Composable Code Generation Applied to MDA

Kirk Schloegel, David Oglesby, Eric Engstrom

MDA Implementers' Workshop
December 2-5, 2003

Work is supported by a grant from DARPA.

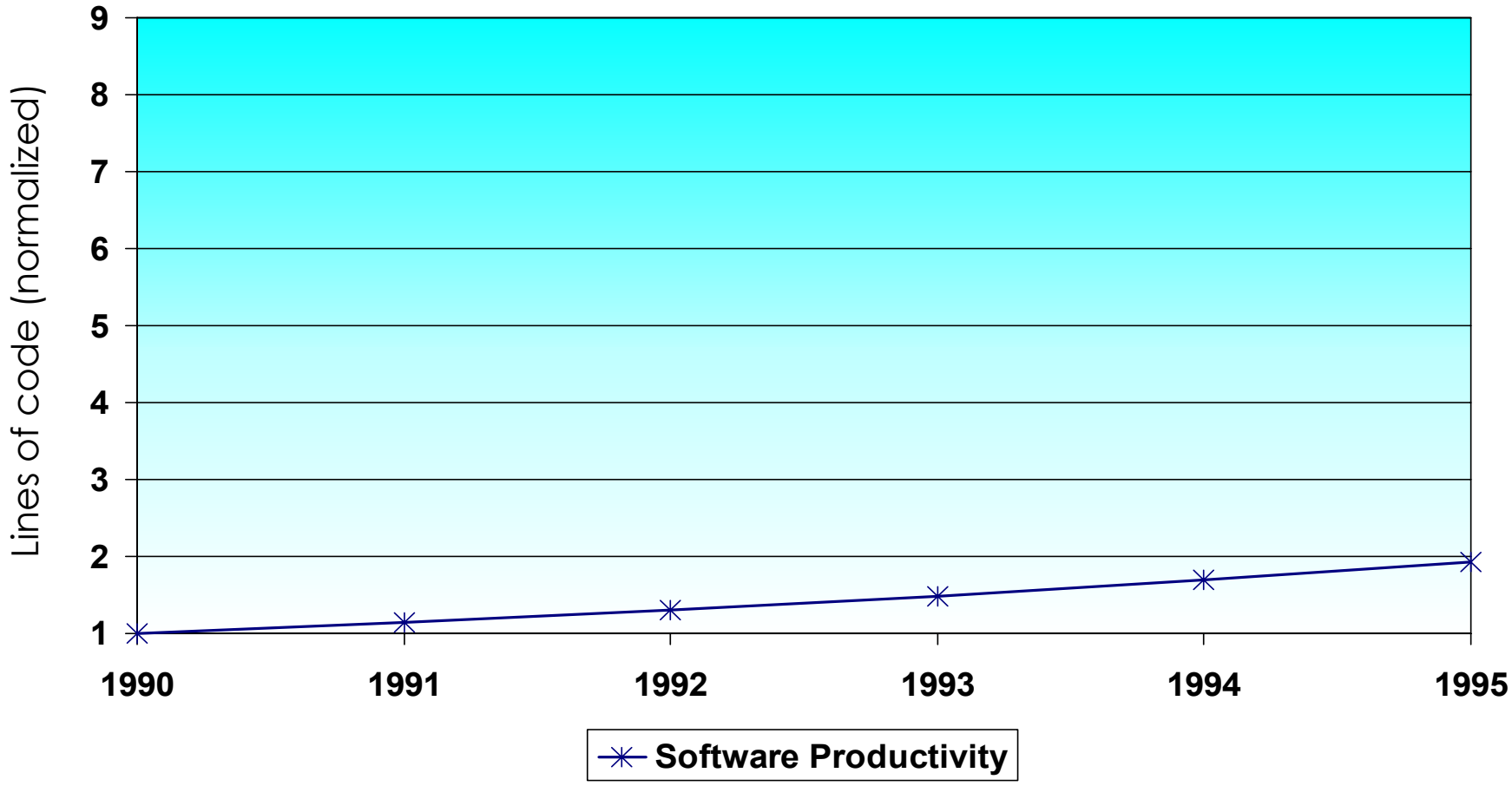
Agenda

- Modeling, Meta-modeling, and Code Generation
- Composable Code Generation
- Experimental Results
- Application to MDA

Software engineering productivity growth

Software engineering productivity is growing at a good rate

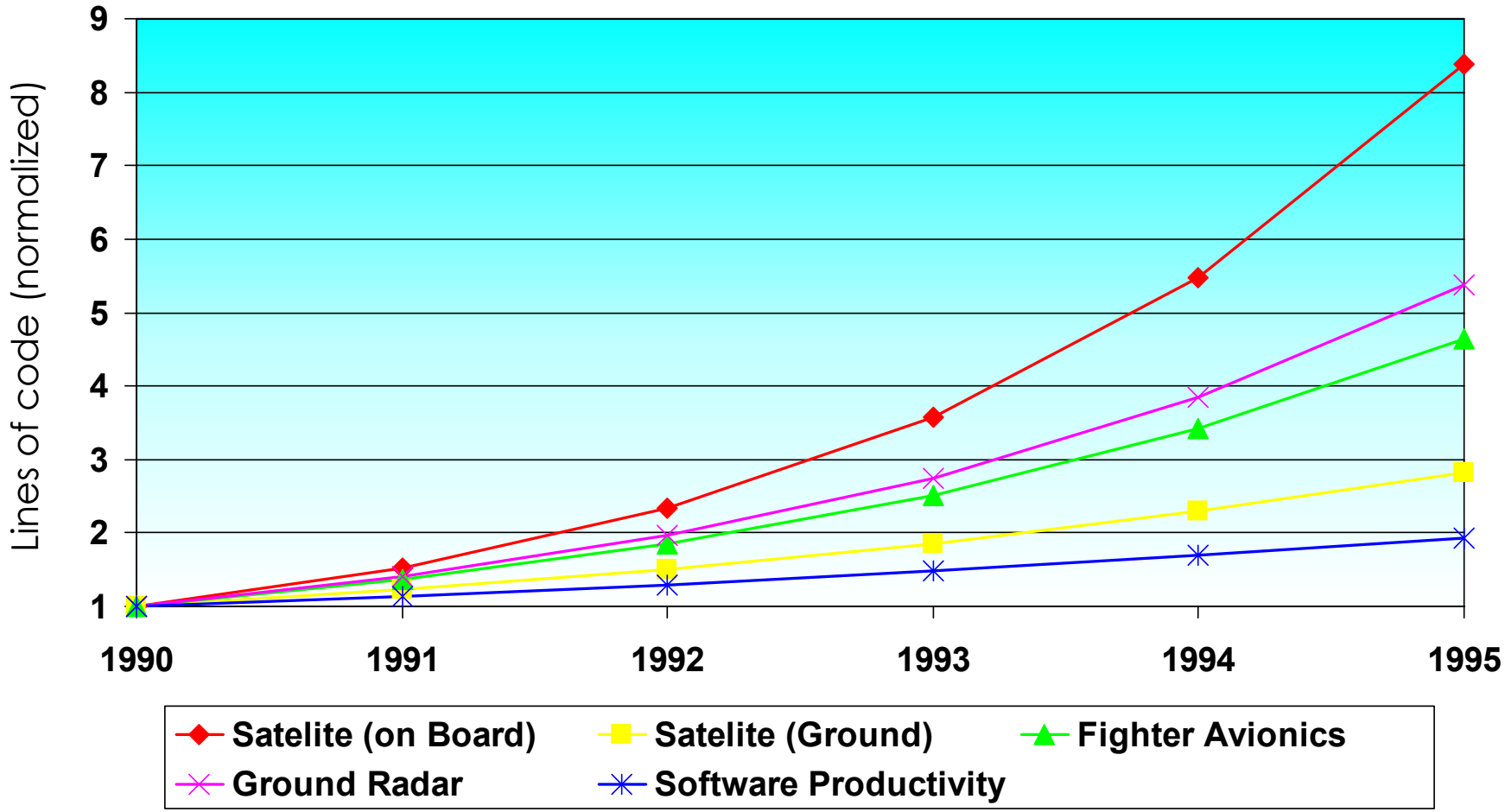
- (~14% per year)



