Preparing for Change

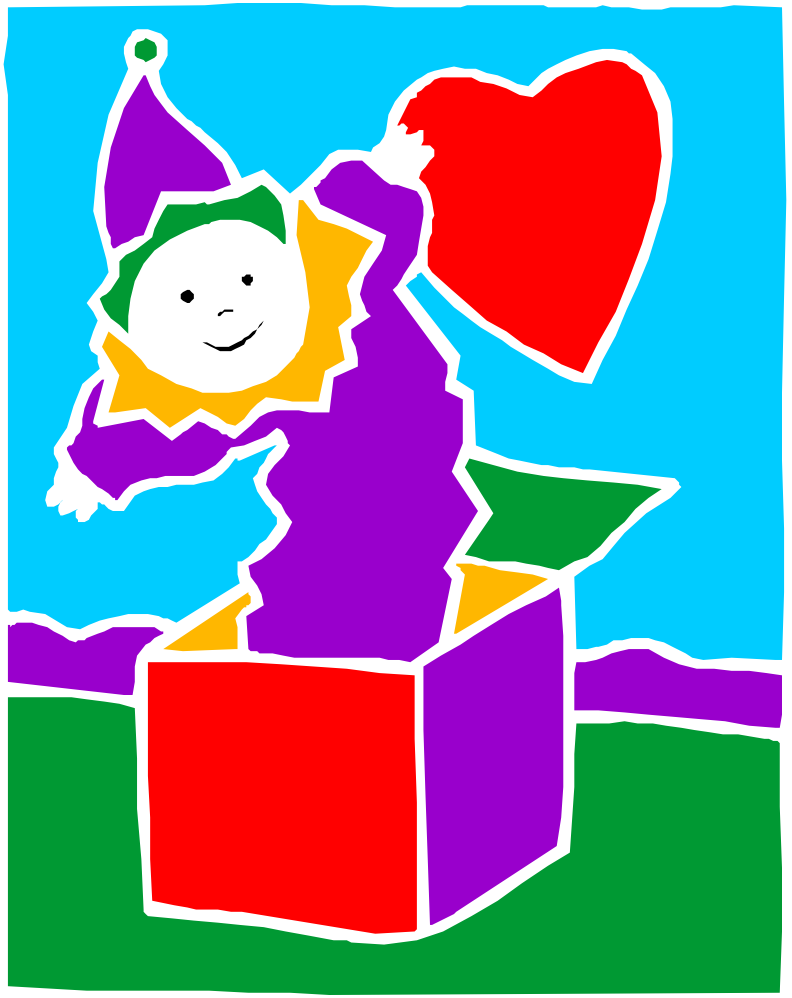
Stephen J. Mellor
Vice-President
Project Technology, Inc.
http://www.projtech.com

PROJECT TECHNOLOGY INC.

