DSL Tools
&
Eclipse Modelling Project

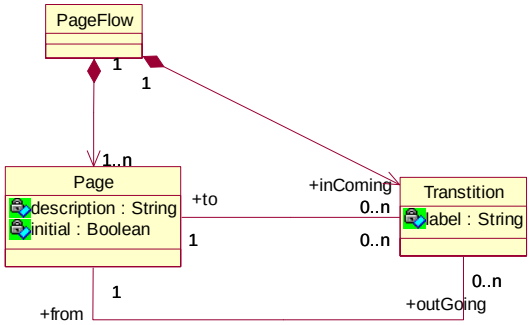
Mini Tutorials

Vicente Pelechano
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

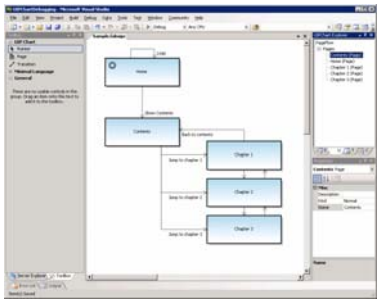
User Interaction Process (UIP)

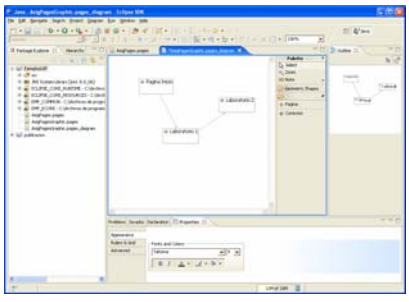
- Build a DSL and a Graphical Designer to specify web sites, wizards, or form-based applications.
- A UIP chart defines a collection of pages and transitions between pages.
- Transitions are indicated by an arrow originating at the page being transitioned from, and ending at the page being transitioned to.
- Each transition is labeled. Transition labels might be used, for example, to provide the names of buttons in generated code.

User Interaction Process (UIP)



User Interaction Process (UIP)





DSL Tools vs Eclipse

Experimental Comparison

Vicente Pelechano
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Contents

- Experiment
- Project Proposals
- Research Questions
- Conclusions

Experiment

- 48 Fifth Year Undergraduate Students of Computer Science Engineering.
- The students were between 22 and 24 years old and had similar backgrounds.
- Students were divided into 2 groups for developing a DSL (including code generation):
  - One Group with DSL Tools (February 2006 Version)
  - One Group with Eclipse (April 2006 Version)

Project Proposals

- PervML Modeler and Code Generator (Supporting MDD of Pervasive Systems).
- J2EE Code Generator (DSL and Code Generation).
- Definition of a Language for Specifying Project Plans and Code Generation of a tracking application.
- Definition of a Language for Modeling Agents and Code Generation in Jade.
- Generating Code in Hibernate from Class Diagrams. Making the Objects Persistents.
- Specification and Automatic Generation of Unit Tests (Junit) from the Class Diagram.
- Code Generation from BPMN Models using Toghether 2006 QVT.
- From the Class Diagram to the Relationa Model. Generating SQL code.
- Definition of a Language for specifying Aspect Oriented Software Architectures.
- Definition of A State Transition Diagram Modeler and Code Generation in C#.
- A DSL for specifying and generating/producing Posters.
- Definition of Language for modeling Surveys and HTML Code Generation.
- **….and much more projects**.

Research Questions

- Q1: Documentation Availability
- Q2: Metamodeling Language Understandability
- Q3: Metamodeling Language Expresivity
- Q4: Language (Metamodel) Designer Usability
- Q5: Graphical Designer Usability
- Q6: Quality of the Resulting Graphical Modeler
- Q7: Graphical Designer Completeness

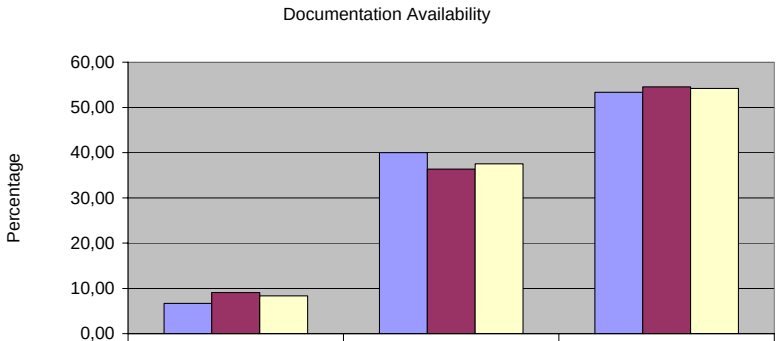
Research Questions

- Q8: Extensibility of the Graphical Designer/Mapper
- Q9: Comparing Generated Editors. DSL vs. Eclipse
- Q10: Maturity and Robustness. DSL vs. Eclipse
- Q11: Complexity in Defining the Code Generator
- Q12: Implementing the Code Generator. Programming Language vs. Template Engine
- Q13: Utility of the Employed Tools
- Q14: Industrial Application
- Q15: Fidelity to the Tool

