

JOHANNES KEPLER
UNIVERSITY LINZ | JKU



Model-Driven Software Development

Bridging the Gap between Requirements, Model, and Code

Univ.-Prof. Dr. Alexander Egyed

Johannes Kepler Universität, Linz (JKU)
http://www.sea.jku.at

1

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

Who am I?

Current Affiliations:

- Professor at **Johannes Kepler University**, Austria 2008
- Head of **Institute for Systems Engineering and Automation** (14 Staff Members)

Doctorate Degree:

- University of Southern California**, USA 2000 (under Dr. Barry Boehm)

Past Affiliations:

- Research Fellow at **University College London**, UK 2007
- Research Scientist at **Teknowledge Corporation**, USA 2000

SEA © 2009 Alexander Egyed

2

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

I was involved in:

Command and Control of Unmanned Air Vehicles
with Northrop-Grumman, Honeywell, Sarnoff

Target Detection and Tracking in a Sensor Grid
with Mitre Corp.

Embedded Avionics Product Line Architecture
with Boeing Company and Carnegie Mellon University (CMU)

Architectural Differencing, Diagnosis, Recovery, and Adaptivity
with Massachusetts Institute of Technology (MIT)

SEA © 2009 Alexander Egyed

3

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

Cost of Software Error

1995 US spending on S/W projects:
US\$ 250 billion

Cost overruns: **US\$ 59 billion**

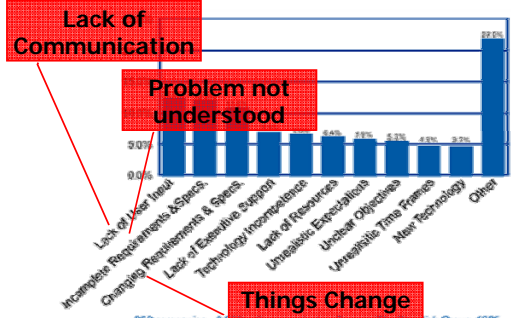
Failure/cancelled projects: **US\$ 81 billion**

SEA © 2009 Alexander Egyed

4

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

Why Software Projects Fail



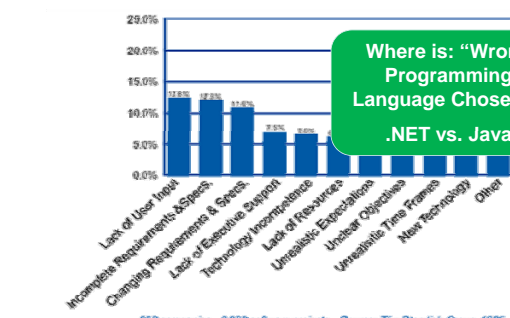
Reason	Percentage
Lack of Communication	29.6%
Problem not understood	17.8%
Things Change	17.8%
Lack of User Input	17.8%
Incomplete Requirements & Specs	17.8%
Changing Requirements & Specs	17.8%
Lack of Executive Support	17.8%
Technology Incompatibility	17.8%
Lack of Resources	17.8%
Unrealistic Expectations	17.8%
Unclear Objectives	17.8%
Unrealistic Time Frames	17.8%
New Technology	17.8%
Other	17.8%

SEA © 2009 Alexander Egyed

5

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

Why Software Projects Fail



Reason	Percentage
Lack of User Input	17.8%
Incomplete Requirements & Specs	17.8%
Changing Requirements & Specs	17.8%
Lack of Executive Support	17.8%
Technology Incompatibility	17.8%
Lack of Resources	17.8%
Unrealistic Expectations	17.8%
Unclear Objectives	17.8%
Unrealistic Time Frames	17.8%
New Technology	17.8%
Other	17.8%

SEA © 2009 Alexander Egyed

6

Software Defects “enjoy” high Visibility

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- 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 capsizes in corners [Mercedes97]
- Ariane 5: code reuse and the dimensions of variables
- Denver Airport Baggage Handling: Reset lost data

SEA © 2009 Alexander Egyed 7

Embarrassing Software Defects

JOHANNES KEPLER UNIVERSITY LINZ | JKU

How many companies can afford to let the customer test their software?