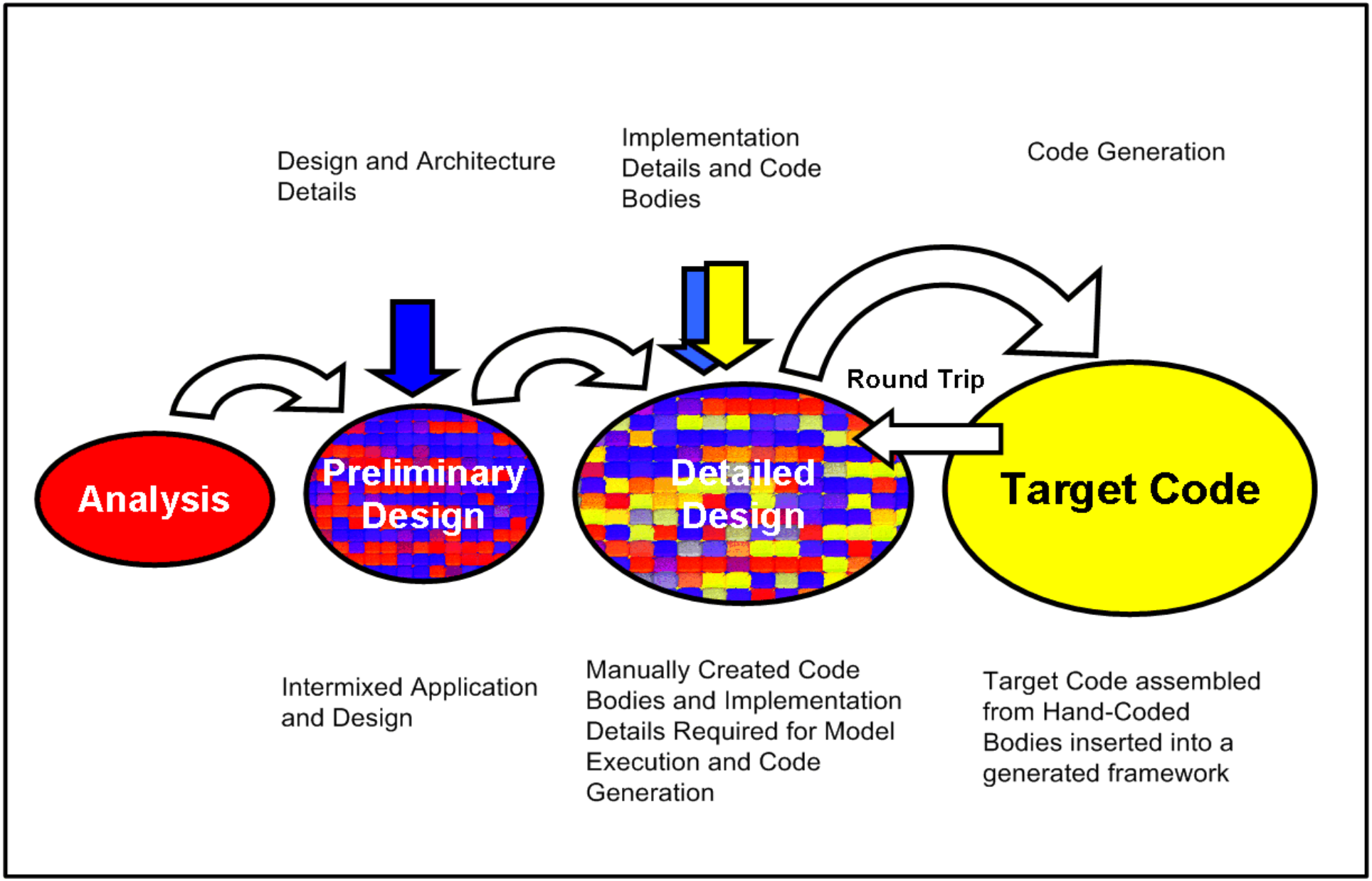


Change is a constant

- Requirements and system functionality evolve continuously.
- Technology changes more rapidly than we can keep up with it.
- Architectural mismatch is a nightmare.
- Staff changes, taking their knowledge with them.
- Deployed systems become the basis of new releases and new systems.



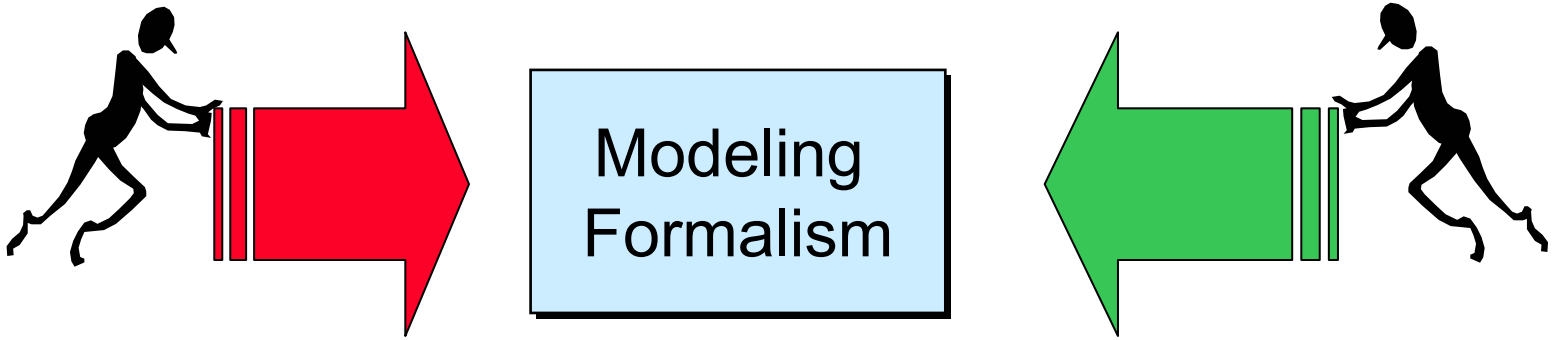
Elaborative development



Two forces

Modeling formalism should be close to the knowledge we're capturing.

Modeling formalism should be close to the implementation.



But this makes a hidden assumption….

…that a *single* model should capture both:

- Knowledge, and
- Software structure

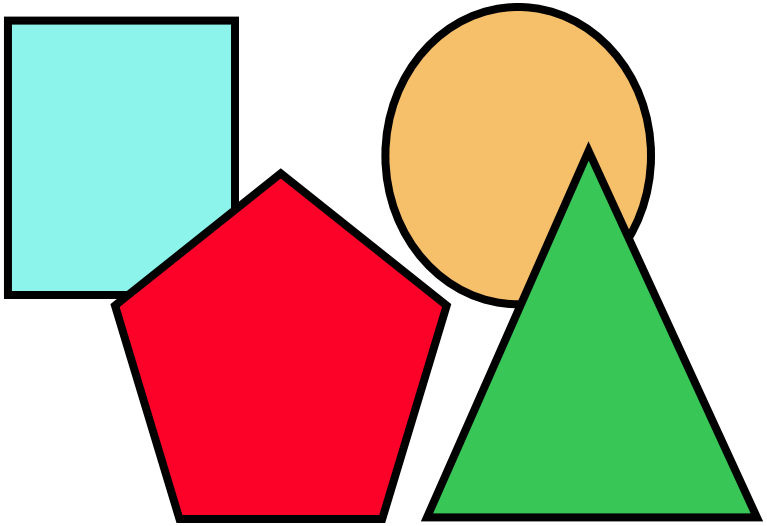
using the *same* modeling formalism.