Results

Q1: Documentation Availability:

Answers: (a) Good (b) Enough (c) Poor

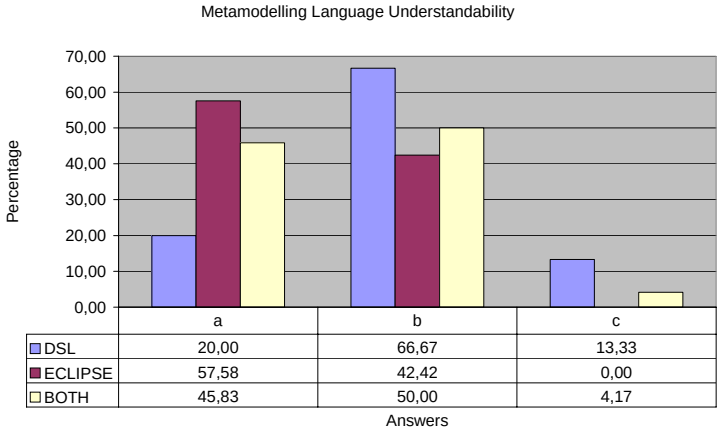


	a	b	c
DSL	6,67	40,00	53,33
ECLIPSE	9,09	36,36	54,55
BOTH	8,33	37,50	54,17

Results

Q2: Metamodeling Language Understandability:

