Model-Integrated Computing: 20 Years and still going…
… but where?
Quo vadis, MIC?

Gabor Karsai
What is it?

- **Programming** and **building** systems **from** and **with** models, where you can define your own modeling language.
  - Higher-order programming
  - Visual programming
  - Systems engineering with models
  - Modeling + Analysis + Generation
  - ....
  - System construction via domain-specific models.
How is it done?

- Figure out what you need… to generate
- Define your own modeling language…. or modify an existing one
- Try out your language
- If you like it, figure out how to translate your models into something what you really need.
- Iterate until you get it right
OK, so what can we do?

- Generate code from models
  - Glue (‘boring’) code
  - Functional code
- Generate non-code from models
  - Docs
  - Other models ← Analysis!
  - Build scripts
  - …???
- Integrate systems via models
How about other, generative approaches?

AOP

- Base code
- Aspect-1 code
- Aspect-2 code
- Joints

- Weaver
- Source code
- Compiler
- Executable

IP

- IP Space
  - Intentions stored as tree-like graphs
  - Visualization, etc.
  - reduction enzymes
  - intermediate code
  - code generator
  - executable

MIC

- Metamodels
  - Language definition
  - Models
  - Model Space
  - Models stored as hypergraphs
  - Visualization
  - Editing engine
  - Transformation engine
  - Component/configurations
  - Component Integration Platform
Economics: Is it worth it?

- With MIC, you build custom languages (with explicit abstractions) and build or integrate tools to solve problems
  - Amortize the cost of language and tool building over the entire software development
- If the language/tools give you more gains than the effort it took to build them, SURE!
What we need to do…

- Use MIC for building real systems
  - More FCS-like projects, not demo systems
    … a.k.a. ‘eat (more) of your own dogfood’.
- Figure our how MIC fits together with existing development techniques
  - Integrating model-generated and hand-written code
  - Testing, continuous integration, version control, etc.

Industry perception about MDA today: it is useless because you model, and then you have to do the same things what you would be doing anyway, and pretty pictures don’t help much.
What we need to do…

- Rethinking the (meta-)toolchain
  - What platform and what language?
  - Scripting and component integration?
  - Relationship to standards (real and de facto)?
  - Doing simple things in a simple way?
  - Weaving models and text(ual languages)
  - Working in the large
What we need to do...

- Build some more foundations
  - Model execution? (as in ‘quick feedback for the designer’)
  - A language ‘lab’ for rapid experimentation?
  - Model debugging?
  - That little matter of semantics…
  - Trust but verify…
In summary...

- MDE is being recognized today but it has not reached its full potential yet
  - Too much snake-oil, too many research prototypes, too few working systems built
- With MIC we have an opportunity
  - But we have to run very hard just to stay in place 😊