SEA © 2009 Alexander Egyed 8

Essential Software Engineering Difficulties (Brooks)

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Complexity**
 - no two software parts are alike
 - complexity grows non-linearly with size
- 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 geometric entity

SEA © 2009 Alexander Egyed 9

Slide taken from Grady Booch

SEA © 2009 Alexander Egyed 10

Models take on Difficulties

JOHANNES KEPLER UNIVERSITY LINZ | JKU

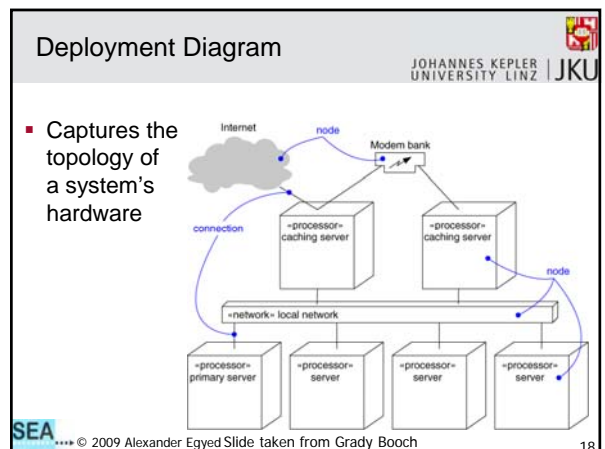
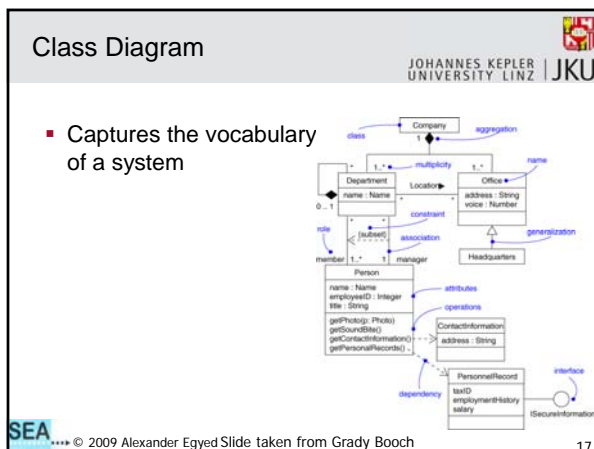
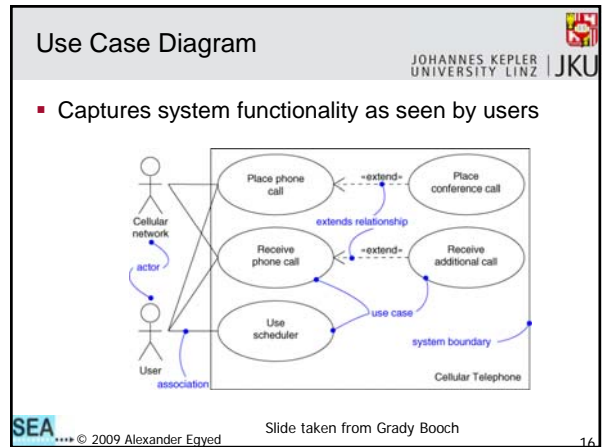
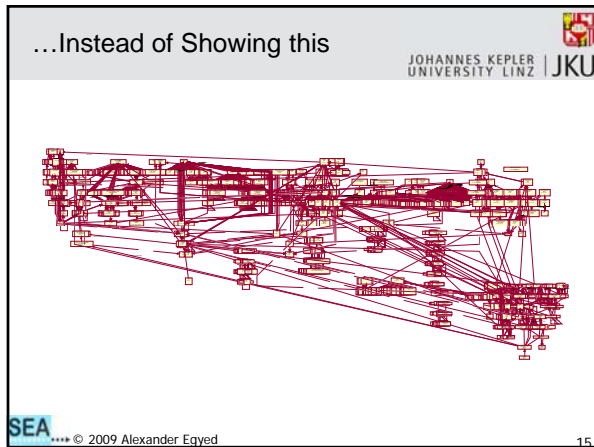
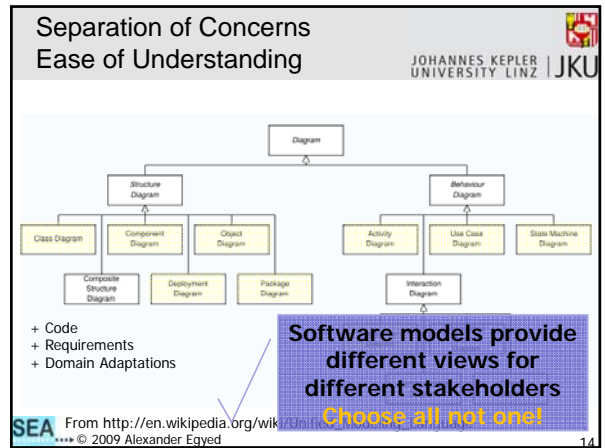
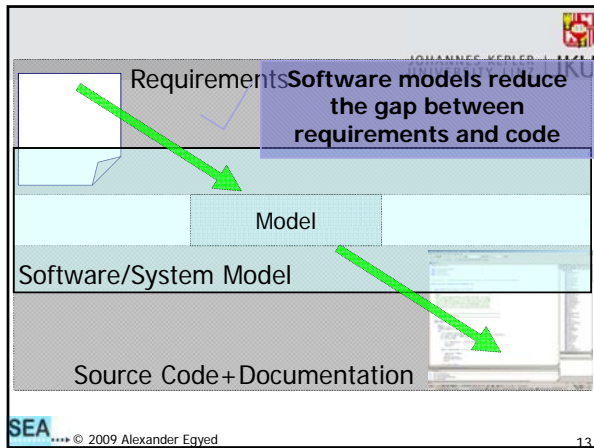
- Complexity ✓ **Software models abstract and simplify**
- Conformity ✓ **Software models show context**
- Changeability ✓ **Software models separate concerns**
- Invisibility ✓ **Software models depict what is otherwise hidden**