Two forces

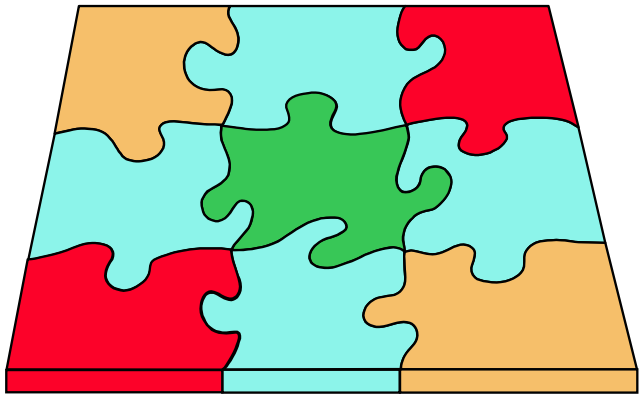
A single model cannot easily be both:

- An accurate and accessible model of knowledge, and
- An efficient and effective expression of the implementation

The structure of the formal knowledge is not necessarily the same as the structure of the software implementation.



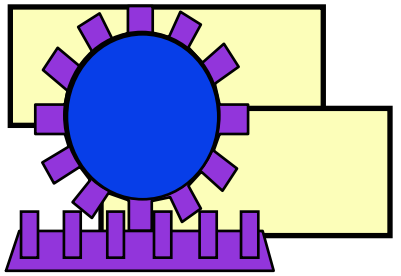
The knowledge

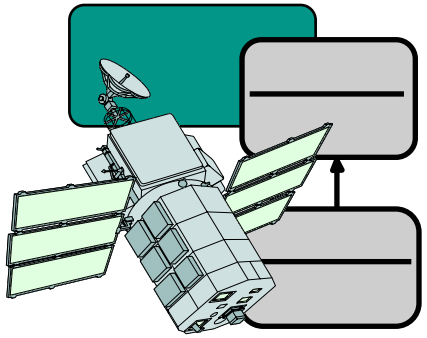


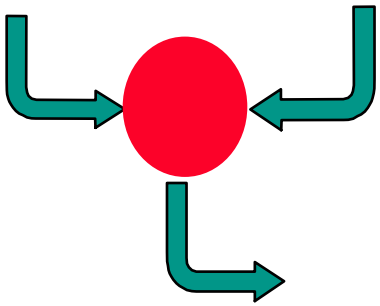
The software

Change the assumptions…

- Separate application from system architecture design.
  - An architecture-independent application, and
  - An application-independent system architecture
- Use the same *executable* modeling language (UML) for all subject matters.
  - The executable model should not imply an implementation.
- Compose the models automatically.
  - This process is a *translation* or a *weaving*.

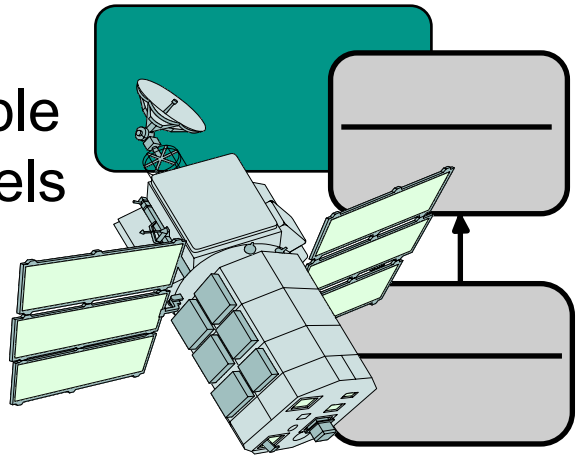






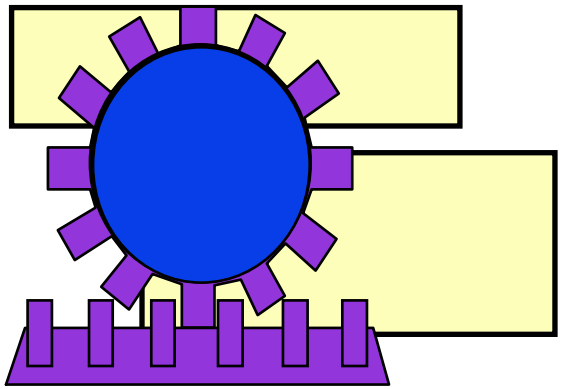
Change the assumptions….

