GEMS and ATLAS

Kaheer Suleman and Sharon Choy

Outline

- GEMS Overview
- ATLAS Overview
- “The Big Picture” – how the pieces fit together
- Demonstration
- Analysis, insights, and lessons learned

GEMS

- Created by Distributed Object Computing Group at Vandebilt University and Siemens Corporation
- Open source project
- Incubation status

GEMS

- Configurable toolkit for creating a domain-specific modeling environment
- Uses Eclipse's Graphical Editing Framework (GEF) and Draw2D plug-in
- Goal – bridge the gap between visual metamodelling (e.g. GMF) tools and Eclipse based tools (EMF)

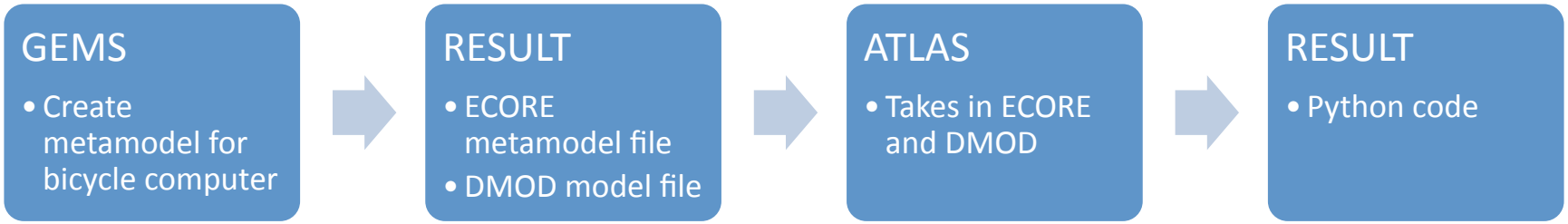
GEMS

- Purpose: to create the metamodel for expressing our bicycle computer
- Resulted in the creation of a plug-in that allowed us to create an instance of a model

ATLAS

- Transformation program composed of rules that define how source model elements are matched
- Mapped elements of our metamodel to Python code

Code Generation Process



Demo

GEMS Insights and Lessons Learned

- Advantages
  - Provides a graphical interface for users to create their metamodel
  - Intuitive and straightforward to use
  - Easy to integrate

GEMS Insights and Lessons Learned

- Disadvantages
  - Installation issues
  - Lack of documentation
  - Last release was 2008
  - Cannot express inheritance in the resulting model
  - Simplicity of GEMS may not allow rich expression
  - Usability issues

ATLAS Insights and Lessons Learned

- Advantages
  - Well-documented
  - Rules and helpers allow users to transform their models into any output language

ATLAS Insights and Lessons Learned

- Disadvantages
  - Integration with GEMS in certain situations
  - Lack of else-if statements in ATLAS programming language

Conclusion

- GEMS provides a graphical interface for generating EMF Model
- ATLAS takes in EMF code (.ecore file)
- Transformation rules done in ATLAS result in Python code