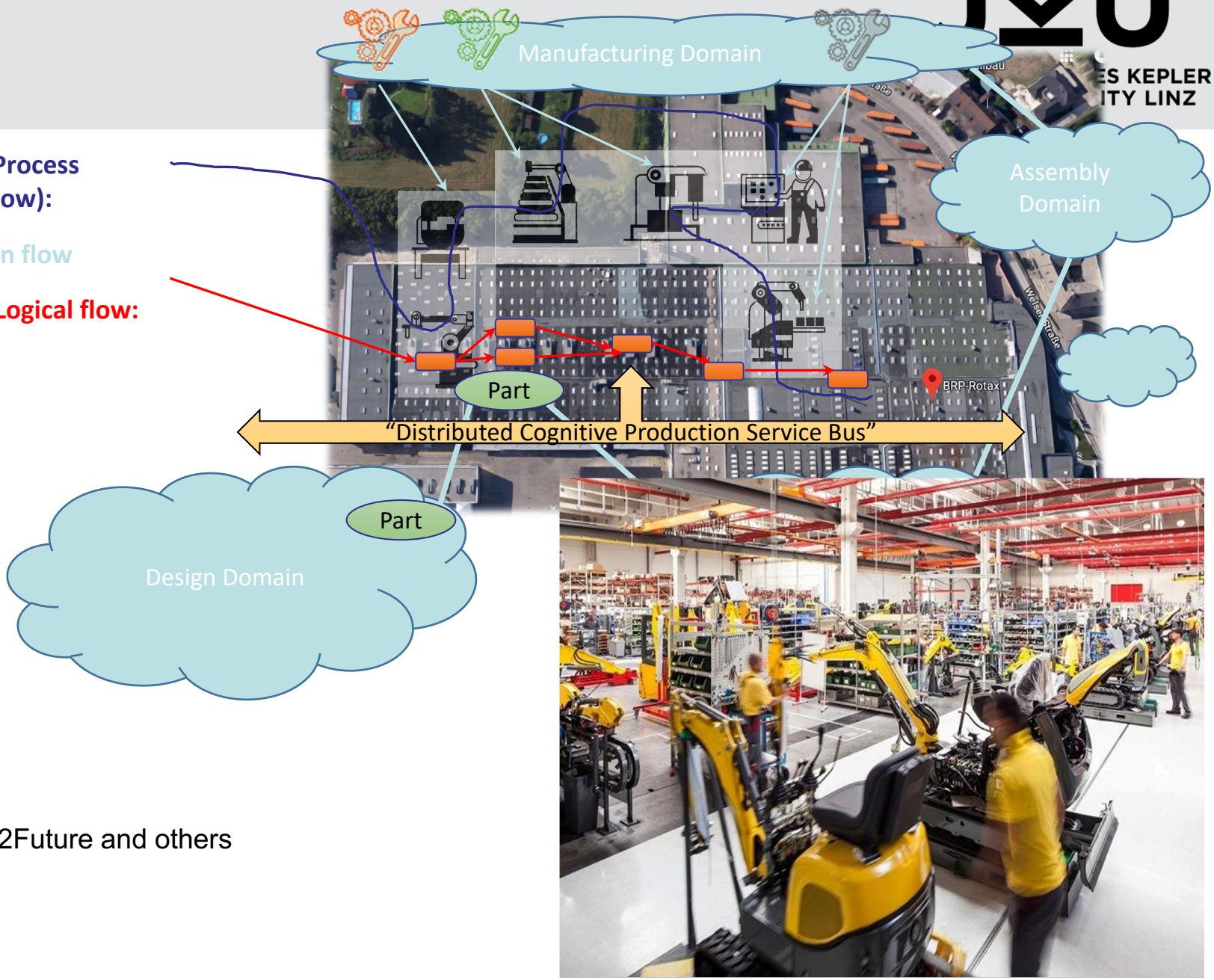
Customer and Third-party Systems

PLC Level 1 (process control) PLC

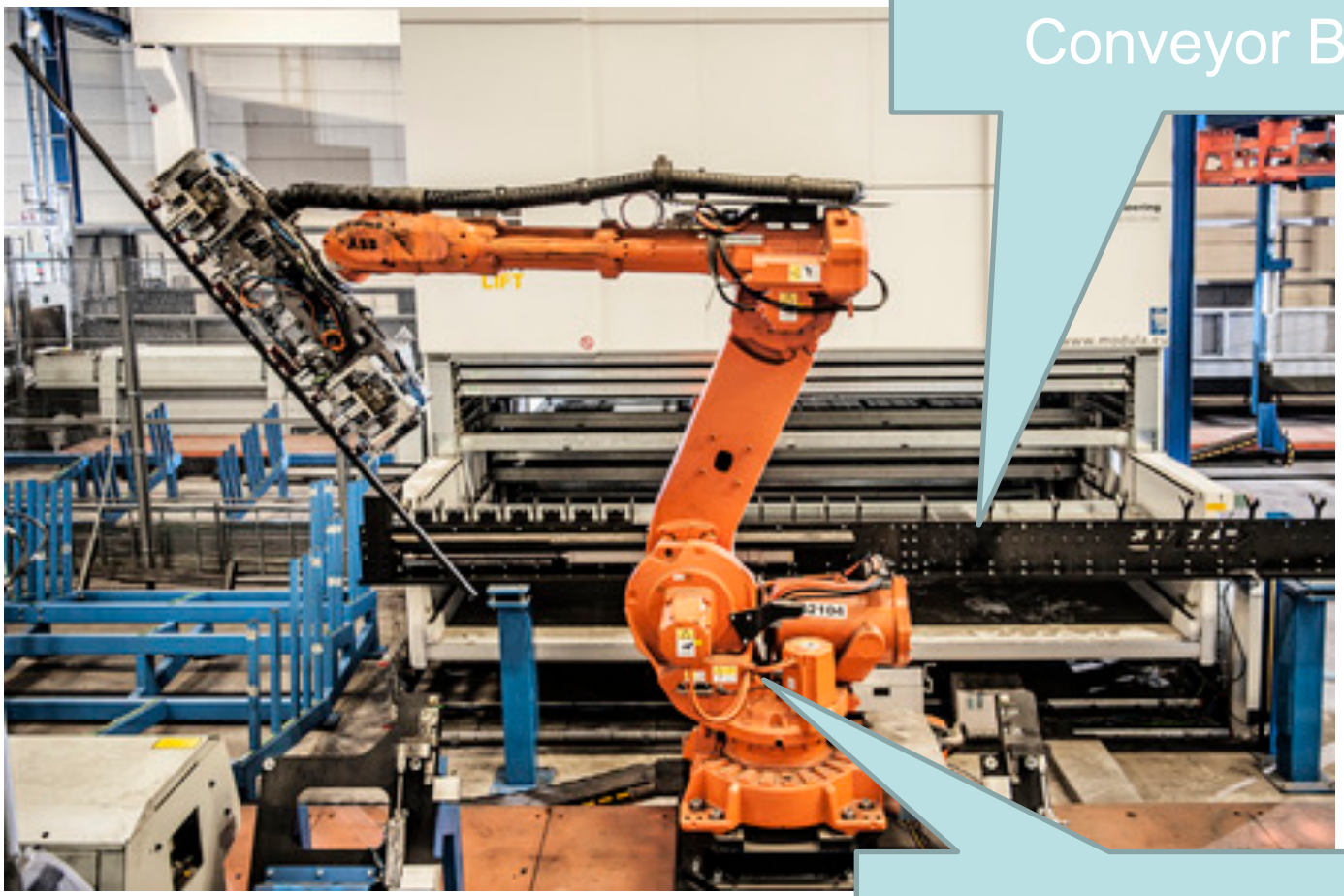
Product / Process (physical flow):

Information flow

Control & Logical flow:



Source: Pro2Future and others


Conveyor Belt
Robotic Arm

Science Part 3, 2nd Floor

JOHANNES KEPLER
UNIVERSITY LINZ

ISSE

Software

Software

JOHANNES KEPLER
UNIVERSITY LINZ

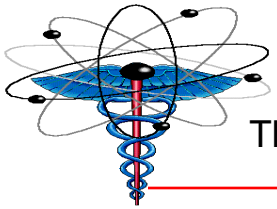
- Modern economies are dependent on software
- More and more systems are software controlled
- Software failures are ever more visible and costly
- Software engineering expenditure represents a significant fraction of GNP in all developed countries

ISSE

Fatal Software Defects

London Ambulance System (LAS)-1992 [Finkelstein96]

Mars Climate Orbiter-1998 [Breitman99]



Therac 25, Medical Linear Accelerator-1985 [Leveson93]

Siemens: Possible Hearing Damage in Some Cell Phones-2004 [Siemens05]

Mercedes A-Class capsize in corners [Mercedes97]

Ariane 5: What happened?

JOHANNES KEPLER UNIVERSITY LINZ

The first flight carried $500M of satellites and was destroyed about 40 seconds after lift off. **The error was in a program segment for converting a floating point number, representing a measurement, to a signed 16 bit integer was executed with an input data value outside the range representable by a signed 16 bit integer**.

The code in question was reused from an earlier vehicle where the measurement would not have become large enough to cause this failure.

https://www.esa.int/Newsroom/Press_Releases/Ariane_501_-_Presentation_of_Inquiry_Board_report

ISSE

Ariane 5 and Software Engineering

JOHANNES KEPLER UNIVERSITY LINZ

- **Programming error?** An incorrectly handled software exception resulted from a data conversion of a 64-bit floating point to a 16-bit signed integer value.
- **Design error?** The design specification accounted for random hardware failures only, and the exception handling mechanism was unable to recover from a random software error.
- **Incorrect analysis of changing requirements?** Failure could have been prevented if the adaptive maintenance team had not adopted the approach of "if it ain't broke, don't fix it."
- **Inadequate validation and verification?** No test to verify that the system would behave correctly "when being subjected to the countdown and flight time sequence and the trajectory of Ariane 5."
- **Project management problem?** The review process for Ariane 5 development was inadequate as no external people were involved

B. Nuseibeh, "Ariane 5: Who Dunnit?," in *IEEE Software*, vol. 14, no. 3, pp. 15-16, May-June 1997.

Mars Orbiter

exploringmars in-depth specials

Metric mishap caused loss of NASA orbiter

September 30, 1999
Web posted at: 4:21 p.m. EDT (2021 GMT)

In this story:

