Explication and Combination of CSCL Scripts and their components

Andreas Harrer
COLLIDE research group
University Duisburg-Essen

December 1, 2005

kaleidoscope
concepts and methods for exploring the future of learning with digital technologies

Abstraction Levels of CSCL Scripts

- Conceptual Level - educational concepts as constructs (Designer's/Practitioner's perspective)
- Operational Level – primitive Constructs needed for Execution (Machine perspective)
- Concrete Level – actions taken in real or simulated processes (User perspective - as demonstrator or learner)

State of the Art

- Scripts embedded (hard-wired) in the graphical user interface
- Informal Script-Notations
- Semi-formal Templates/Tables for Educational Processes
- Educational Modelling Language
- Learning Design (IMS/LD)
  - Pedagogically neutral
  - Structured format (XML)
  - Executable with „Players"

The Problems of Formalization and Operationalization

- Only explicit representation of a script facilitates re-use
- Description has to be formal to become operational
- Complex situations, such as role changes and dynamic group formation are even hard to specifiy
- Educational Rationale is not clearly visible with current notations

- Do we need a programming language for CSCL scripts?

There are scripts and other scripts...

- Macro-Scripts (Dillenbourg) that shape the context of an activity at a course granularity
- Micro-Scripts (Tübingen/München) that give a scaffold for cognitive procedures, i.e. fine-grained
- One of our aims in COSSICLE is to create scripts from building blocks
- Hierarchical Structures of Macro-/Micro-Scripts are a natural choice, but how to integrate them seamlessly, how to bridge the gap?

Example Combination

- Macro-Script: Argue Graph
  goal: students develop pedagogical rational for different design alternatives by argumentation with partner that has controverse opinions
- Micro-Script: Argument Constrcution
  goal: help students to learn how to argument and integrate perspectives
- Micro-Script can help students in argumentation, if they lack the procedures for that

Problems identified yesterday

- How to distinguish sequential and parallel activities?
- How to specify association of roles and activities?
- How to specify alternatives?

Desired Properties for the Specification Language

- Language uses concepts the practitioner is familiar with and that relies on a theory
- Enables educational modelling at different levels of granularity – composition and structuring of scripts/components
- Provides mappings to operational level – constructs have a semantic mapping to operational models

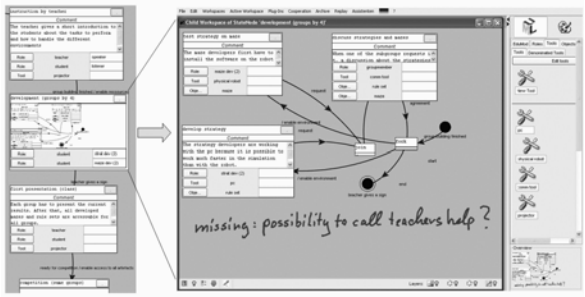
Comparison of Modelling Approaches

Conducted in a Computer Science Seminar „Process-oriented Modelling Languages“ with the Micro/Macro-Script as practical example

- Automata-based approaches (finite, statechart)
- Activity diagrams (UML)
- ARIS (Architecture for Information Systems)
- IMS/LD
- Collaborative Learning Flow Patterns

Hierarchical models reduce complexity, gap between script description and its instantiation at runtime

Proposal 1 – Graphical Modelling with Statecharts

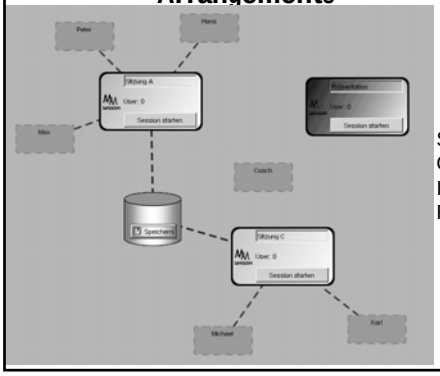


Structuring of Activities, Operational model available

Modelling of Scripts with Statecharts

- Modelling in Layers
  - general process structure of the activities (phases)
  - Each phase state is refined as a parallel composition of the different roles involved in an activity
  - Each role can be further refined into substates representing different behaviour available to the role at the moment. Actions available to the role cause transitions to follow-up states, including loops to come back to behaviour already taken.

