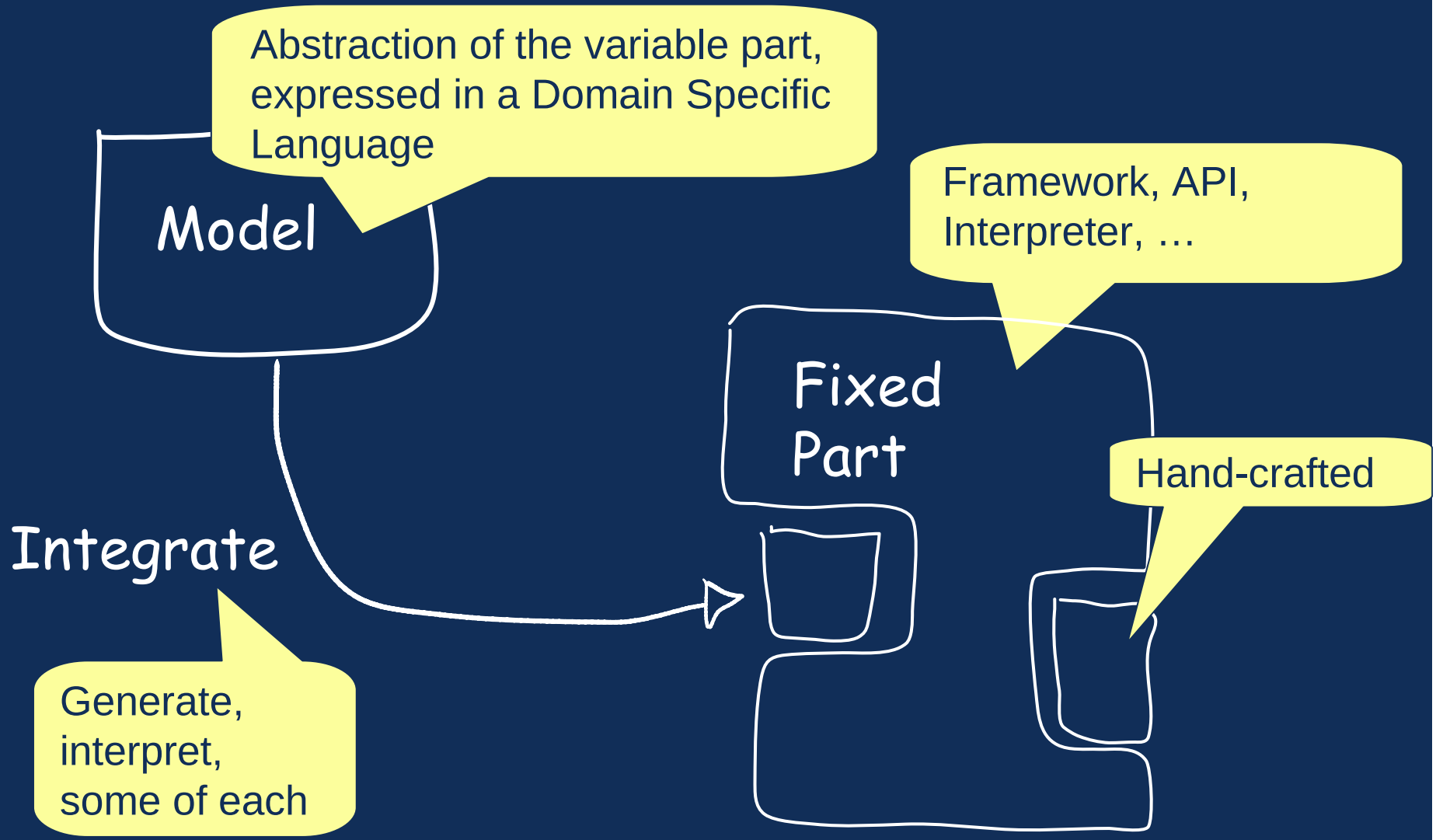
Domain Specific Development

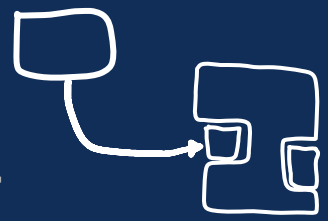
Stuart Kent, Senior Program Manager,
Visual Studio Team System, Microsoft

Domain Specific Development Pattern



Contents

- ✓ Domain Specific Development Pattern
- Demo 1 – Wizard UIP example
- DSD – a Software Factory Pattern

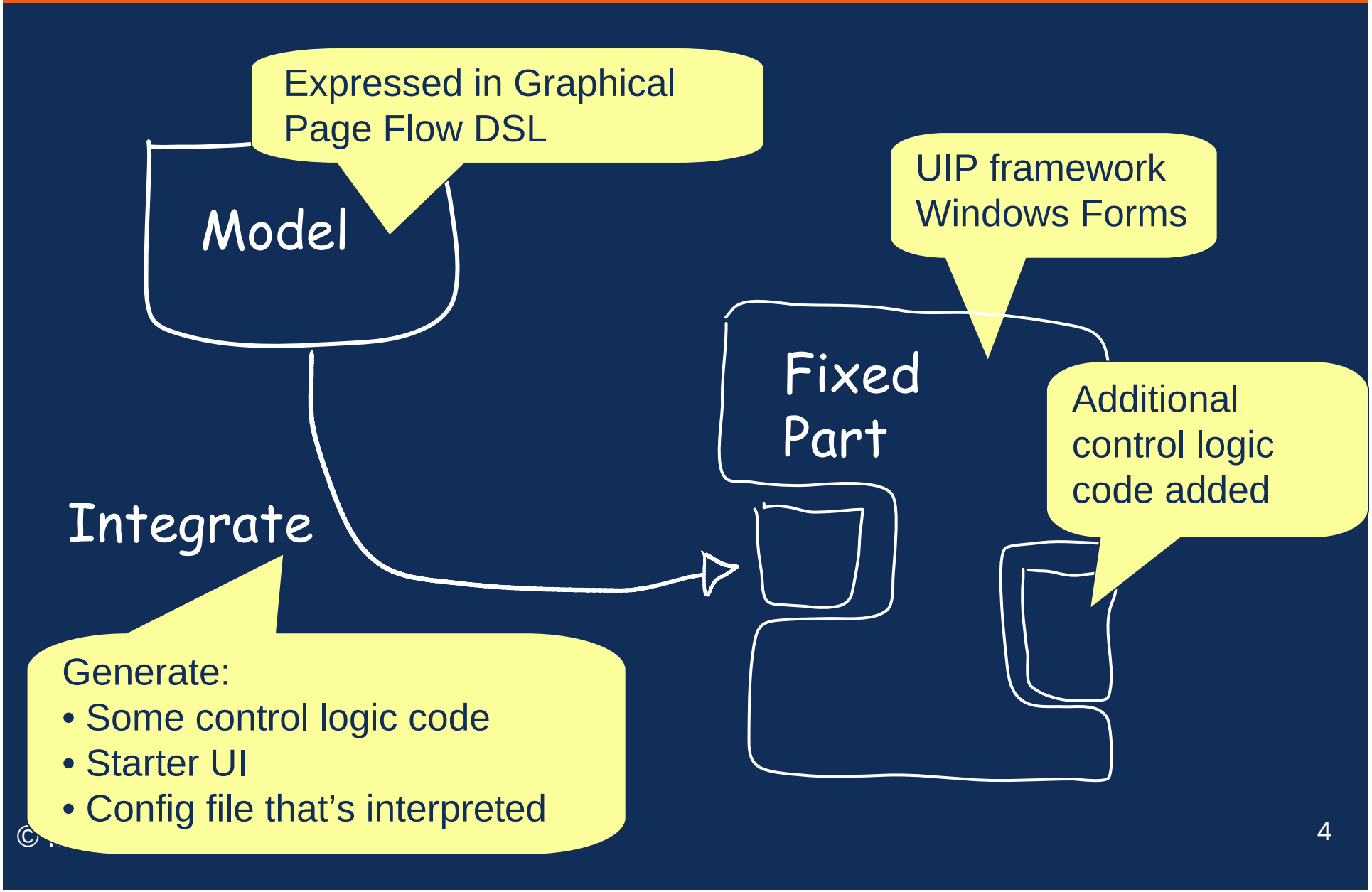


- Domain Specific Languages (DSLs)
- Integration: Generation versus Interpretation from/of models
- Costs & benefits of the DSD approach