SEA © 2009 Alexander Egyed 11

Requirements

Source Code+Documentation

SEA © 2009 Alexander Egyed 12



Sequence Diagram

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Captures dynamic behavior (time-oriented)

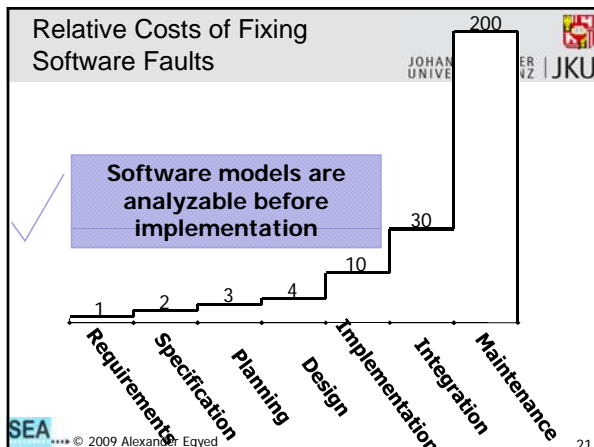
SEA © 2009 Alexander Egyed Slide taken from Grady Booch 19

Statechart Diagram

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Captures dynamic behavior (event-oriented)

SEA © 2009 Alexander Egyed Slide taken from Grady Booch 20



The Benefits of Modeling

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Overcome the requirements/code gap
- Reduce the Cost of Fixing Errors
- Exploring the Problem
- Separation of Concerns
- Ease of Understanding
- Maintenance
- To "hide" implementation details
- To "emphasize" limitations and assumptions
- To think the problem through
- Assess Requirement Changes
- Correct Implementation guarantees
- Analyzable
- Integrate disciplines

SEA © 2009 Alexander Egyed 22

BUT!

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Modeling tools are like a hammer

- Can do good but, if used wrong, they are useless!
- Require training, guidance, and automation

Have witnessed many companies "introducing" modeling only to fail because they do not understand model-driven software development

SEA © 2009 Alexander Egyed 23

Requirements

If software models no longer reflect the code or requirements...