Sources: US Dept. of Labor, Bureau of Labor Statistics,
Center for Management of Technology and Entrepreneurship, University of Toronto.

Embedded software complexity growth

Complexity of embedded software seems to be growing at a faster rate!



Sources: US Dept. of Labor, Bureau of Labor Statistics,
Center for Management of Technology and Entrepreneurship, University of Toronto.

Honeywell's response (10+ years ago)

Honeywell has a rich portfolio of embedded system products

- Thermostats & Home Security (small scale)
- Flight Management & Guidance Systems (mid scale) Real-time & Safety Critical!
- Industrial Controls (large scale)

Honeywell Labs responded to these trends in the early 90s

- In a manner that looks a lot like MDA applied to embedded systems

- Use of domain-specific modeling tools to generate "code"
  - Petri Nets (PIM) to systems of equations (PSM)
  - Hardware architectures (PIM) to HDLs (PSM)
  - Coad-Yourdon diagrams (PIM) to Ada specification files (PSM)
  - Many notations (PIM) to documentation (PSM) Humans are the platform

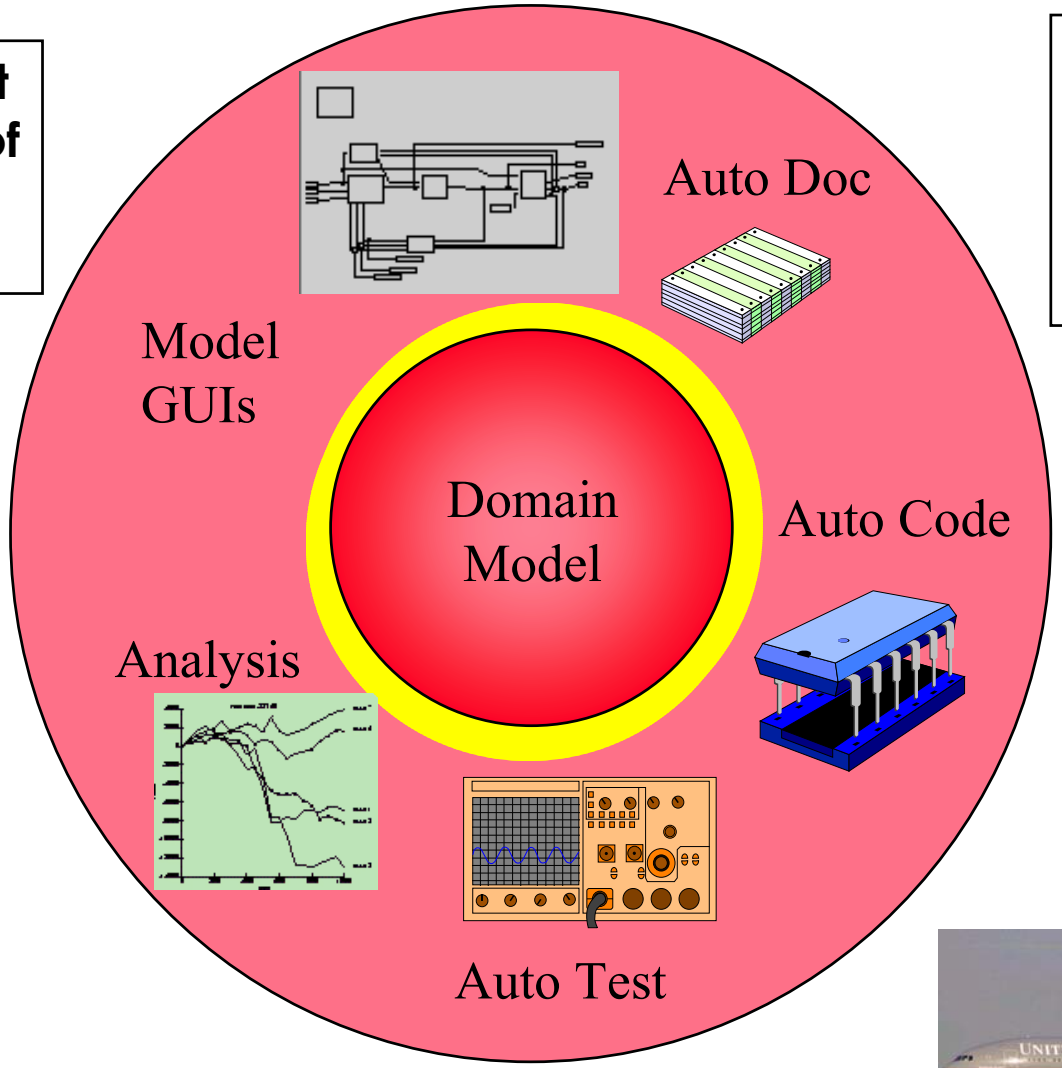
Problem: developing domain-specific modeling tools is costly

- What about a tool for developing tools?
  - Meta-models (PIM) to domain-specific modeling tools (PSM)

Domain Modeling Environment

Rapid development and customization of domain-specific modeling tools!