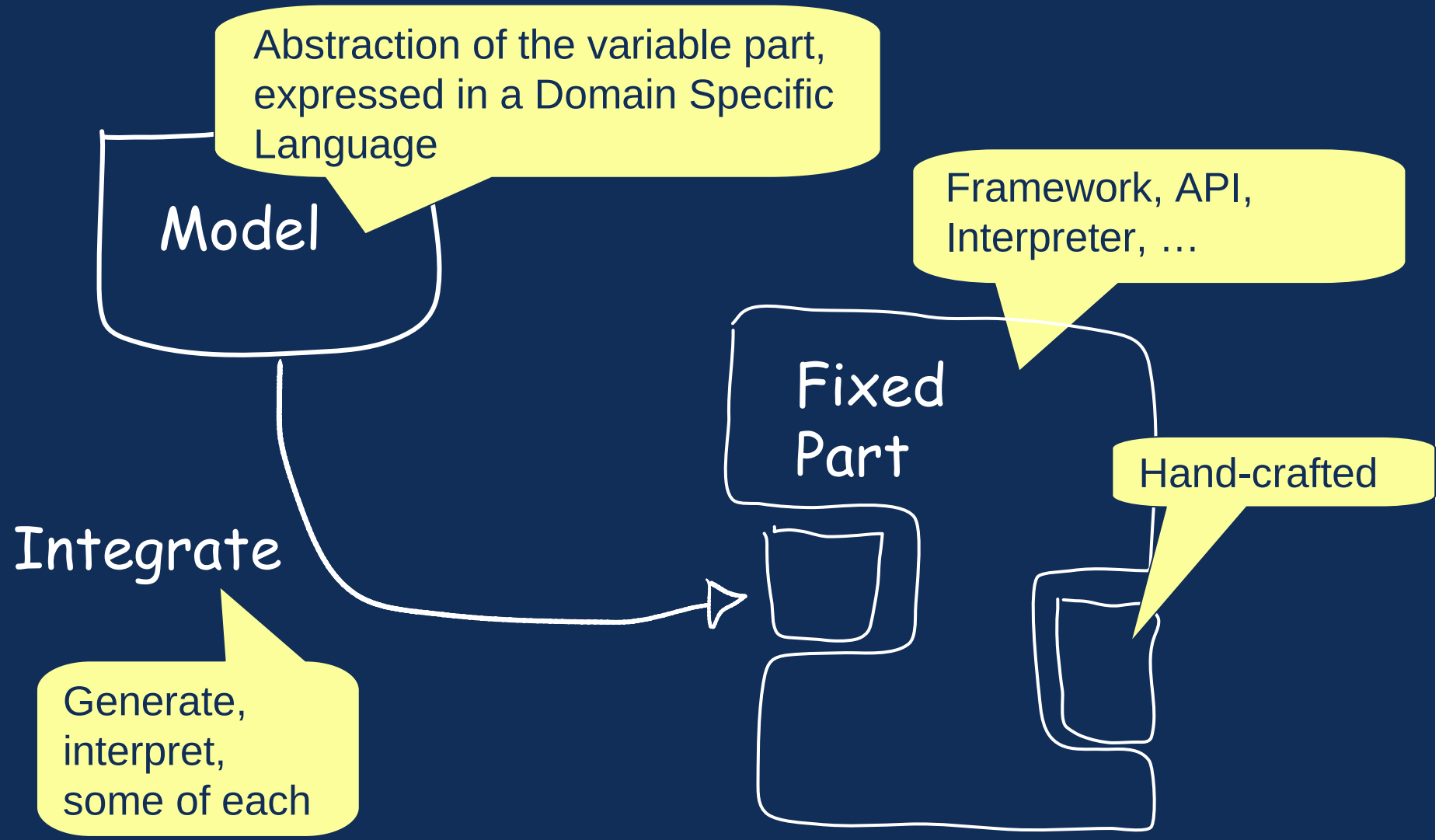
- DSL Tools in Visual Studio
- Demo 2 – Building and deploying a DSL

- What Next? – DSL Tools
- What Next? – Software Factories

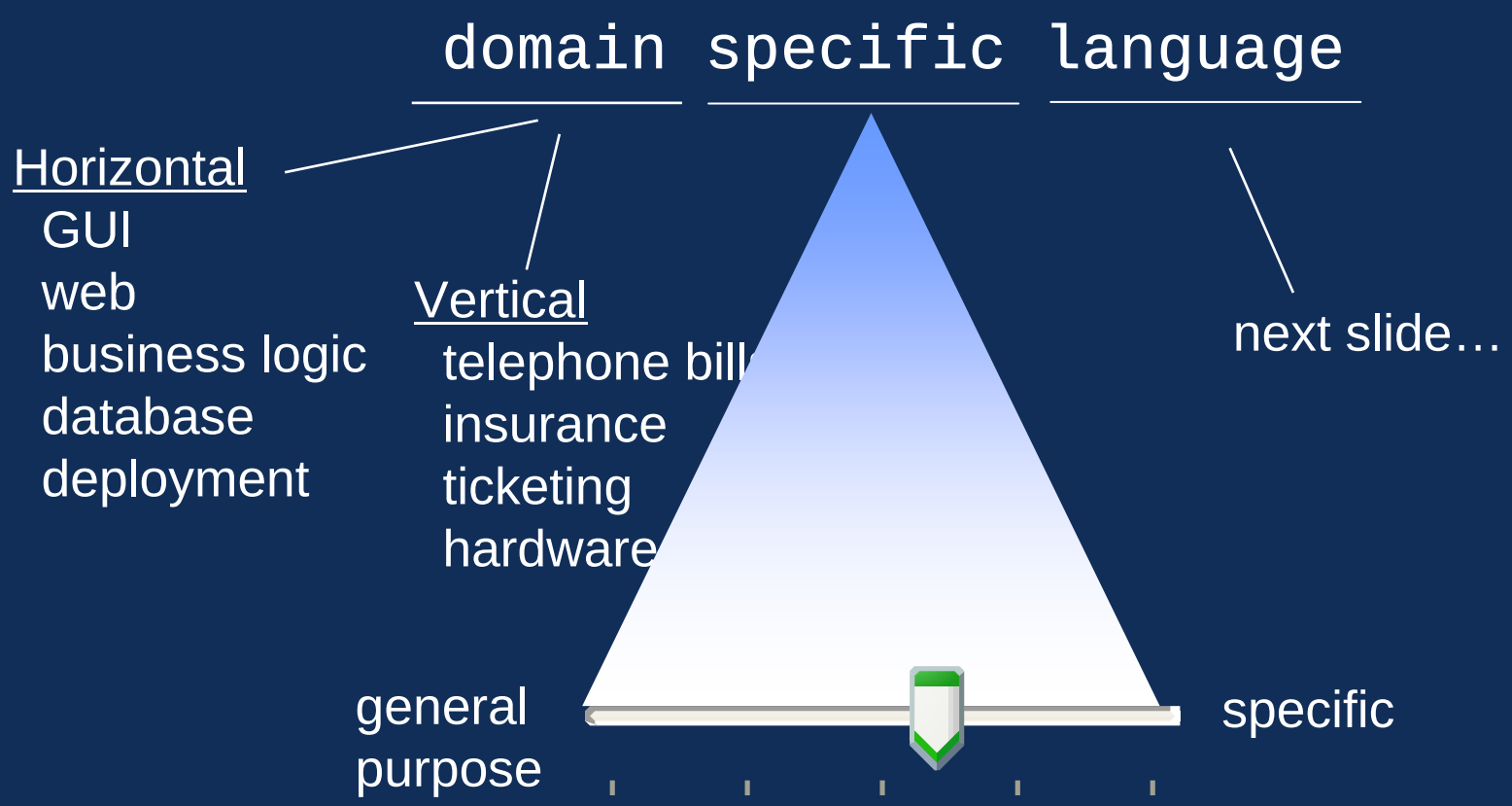
Demo 1 – Wizard User Interaction Process