Answers: (a) Easy (b) Acceptable (c) Difficult

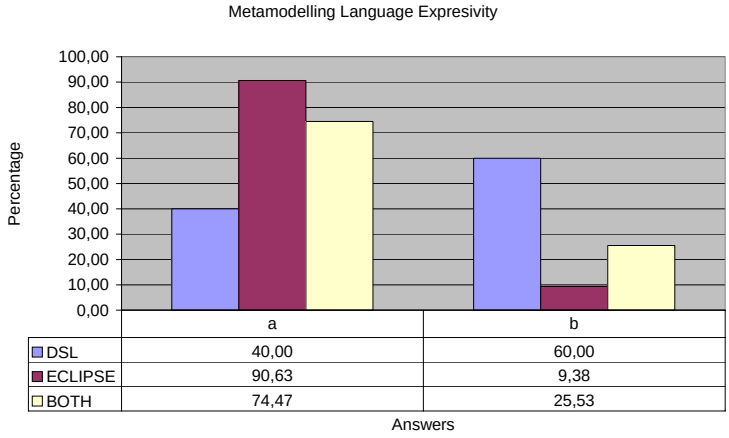


	a	b	c
DSL	20,00	66,67	13,33
ECLIPSE	57,58	42,42	0,00
BOTH	45,83	50,00	4,17

Results

Q3: Metamodeling Language Expresivity

Answers: (a) Enough (b) Not Enough

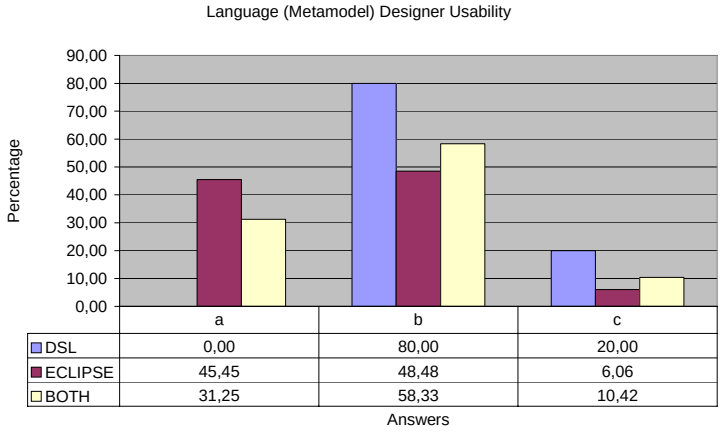


	a	b
DSL	40,00	60,00
ECLIPSE	90,63	9,38
BOTH	74,47	25,53

Results

Q4: Language (Metamodel) Designer Usability

Answers: (a) Easy (b) Acceptable (c) Difficult

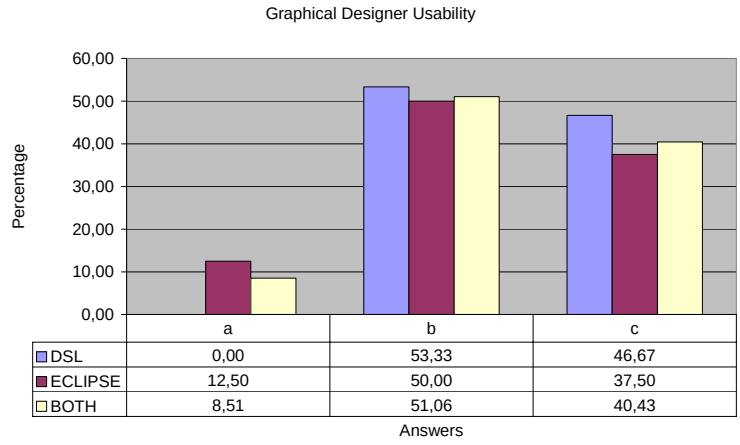


	a	b	c
DSL	0,00	80,00	20,00
ECLIPSE	45,45	48,48	6,06
BOTH	31,25	58,33	10,42

Results

Q5: Graphical Designer Usability

Answers: (a) Easy (b) Acceptable (c) Difficult

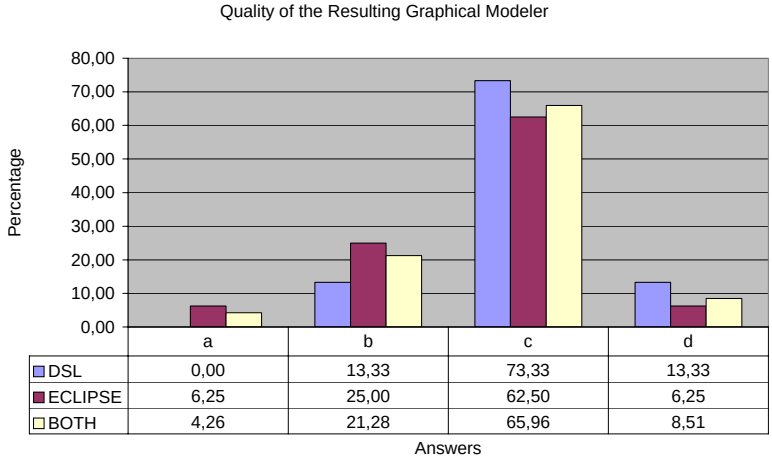


	a	b	c
DSL	0,00	53,33	46,67
ECLIPSE	12,50	50,00	37,50
BOTH	8,51	51,06	40,43

Results

Q6: Quality of the Resulting Graphical Modeler

Answers: (a) Better than expected (b) As Expected (c) I miss some details (d) Poor

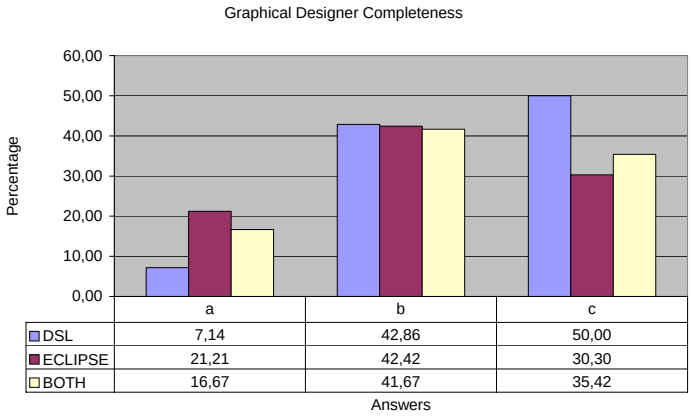


	a	b	c	d
DSL	0,00	13,33	73,33	13,33
ECLIPSE	6,25	25,00	62,50	6,25
BOTH	4,26	21,28	65,96	8,51