Powerful scripting language for implementing constraints and semantics!



COTS-tool-based artifact generation environment!

Honeywell

That was 10 years ago

In the meantime,

- many other meta-modeling tools have been developed
  - GME (Vanderbilt), MetaEdit (MetaCase Consulting)

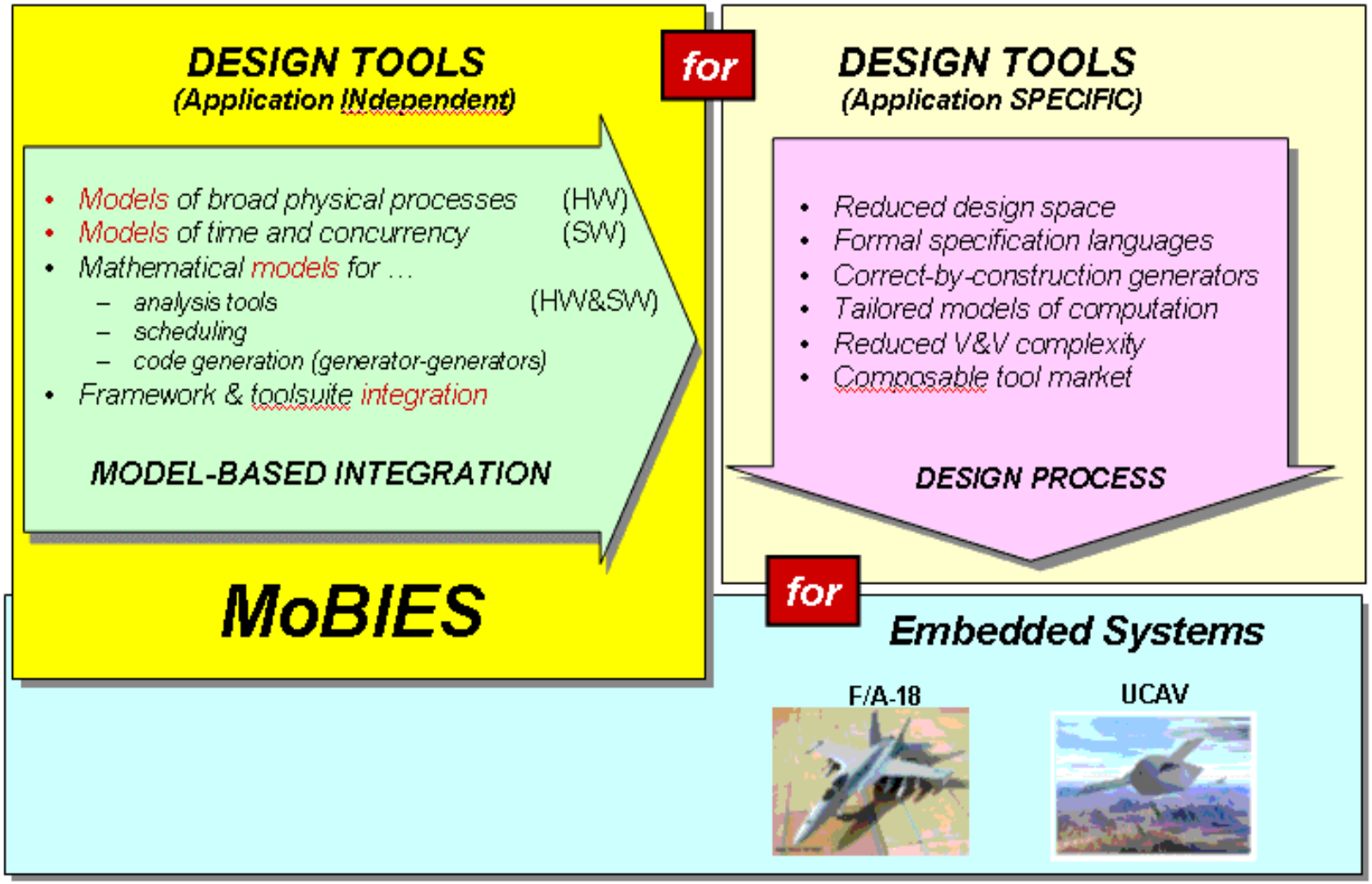
- At Honeywell, a lot of work has gone into using meta-modeling to develop domain-specific modeling notations and tools

Recently (~2001 - 2003)

- We participated in DARPA's MoBIES program with the goal to extend DOME's capabilities.

DARPA MoBIES Program Description

MoBIES is developing interoperable tools to design and test complex computer-based systems such as avionics, weapons, and communications systems. These tools will simplify the design of complex embedded systems by focusing on the pre-production environment rather than after-the-fact integration. The approach is to customize the design tools used by applications engineers so that controller design and systems integration can be more fully automated and the errors thereby reduced.



Honeywell's Challenges in the MoBIES Program

Create/customize a high-level design tool comprised of multiple integrated modeling notations/views

- Enable distributed and real-time embedded designs to be specified by systems of disparate, yet interacting, models

Generation of middleware configuration code (PSM) and tool interchange data (PSM) from

- Event flow models (PIM), Process-thread-component maps, Hardware models

Enable model reuse

- Specifically, enable the capture and reuse of design patterns as modeling constructs