Model

Software/System Model

Source Code+Documentation

SEA © 2009 Alexander Egyed 24

Model-Driven Software Development

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Change in paradigm

- => Modeling is the "creative" part (with automated refinement to code)
- => Coding is filling in details not in models
- => multi-view "programming"

Living Models

- Not documentation but path to solution
- Used/Useful throughout the software life cycle

SEA © 2009 Alexander Egyed 25

Entire Software Life Cycle

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- Software Cost = Development + Maintenance
- Maintenance is 60-80% of SW costs
- Quicker development is not always preferable

Software development is long term: models help remember important facts

SEA © 2009 Alexander Egyed 26

Success Stories

JOHANNES KEPLER UNIVERSITY LINZ | JKU

HP Product Reuse Investment and Payoff

Software modeling done wrong is a burden. Software modeling done right saves money, effort, and embarrassment

SEA © 2009 Alexander Egyed Slides taken from Barry Boehm 27

Model-Driven Software Development

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Automation:

- Ensuring Correctness
 - E.g., spell checker for modeling
- Propagate Changes
 - E.g., traceability, impact of a change
- Facilitate Reuse
 - E.g., product lines

SEA © 2009 Alexander Egyed 28

Change, Inconsistencies, and their Impact

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Changes cause Inconsistencies and we can help the developer detect and fix them!

SEA © 2009 Alexander Egyed 29

What Engineers Need

JOHANNES KEPLER UNIVERSITY LINZ | JKU

- (1) Identify Inconsistencies Quickly [ICSE 2006]
- (2) Fixing Inconsistencies Quickly [ASE 2008][ICSE 2007]

SEA © 2009 Alexander Egyed Anthony Finkelstein 30

Model Analyzer Approach

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Demo...

Model and Inconsistencies

SEA © 2009 Alexander Egyed 31

Some Consistency Rules

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Rule 1: Name of message must match an operation in receiver's class
`operations=message.receiver.base.operations`
`return (operations->name->contains(message.name))`

Rule 2: Sequence of object messages must correspond to events

Rule 3: Calling direction of association must match calling direction of messages

...

Rule 100+

Many rules – impossible to maintain consistency manually

SEA © 2009 Alexander Egyed 32

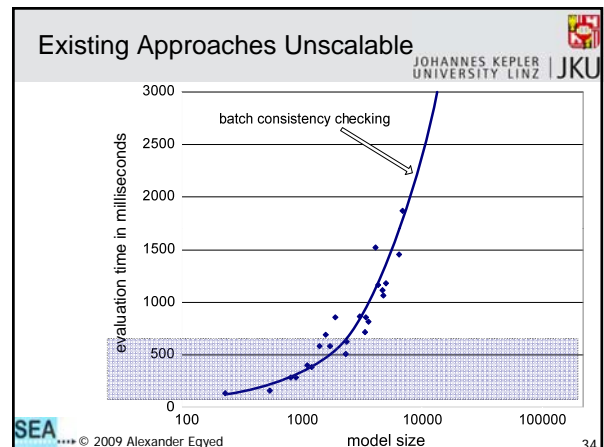
Model Analyzer Approach

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Demo...