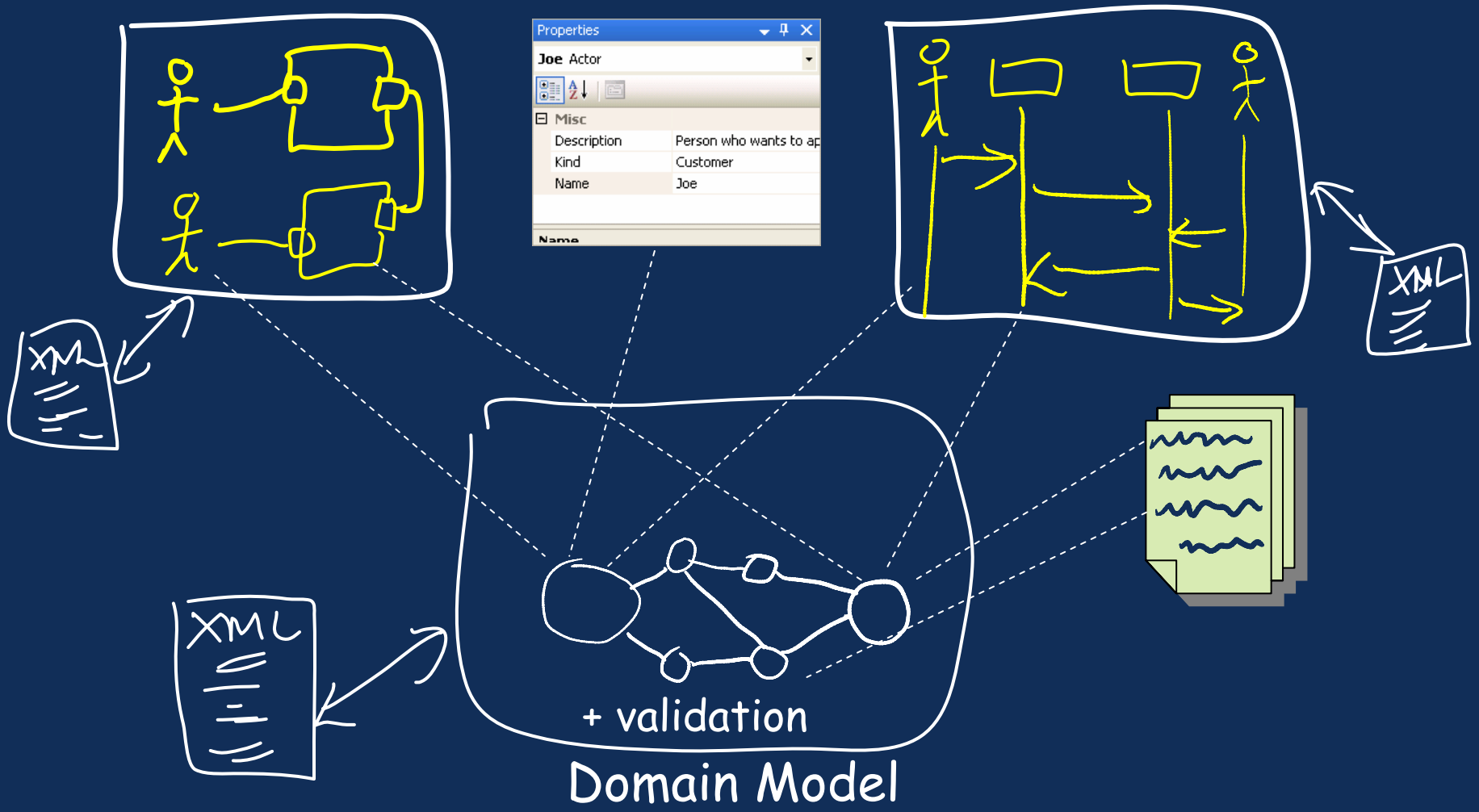
Reminder – DSD Pattern



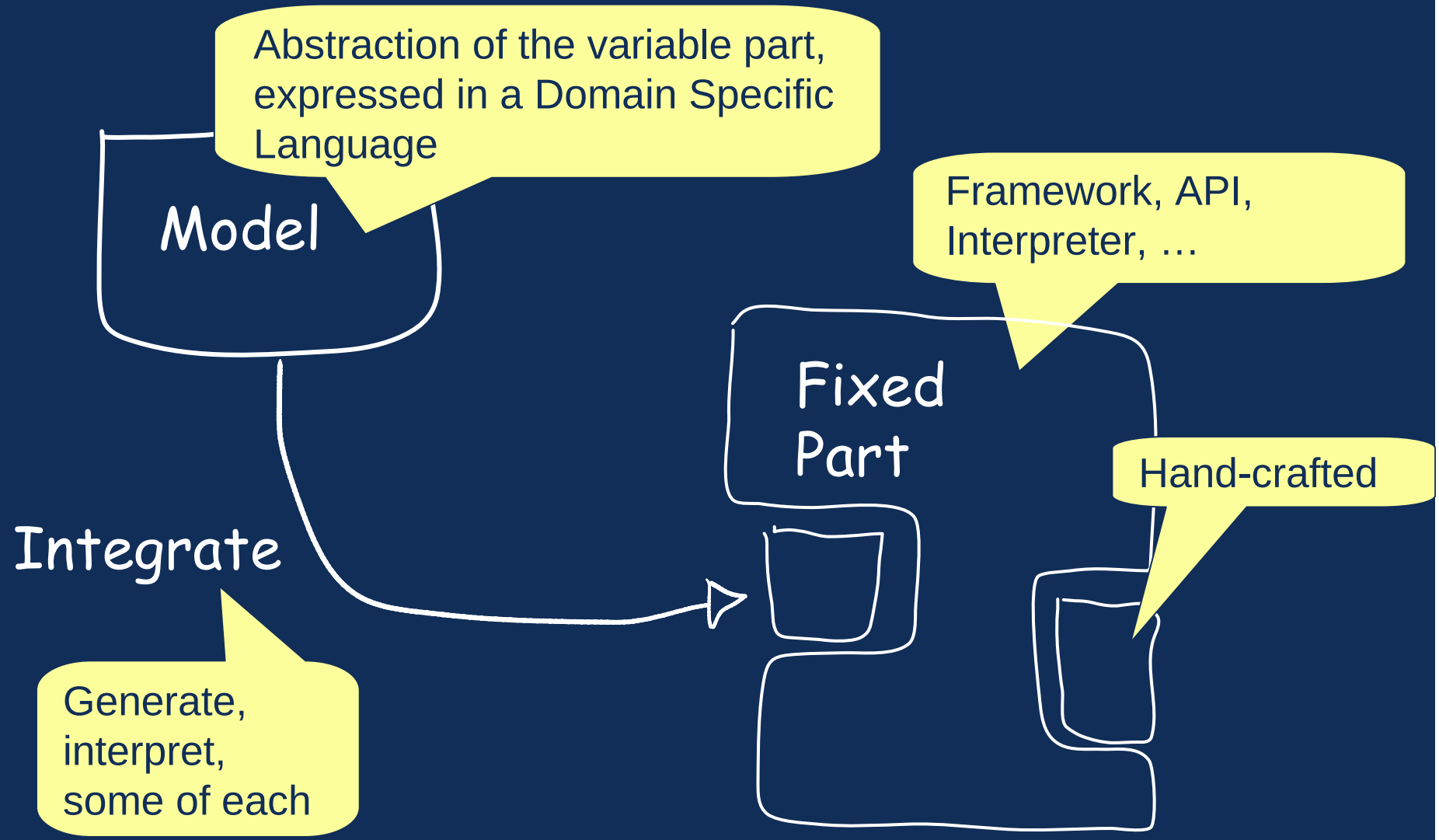
Domain Specific Language Classification



Parts of a DSL



Reminder – DSD Pattern



What does “Integrate” mean?

- Generate
  - Generate code that talks to framework APIs
    - E.g. completely new classes
  - Generate code that completes a framework
    - E.g. C# partial classes
- Interpret
  - The model file itself, or some data pushed into a live environment from the model (e.g. xml configuration file, data in a database) is interpreted
- A Combination of the above

- Easier to produce initially
- More customization options

- Behavior can be changed at run-time
- Fewer artifacts to keep in sync

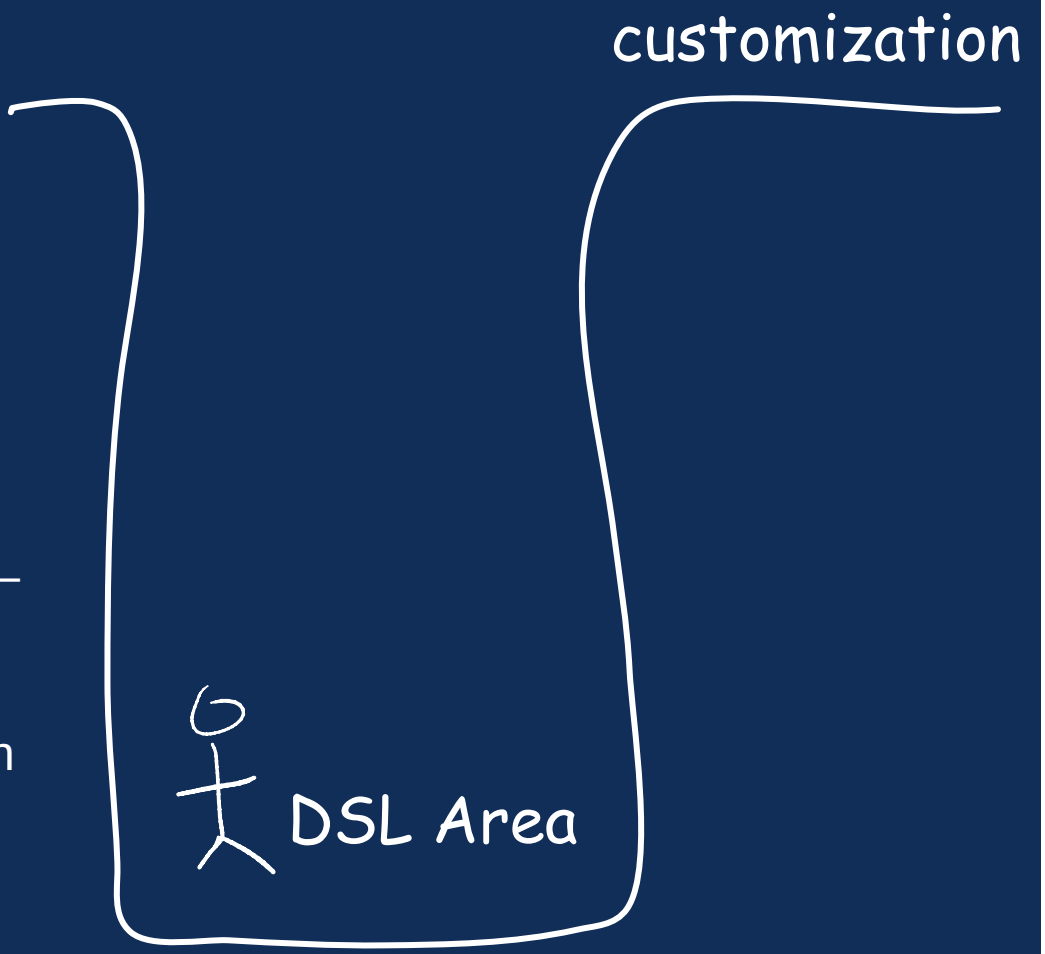
Customization (1)

What?