Code Generation Challenges

Generate code cooperatively across modeling notations

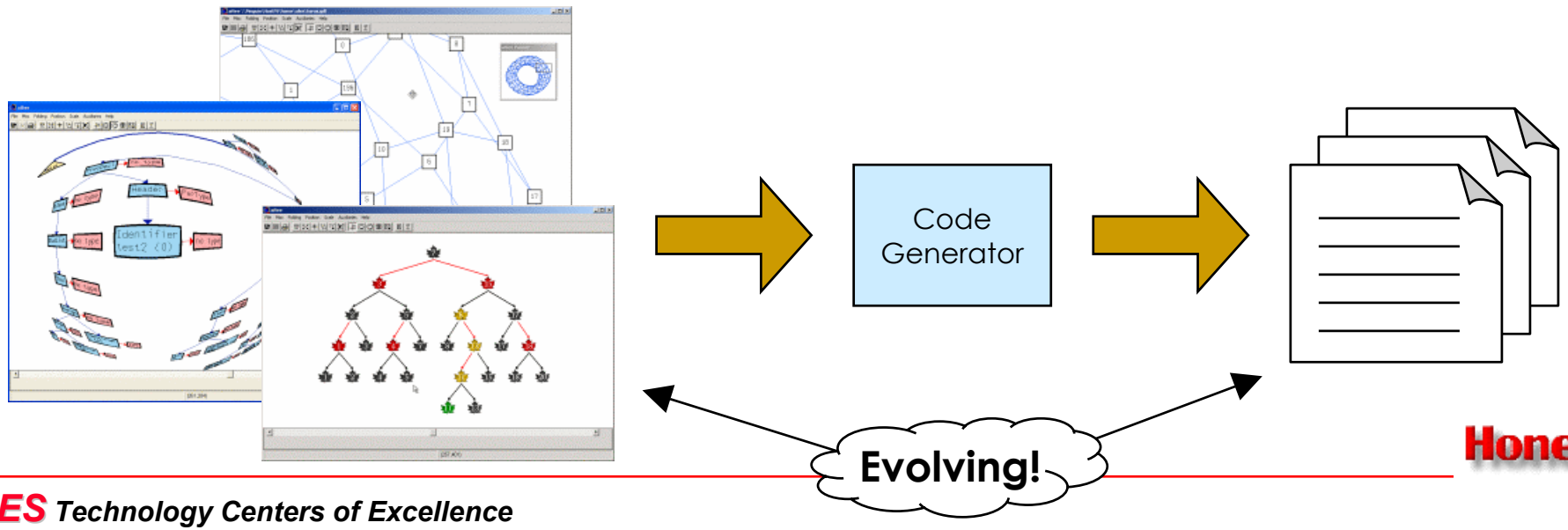
- Domain-specific modeling notations have few, deep concepts
- They are not usually extensible
- The data required to generate code is not specified in any one notation

Generate to Interchange Formats (IF)

- Data formats that have been agreed upon by all participants
- Sometimes the necessary data is not specified in any of the models

Both the modeling environment and the IFs are (rapidly) evolving

- Requires an agile code generation framework



CCG -- Key idea

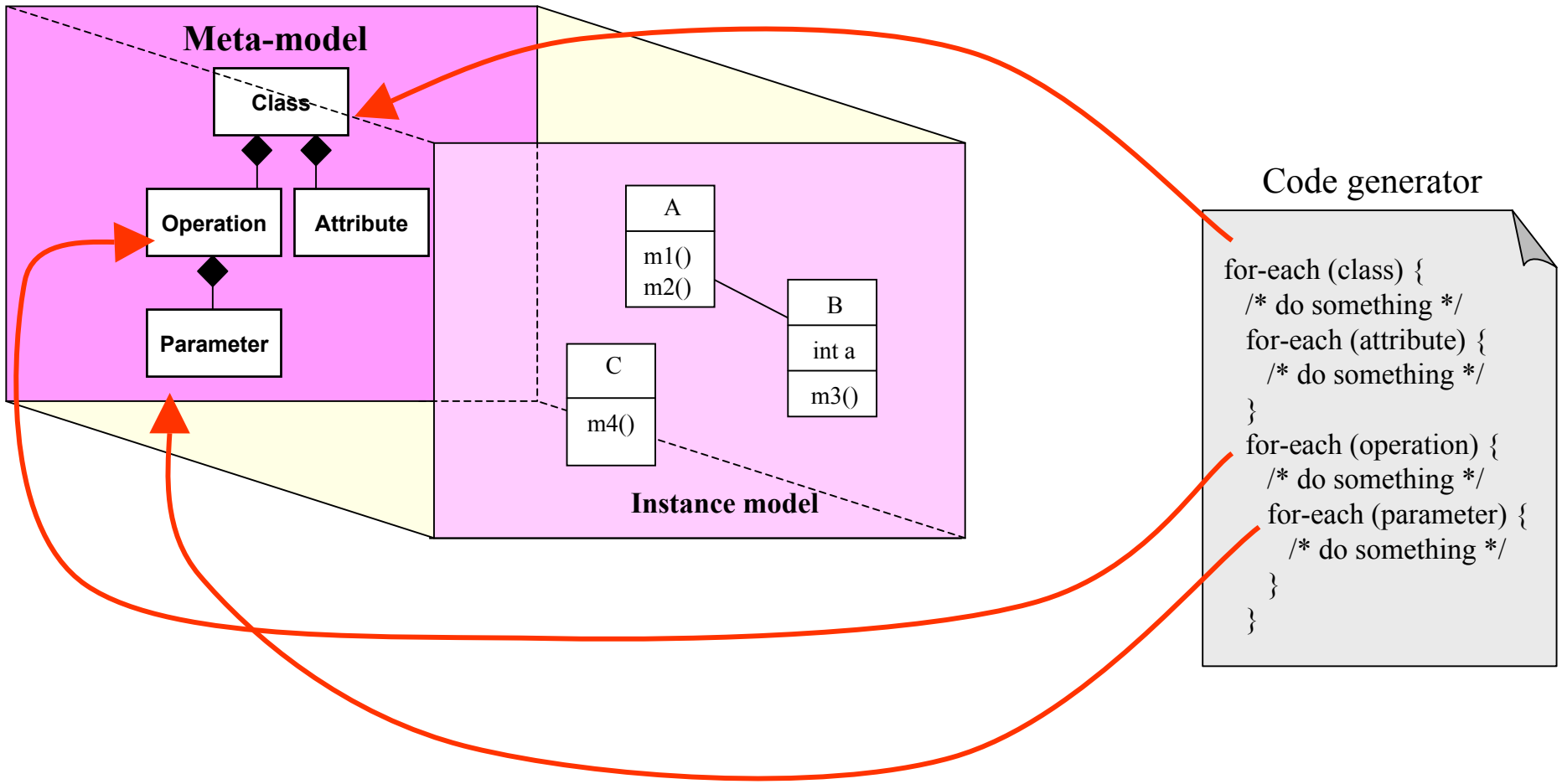
Current model-based code generators

- Monolithic programs that are external to the modeling tool
  - A dual hierarchy typically results