Metric system used by NASA for many years

Error points to nation's conversion lag

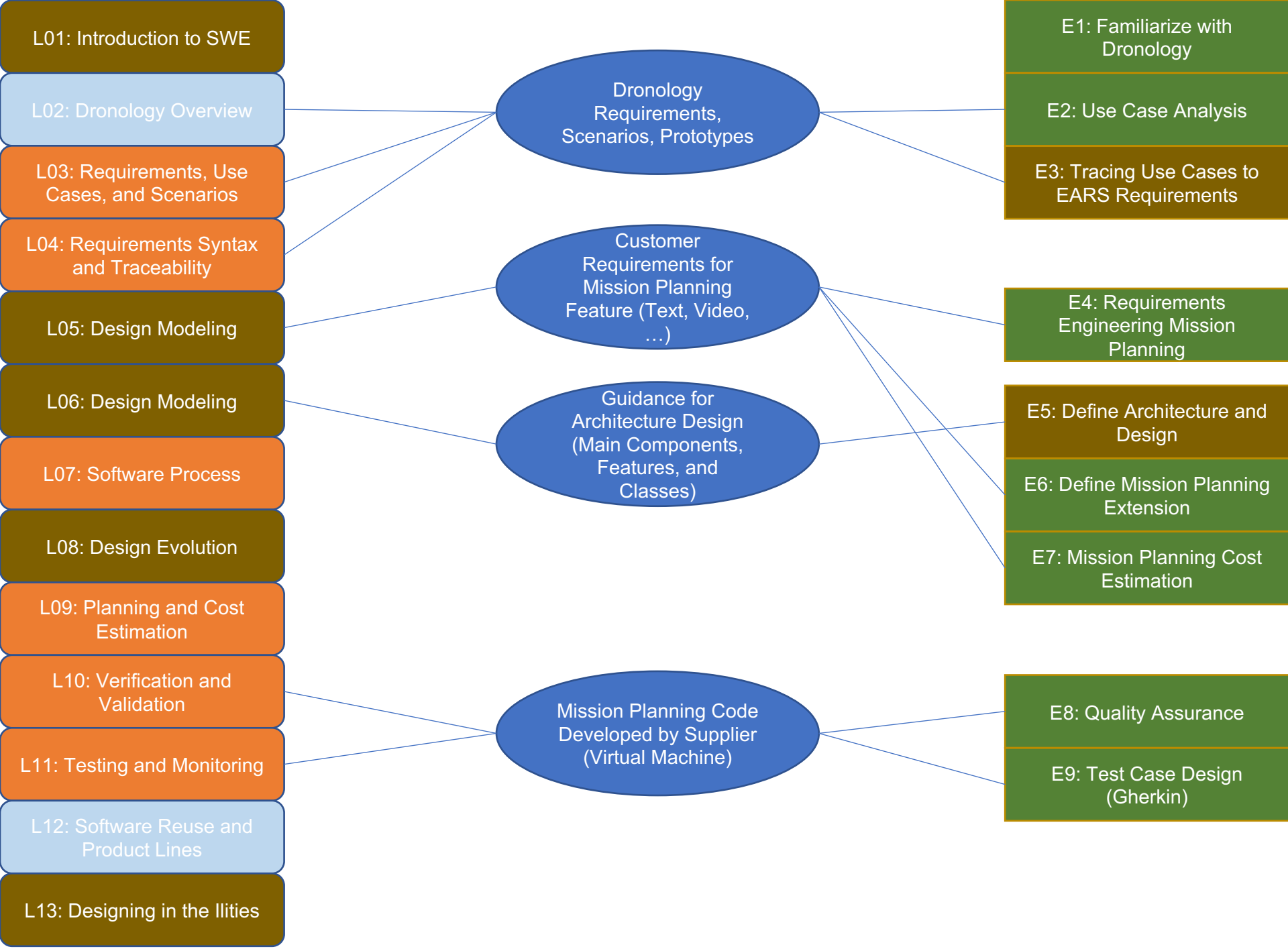
RELATED STORIES, SITES

NASA's Climate Orbiter was lost September 23, 1999

By Robin Lloyd
CNN Interactive Senior Writer

(CNN) -- NASA lost a $125 million Mars orbiter because a Lockheed Martin engineering team used English units of measurement while the agency's team used the more conventional metric system for a key spacecraft operation, according to a review finding released Thursday.

Course


L01: Introduction to SWE
L02: Dronology Overview
L03: Requirements, Use Cases, and Scenarios
L04: Requirements Syntax and Traceability
L05: Design Modeling
L06: Design Modeling
L07: Software Process
L08: Design Evolution
L09: Planning and Cost Estimation
L10: Verification and Validation
L11: Testing and Monitoring
L12: Software Reuse and Product Lines
L13: Designing in the Ilities
Dronology Requirements, Scenarios, Prototypes
Customer Requirements for Mission Planning Feature (Text, Video, …)
Guidance for Architecture Design (Main Components, Features, and Classes)
Mission Planning Code Developed by Supplier (Virtual Machine)
E1: Familiarize with Dronology
E2: Use Case Analysis
E3: Tracing Use Cases to EARS Requirements
E4: Requirements Engineering Mission Planning
E5: Define Architecture and Design
E6: Define Mission Planning Extension
E7: Mission Planning Cost Estimation
E8: Quality Assurance
E9: Test Case Design (Gherkin)

Lectures

JOHANNES KEPLER UNIVERSITY LINZ

05/10/2022	Introduction to SWE+Dronology Overview
12/10/2022	Requirements, Use Cases, and Scenarios
19/10/2022	Requirements Syntax and Traceability
26/10/2022	holiday
02/11/2022	holiday
09/11/2022	Design Modeling
16/11/2022	Design Modeling
23/11/2022	Software Process
30/11/2022	Design Evolution
07/12/2022	Planning and Cost Estimation
14/12/2022	Verification and Validation
21/12/2022	holiday
28/12/2022	holiday
04/01/2023	holiday
11/01/2023	Testing and Monitoring
18/01/2023	Software Reuse and Product Lines
25/01/2023	Designing in the Ilities
01/02/2023	Exam

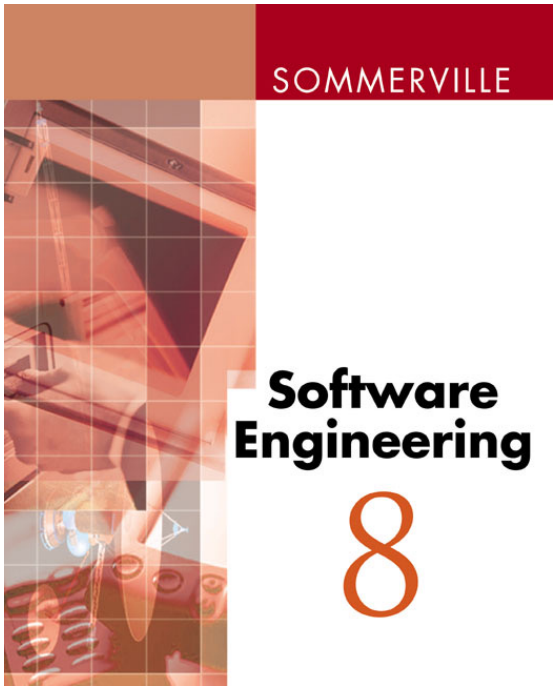
Slides can be downloaded from the KUSSS and MOODLE (exercises)
Also please register for the course and the exam there!

ISSE

Recommended Reading

JOHANNES KEPLER UNIVERSITY LINZ

Ian Sommerville
„Software Engineering“



ISSE

Recommended Reading

https://www.computer.org/education/bodies-of-knowledge/software-engineering

Software Engineering

Margaret H. Hamilton: The Scientist who invented the term "Software Engineering"

JOHANNES KEPLER UNIVERSITY LINZ

*"I **fought to bring the software legitimacy** so that it—and those building it—would be given its due respect and thus I **began to use the term 'software engineering'** to distinguish it from hardware and other kinds of engineering, yet treat each type of engineering as part of the overall systems engineering process. When I first started using this phrase, it was considered to be quite amusing. **It was an ongoing joke for a long time**. They liked to kid me about my radical ideas. Software eventually and necessarily gained the same respect as any other discipline"*

Margaret Hamilton at MIT during the Apollo 11 mission

ISSE

https://www.computer.org/publications/tech-news/events/what-to-know-about-the-scientist-who-invented-the-term-software-engineering

Software Engineering

JOHANNES KEPLER UNIVERSITY LINZ

- Still a young discipline
- NATO Conferences
  - —Garmisch, Germany, October 7-11, 1968
  - —Rome, Italy, October 27-31, 1969
- The reality is finally beginning to arrive
  - —Computer science as the scientific basis
    - Other scientific bases?

ISSE

What is Software Engineering?

JOHANNES KEPLER UNIVERSITY LINZ

Application of a systematic, disciplined, quantifiable approach to the **development**, **operation**, and **maintenance** of software; that is, the **application of engineering to software**.

(ISO/IEC TR 19759:2016, Software Engineering — Guide to the Software Engineering Body of Knowledge)

ISSE

Software Engineering

JOHANNES KEPLER UNIVERSITY LINZ

- Adopting a systematic approach and using appropriate tools and techniques … ?
- … is depending on
  - —the *problem* to be solved,
  - —the development *constraints*, and
  - —the *resources* available

- A key software engineering principle
  - —Better
  - —Cheaper
  - —Faster

} pick any *two*

ISSE

Essential Software Engineering Difficulties (Brooks)

JOHANNES KEPLER UNIVERSITY LINZ

- Complexity
  - —no two software parts are alike
  - —complexity grows non-linearly with size due to interactions
- Conformity
  - —software is always required to conform
  - —often the “last kid on the block”
- Changeability
  - —software is viewed as infinitely malleable
  - —change originates with new applications, users, machines, standards, laws
- Invisibility
  - —the reality of software is not embedded in space
  - —software is not representable as a familiar geometric entity

http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

Hidden Complexity/ Hidden Software

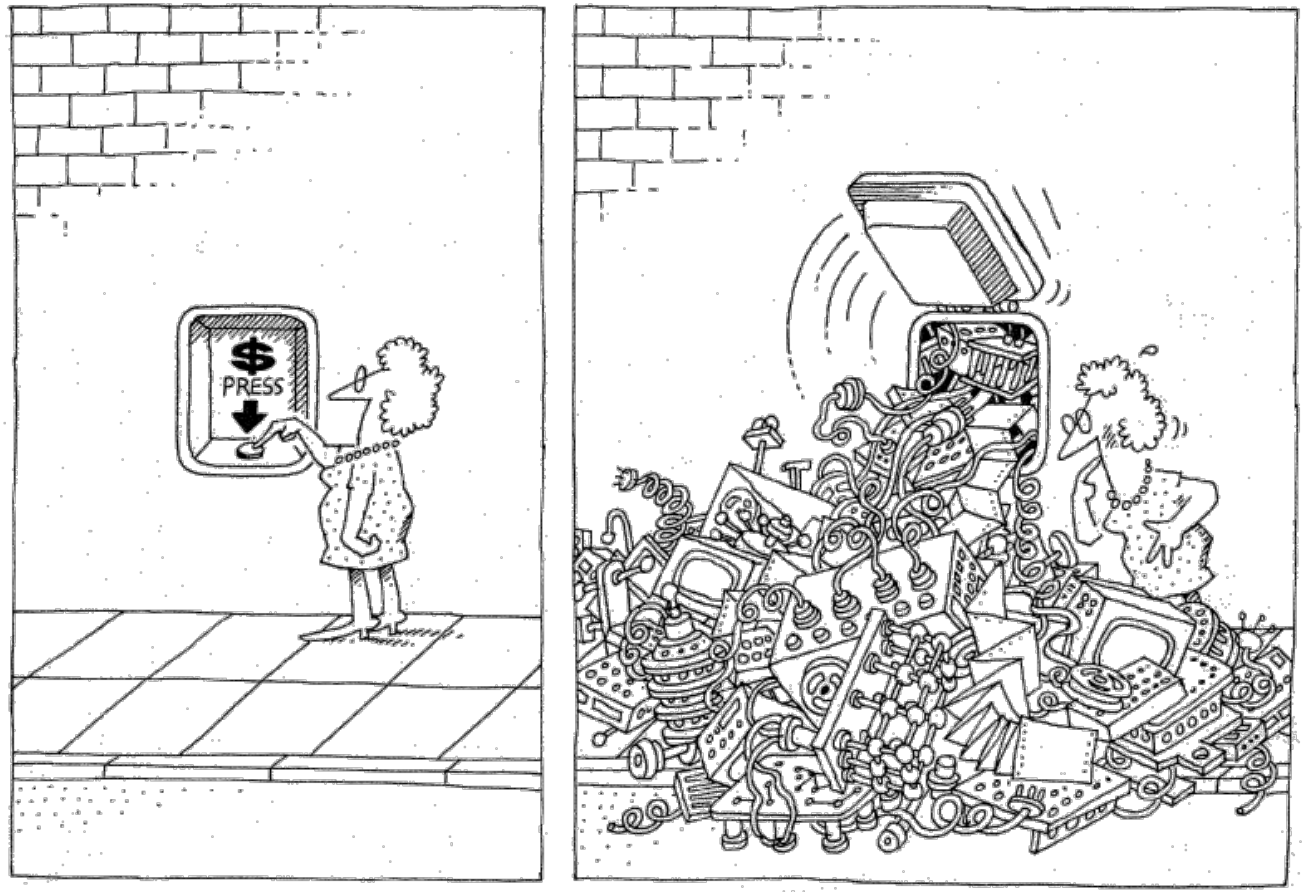


Figure taken from a presentation by Grady Booch 2007

Complexity

JOHANNES KEPLER UNIVERSITY LINZ

"Software entities are more complex for their size than perhaps any other human construct" (F. Brooks)

- No two software parts are alike
- Complexity grows non-linearly with size due to interactions

- Royce: "for a $5M procurement, need a 30-page spec for hardware, and a 1500-page spec for software"

Slides taken from Barry Boehm

Conformity

JOHANNES KEPLER UNIVERSITY LINZ

- Software has to interface with existing systems
- Software "must conform because it is perceived as the most conformable" (Brooks), often the "last kid on the block"

Example: Intel Pentium Bug

- X = 4195835, y = 3145727, and z = x – (x/y)*y.
- Computed exactly, the answer, z, should be zero. The Pentium chip gave 256 as the answer.
- Various software patches were produced by manufacturers to work around the bug.
- Spectre and Meltdown are more recent examples requiring software patches

https://en.wikipedia.org/wiki/Pentium_FDIV_bug#Software_patches

Changeability

- All successful software gets changed. When a client asks ‘can you make it do ‘, the answer, ultimately, is ‘yes, we can’.