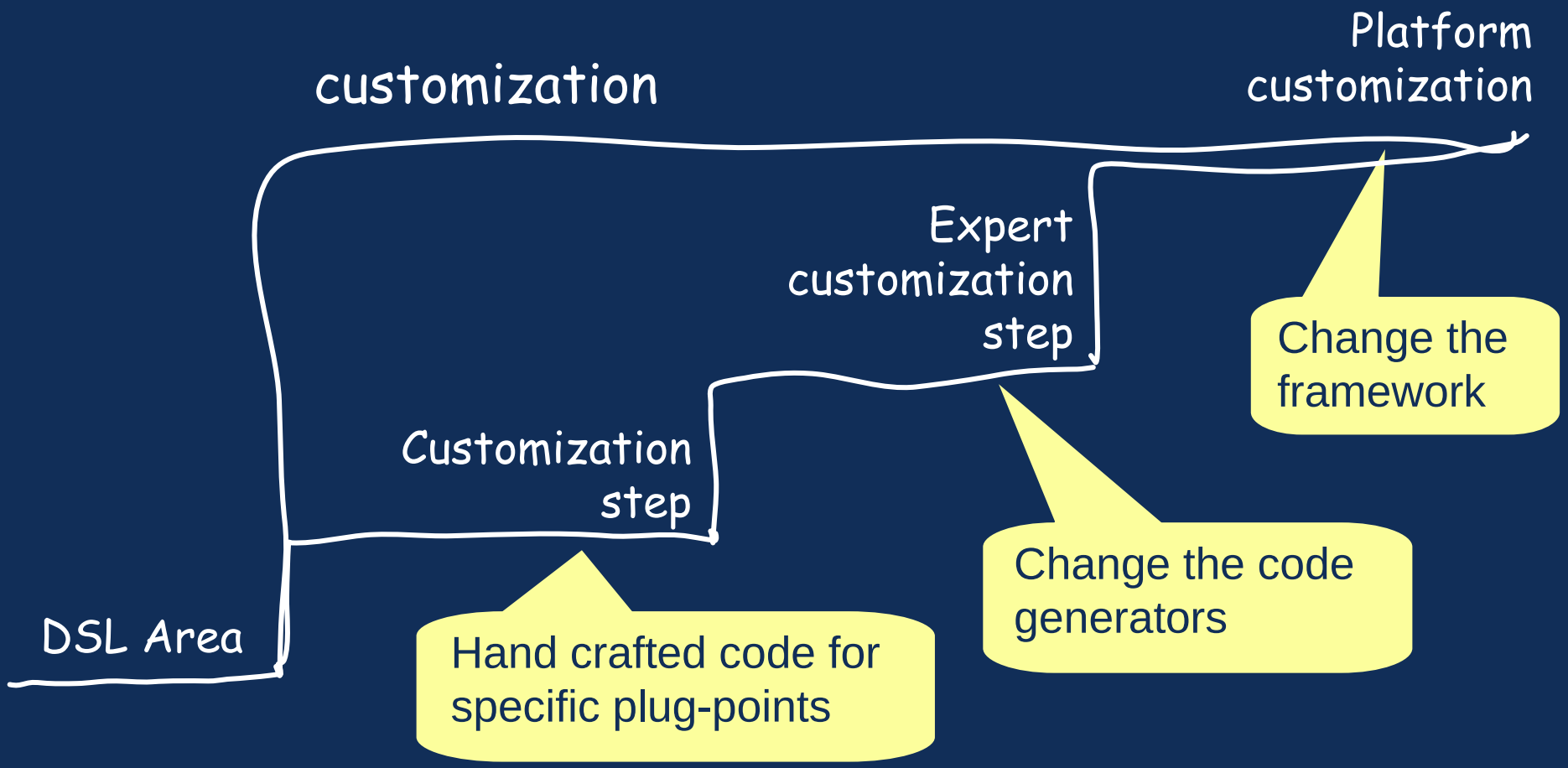
- Customize/extend the behavior expressed by the model + fixed part

Why?

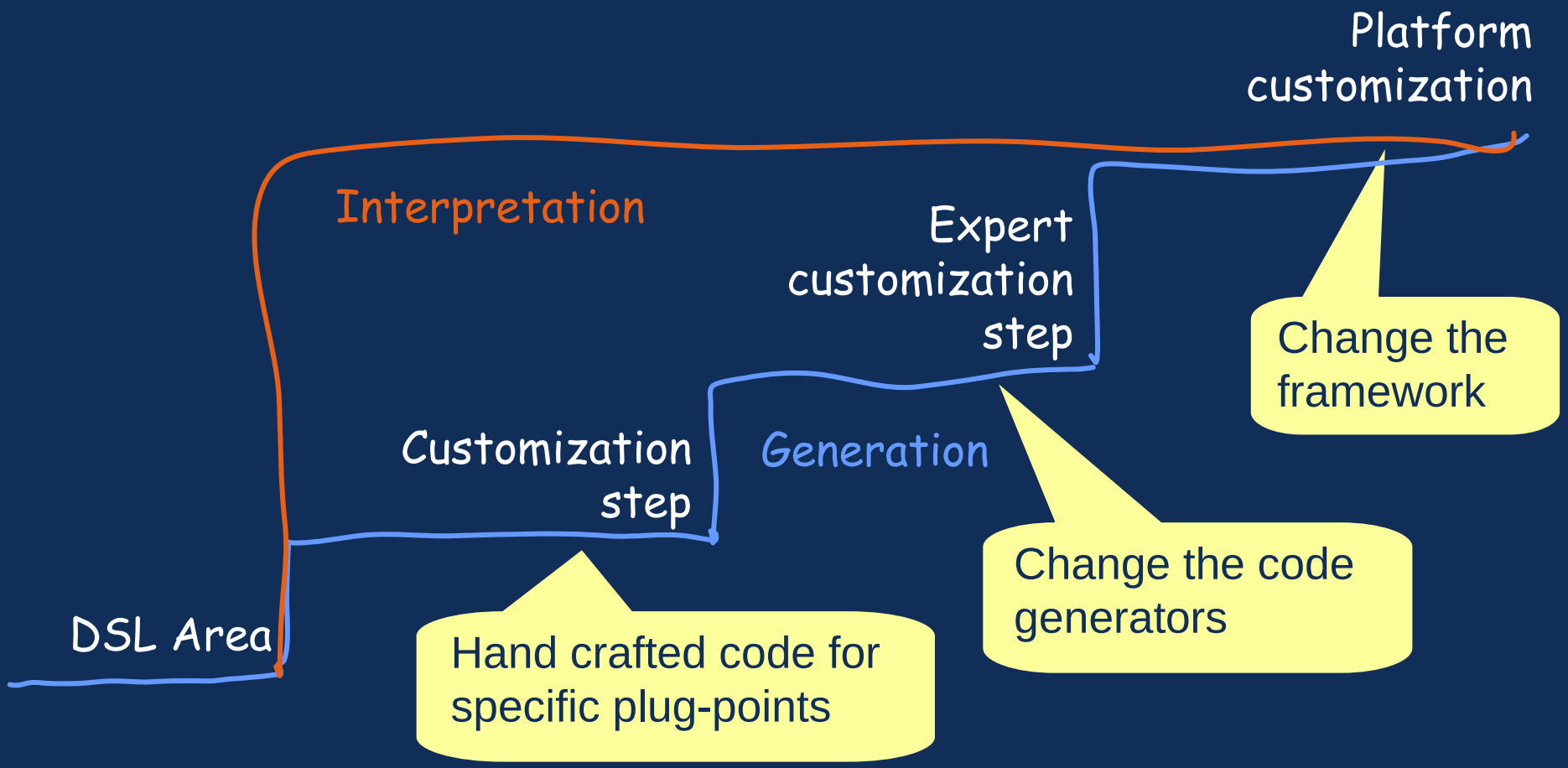
- However good the fixed part, not every point of variability will have been predicted
- Some things (e.g. complex logic) are just easier to express in code – avoids reinventing programming languages in a DSL
- There will be undiscovered bugs in the code generators / interpreter / framework which need to be fixed somehow



Customization (2)



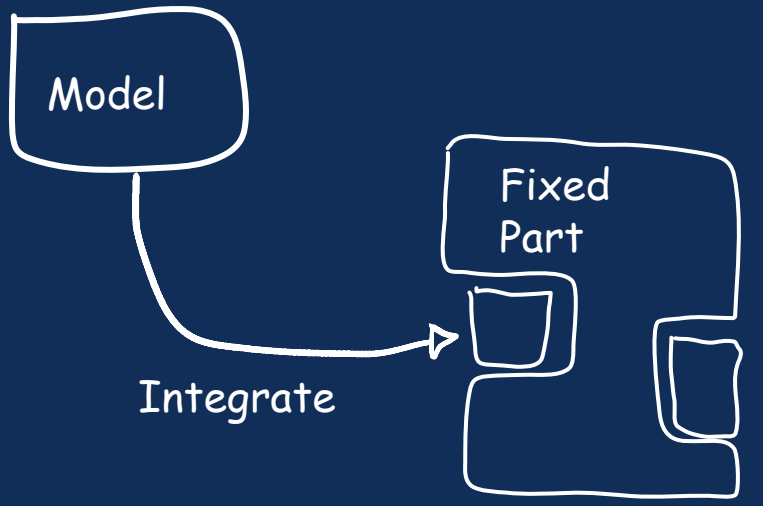
Customization (2)



Domain Specific Development – Cost / Benefit

Benefits

- Fewer copies of the same code to maintain
- Fix bug in code generators or framework fixes for all
- Change in requirements (e.g. business rules) = change in model
- Change in technology = change in code generators / framework
- Better communication with stakeholders



Cost

- Creating/maintaining the DSL / code generators / framework potentially across many projects and versions

Goal of Microsoft DSL Tools is to bring down this cost

DSL Tools in Visual Studio

What?