Results

Q7: Graphical Designer Completeness

Answers: (a) Complete (b) Acceptable (c) Not Enough

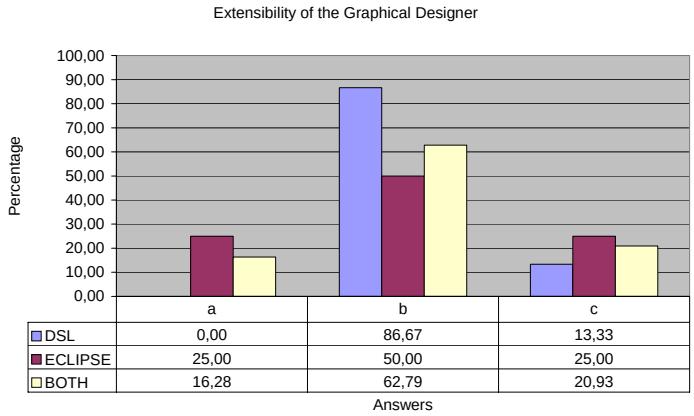


	a	b	c
DSL	7,14	42,86	50,00
ECLIPSE	21,21	42,42	30,30
BOTH	16,67	41,67	35,42

Results

Q8: Extensibility of the Graphical Designer

Answers: (a) Easy (b) Need Some Extra Work (c) Impossible

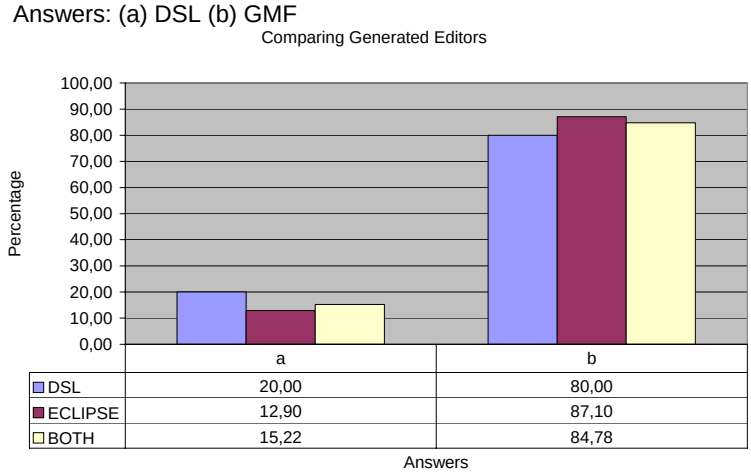


	a	b	c
DSL	0,00	86,67	13,33
ECLIPSE	25,00	50,00	25,00
BOTH	16,28	62,79	20,93

Results

Q9: Comparing Generated Editors. DSL vs. Eclipse

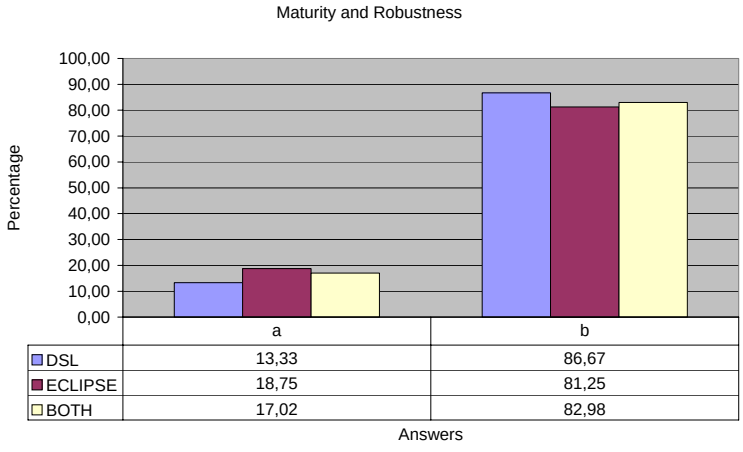
Answers: (a) DSL (b) GMF



Results

Q10: Maturity and Robustness

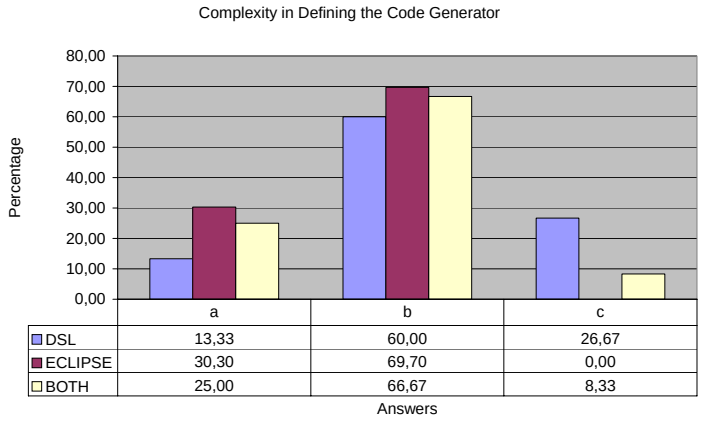
Answers: (a) DSL Tools (b) Eclipse



Results

Q11: Complexity in Defining the Code Generator

Answers: (a) Easy (b) Medium (c) Difficult

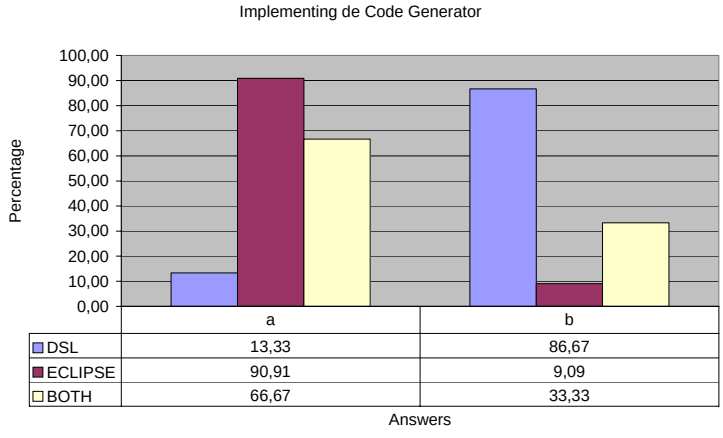


	a	b	c
DSL	13,33	60,00	26,67
ECLIPSE	30,30	69,70	0,00
BOTH	25,00	66,67	8,33

Results

Q12: Implementing the Code Generator

Answers: (a) Template Language (b) Any Other Programming Language

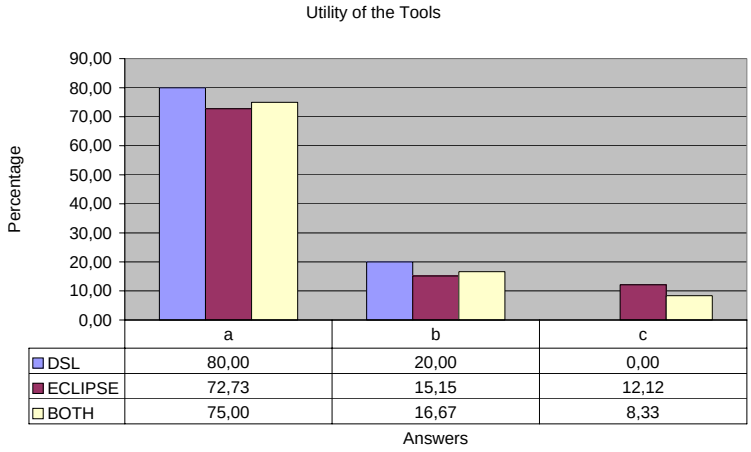


	a	b
DSL	13,33	86,67
ECLIPSE	90,91	9,09
BOTH	66,67	33,33

Results

Q13: Utility of the Employed Tools

Answers: (a) Yes (b) Not sure (c) Not

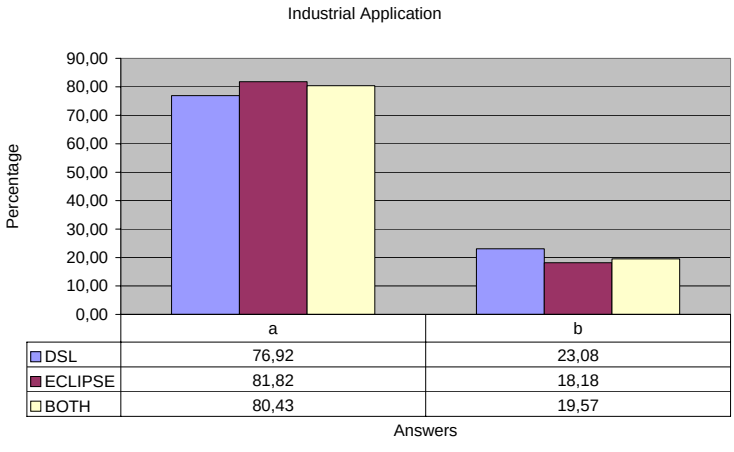