Benefit and Doom

- Implication: Software is ever expanding
  - ⇒ Can we resist the temptation?
  - ⇒ We know from history that **things break** when they get **too big**

Changeability

JOHANNES KEPLER UNIVERSITY LINZ

Vasa Ship und the story about failed change management

Invisibility

JOHANNES KEPLER UNIVERSITY LINZ

"In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable." (Brooks)

Barry Boehm's anecdote:

- Weights engineer on a spacecraft. Budget for software $3,000,000. So, how much weight is that? The Engineer was surprised to learn: None
- A week later, the weights engineer came back with a deck of punch cards. "Is this the software?" he asks. The software engineer said yes. The weights engineer said he would weigh the deck to determine the weight of the software. The software engineer said, the software is the "holes".

ISSE

Software Engineering vs Programming

JOHANNES KEPLER UNIVERSITY LINZ

- Teams of developers
- Complex systems
- Very long lifespan
- Many stakeholders:

Architect ≠ Developer ≠ Manager ≠ Tester ≠ Customer ≠ User

- System families
- Reuse to amortize costs
- Maintenance accounts for over 60% of overall development costs

- Single developer
- Small applications
- Short lifespan
- Few stakeholders:

Architect = Developer = Manager = Tester = Customer = User

- One-of-a-kind systems
- Built from scratch
- Minimal maintenance

ISSE

Growing Complexity: Software and Systems

JOHANNES KEPLER UNIVERSITY LINZ

Slide taken from a presentation by Rudolf Haggenmüller

- Ultra-Large Scale Systems
- Systems of Systems
- Software-Intensive Systems

...

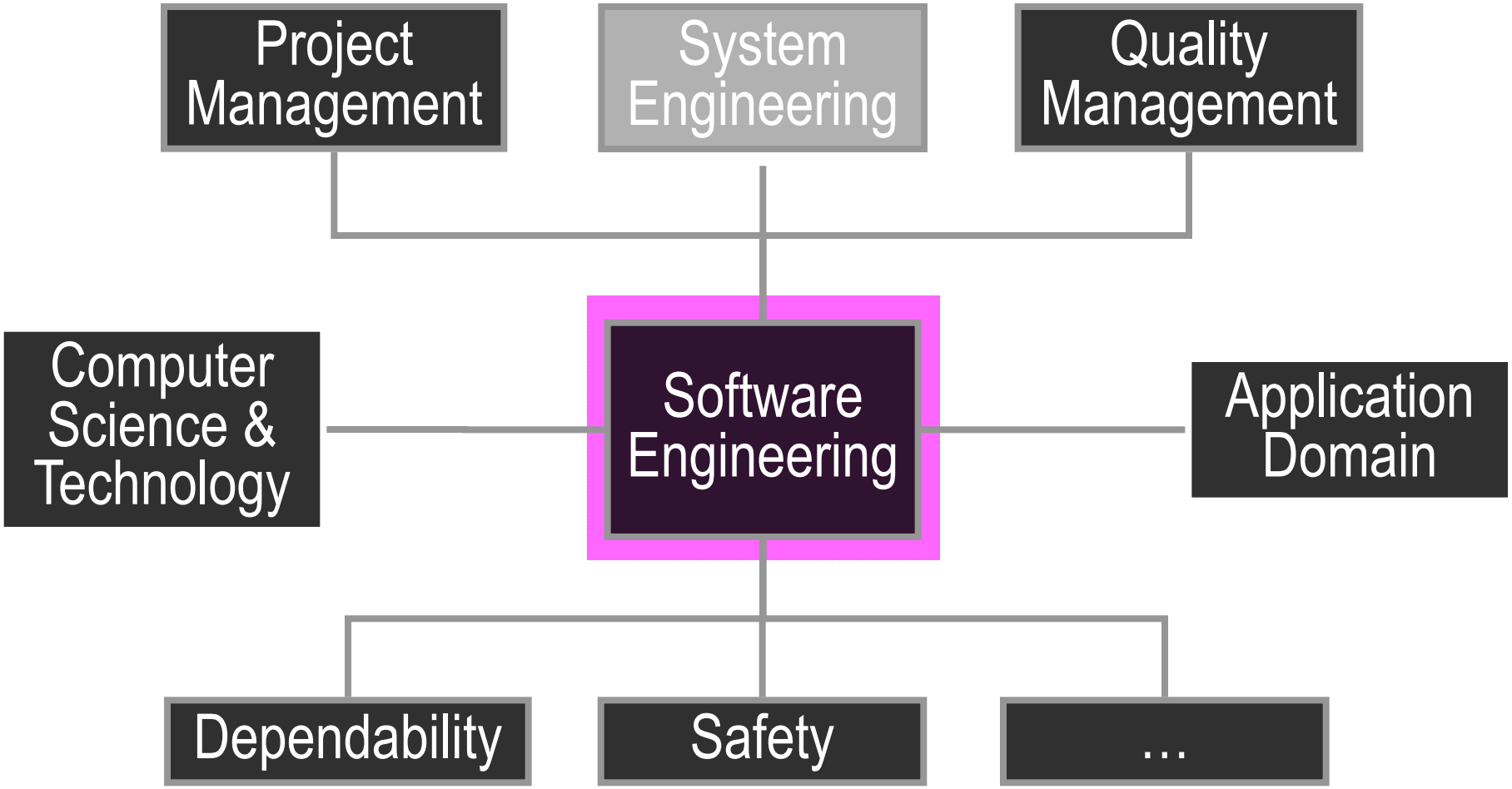
→ Systems Engineering

... *the Grand Challenge in Informatics* [Broy 2006]

- Increasing functionality and complexity
- Multi-disciplinary software development teams
- Important role of quality aspects
- Economy and cost-effectiveness of software development
- Tailored software systems
- Customizability during deployment

Software and Systems Engineering

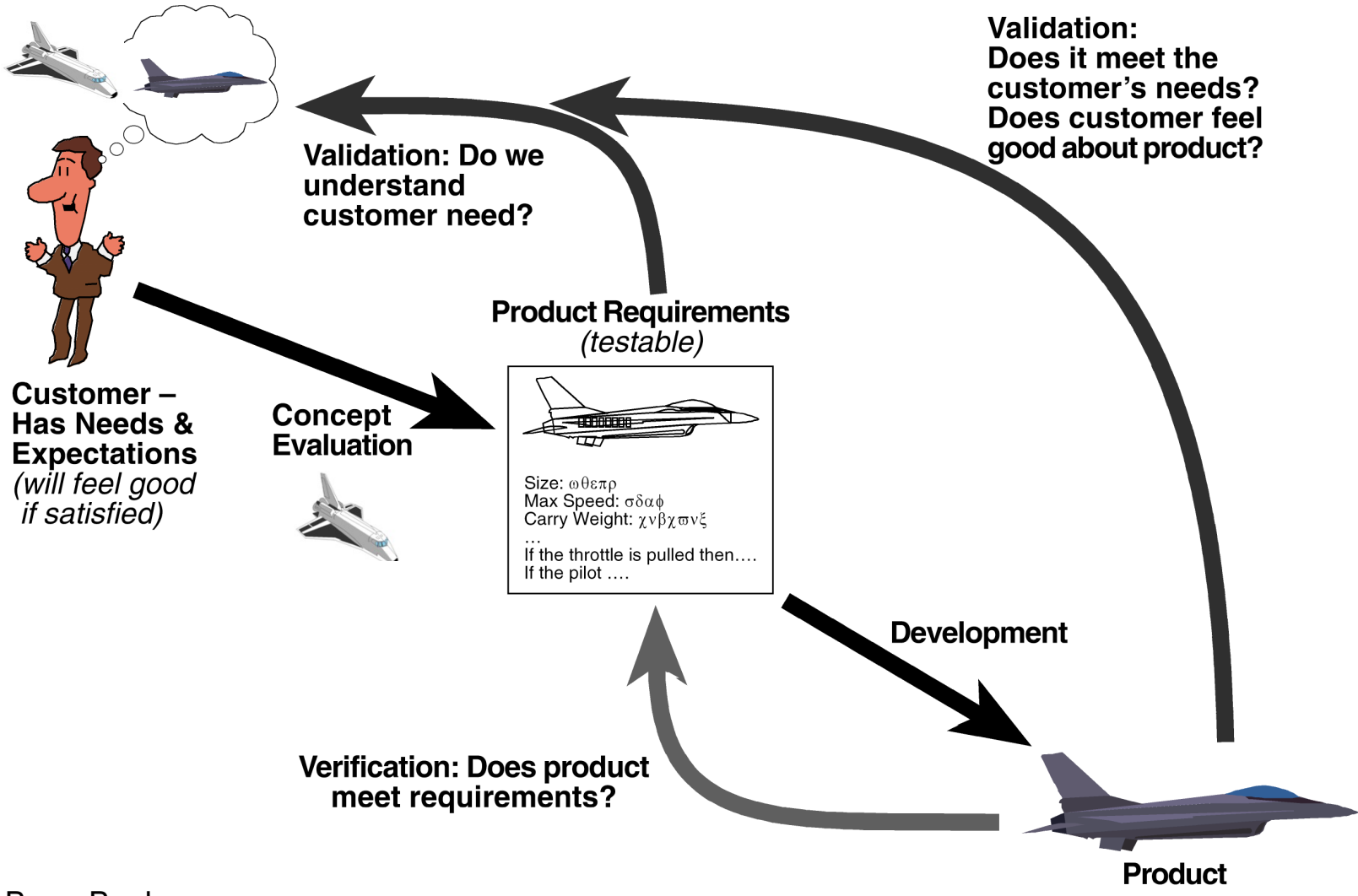
JOHANNES KEPLER UNIVERSITY LINZ



ISSE

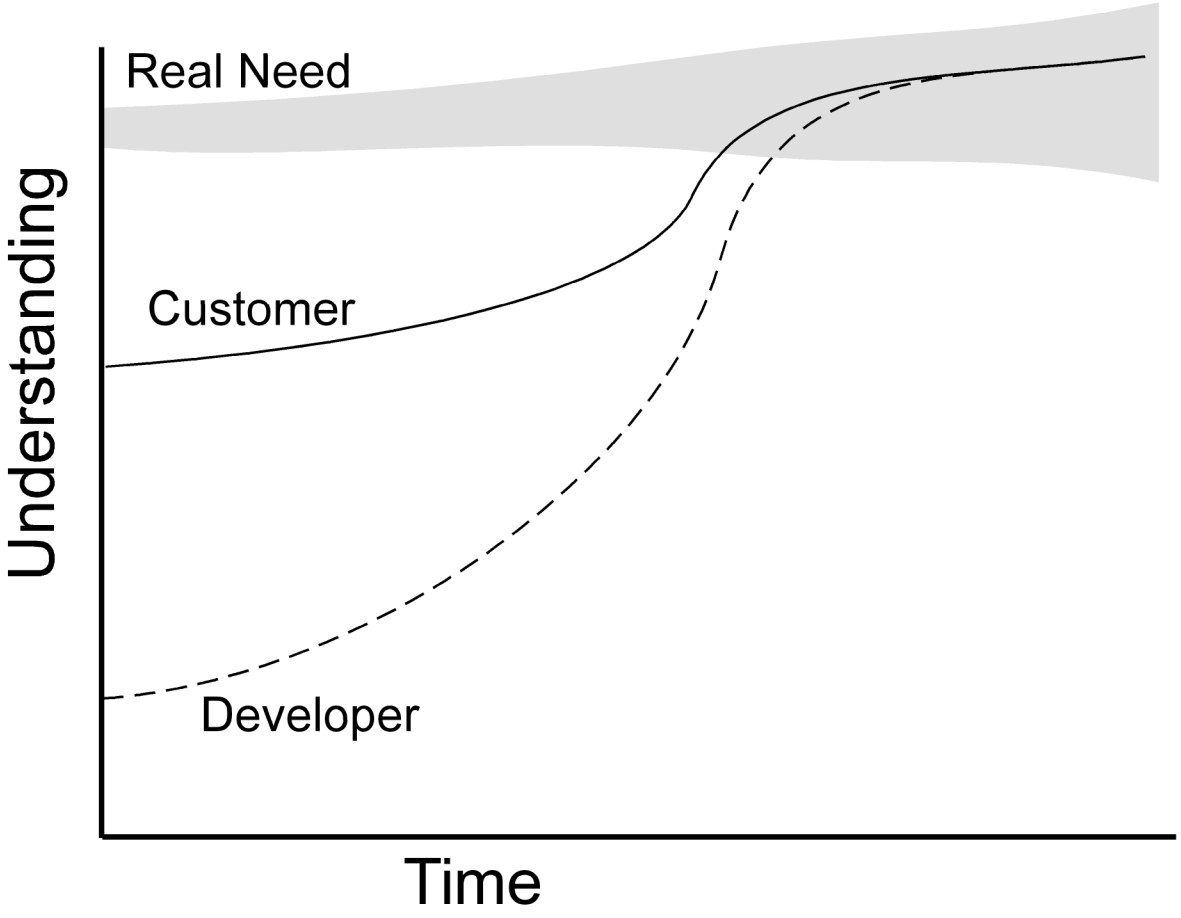
[IEEE SESC, 1998]