Composable Code Generation (CCG)

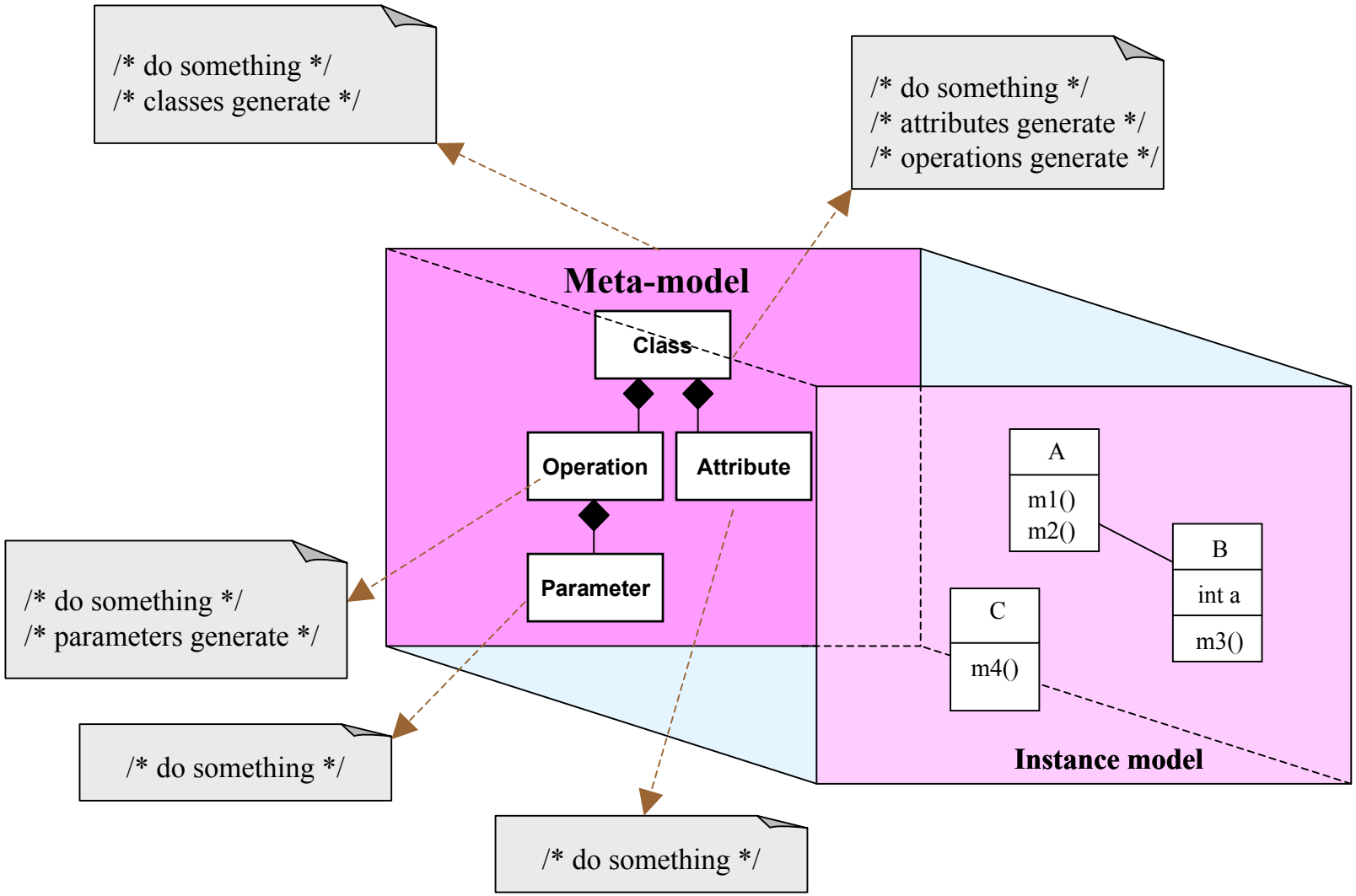
- A framework for adding automation to domain-specific modeling tools
- The key idea is that CCG applies object-oriented programming techniques to code generation
  - Modeling entities are like objects, each has a meta-entity (i.e., similar to a class)
  - A natural message to pass to a modeling entity is "provide your *Java* code generator"
  - Entity-specific code generators can be attached to the meta-data that define specific types of modeling entities.

Dual hierarchy example



The looping structure of the code generator depends upon the compositional structure of the modeling domain.

Composable code generation



Generator specialization

Meta-models are models

If you can attach generators to entities in meta-models, can you also attach them to entities in other types of models?

- Yes

What about to collections of entities?

- Yes

What about dynamically assigning generators based on model and/or entity properties?