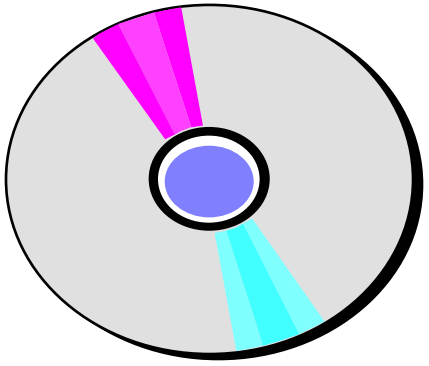
Executable and Translatable UML models



Translation according to rules

Separation of application from architecture



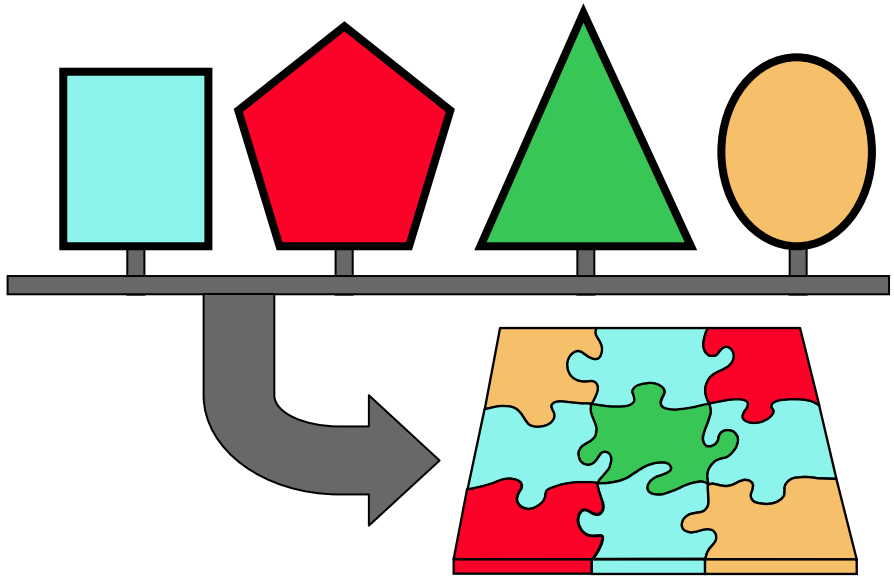


Model compilers

A model compiler compiles each model according to a single set of rules so that the various subject matters *are known to fit together*.

System dimensions include:

- Concurrency and sequentialization
- Multi-processing & multi-tasking
- Persistence
- Data organization choices

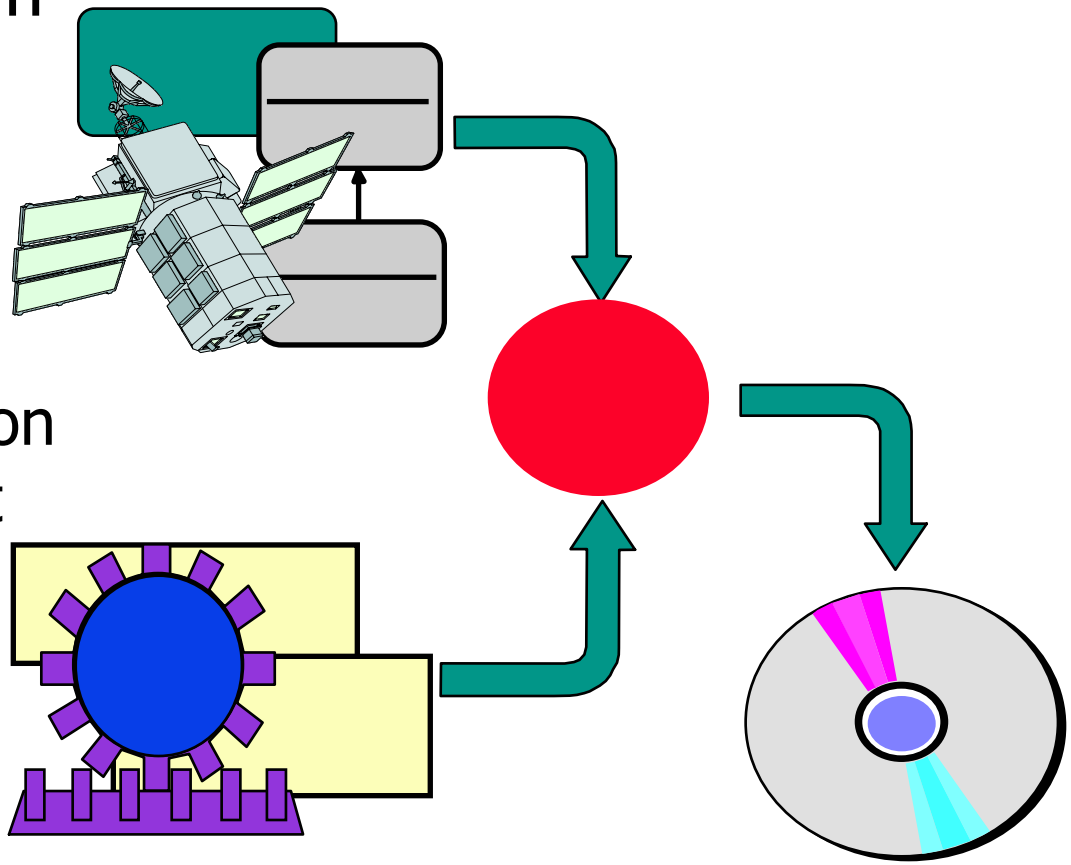


A model compiler realizes <u>design-time composability</u>.