Ultimate Goal — Satisfied Customer



Slides taken from Barry Boehm

Developer's Problem — Understanding the Problem



- Need *Software Engineering Method*(s) that reduce time & effort to
  - Understand & solve the problem
  - Maintain the solution

Chinese Whispers ("Stille Post")

JOHANNES KEPLER UNIVERSITY LINZ

User tells Customer

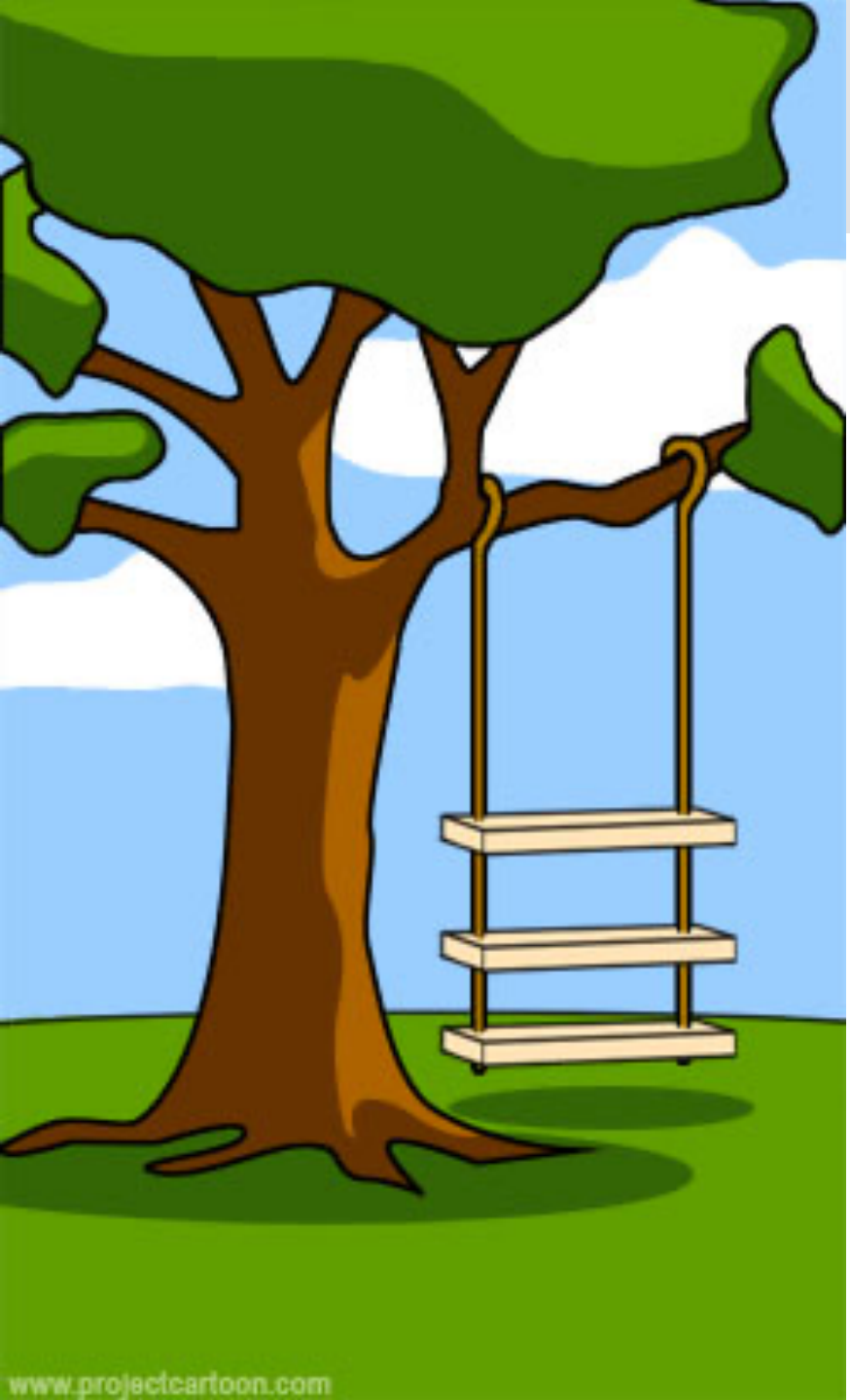
Customer talks to Project Leader

Project Leaders talk to Software Analysts

Software Analysts talk to Programmers

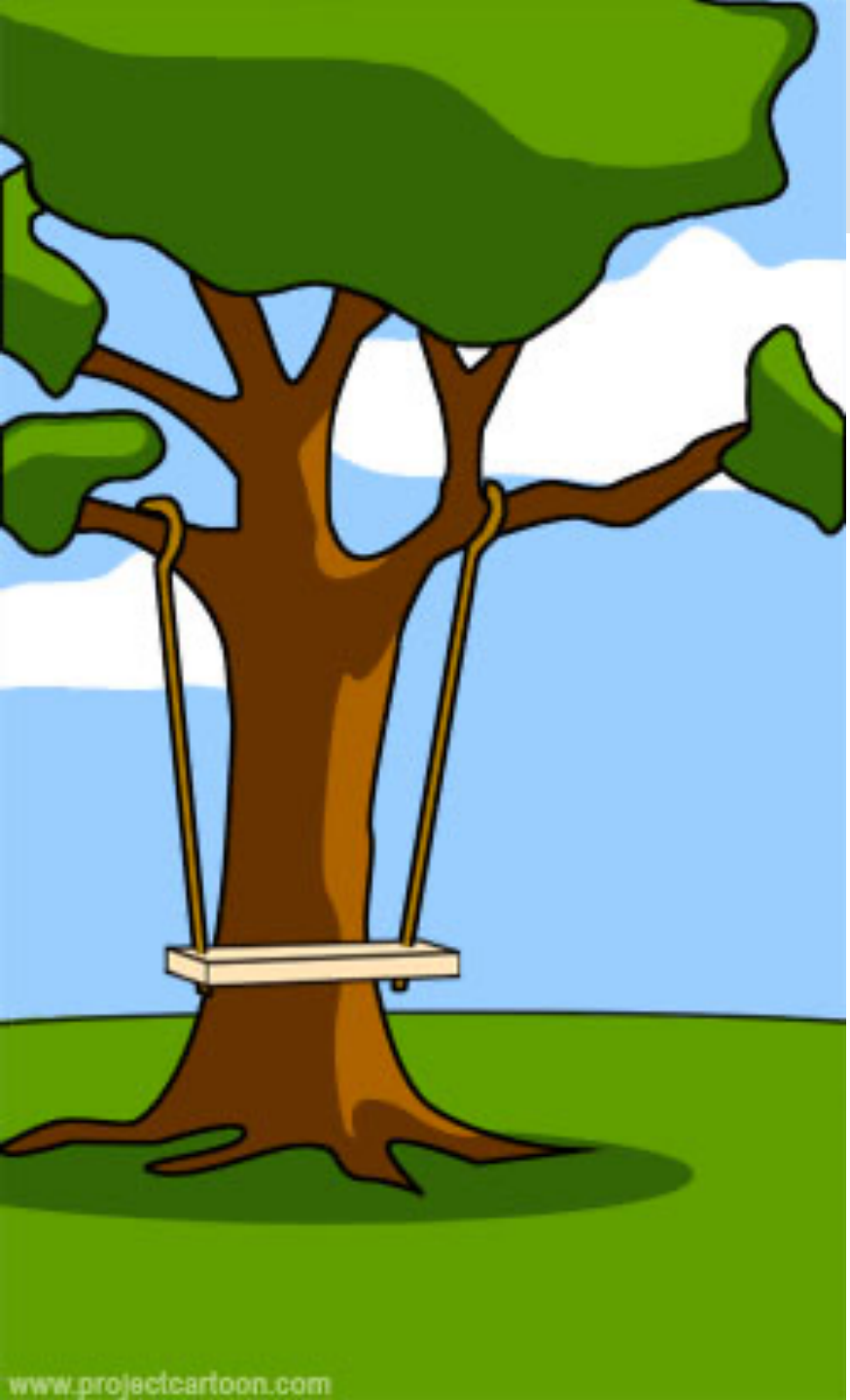
Do you think the programmer understood what the user asked for?