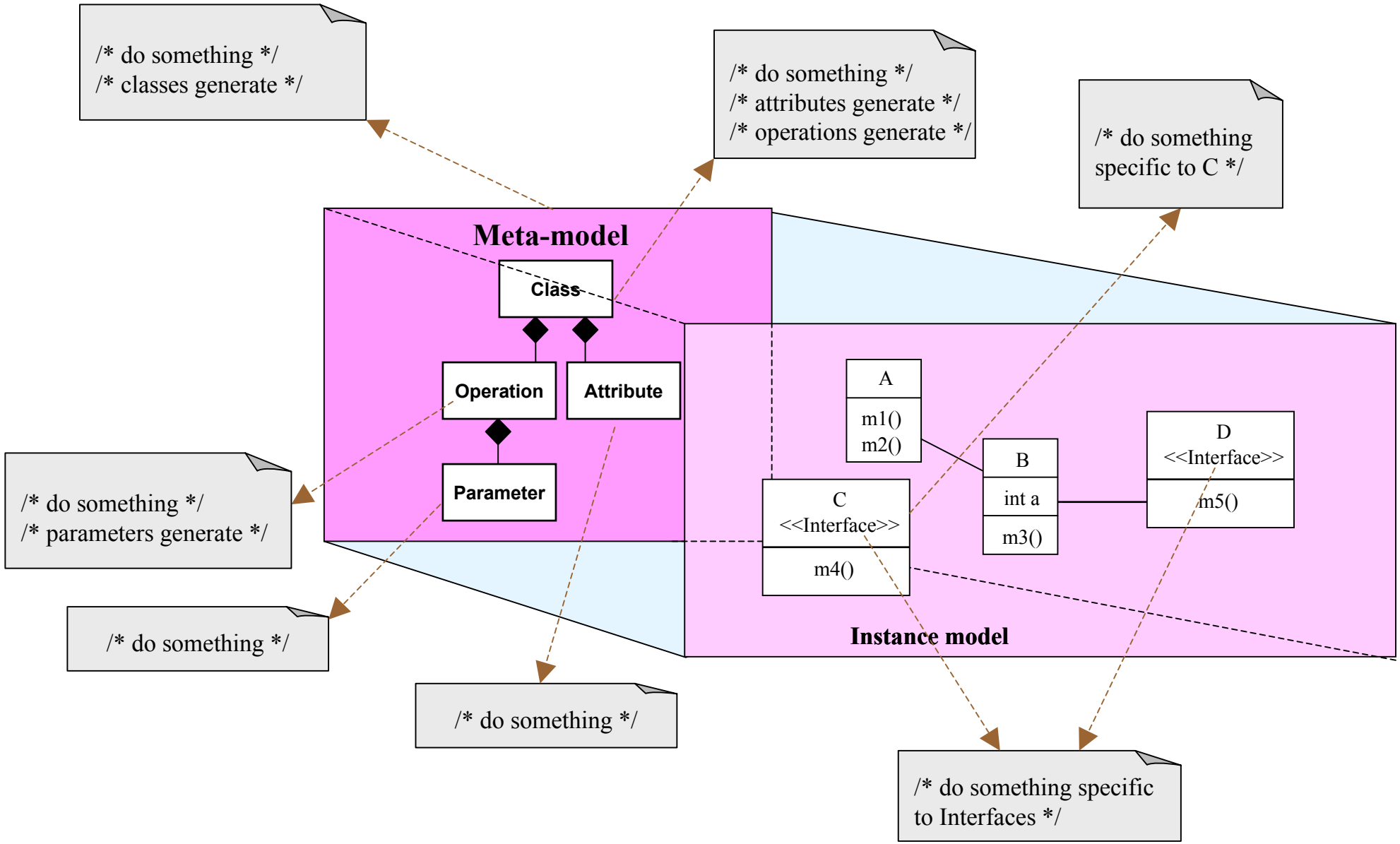
- Yes

Given these capabilities, collisions can occur

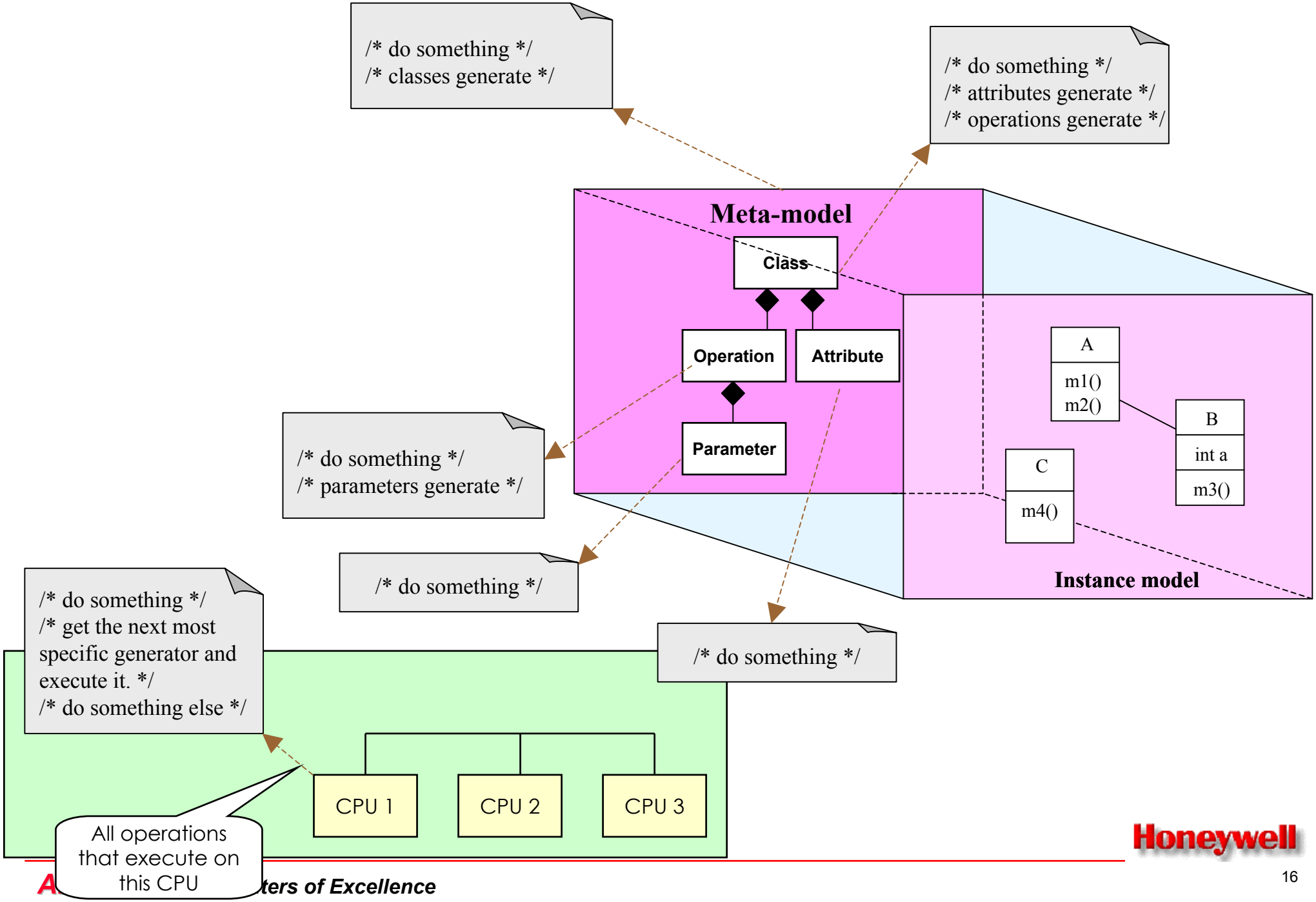
- The most specific generator is typically selected from a set
  - We refer to this as *generator specialization*