New Consistency Rules easily definable

SEA © 2009 Alexander Egyed 33



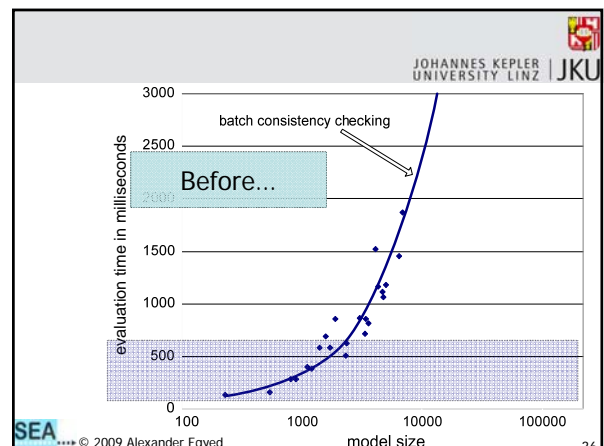
Model Analyzer Approach

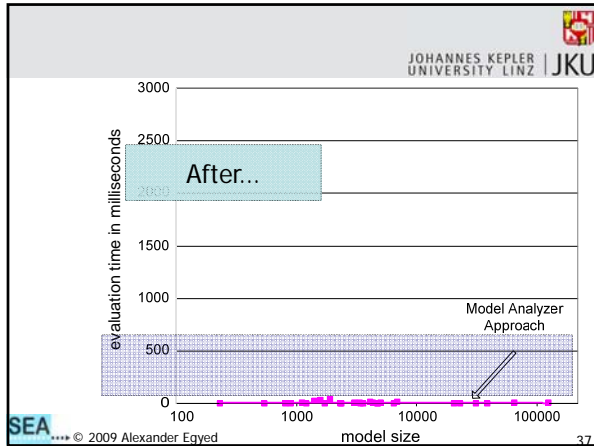
JOHANNES KEPLER UNIVERSITY LINZ | JKU

Demo...

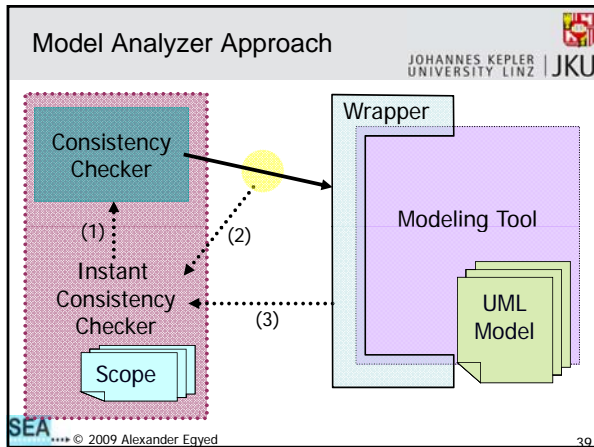
Changes and Consistency Checking

SEA © 2009 Alexander Egyed 35

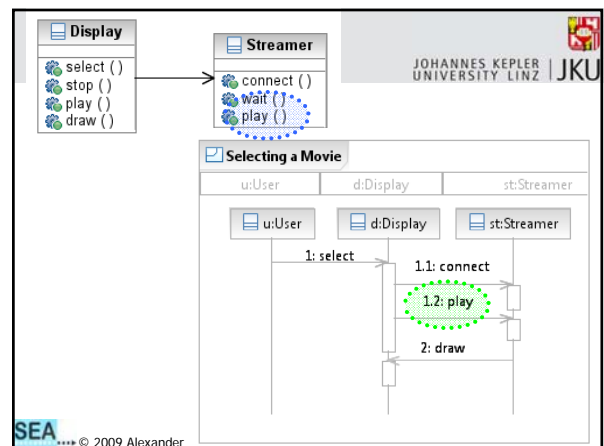
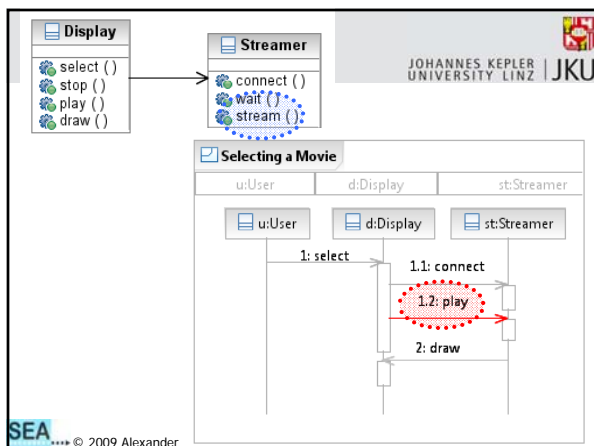




- ### Or Unusable
- JOHANNES KEPLER UNIVERSITY LINZ | JKU
- 1) Must use an obscure (and limited) rule language (who understands that language?)
 - 2) Must annotate constraints (who has the time and experience to do that?)
- SEA © 2009 Alexander Egyed 38



- ### What Engineers Need
- JOHANNES KEPLER UNIVERSITY LINZ | JKU
- (1) Identify Inconsistencies Quickly [ICSE 2006]
 - (2) Fixing Inconsistencies Quickly [ASE 2008][ICSE 2007]
- Or, how can you help an engineer resolve inconsistencies if arbitrary constraints can be defined
- SEA © 2009 Alexander Egyed Anthony Finkelstein 40



Model Analyzer Approach

JOHANNES KEPLER UNIVERSITY LINZ | JKU

Demo...