Requirements and functionality evolve…

Separating design from application analysis:

- facilitates application iteration without touching the design
- reduces the verification gap between concept and code.



independent evolution of analysis and design

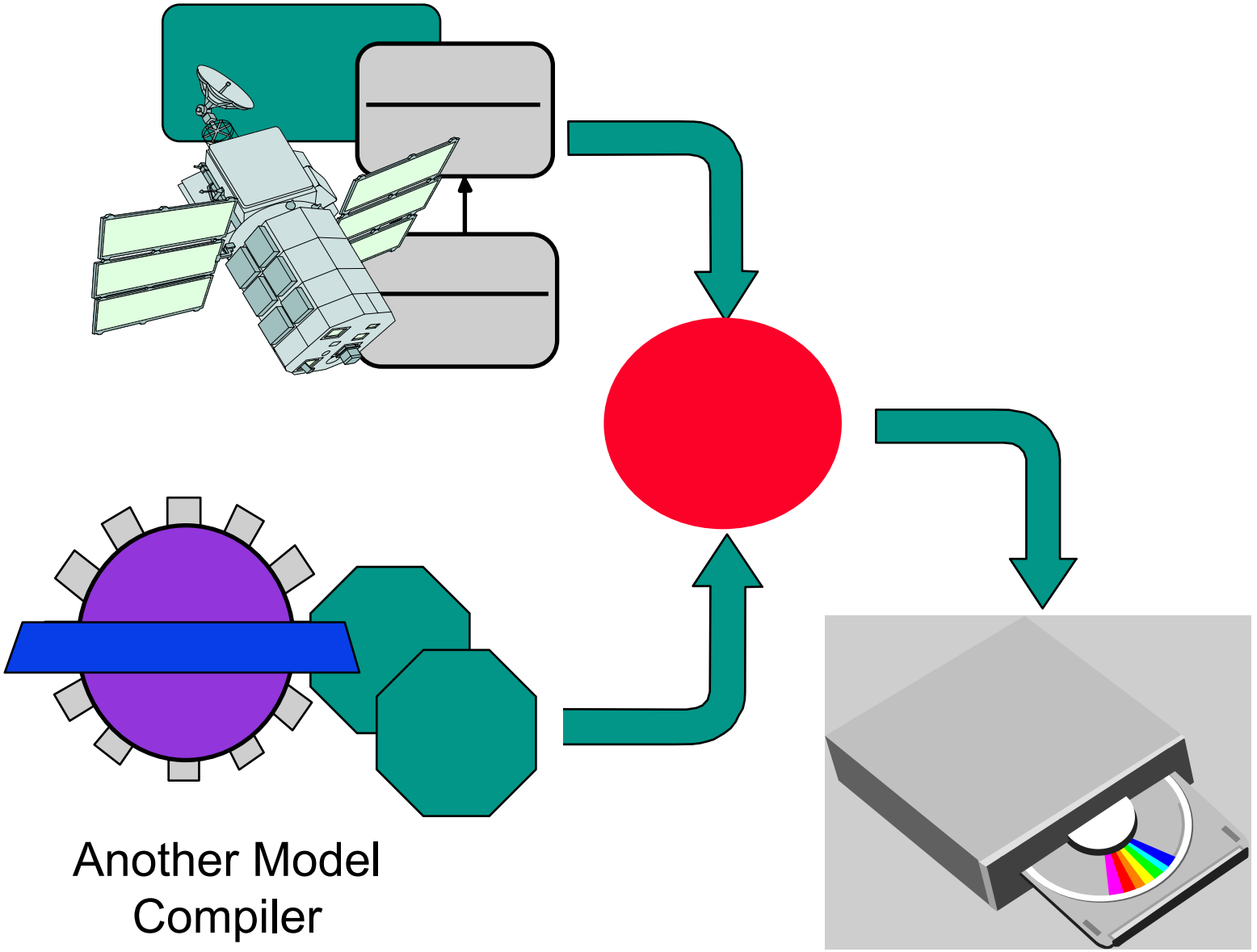
Technology changes rapidly…

Separating application analysis from design facilitates :

- design iteration without touching the application
- migration to new target platforms and technologies.

Performance optimization and design tuning re-implements the application without changing the application