- Tools for authoring and deploying DSLs and template-based text artefact (e.g. code) generators

Where?

- Now installed as part of the Visual Studio SDK

  http://msdn.microsoft.com/vstudio/extend/

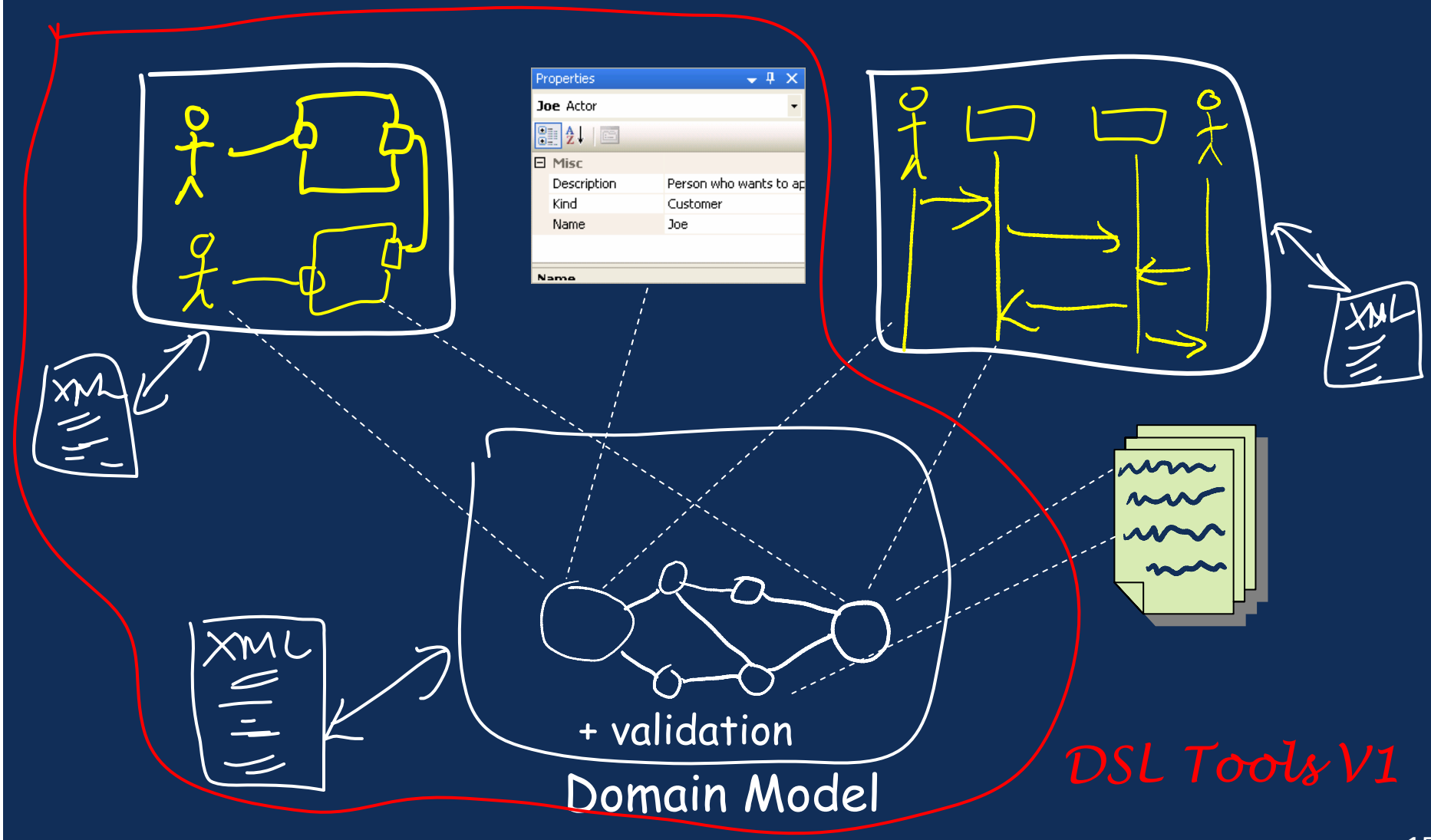
- Also see

  http://msdn.microsoft.com/vstudio/DSLTools/

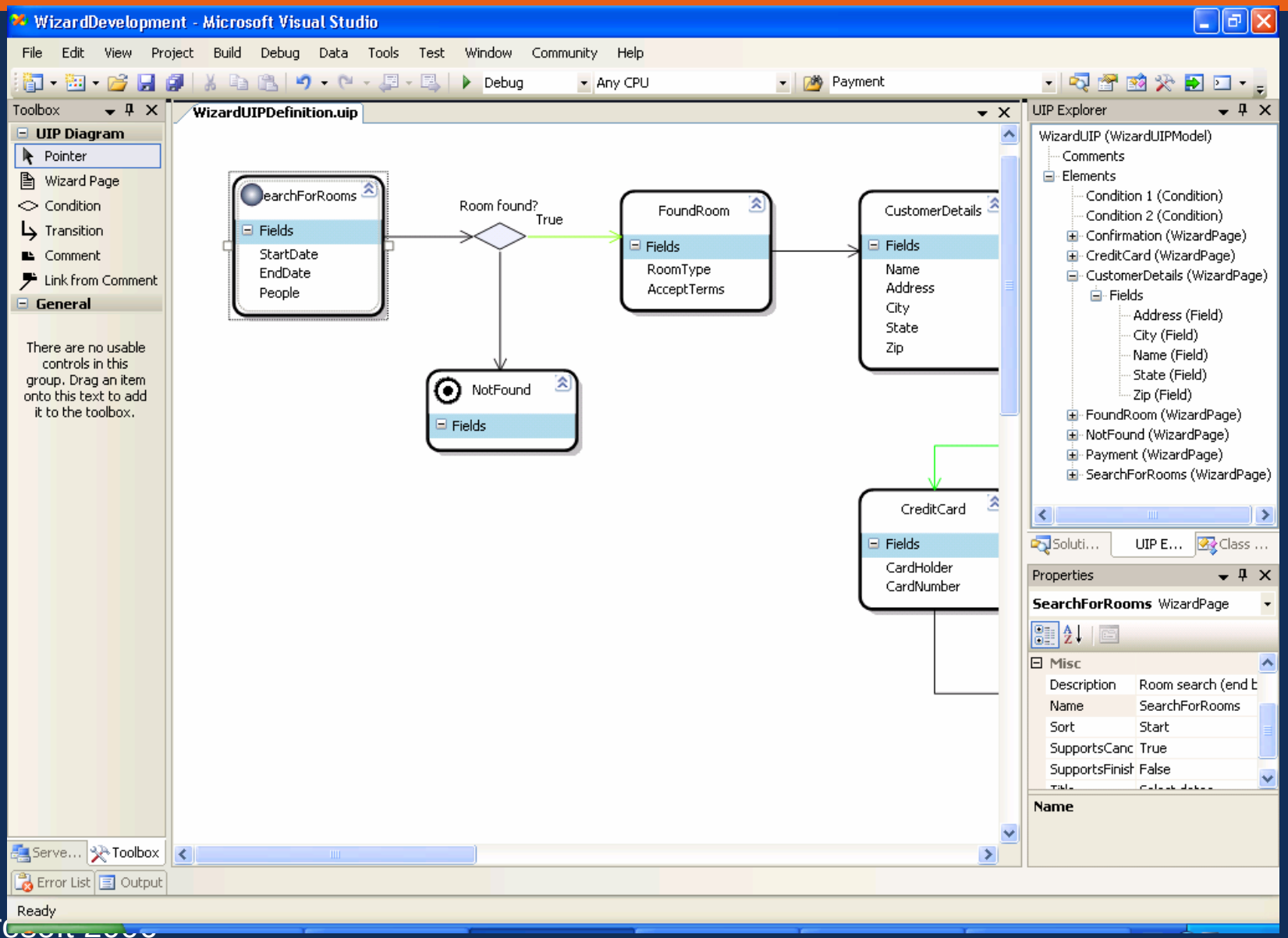
When?