Finding Locations where to Fix Inconsistencies

© 2009 Alexander Egyed

Locations for fixing this inconsistency:

- 1) rename message *play*
- 2) Change receiver of message *play*
- 3) add a new method to the class *Streamer*
- 4) change the ownership of object *st*
- 6) rename method *connect*
- 7) rename method *stream*
- 8) rename method *wait*
- 9) delete message *play* (makes rule obsolete)

© 2009 Alexander

Choices for fixing this inconsistency:

- 1) rename message *play* to *stream*
- 2) Change receiver of message *play* to object *d*
- 3) add a new method *play* to the class *Streamer*
- 4) change the ownership of object *st* to *Display*
- 6) rename method *connect* to *play*
- 7) rename method *stream* to *play*
- 8) rename method *wait* to *play*
- 9) delete message *play* (makes rule obsolete)

© 2009 Alexander

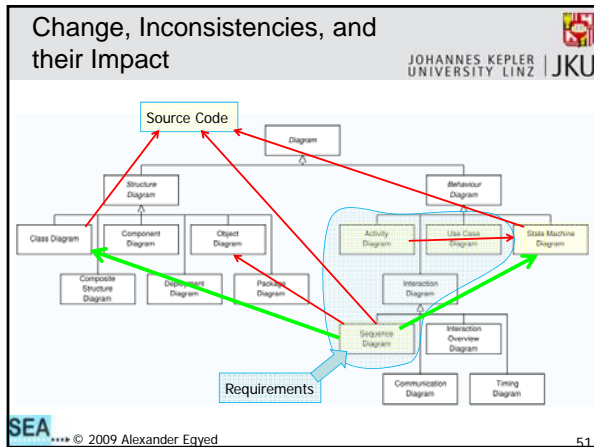
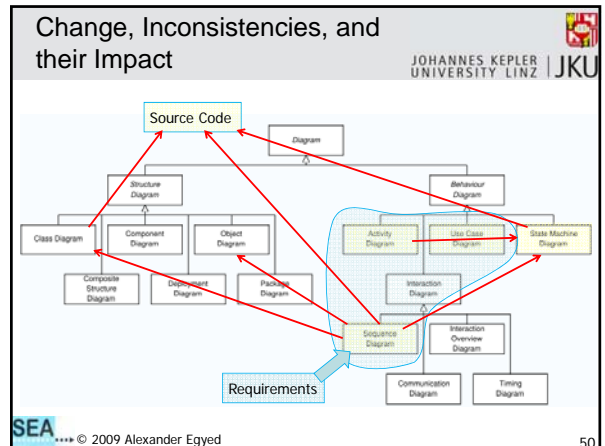
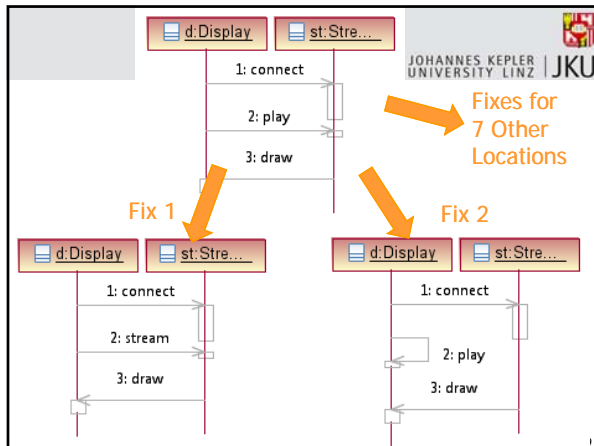
Demo...

Problem: Wrong Choices! Why is 'stream' the only valid choice?

© 2009 Alexander Egyed

© 2009 Alexander

© 2009 Alexander



These success stories required innovation, commitment, and diligence

something we can help you with!

Hall of Fame Inductee (SPLC2)

Questions?

Univ.-Prof. Dr. Alexander Egged
alexander.egyed@jku.at

SEA ***** © 2009 Alexander Egged 52