Mapping between aspect-oriented requirements, domain analysis, and architecture

Group 4:
Ana (amm@di.fct.unl.pt),
Istvan (nagyist@cs.utwente.nl),
Andrew (A.Dingwall-Smith@cs.ucl.ac.uk),
Gabriel (gabriel.amoros@sema.es),
Tommi (tjm@cs.tut.fi)
Workflow

• Started with 25 questions
• Ended up 21 refined questions
• Divided the questions to 4 categories
  – Formal vs. informal mapping
  – Mapping process
  – Tools, techniques, and languages
  – Pros and cons of mapping
Formal vs. informal mapping

• Typical questions
  – How can an aspect be formalized at requirements level?
  – Should the mapping be an informal or formal?

• Typical solutions
  – Formal specification languages for AO A2D
    • Common basis for terminology (int), reasoning (+), not here (-)
  – Case studies, pilot projects
    • Street credibility (int, +), long and hard road (-)
  – Prototyping
    • Down-to-earth (+), stuck with what we have (-)
  – The issue of domain-specific practices
    • Freedom of choice (+), divergence in practices (-)
Mapping process

• Typical questions
  – To what is a requirements concern mapped onto?
  – Does the mapping constrain the starting point?

• Typical solutions
  – Successive refinements in a formalism
    • Incrementality (+, int), preserving behavior (+, int),
      verification (+), need the formalism (-), need refinement policy
      (-), evolution?
  – Program constructs and other artifacts and links
    • Pragmatic (+), may not give many new ideas (-)
  – Catalog of good/bad mappings
    • Applies to many mappings (int, +), very pragmatic (int, +),
      domain-specific (-)
Tools, techniques, and languages

• Typical questions
  – Language features to support mapping
  – Can a mapping be created automatically

• Typical solutions
  – Animators, compilers, preprocessors, interpreters, decisionmaking systems, knowledge data bases, theorem provers, visualizers
    • Having all these tools would be great (int), available tools determine the advocated formalisms (-), no way to understand how the tools fit together (-)
  – Languages with new flavors of modularity
    • Lots of research challenges (int), lots of PhDs (very int), lots of EU projects (very very int)
Pros and cons of mapping

• Typical questions
  – Benefit ratio of mapping/coding
  – Pros and cons of mapping in the first place?

• Typical solutions
  – Requires testing of many (if not all) of the approaches discussed in the earlier slides
    • Better predictability (+), improved understanding of software (int, +), not an add-on to existing practices (-)