Using a logic meta language to express cross-cutting through dynamic joinpoints.

Kris Gybels
Programming Technology Lab
VUB
kris.gybels@vub.ac.be
Why?

- Cross-cut languages: AspectJ, Composition Filters, HyperJ, ...
- Aspects ‘see’ a program in terms of joinpoints
- Information on joinpoints usually limited
- Why not exploit the full program to express cross-cutting?
Application 1: open weaver

- Aspect programmer can take on part of the weaver’s job: extracting joinpoints from code
- Paper example: programming conventions
- Benefit: not hard coded in weaver
Application 2: decoupling

• Observer aspect: how does the cross-cut select modifying messages?
• Typical: enumeration in cross-cut
• Problem: couples aspect to specific class
• Idea: use LMP to extract modifying methods
Dynamic values on joinpoints

• Presents more dynamic information about cross-cutted program
• Help separate cross-cut expression from action
Possible points of discussion

• Useful to capture patterns in cross-cuts?
• LMP good way to do it?
• Dynamic (executional) information on joinpoints useful?
• Useful in non-AspectJ cross-cut language?
• Suggestions for experiments