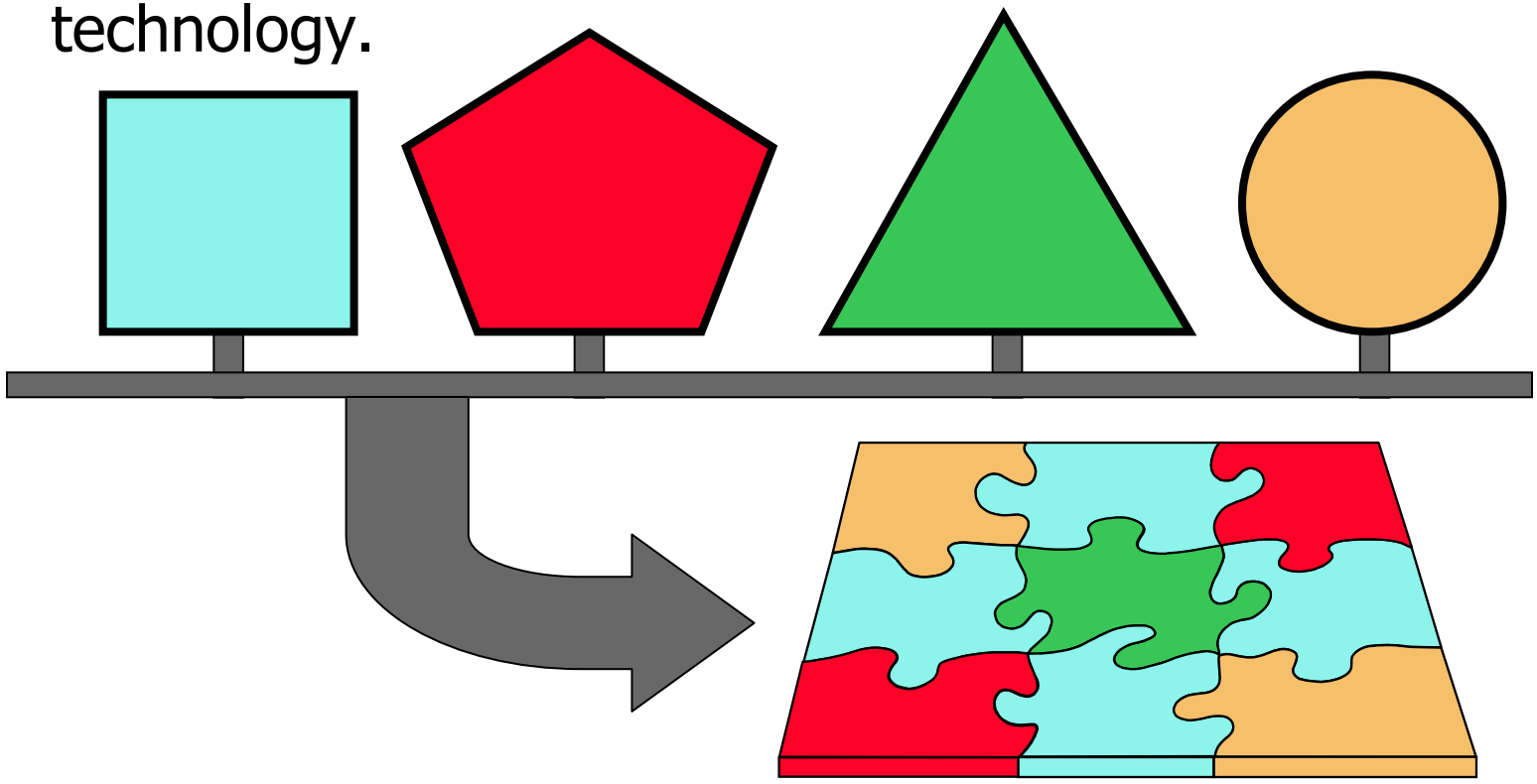
Technology changes rapidly…



Architectural mismatch is a nightmare

Weaving system functionality and software architecture design together late in the process:

- allows components to be defined without making premature decisions about specific implementation technology.

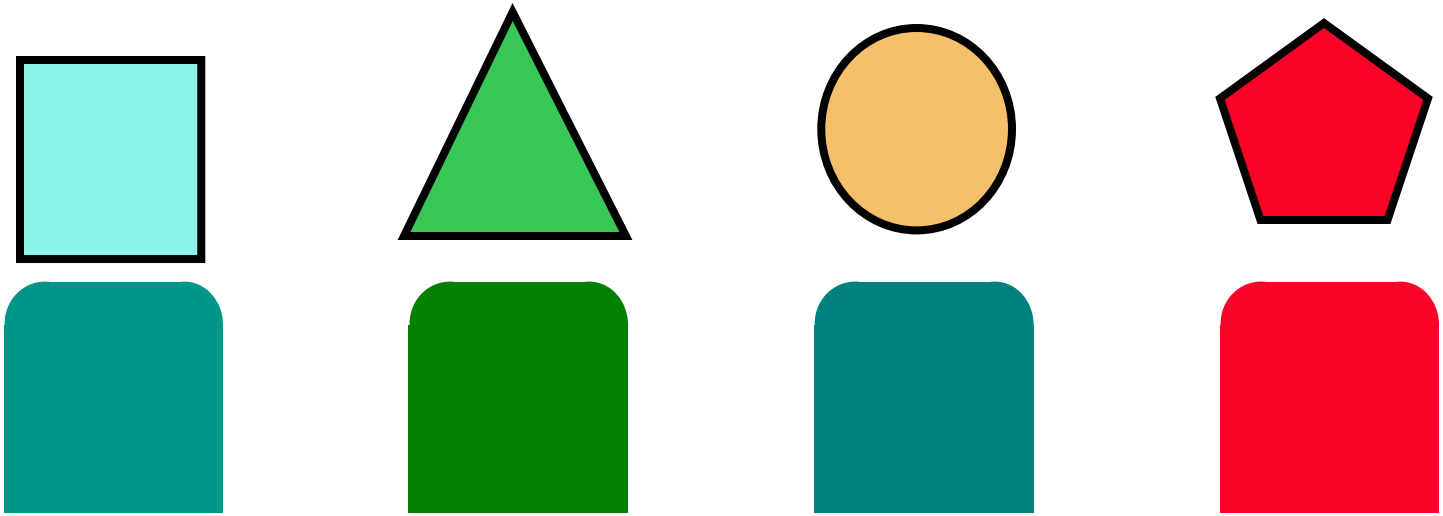


Staffing changes

Separating subject matters:

- promotes formalization of knowledge
  - that can then be transferred
- eases transfer of knowledge
  - at a higher level of abstraction (models, not code)

Subject matter separation is a more general form of separation of analysis and design

