

# Model-Driven and Framework-Supported Augmented Reality Development

Arnd Vitzthum

University of Munich

Arnd.Vitzthum@ifi.lmu.de

# Introduction

- Augmented Reality: Enriches the real world with virtual information
- Lack of higher-level development concepts and tools



- Solution approach: SSIML/AR
  - *Scene Structure and Integration Modelling Language* for AR
  - Visual modeling language
  - Platform-independent abstract design of AR-applications
    - 3D-world structure, integration of application logic and 3D-content
  - Specification prior to implementation
  - Automatic model-code-mapping
  - Intended for use in task-focused domains (e.g. assembly/maintenance)

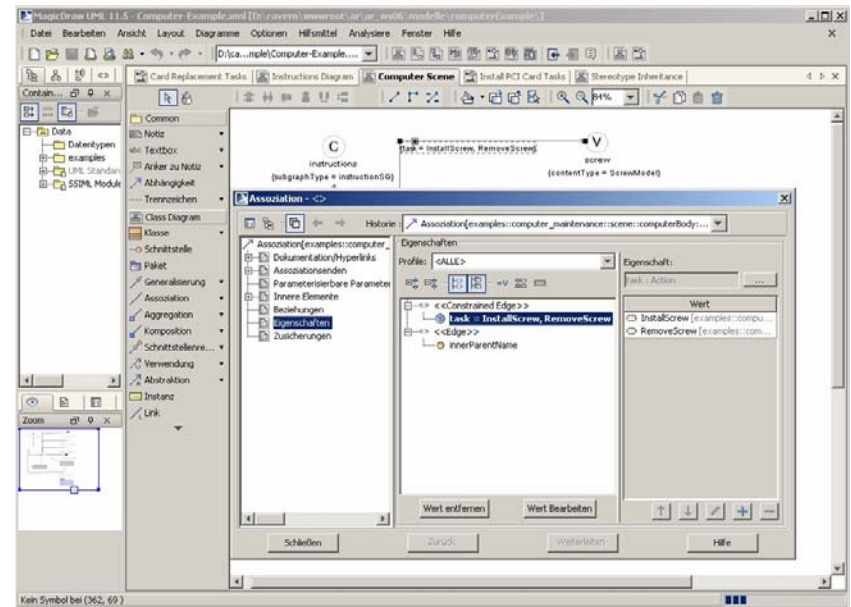
# Outline

- Part I: A Brief Overview of SSIML/AR
- Part II: Informal Evaluation of the Approach

# Development Workflow & Tool Support

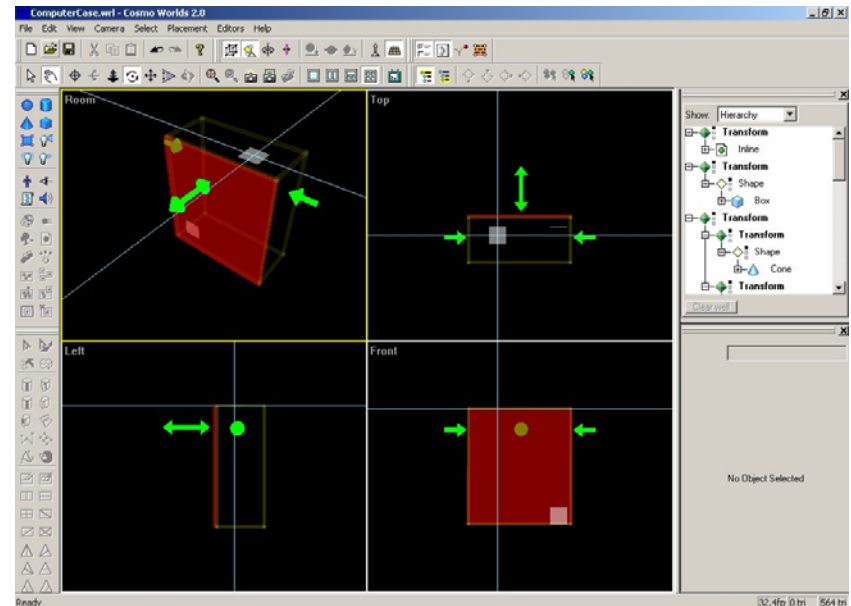
## Model creation

- UML2-Profile based on the SSIML/AR metamodel
- Profile integration into UML case tool NoMagic MagicDraw