ISSE



JOHANNES KEPLER
UNIVERSITY LINZ

How the
Customer
explained it

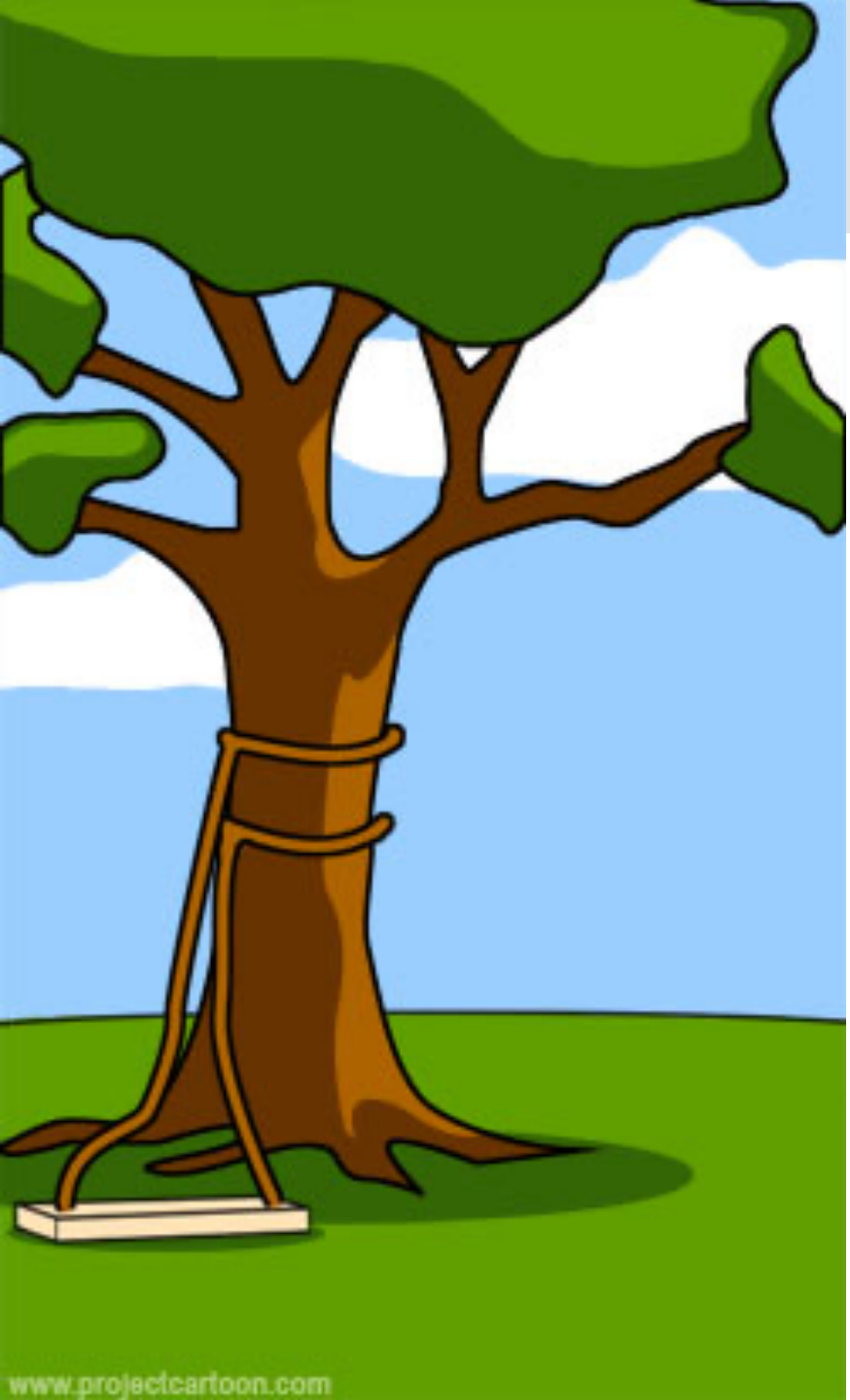


JOHANNES KEPLER
UNIVERSITY LINZ

How the
Project Leader
understood it

JOHANNES KEPLER
UNIVERSITY LINZ

How the
Analyst
designed it



JOHANNES KEPLER
UNIVERSITY LINZ

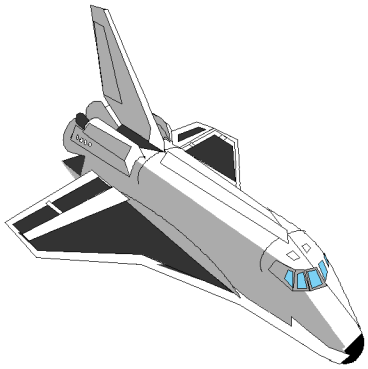
How the
Programmer
implemented it

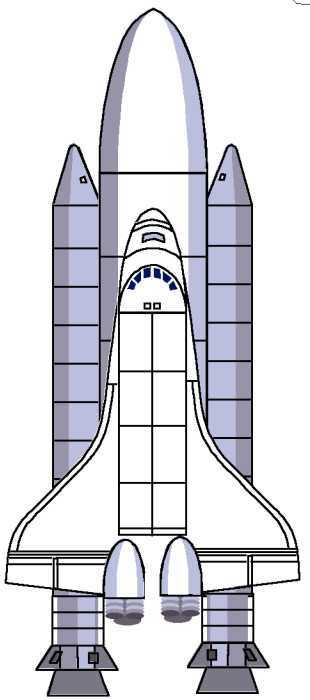
JOHANNES KEPLER
UNIVERSITY LINZ

What the
User
really needed

Developer's Problem

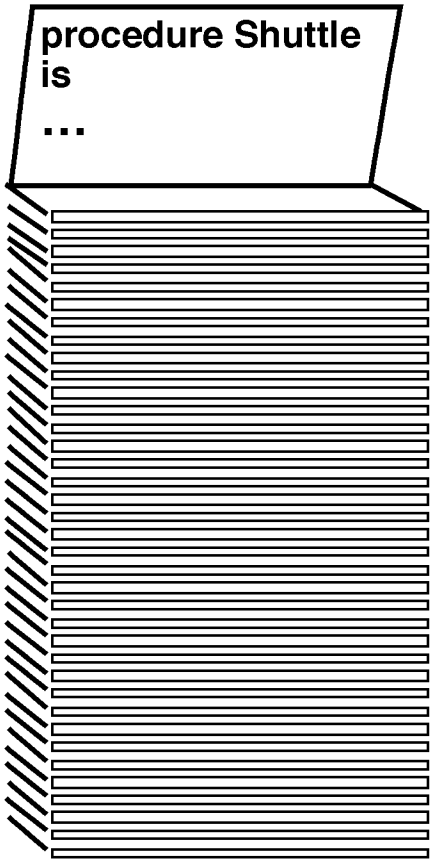
Real World





Need to take from concept to solution!

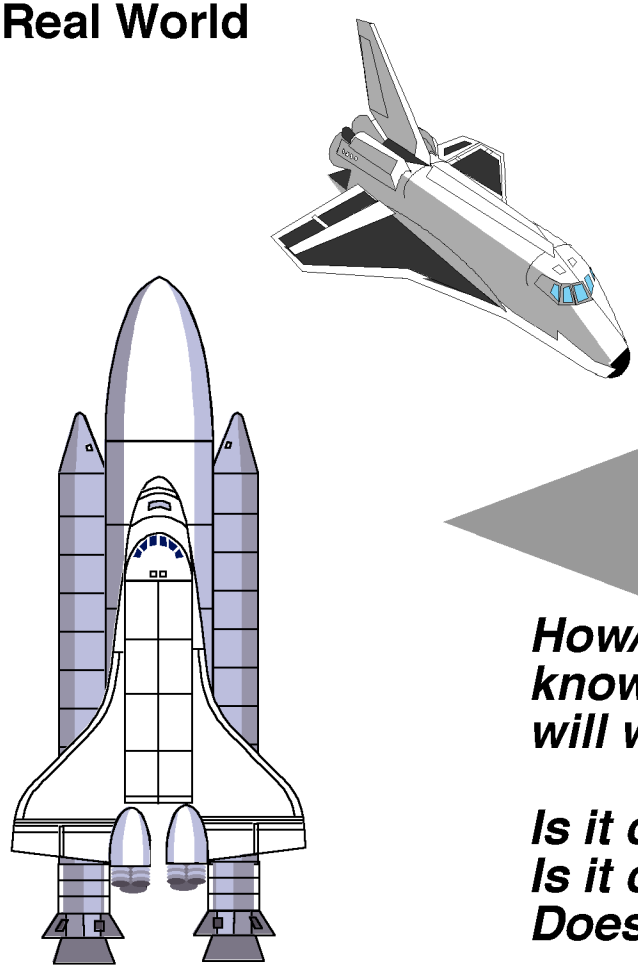
Solution Application



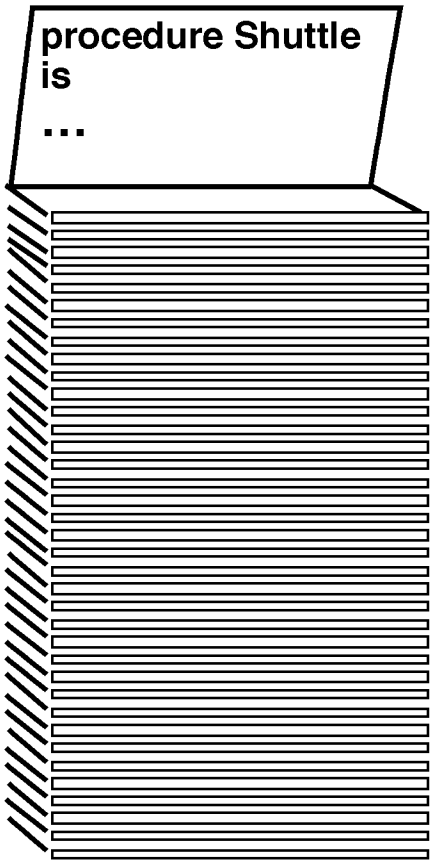
Developer's Problem (cont.)

JOHANNES KEPLER UNIVERSITY LINZ

Real World



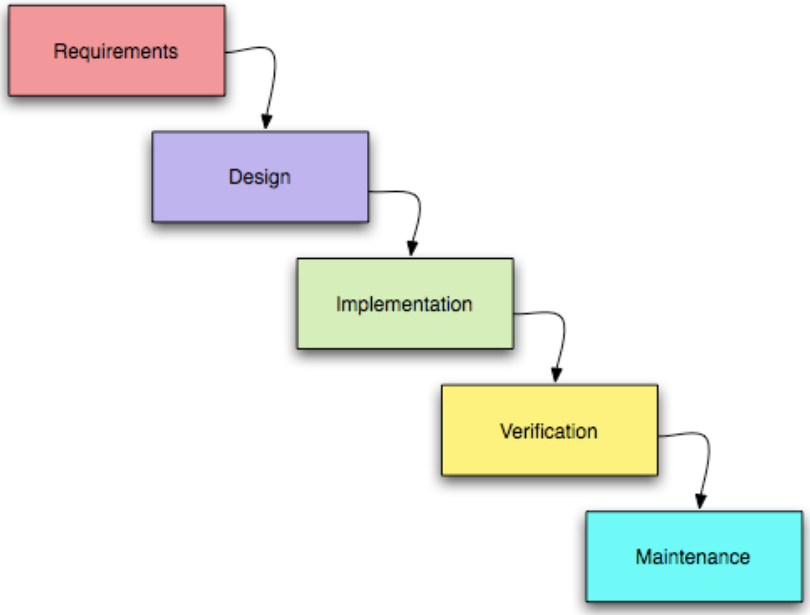
Solution Application

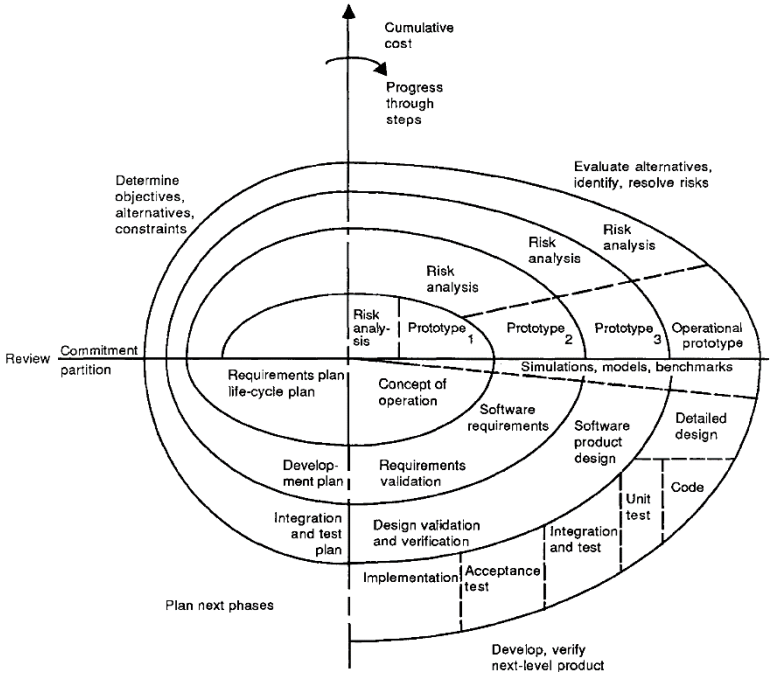


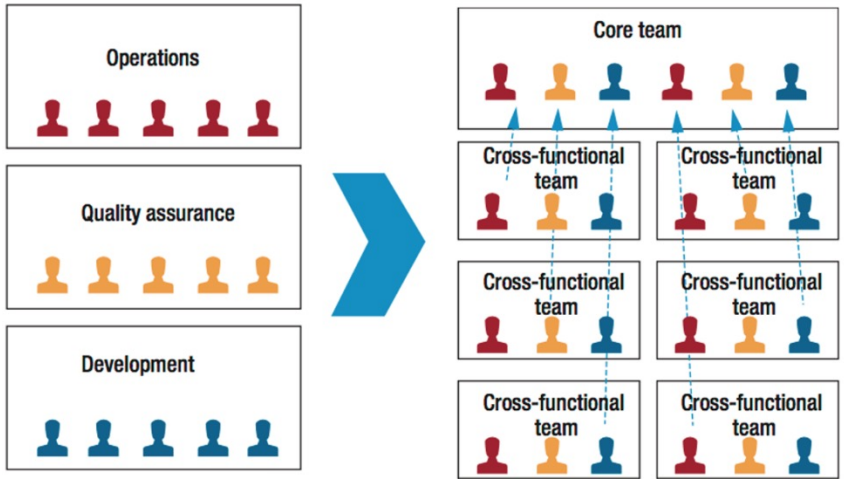
How/When do we know that program will work?

Is it complete?
Is it correct?
Does it meet goals?