About to release Version 1 (Sept'06)

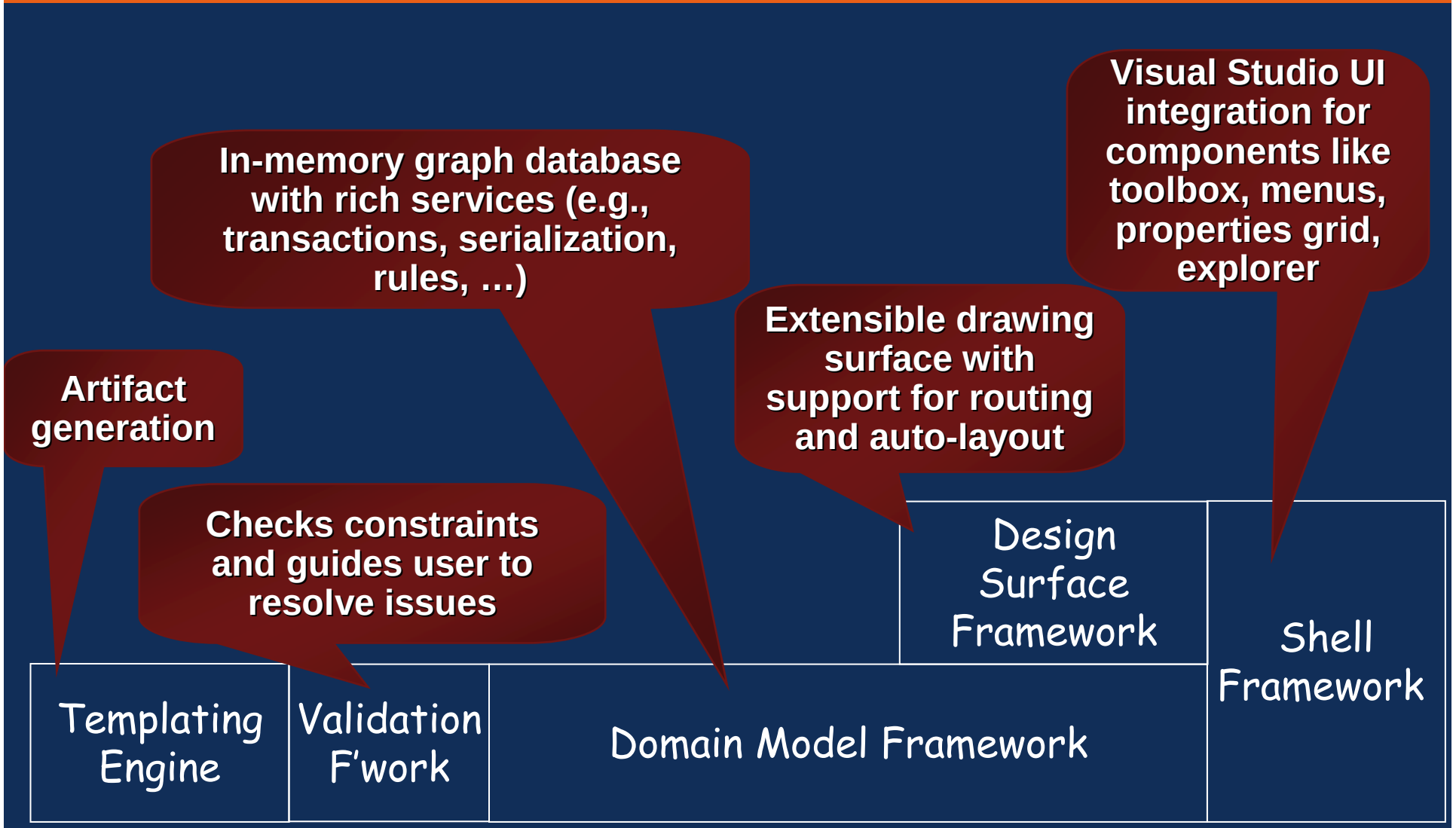
Reminder – Parts of a DSL



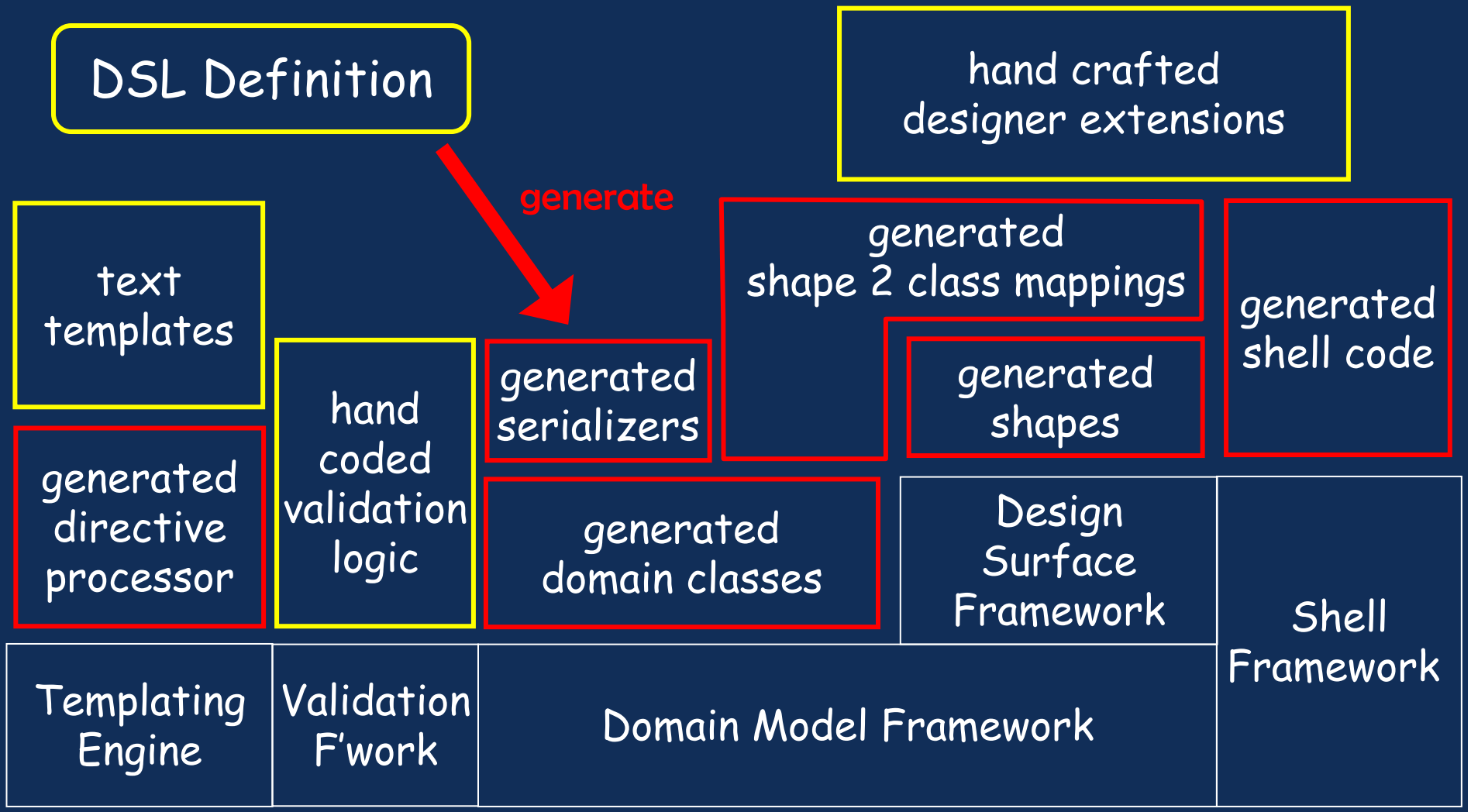
Designer for a DSL in Visual Studio



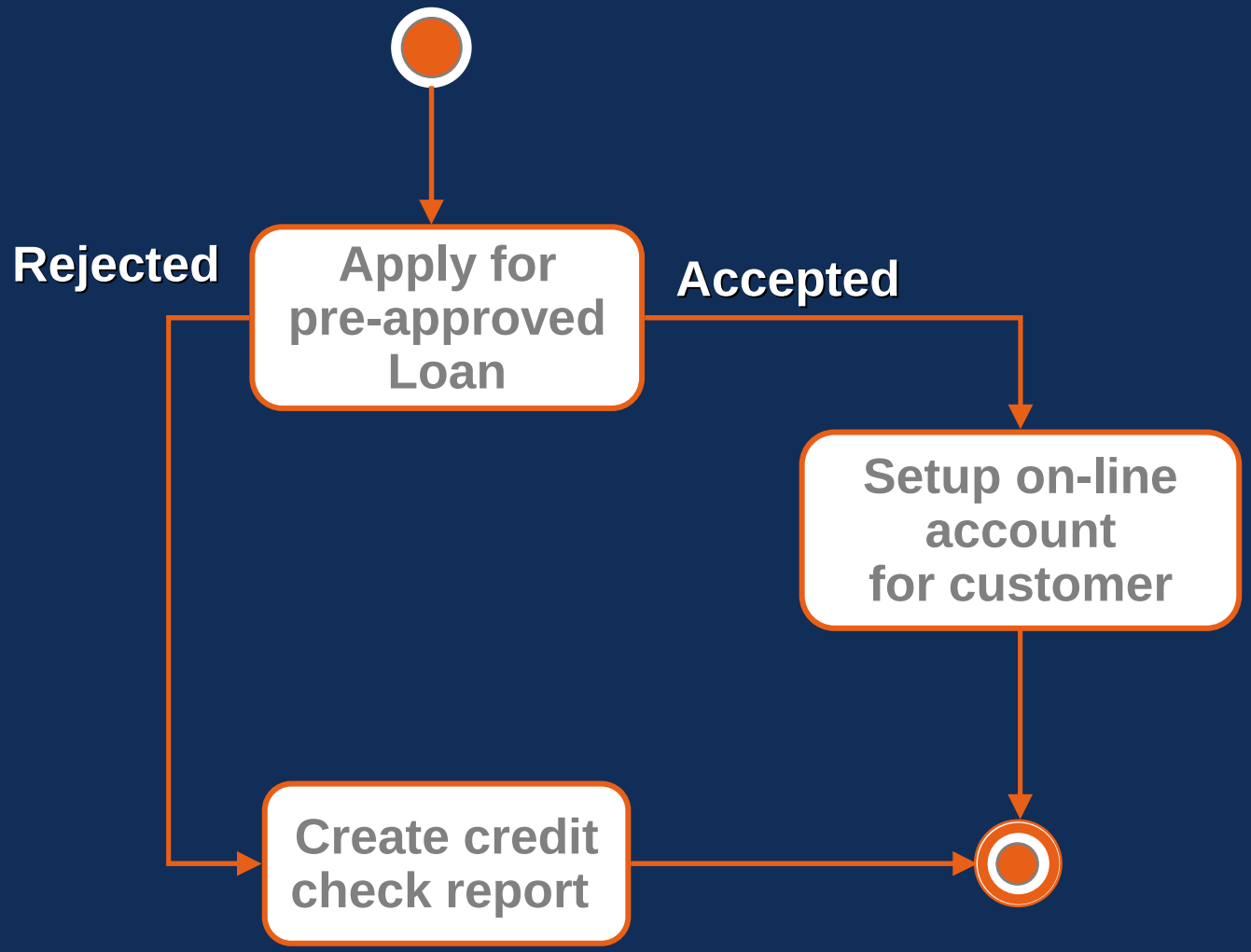
DSL Tools – Modeling Platform



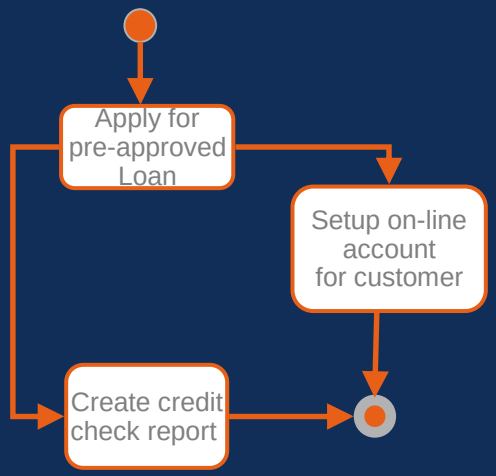
DSL Tools – Architecture of a Designer

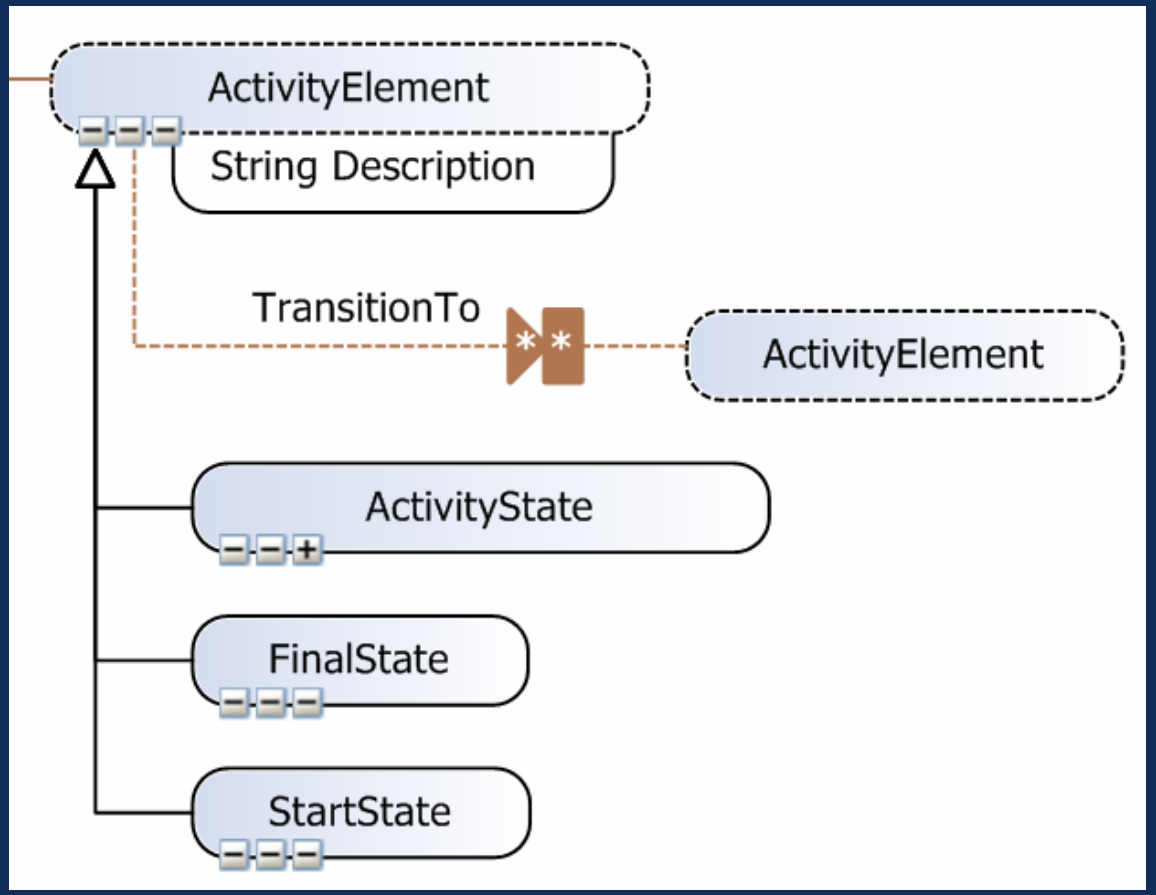


A DSL for Activities

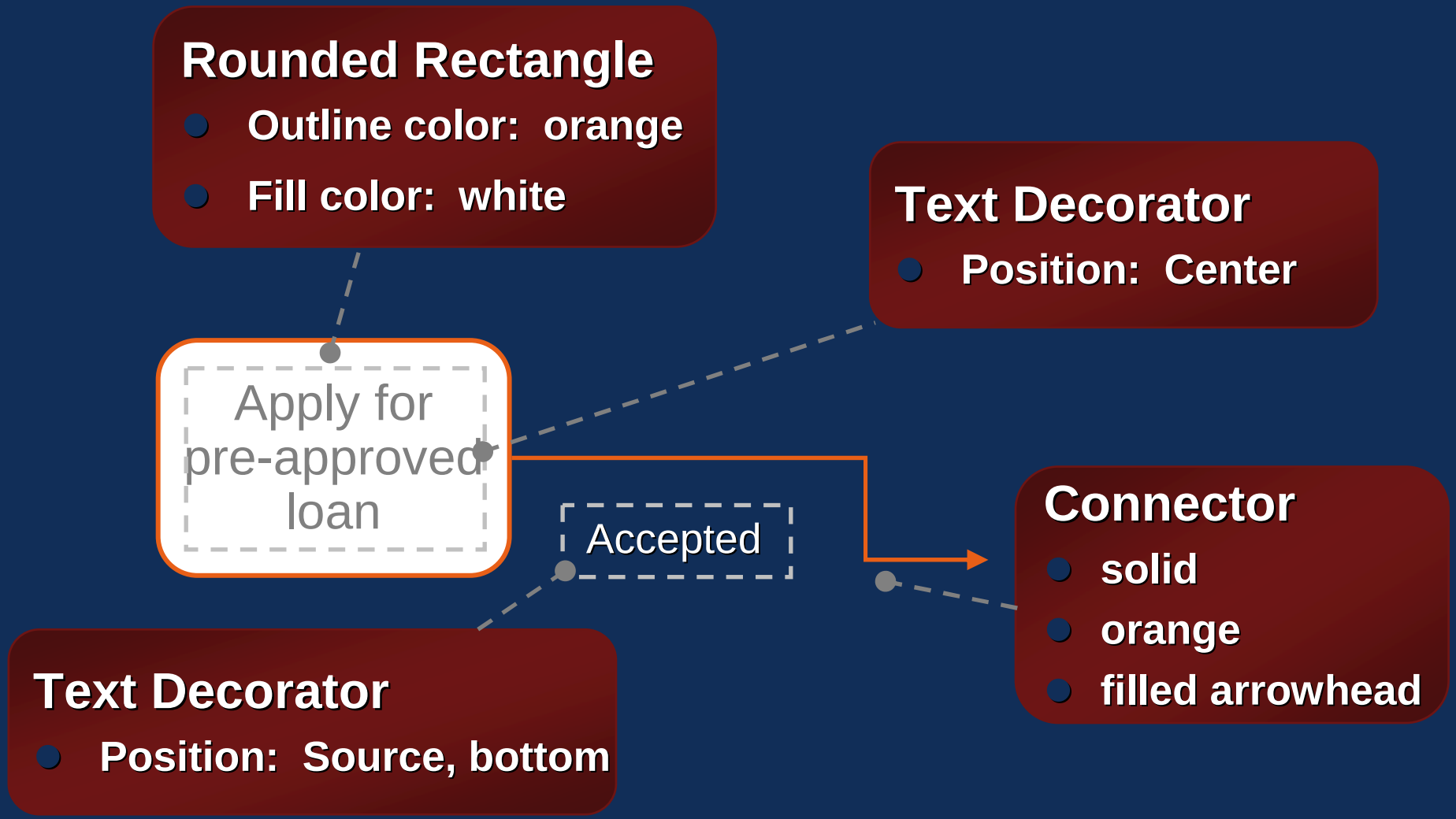


Defining a DSL, Step 1: Define Domain model



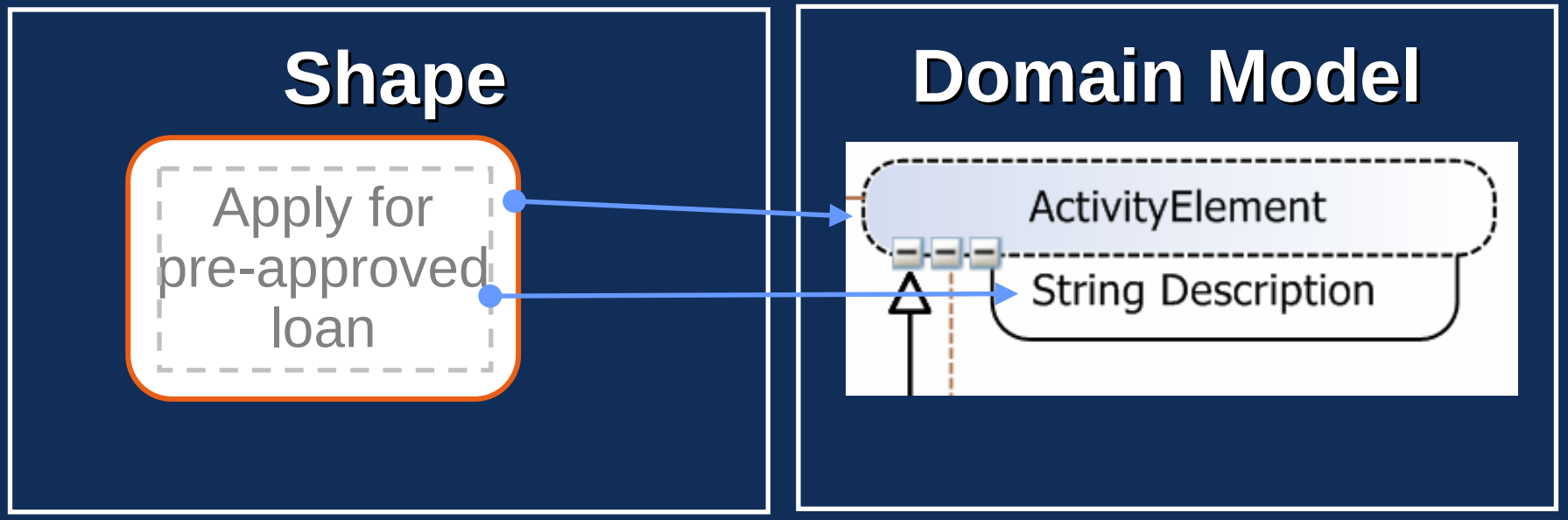


Defining a DSL, Step 2: Define Notation

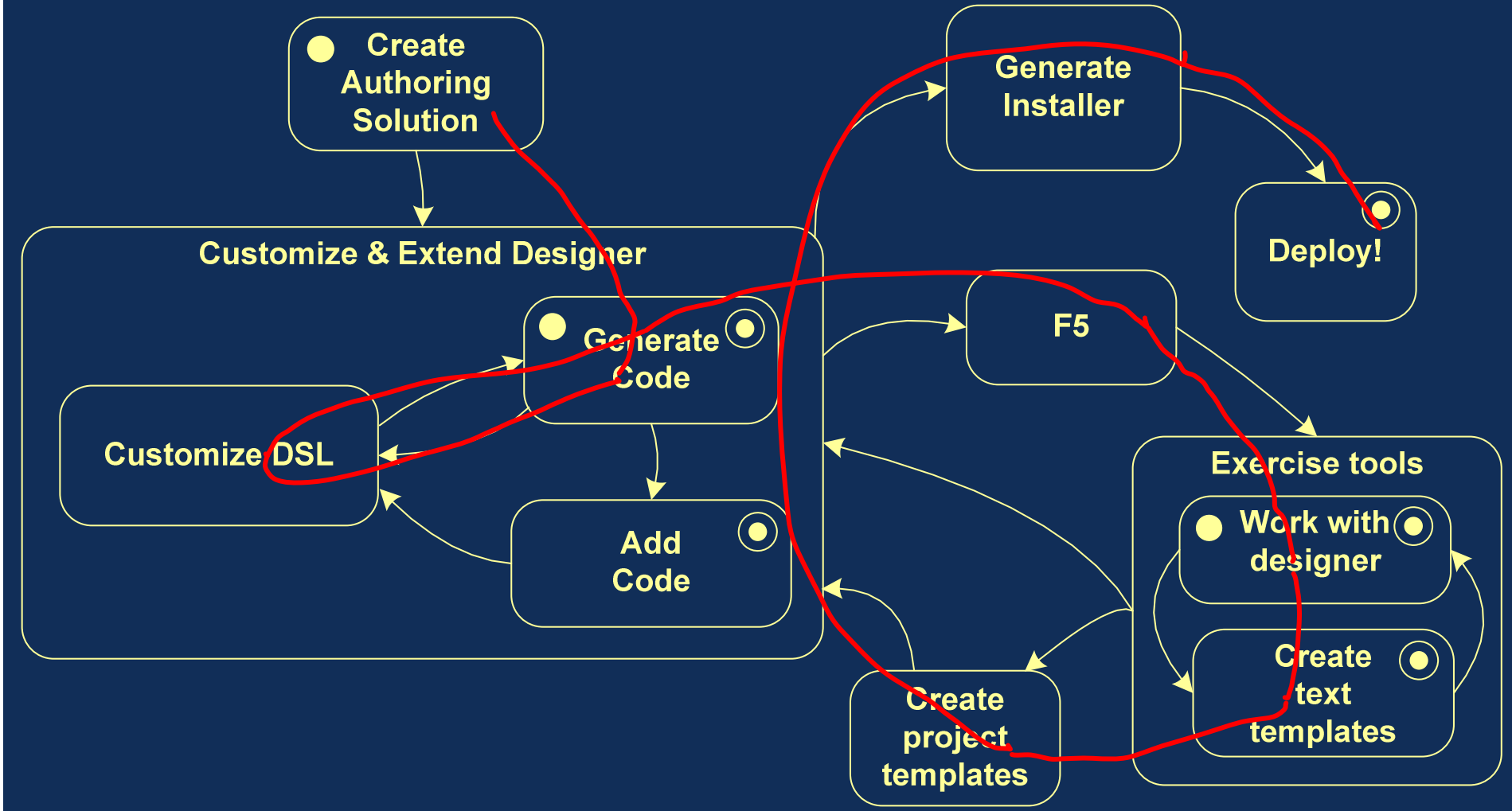


Defining a DSL, Step 3

Define visualization of domain model via notation elements

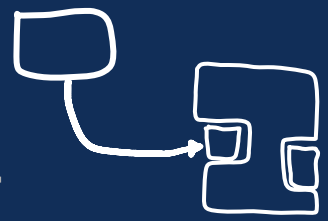


Demo 2 – Building & Deploying a DSL



Contents

- ✓ Domain Specific Development Pattern
- ✓ Demo 1 – Wizard UIP example
- ✓ DSD – a Software Factory Pattern