## Automatic code generation

- Models saved in XML Metadata Interchange format (XMI)
- XSLT-based translation of a model into Java, XML and VRML97 code skeletons
- Use of VRML authoring tools (White Dune, Cosmo Worlds) to complete scene (transformation values, geometries)



# Solution Approach

- Advantages
  - Semi-formal specification
  - Models as communication aid
  - Models as documentation
  - Consistency of different code components
  - Reduction of implementation errors
- Related work (selection):
  - DART: AR authoring for designers (MacIntyre et. al. 2004)
  - ASUR: Interactions between users and objects (Dubois et. al. 2002)
  - APRIL: Specification of AR presentation flows (Ledermann 2004)

# SSIML/AR – A Brief Overview

- Taskflow models



- Further possibilities

- Hierarchical decomposition of tasks (subtasks)
    - Optional/Alternative tasks

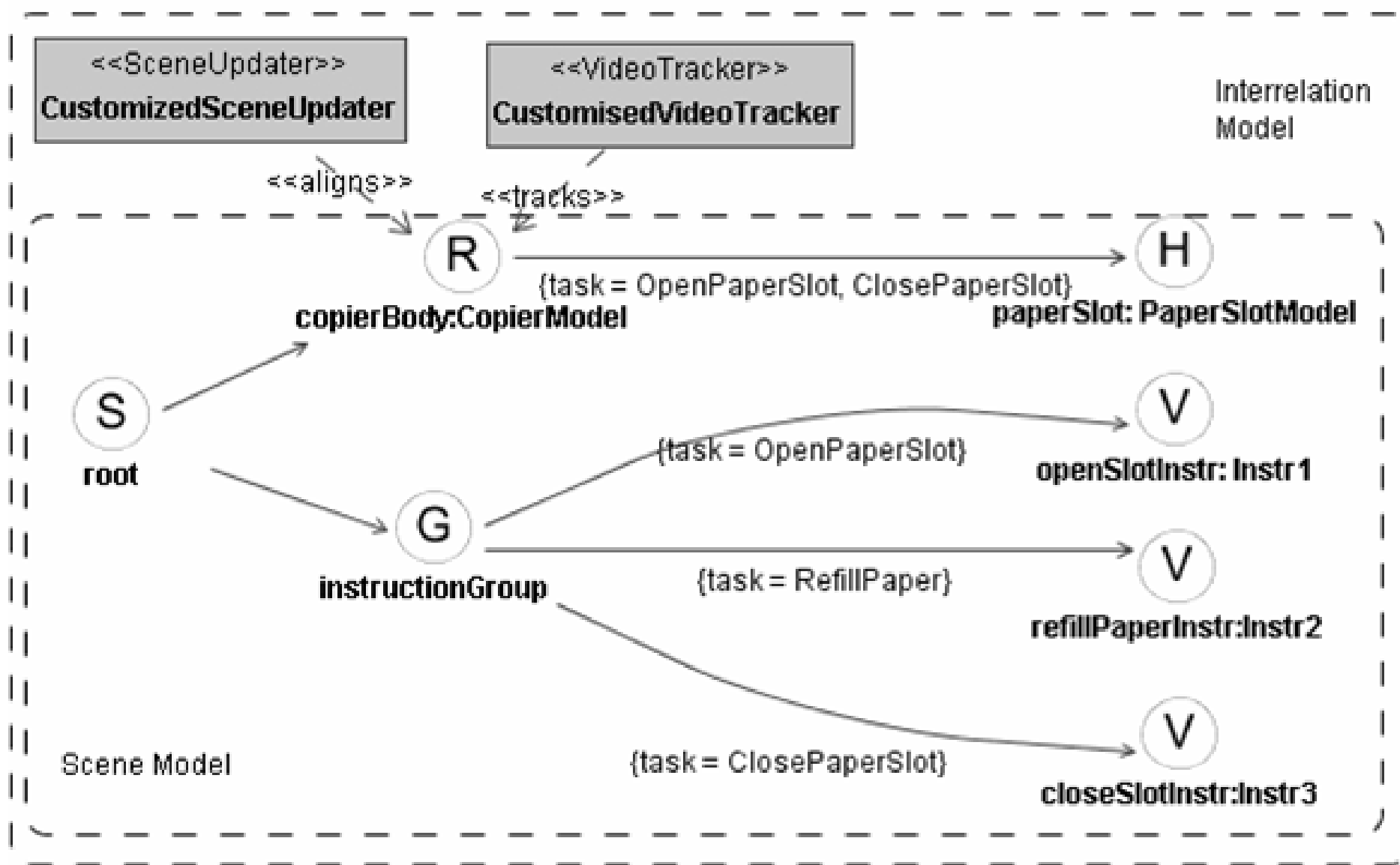
- Scene model

- Scene graph-oriented structure
  - Node types: *Group*, *RealObject*, *VirtualObject*, *HybridObject*
  - (*Task-constrained*) edges between nodes
  - Encapsulation of reusable subgraphs in *Composed Nodes*