Honeywell

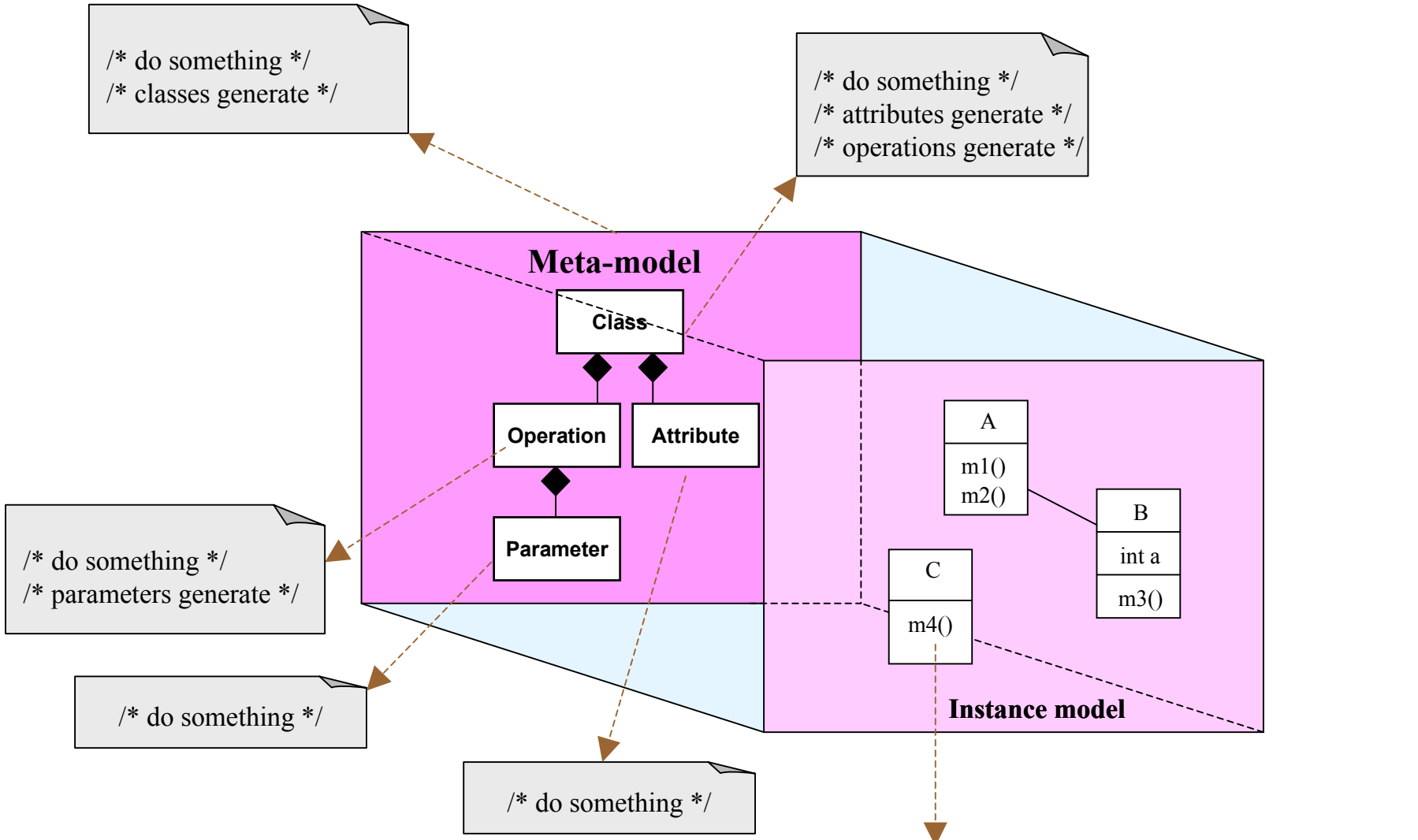
Generator specialization example



Cross-domain specialization



Multi-tool cooperative code generation



MoBIES Results

We developed a multi-model design capability

- for an open experimental platform based upon a jet fighter weapon & navigation system

Our design environment generated XML code for

- middleware configuration
- import into event-dependency and timing analysis tools

We wrote 35 CCG code generation routines in support of these requirements

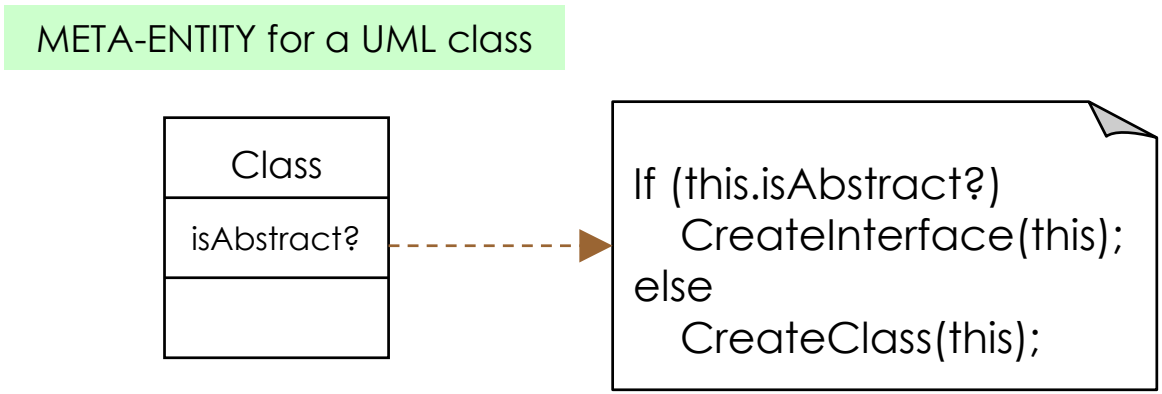
- Average length: 51 lines
- Min length: 6 lines
- Max length: 199 lines

- Generated code from mid-sized models (hundreds of entities)

Application to MDA (1)