- ✓ Domain Specific Languages (DSLs)
- ✓ Integration: Generation versus Interpretation from/of models
- ✓ Costs & benefits of the DSD approach

- ✓ DSL Tools in Visual Studio
- ✓ Demo 2 – Building and deploying a DSL

- What Next? – DSL Tools
- What Next? – Software Factories

What Next? – DSL Tools

- Key problem is how to deal with large-scale models
  - Possible solutions
    - Visualize the model through multiple diagrams, possibly involving many designers
    - Better facilities for expand/collapse, exploiting model structure (e.g. trees)
    - Zooming in/out of diagrams
    - Drill-in/drill-out with different notations being used for different levels of detail
    - Use of autolayout
    - Break up model into multiple files to support finer-grained source control
    - Small designers integrated into software factories
    - What else?
- Also new notations, better authoring, …

What Next? – Software factory platform

- DSL Tools become part of a wider Software Factory platform
- A software factory is a configuration of a software development environment aimed at developing systems in a particular domain
  - E.g. you may have a software factory aimed at building web front ends for enterprise systems
- Software factories might need further customization on a per project basis, or indeed constructed for a specific project (generative programming)
  - E.g. code generators may need to be adjusted to account for existing legacy systems
- Software factories integrate:
  - DSLs associated with viewpoints
  - Code generation actions
  - Discovery actions
  - Orchestration & coordination of DSLs, mappings between DSLs and actions
  - Guidance for users of the factory, that integrate with the orchestration & coordination of actions & DSLs
- We (Microsoft) are working on this now