- Interrelation model

- Comprises scene model and application components (e.g. *Tracker*, *SceneUpdater*)
  - Interrelations between application components and scene nodes

# SSIML/AR - Example Model



# AR-Framework & Code Generation

- ***Simple AR-Framework*** provides basis for generated Java-Classes
- Default implementation: *jARToolkit* (video capturing, tracking), *Java3D* (rendering)
- Input: Mouse, Keyboard, Speech
- Focus on comprehensibility, usability, extensibility
- Generated main class runs overall application
  
- Taskflow Model
  - Translated into XML-encoded taskflow description
  - Loaded and interpreted by the AR-Framework
  - Manual task switching

# Informal Evaluation

- Questions
  - Can model driven development with visual models speed up AR development?
  - Can especially SSIML/AR speed up AR development?
  - In which phase of the development?
  - Under which conditions (project size, domain etc.)?
  - How the approach can be enhanced?

# Evaluation - Realization

- Practical course on AR programming with 6 students
- No previous AR programming experiences
- Training phase:
  - 3D matrix calculations (pen and paper)
  - AR programming exercises with ARToolkit (C-based) and jARToolkit (Java-based)
    - Single/multi-pattern-tracking; 3D-transformation hierarchies and scene graphs, 3D-model loading and integration
- Project phase:
  - 2 teams, each consisting of 3 members
  - Project topic for both teams: *AR-supported replacement of damaged PC components*
  - *Approach “C”*: Use of ARToolkit and traditional development techniques
  - *Approach “SSIML”*: Use of SSIML/AR and AR-Framework
  - Random assignment of a team to an approach → *Team C, Team SSIML*
  - Each team had an own supervisor