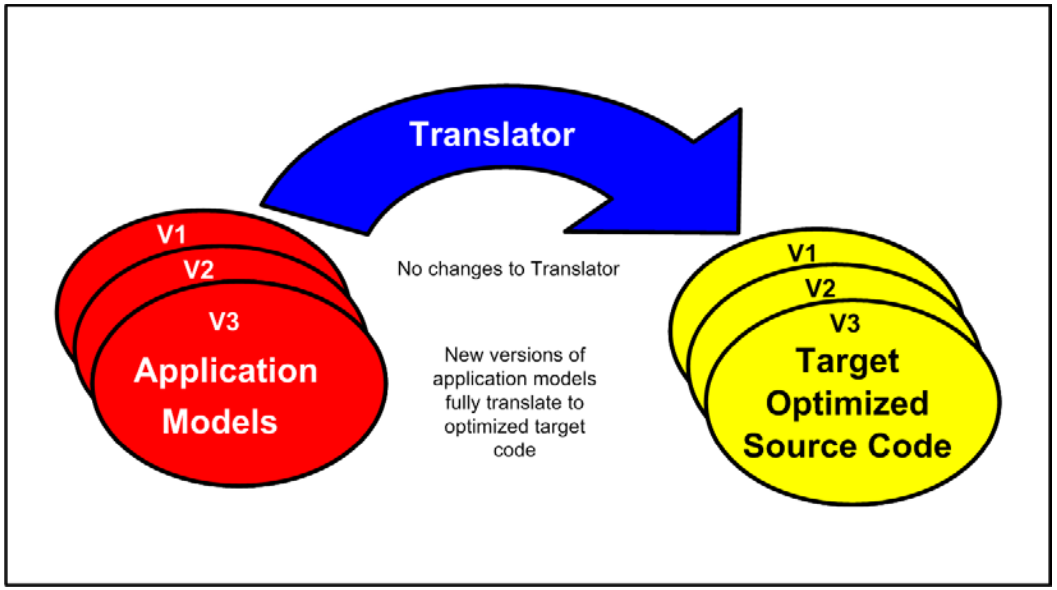


Deployed systems become new systems

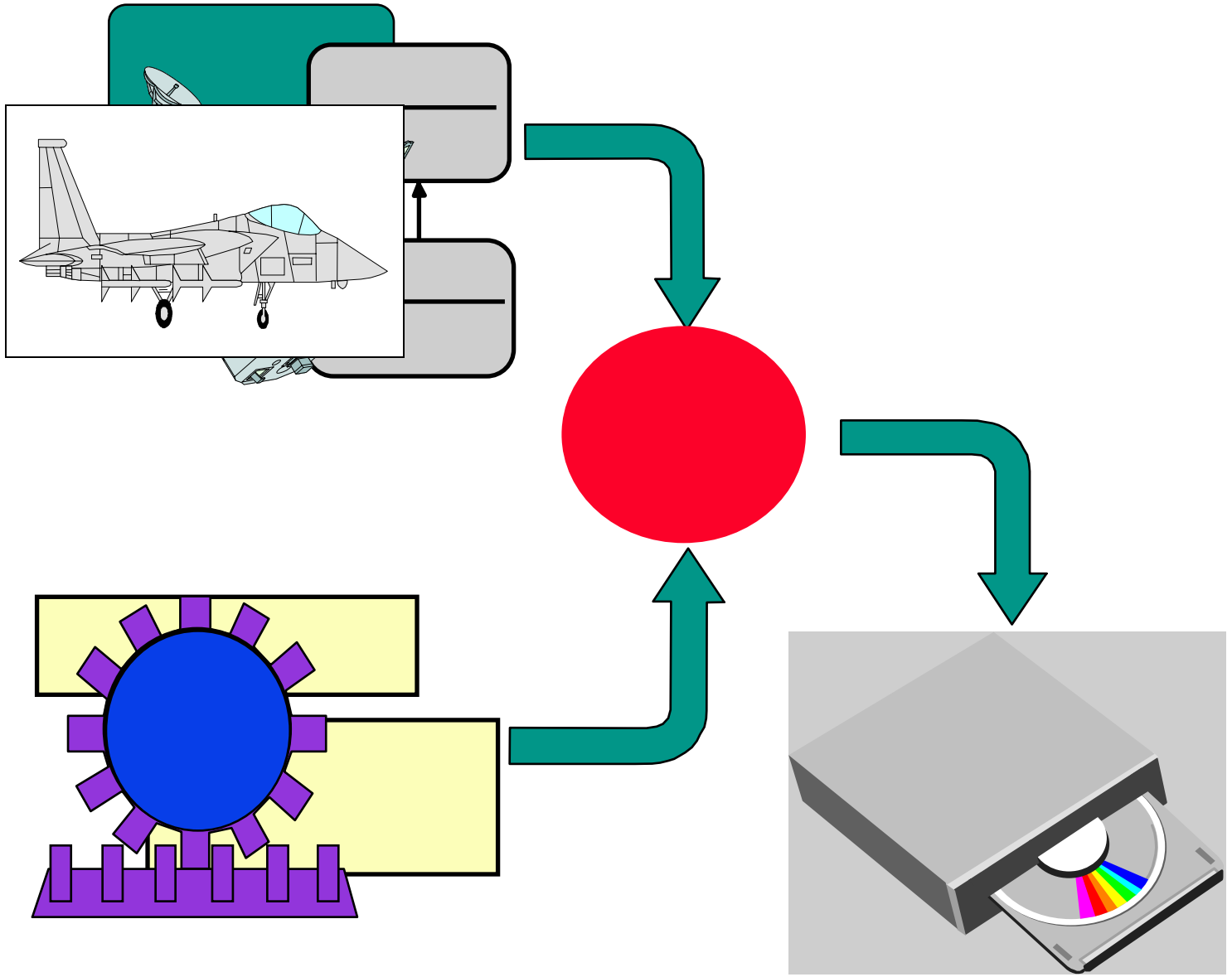
Intertwingling implementation technology and application functionality inhibits reuse.