Examples of Software Life Cycles

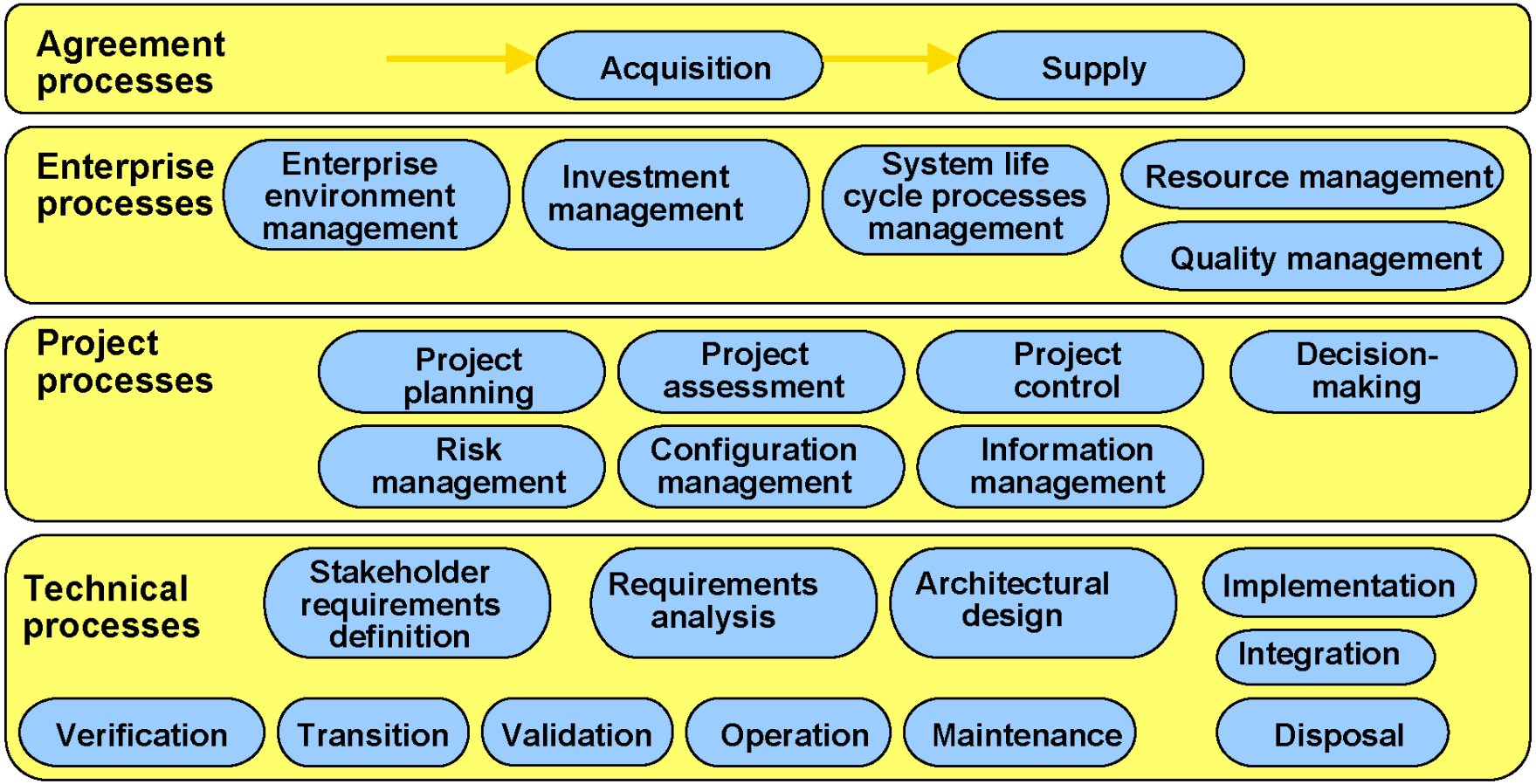






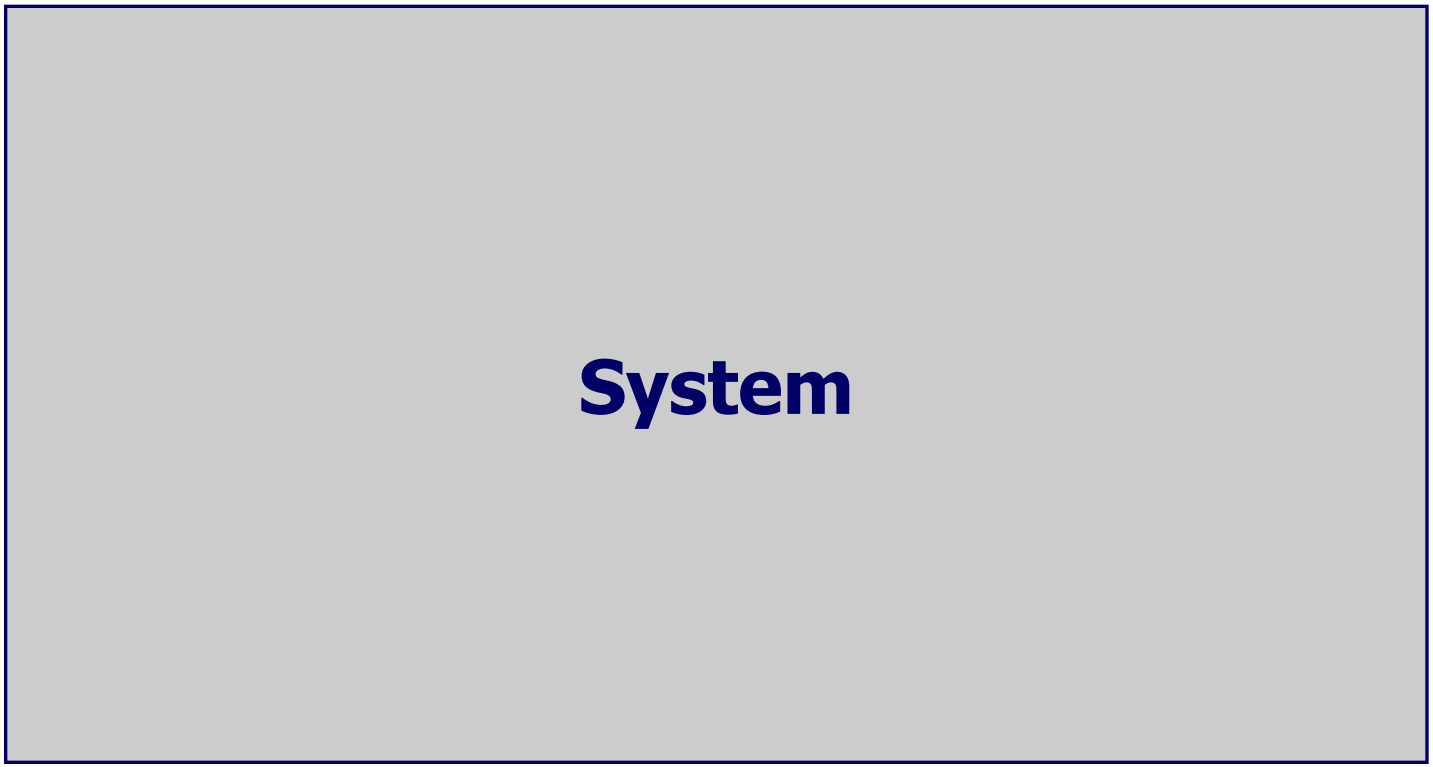
Typical Process Areas

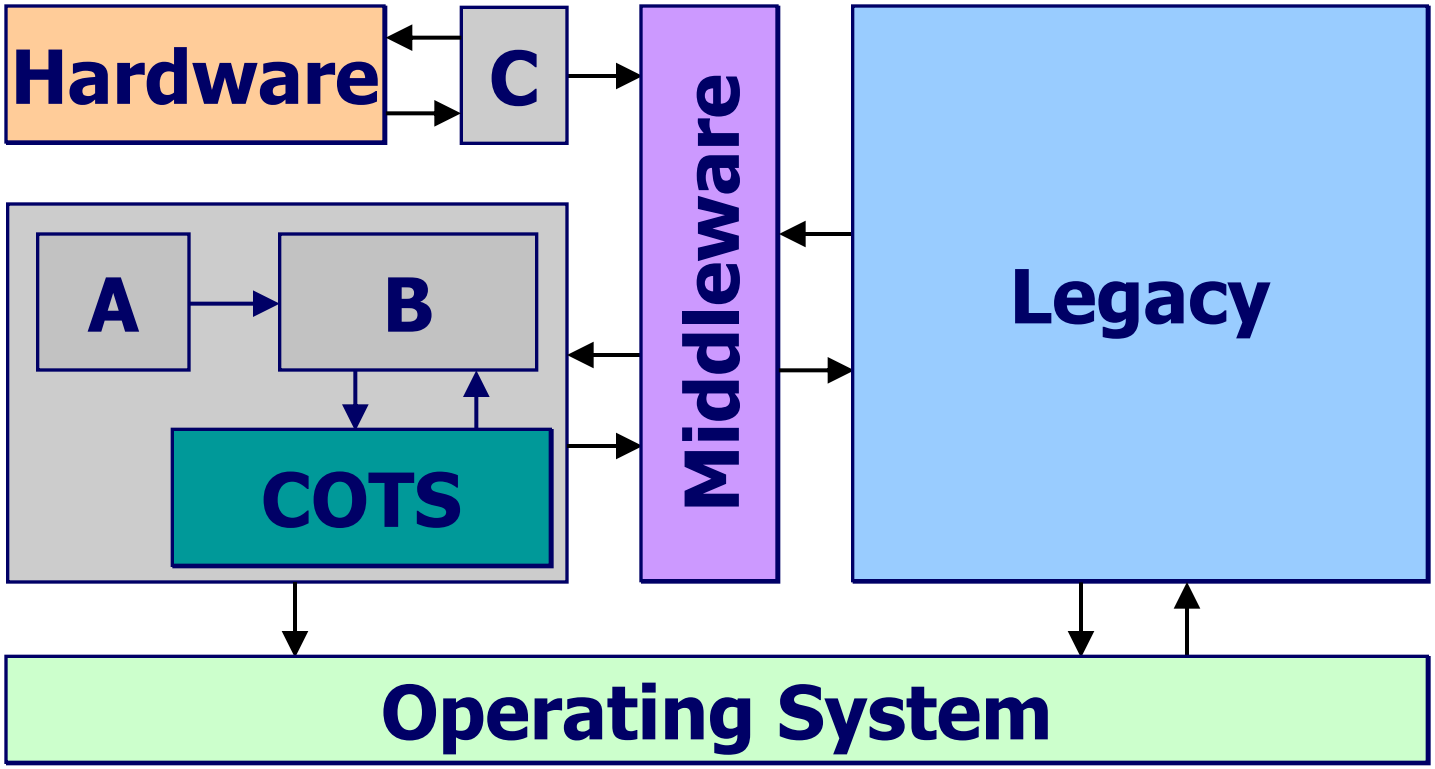
JOHANNES KEPLER UNIVERSITY LINZ



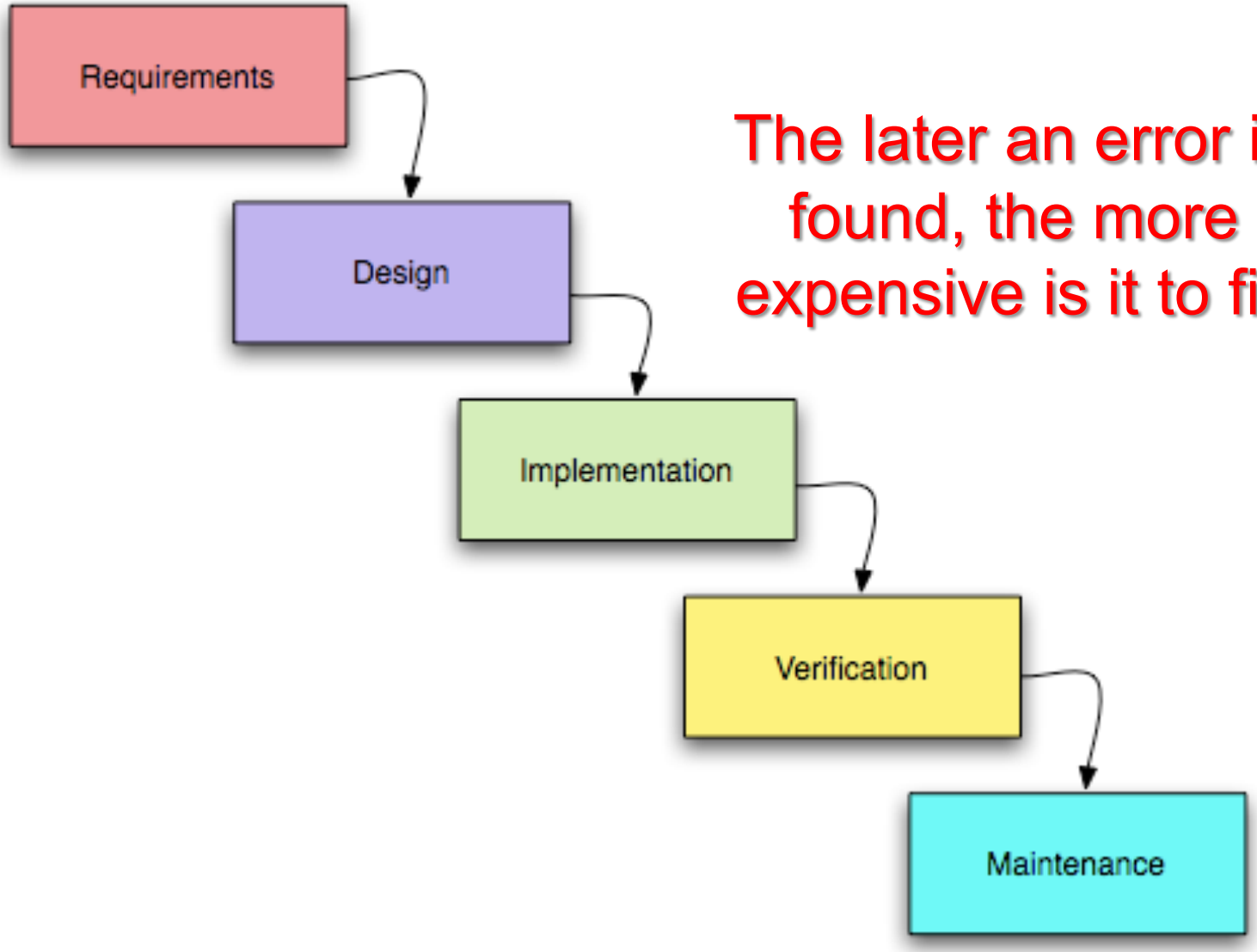
ISSE

ISO/IEC 15288 process framework


System


Hardware
C
Middleware
Legacy
A
B
COTS
Operating System

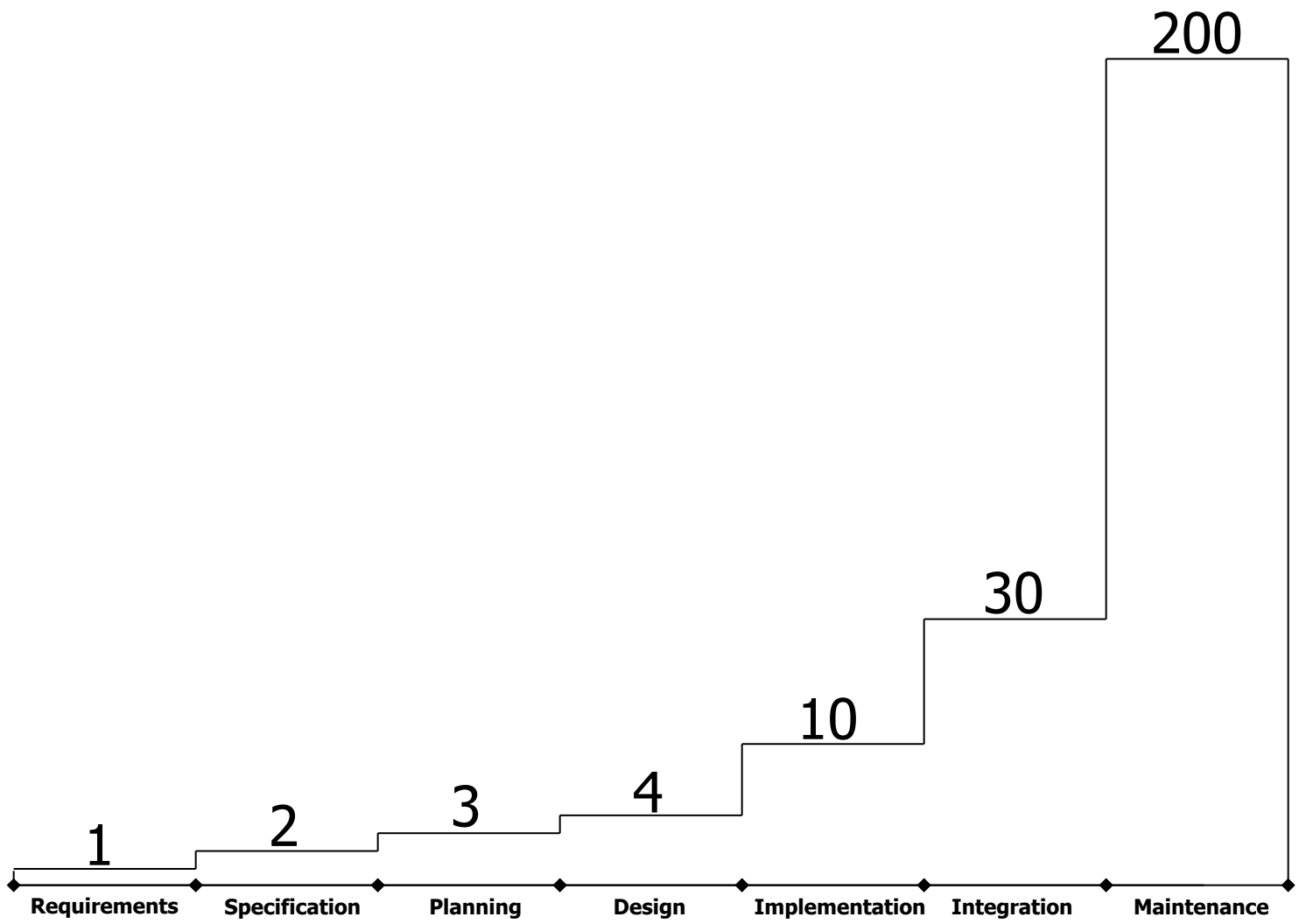
Waterfall Model



The later an error is found, the more expensive is it to fix

Slide based on Wikipedia

Relative Costs of Fixing Software Faults



Software costs

- Software costs often dominate system costs. The costs of software on a PC are often greater than the hardware costs
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs
- Software engineering is concerned with cost-effective software development

Economic and Management Aspects of SE

JOHANNES KEPLER UNIVERSITY LINZ

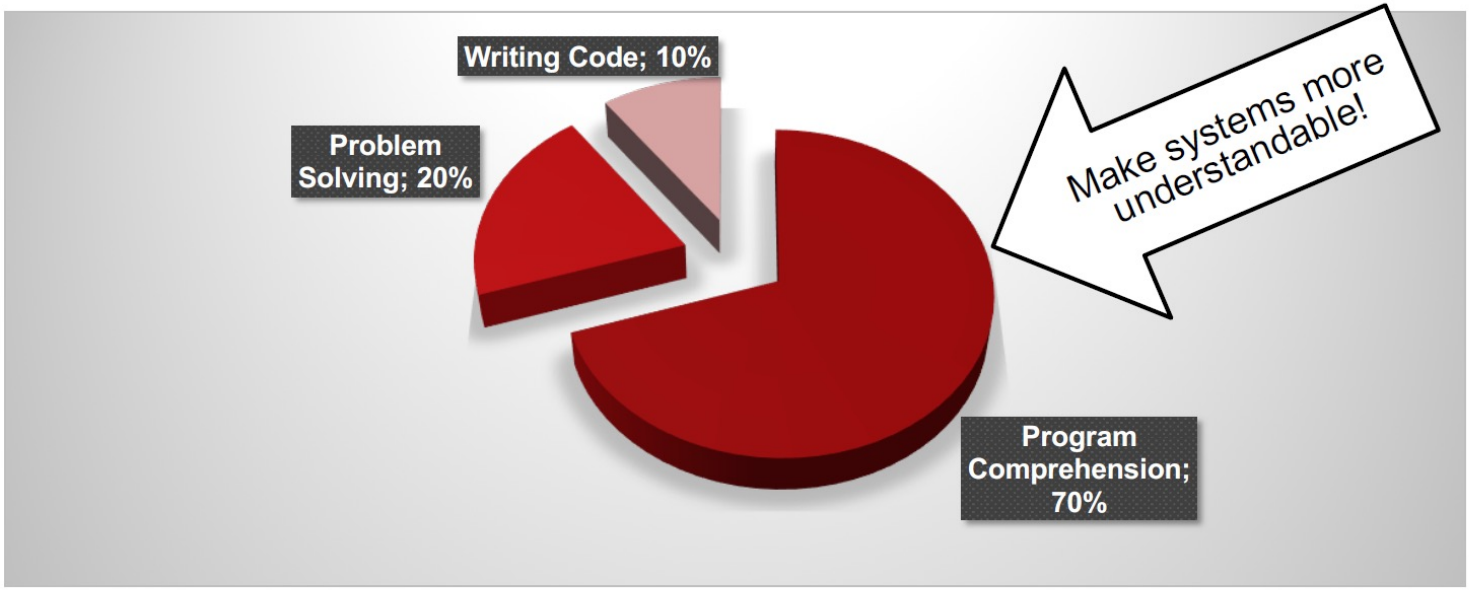
- **Software production = development + maintenance (evolution)**

- Maintenance costs > 60% of all development costs
  - —20% corrective
  - —30% adaptive
  - —50% perfective

- Quicker development is not always preferable
  - —higher up-front costs may defray downstream costs
  - —poorly designed/implemented software is a critical cost factor

ISSE

How Developers Spend their Time?