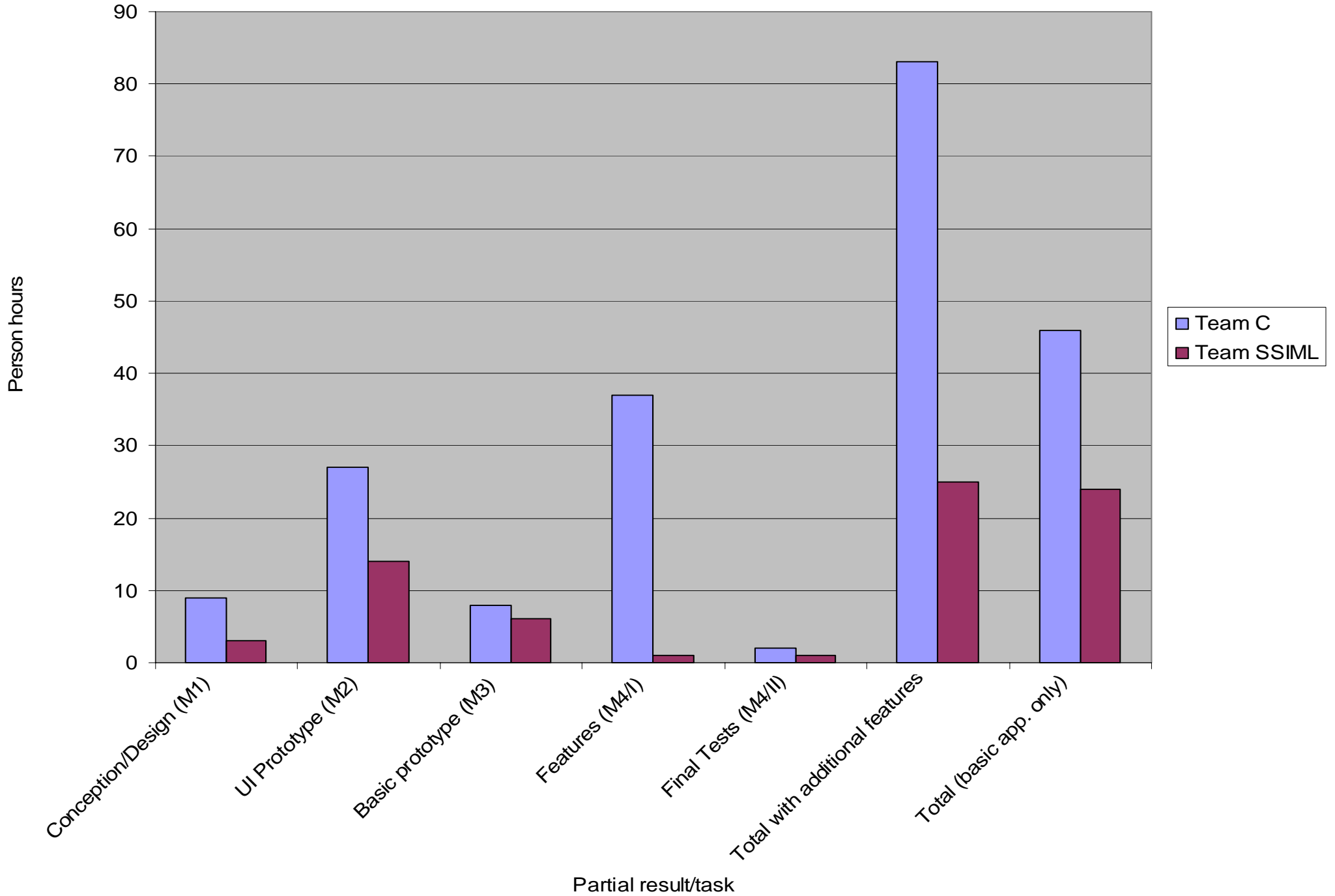
# Project Steps

- Introduction
  - Team C: 1 hour
  - Team SSIML: 4 hours
- 4 Milestones in 5 weeks for both teams:
  - Application conception and design
  - AR user interface prototype (including correct transformations and geometries)
  - First running application prototype
  - Final tested application + additional features
- 4 Milestone meetings: rework old results + new partial results
- Time estimates for each milestone
- Project ending: questionnaires/interviews

# Project Results

- No apodictic statements, only hints!
  - Small number of teams and team members
- Approaches of both teams very different → comparison of partial results (milestones)
- Produced SLOCs
  - Team SSIML: approx. 300 SLOCs (VRML, Java, XML)
    - Most of the SLOCs were generated automatically
    - Most application logic already contained in framework
  - Team C: approx. 1000 SLOCs
    - No C/C++ code was automatically generated
    - No higher-level or scene graph-API was used
    - Implementation of optional features (animations, task switching via marker recognition)