	a	b	c
DSL	80,00	20,00	0,00
ECLIPSE	72,73	15,15	12,12
BOTH	75,00	16,67	8,33

Results

Q14: Industrial Application

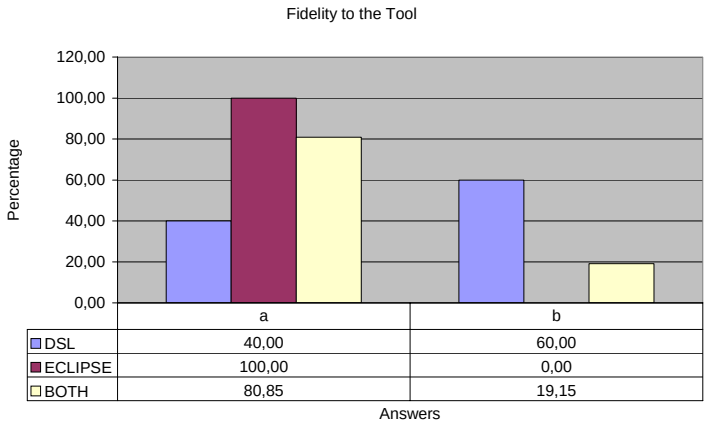
Answers: (a) Yes (b) No



	a	b
DSL	76,92	23,08
ECLIPSE	81,82	18,18
BOTH	80,43	19,57

Results

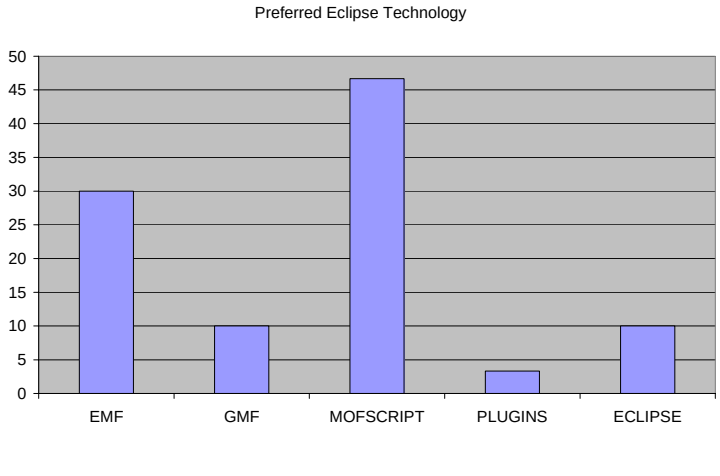
Q15: Fidelity to the Tool
Answers: (a) Same Tool (b) The Other



	a	b
DSL	40,00	60,00
ECLIPSE	100,00	0,00
BOTH	80,85	19,15

Results

Extra Info for Eclipse Users.
Preferred Technology



Conclusions

- Both Tools should publish more Documentation (They are on the way).
- eCore and EMF are easier to understand than the proprietary notation provided by the DSL Tools. However, in the end, both could be understood without any problem.
- eCore is expressive enough to build language models. DSL Tools notation is a little bit more difficult to understand than eCore and it is difficult to build the metamodel.
- EMF metamodel designer is more usable than the one provided by the DSL Tools.

Conclusions

- GMF and DSL Tools Graphical Designer are difficult to use, however only in the case of GMF some students say that it is Easy to use (12%).
- The Generated Graphical Modelers seem incomplete in both cases.
- GMF and DSL Tools Designer need to be improved in order to provide more mechanisms for defining and producing professional Graphical Designers. However, GMF-Eclipse reaches a high degree of acceptability (63% say that is complete enough) compared to DSL Tools.
- Most of the time, an extra Work is needed to build professional Editors.

Conclusions

- DSL Tools and Eclipse Users prefer the Editors that have been generated using GMF.
- DSL Tools and Eclipse Users think that Eclipse Tools are more mature and robust.
- The task of defining/implementing a Code Generator has been rated as a medium degree of difficulty. It is important to note that only DSL Tools users (26%) consider this task as a difficult one.
- Eclipse users prefer *MOFScript* to build the code generator. However, DSL Tools users prefer a different programming language to the one (the template language) provided by the Tool.

Conclusions

- Eclipse and DSL Tools users think that both tools are very useful.
- Most students are thinking about using these tools in their professional careers.
- Eclipse users are 100% faithful to this environment, however 60% of the DSL Tools users would migrate to Eclipse.
- MOFScript and EMF are considered the best tools currently provided by Eclipse for supporting MDD.