Natural Language Generation from Class Diagrams

Håkan Burden and Rogardt Heldal
Software Engineering

Chalmers University of Technology and University of Gothenburg
Motivating example

It is important that a flight can have more than one flight number to enable code sharing, a multimillion-pound business that affects an alliance of airlines.

Arlow et. al. Literate Modelling – Capturing Business Knowledge with the UML, 1999
Model-Driven Architecture*

CLASS ( name STRING, ...
    visibility STRING )

.marked PIM

PSM

Marks and mappings

public class Flight {
    ...
}

private class FlightNumber {
    code : string
    ...
}
Natural Language Generation: NLG*

1) Text planning
   Which information and in what order

2) Sentence planning
   Which words and sentence structure

3) Linguistic realisation
   * An aircraft – many aircraft*

* Reiter and Dale Building Natural Language Generation Systems, 2000
Grammatical Framework*

• Abstract syntax is Text planning:
  ➢ fun Addition : Int × Int → Int

• Concrete syntax is Sentence planning:
  ➢ lin Addition x y = x "+" y
  ➢ lin Addition x y = x "plus" y
  ➢ lin Addition x y = "the sum of" x "and" y
  ➢ lin Addition x y = "summan av" x "och" y

* Aarne Ranta Grammatical Framework, Programming with Multilingual Grammars, 2011
PIM to Grammar

CLASS (  
  name STRING,  
  ...  
  plural STRING  
)

.select all instances of CLASS  
.for each instance

fun ${instance.name} : ClassName

lin ${instance.name} = mkN ${instance.name}  
  ${instance.plural}
Result

An Airport has a name, an airport code and an address.
An Aircraft can get next Flight and get Airline.
A Flight is identified by one or more Flight Numbers.
The relationship between a Flight and a Flight Number is specified by an Airline.
A Flight can have more than one Flight Number since code sharing is a multimillion-pound business, affecting an alliance of airlines.
Conclusion

• Grammars are generated in the same way as code
• putting NLG within an MDA-context

• The grammars give text and sentence planning
• GF takes care of linearisation
• Tailored lexicon for our specific needs
Future work