[L17] Lilienthal, Carola. Langlebige Software-Architekturen: Technische Schulden analysieren, begrenzen und abbauen. dpunkt. verlag, 2017.

To Satisfy A Customer, We Must…

JOHANNES KEPLER UNIVERSITY LINZ

- Build a software product that satisfies a human need & meets:

Operational System Objectives*	Life–cycle Objectives
•**Reliability** •**Efficiency** •**Suitability**[1] • **Must be verifiable & testable**	•**Understandability** •**Adaptability** •**Portability** •**Re–usability** •**Tunability** •**Plasticity** •**Maintainability**
Development Process's (Economic) Objective •**Cost–Effectiveness** •**Productivity** •**Return on investment**	

[1] Also called "appropriateness"

i.e. achieve "Software Engineering & System Building Goals"

ISSE

What are the Attributes of Good Software?

- Software should deliver the required functionality and performance, and should be:
- Maintainability
  - —Software must evolve to meet changing needs
- Dependability
  - —Software must be trustworthy
- Efficiency
  - —Software should not waste system resources
- Usability
  - —Software must be usable by the users for which it was designed
- There are many others!

Exercises

Key points

JOHANNES KEPLER UNIVERSITY LINZ

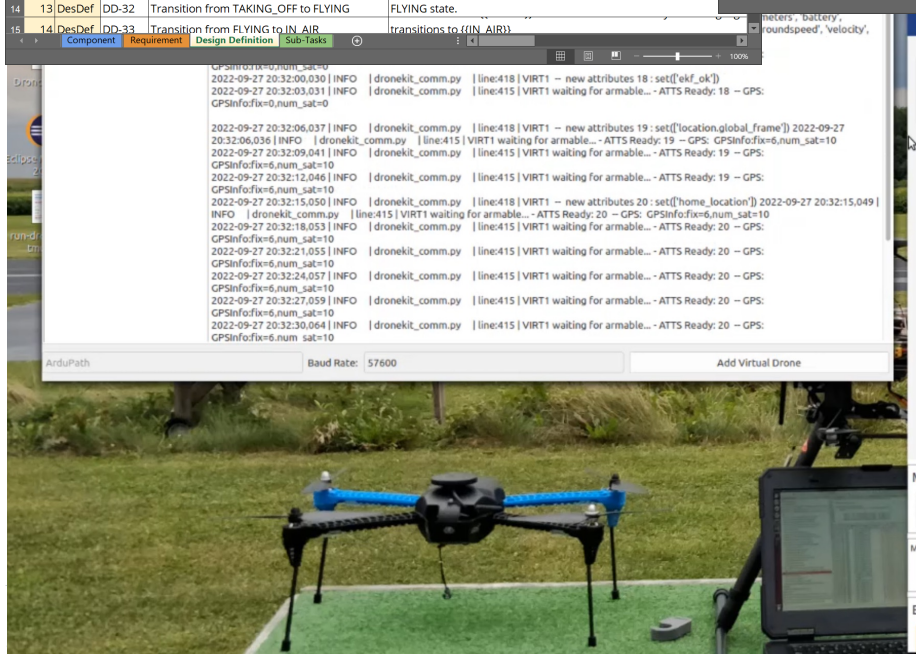
2022-06-20 Dronology dataset.xlsx - Excel