Proposal 2 – Graphical Specification of Group Arrangements

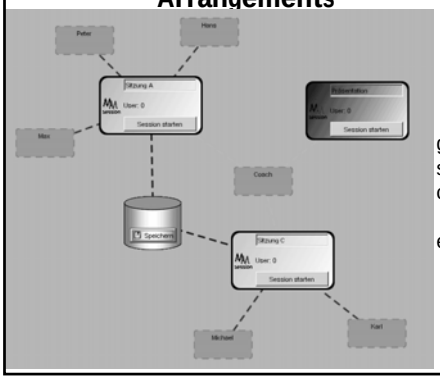


Structuring of Groups, Interaction Spaces, Results,

Proposal 2- Mapping rules to LD

- Each session is mapped to an IMS/LD "learning-activity"
- Each client node or slot node is mapped to an IMS/LD role of type "imsld:learner", the implict teacher to "imsld:staff"
- The whole classroom scenario graph of our visual language format is mapped to an IMS/LD "act"
- For each learner (client or slot node) an IMS/LD "role-part" is created within the act with the respective "role-ref"
- In each "role-part" a "learning-activity-ref" is added for each session the role participates in

Proposal 2 – Graphical Specification of Group Arrangements

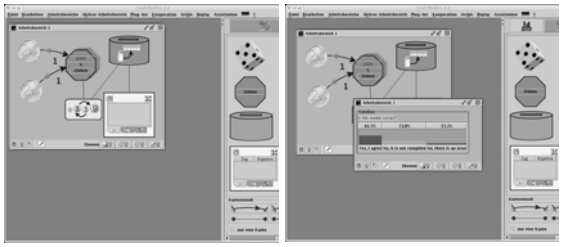


graph – act
session – activity
client – role + role-part
edge – activity-ref in role-part

Proposal 2 – Graphical Specification of Group Arrangements

Multiple Arrangements define process sequence

Proposal 3 – Learning Design by Example

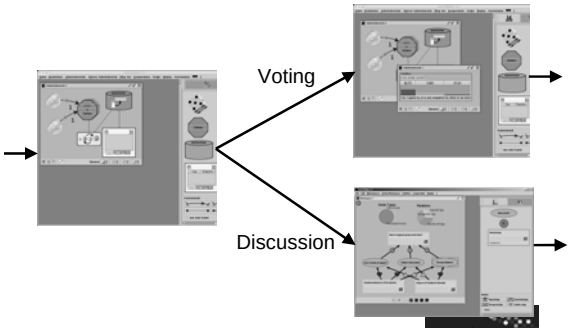


User Actions: 1. Add Voting Tool, 2. Conduct Voting

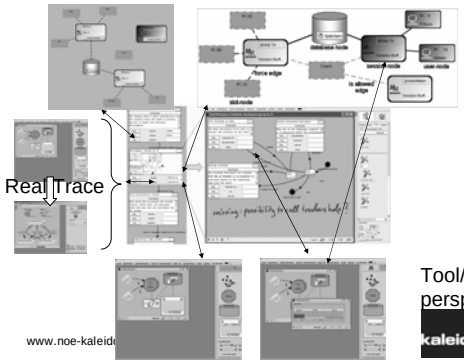
Proposal 3 – Learning Design by Example

- Specification of the process by "doing it by example"
- Multiple examples define variations and pathways
- Configuration/design is re-usable for other processes
- Authoring is needed to create "meaning" for the conducted action, e.g. "decision making phase facilitated by voting tool"

Proposal 3 – Learning Design by Example



Integration of multiple perspectives



Challenges with Script Components

- Which are the constraints for using different components with each other?
- Which components are refinable and which are not?
- How can we manage the seamless integration of components, e.g. artifacts flowing from one component into the next?

Organizing Script Components

- Transferral of the concept „pattern language“
- Pattern language describes which components have relations to each other
  - Component A refines Component B
  - Component A can be used in sequence with Component B
  - Component A is prerequisite of Component A
  - Component A is alternative to Component B
  - Component A is of type B (Families of Components/Scripts)

Thank you!

Please feel free to comment and ask...