Model Type Mappings

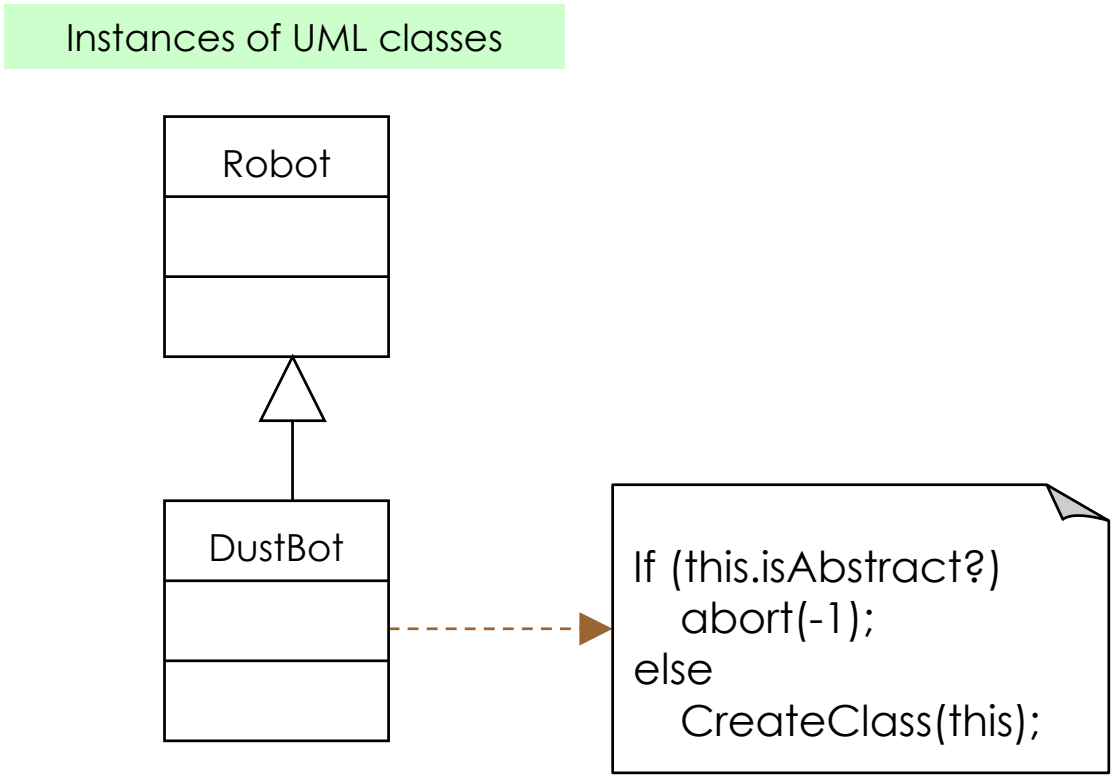
- Model types (meta-entities define model types) can have type-specific generators that specify how they map to types in different notations.
- These can be keyed to properties



Application to MDA (1)

Model Instance Mappings

- Model instances can have instance-specific generators that specify how they map to different notations

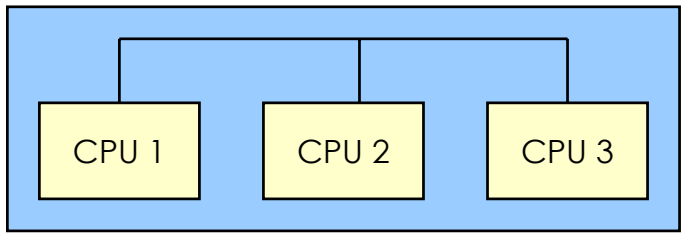


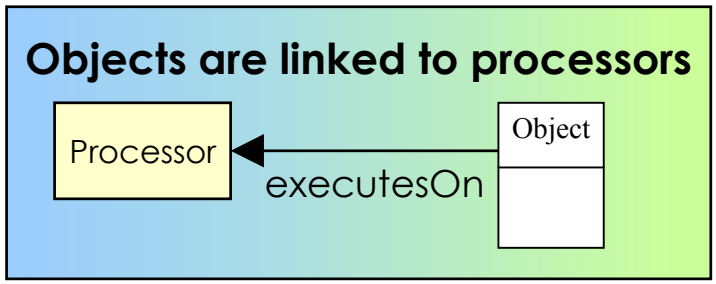
Application to MDA (1)

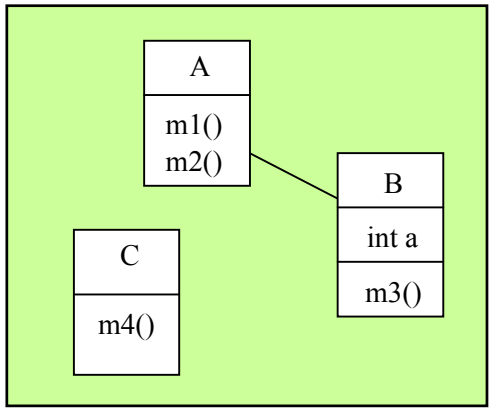
MDA notions

- Marks
  - Marks are essentially property definitions. These define name-type pairs and are specified on the meta-entity. Modeling entities have values for these.
    - e.g., All entities of type "Polygon" have the property "Number of sides" of type integer. Instances of type Polygon have a value for this property.

- Marking Models
  - Additional properties (aka marks) can be assigned to entity types from outside of their meta-models using the "Further defines" [1] cross-domain relationship.
  - Generators can be selected dynamically by model and instance properties.







[1] K. Schloegel, D. Oglesby, E. Engstrom, D. Bhatt. *A New Approach to Capture Multi-model Interactions in Support of Cross-domain Analyses*, 2001.

Application to MDA (2)

MDA notions

- Templates
  - Any entity-specific generator
- Model Merging
  - Traversal of models and generation of code from multiple notations is supported
    - This capability was crucial for the code generated under the MoBIES program
- Additional Information
  - Additional data for code generation can be imported dynamically from external tools and models, user input, or may be specified statically (i.e., hardcoded) in generators
- Patterns
  - We have developed a mechanism for the capture and reuse of design patterns that we refer to as archetypes [2].
  - Archetypes can have their own archetype-specific generators that override base generators

[2] D. Oglesby, K. Schloegel, D. Bhatt, and E. Engstrom. *A Pattern-based Framework to Address Abstraction, Reuse, and Cross-domain Aspects in Domain Specific Visual Languages*. In Proc. of OOPSLA 2001, 2001.

DOME is publicly available and open source

We are currently working on porting (and extending) DOME to Java

http://www.htc.honeywell.com/dome/