Person hours/task



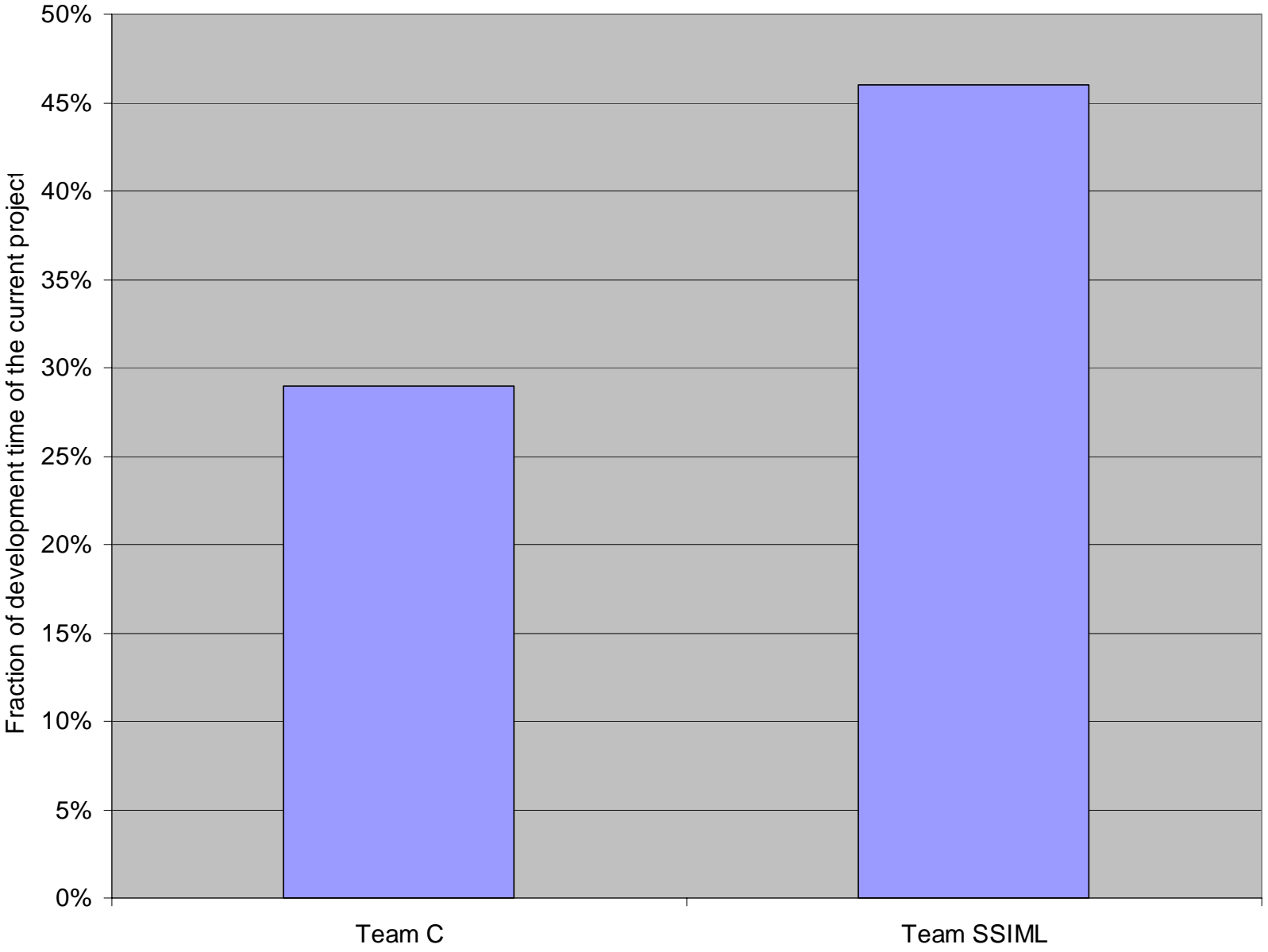
# Results - Questionnaires

- Prior knowledge of all participants
  - Previous experiences with software engineering principles and visual modeling languages (UML)
- After training phase
  - All participants felt able to produce small/mid-size AR applications with C/C++ or Java
- After project phase:
  - Level of difficulty of the project rated 4 - 5 on a scale from 0 - 10
- Team C
  - Design with UML prior to implementation was found to be useful
  - Some problems with C/C++ itself
  - But: No significant time savings with Java

# Results – Questionnaires – Team SSIML

- SSIML/AR - visual modeling language
  - Medium complexity
  - Profitable in the present scenario
  - Valuable for similar small/mid-size projects
- Code generation
  - Generated code skeletons were found to be helpful, comprehensible, well-structured, consistent, and easily extendable/adaptable
  - Minimal changes in generated Java code
- Criticism/Disadvantages
  - Only forward engineering supported
  - Students were rather sceptical about using SSIML/AR
    - in other AR domains
    - in big projects (scalability of the approach)
  - No advantages for maintainability/extensibility

Development time of a similar application (without additional features)



# Results – Team Interviews

- Team organization
  - No developer roles → Developer roles useful in larger projects
  - Group work/discussions
  - Pair programming (Team SSIML/AR)
  - Individual work only for some special subtasks (e.g. creation of certain 3D models)
- Project organization
  - Milestone meetings valuable

# Conclusions

- Lessons learned:
  - Clearer definition of required results → better comparable results
  - Next evaluation with a slightly larger scenario
- Design prior to implementation useful for AR development in general
- SSIML/AR
  - Application profitable in small/mid-size AR projects in task-focused domains
  - Encourages structured development
  - Rapid development of fully functional prototypes possible
    - Refinement via programming
  - Automatic code generation/AR framework
    - Seamless transition to implementation
    - Error reduction
    - Time savings
- Main drawback
  - Only forward engineering possible

Thank you

Questions?

# References

- Emmanuel Dubois, Paulo Pinheiro da Silva, and Philip D. Gray. Notational Support for the Design of Augmented Reality Systems. Proc. DSV-IS 2002, pages 74-88, 2002
- Florian Ledermann. An Authoring Framework for Augmented Reality Presentations. Diploma thesis, Vienna Technical University, 2004
- Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. Proc. UIST 2004, pages 197-206, 2004