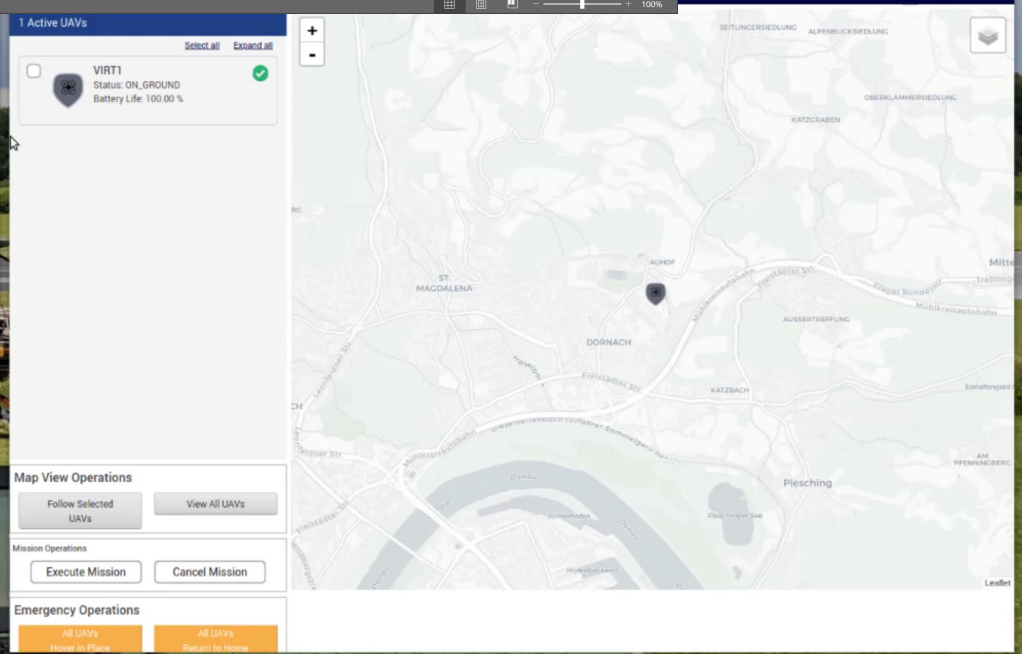
Nr.	Type	ID	Name	Description
1	Req	RE-8	UAV State transitions	When requested the _VehicleCore_ shall shall transition the UAV betv
2	Req	RE-9	ListenerNotifications	If a client registers for notifications the _UIMiddleware_ shall send nc
3	Req	RE-25	UAV Identification	The _VehicleCore_ shall assign a unique ID to each activated UAV.
4	Req	RE-28	UAV Flight Plan Execution	The _SingleUAVFlightPlanScheduler_ shall only execute one flight pla
5	Req	RE-36	Only one instance of each registered drone	The _UAVActivationManager_ shall ensure that only one instance of e
6	Req	RE-38	UAV type specification assignment	The _VehicleCore_ shall associate a {{UAV_TYPE}} for each UAV defini
7	Req	RE-77	Waypoint definition	Each waypoint shall be specified using coordinates that include longi
8	Req	RE-80	Visiting waypoints during flight plan	When a flight plan is executed the _VehicleCore_ shall send the next
9	Req	RE-86	Managed flight plan queue	The _SingleUAVFlightPlanScheduler_ shall maintain a managed list of
10	Req	RE-100	Removal of planned flight	If requested the _SingleUAVFlightPlanScheduler_ shall unassign the f
11	Req	RE-101	Assigning new flight plan upon completion of	If a UAV has pending flight plans then when it reaches the final wayp
12	Req	RE-103	Remove pending plans when return-to-home	When a UAV's current flight is cancelled the _SingleUAVFlightPlanSch
13	Req	RE-108	UI middleware accepts registrations	The _UIMiddleware_ shall provide an interface for registering UI clier
14	Req	RE-110	UAV controlled by only one UI client at a time	The _UIMiddleware_ shall only assign one control-client at a time for
15	Req	RE-112	UI binding to a UAV	If a command is issued by the _VaadinUI_ for an UAV the _VaadinUI_
16	Req	RE-114	Display map in UI	When the _RealTimeFlightUI_ is loaded a map shall be displayed.
17	Req	RE-120	Real-Time Flight UI	The _RealTimeFlightUI_ shall display the location of active UAVs as ico
18	Req	RE-124	Object Avoidance	When a stationary or moving object is within the proximity of a UAV t
19	Req	RE-125	Reach destination while avoiding objects	If a UAV takes evasive actions the system will ensure that it will event
20	Req	RE-126	Collision avoidance activation	The _ObjectAvoidance_ system only issues commands within the cap
21	Req	RE-127	Onboard obstacle avoidance	When a UAV has an active onboard Obstacle Avoidance then the _OA
22	Req	RE-160	Activation of an additional UAV detected by	When a _GCS_ requests activation of a newly detected UAV the _GCS
23	Req	RE-161	UAV activation	The _UAVActivationManager_ shall only activate UAVs that are registe
24	Req	RE-235	Allow reconnection of GCS to the UAV after	When the _GCS_ loses the connection to an UAV the _GCS_ shall atter
25	Req	RE-424	Return To Home without detours	When the ""return to home"" command is issued the _FlightManager
26	Req	RE-501	Flight route defined as waypoints	The _FlightRouteManager_ shall define flight routes as an ordered se
27	Req	RE-503	Valid Flight Route	The _FlightRouteManager_ shall ensure that all flight routes represer
28	Req	RE-505	Flight route persistence	The _FlightRouteManager_ shall persist flight routes.
29	Req	RE-508	Specify takeoff altitude	When a flight route requires a customized takeoff altitude that differ
30	Req	RE-510	Coordinate system conversion	When requested the _CoordinateSystem_ shall transform coordinate

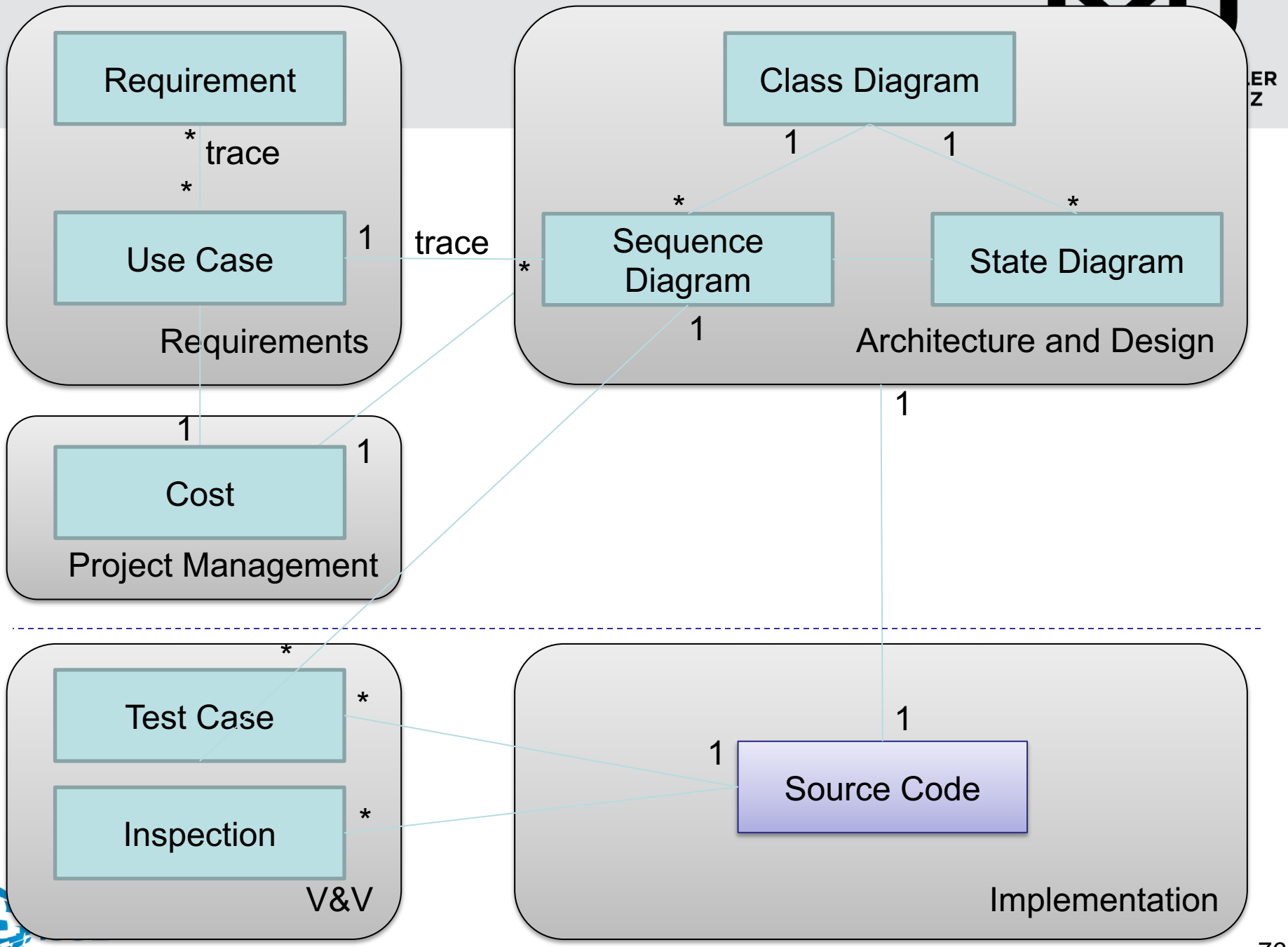
Component | Requirement | Design Definition | Sub-Tasks

Nr.	Type	ID	Name	Description
1	DesDef	DD-11	Client registers for flight route events	A client shall register with the _UIMiddleware_ to receive flight rc notifications whenever a flight route is created modified or delet
2	DesDef	DD-18	Client registers for UAV type specification events	A client shall register with the _UIMiddleware_ to receive UAV typ events whenever a new UAV type specification is created modifi
3	DesDef	DD-19	Client registers for UAV configuration event	A client shall register with the _UIMiddleware_ to receive notifica new UAV type configuration is created modified or deleted.
4	DesDef	DD-20	Client registers for flight plan events	A client shall register with the _UIMiddleware_ to receive notifica new flight plan is activated or executed
5	DesDef	DD-21	Client registers for UAV activation eventsClient registers for UAV activation events	A client shall register with the _UIMiddleware_ to receive notifica UAV instances are activated or deactivated.
6	DesDef	DD-22	Client registers for emergency events	A client may register with the _UIMiddleware_ to receive notifica unexpected events occur.
7	DesDef	DD-23	Coordinate Specification	All waypoints shall be specified using degrees longitude and lati Degrees format (e.g. 38.6556758 -77.7872153)
8	DesDef	DD-24	Coordinate Units	Altitude shall be specified in meters
9	DesDef	DD-26	Activation state is ON_GROUND	When a UAV is initially activated on the ground it is in the {{ON_
10	DesDef	DD-27	UAV Flight Plan Management	Each UAV has zero or more pending flight plans.
11	DesDef	DD-30	Transition from ON_GROUND to AWAITING_TAKEOFF_CLEARANCE	When a flight plan is assigned to a UAV in the {{ON_GROUND}} s the {{AWAITING_TAKEOFF_CLEARANCE}} state.
12	DesDef	DD-31	Transition from AWAITING_CLEARANCE to TAKING_OFF	When permission is granted to UAV in the {{AWAITING_TAKEOFF state it transitions to {{TAKING_OFF}} state.
13	DesDef	DD-32	Transition from TAKING_OFF to FLYING	FLYING state.
14	DesDef	DD-33	Transition from FLYING to IN_AIR	transitions to {{IN_AIR}}

Component | Requirement | Design Definition | Sub-Tasks






Requirement
*
trace
*
Use Case
1
Requirements
trace
*
Class Diagram
1
1
*
*
Sequence Diagram
State Diagram
1
Architecture and Design
1
1
Cost
1
Project Management
*
Test Case
*
Inspection
*
V&V
1
1
Source Code
Implementation

	lectures focus		Excercises
05/10/2022	Introduction to SWE+Dronology Overview	Individual	E1: Familiarize with Dronology (Read ICSE Nier Paper)
12/10/2022	Requirements, Use Cases, and Scenarios	Individual	E2: Use Case Analysis
19/10/2022	Requirements Syntax and Traceability	Team	E3: Tracing Use Cases to EARS Requirements
26/10/2022	Holiday		
02/11/2022	Holiday		
09/11/2022	Design Modeling	Individual	E4: Requirements Engineering Mission Planning (Input: Screen Cast of Mission Planning UI Prototype / Requirements)
16/11/2022	Design Modeling	Team	E5: Define Architecture and Design (without considering E4; Input: Architecture Documentation for Main Components)
23/11/2022	Software Process		
30/11/2022	Design Evolution	Individual	E6: Define Mission Planning Extension
07/12/2022	Planning and Cost Estimation	Individual	E7: Mission Planning Cost Estimation (Provide a Rationale for the Cost Parameters)
14/12/2022	Verification and Validation	Team	E8: Quality Assurance (Inspection, Check System Against E6 of Another Team Member)
21/12/2022	Holiday		
28/12/2022	Holiday		
04/01/2023	Holiday		
11/01/2023	Testing and Monitoring	Individual	E9: Test Case Design (Gherkin)
18/01/2023	Software Reuse and Product Lines	Individual	Rather In-Class Exercise
25/01/2023	Designing in the Ilities		

Concluding

Key points

- Software engineering is an engineering discipline which is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation. Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities which are involved in developing software products. Basic activities are software specification, development, validation and evolution.
- Methods are organised ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

In Short

JOHANNES KEPLER UNIVERSITY LINZ

- In short…
  - —Good programming skills will get you a job
  - —Good software engineering skills will get you a career
- Start with this course
- Continue with the Master's in Software Engineering
- Talk to us about your Project and Thesis

ISSE

Challenge Yourself

JOHANNES KEPLER UNIVERSITY LINZ

- Project in Software Engineering
- Bachelor's Thesis (Projektpraktikum)
- Master Thesis

- **You can be a part of this!**

ISSE