A component that does not depend on implementation details:

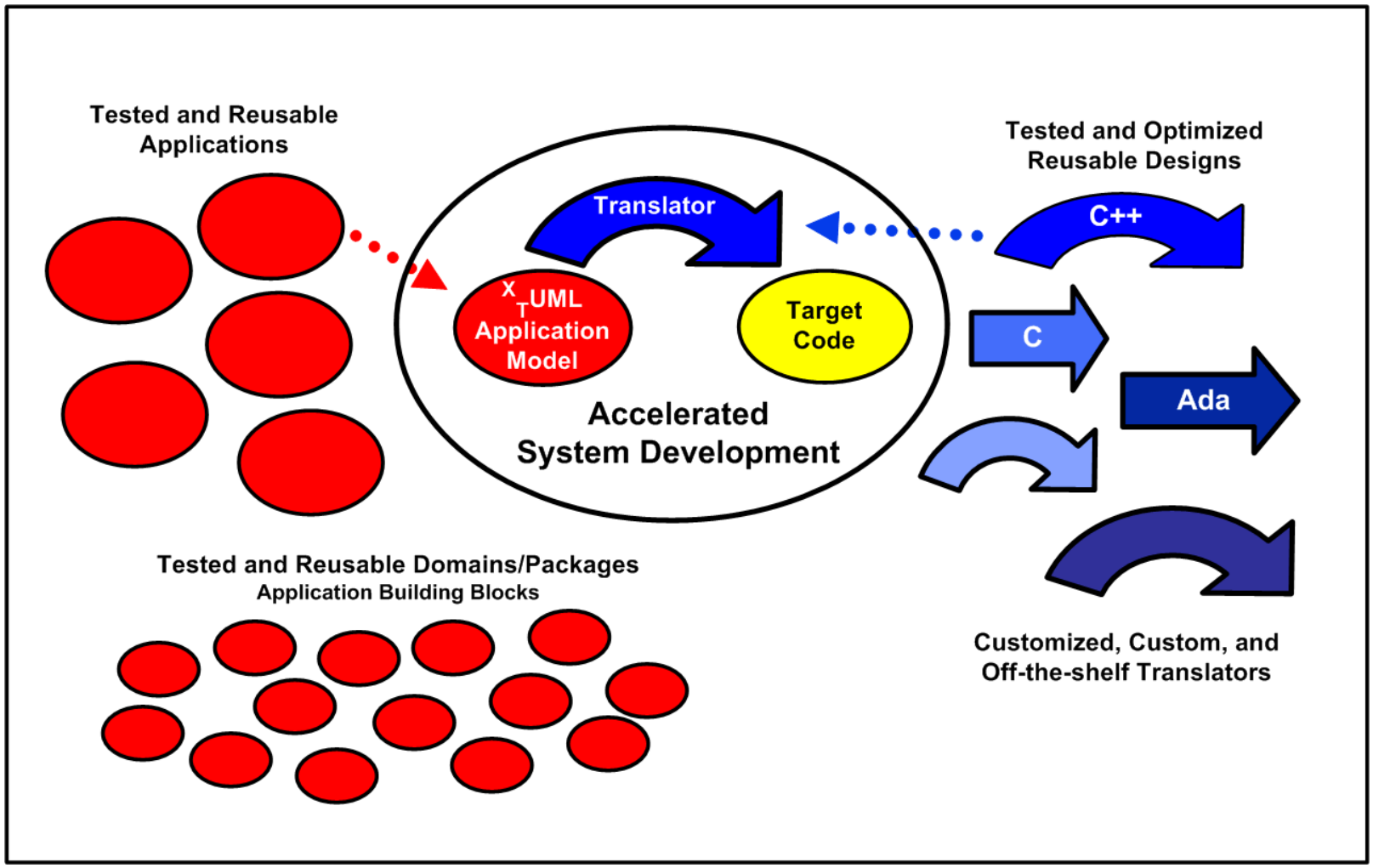
- Maximizes reuse across product configurations and families



Large scale reuse of model compilers…



Large-scale application component reuse



Prepared for change?

- Independent evolution of application and design
- Easier migration to new target platforms and technologies
- Large-scale application component reuse
- Formalization of application knowledge
- Easier transfer of knowledge to new staff
- Staff specialization
- Facilitated iteration of the application without redesign of the system architecture
- Reduction of the verification gap
- A mechanism to deal with architectural mismatch

Questions?

For more information, see:

www.projtech.com

(I shall also be available at Project Technology's Booth #439 through Thursday.)

PROJECT TECHNOLOGY INC.