



Market Readiness for Language Engineering

KISS Workshop, 14 April 2009
ASWEC, Gold Coast, Australia

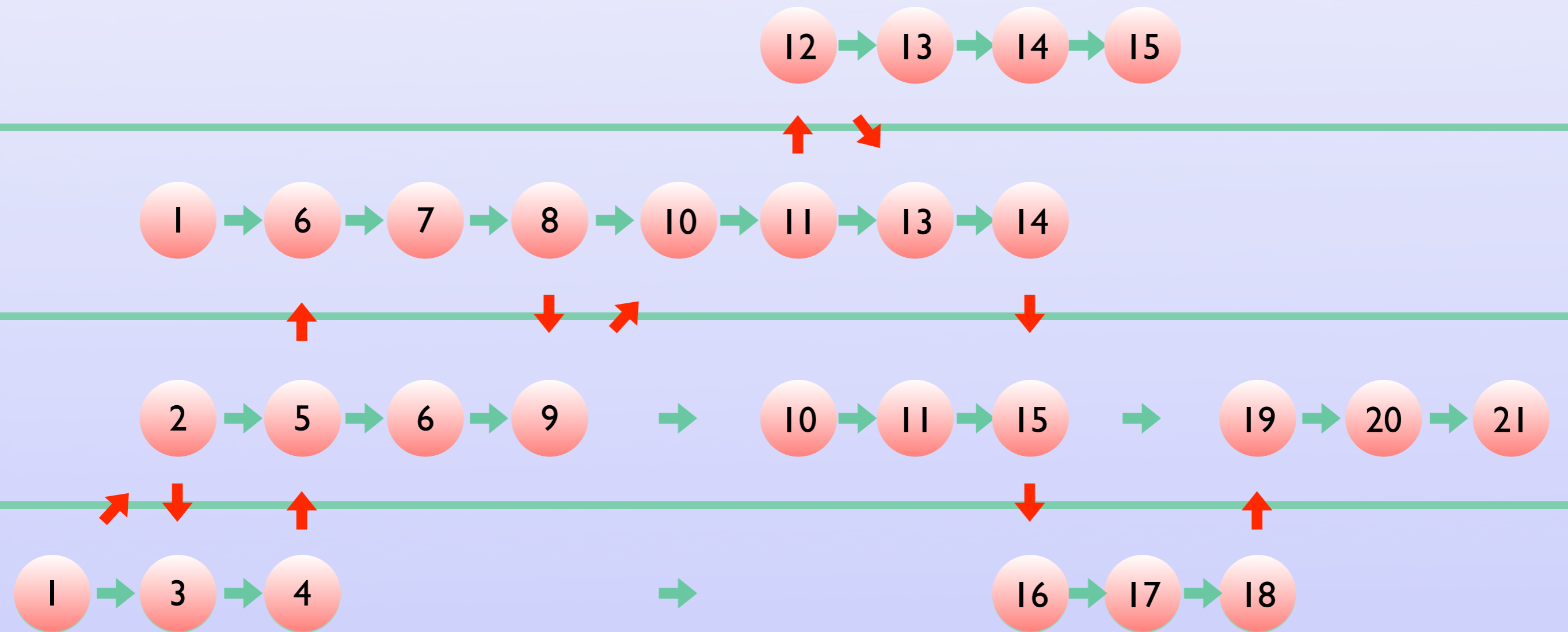
Demand for software product line engineering fuelled by

- organizations that produce highly-configurable software intensive products
- high total cost of ownership of legacy software across the customer base
- slow time-to-market when maintaining traditional code
- configuration files are everywhere, and quality and maintainability is low
- traditional requirements analysis techniques and traditional code are failing ...

Weaknesses of workflow-centric requirements analysis

- Which areas of knowledge are required to run a specific business?
- Can all knowledge be documented in the form of a process?
- Is the role of software limited to supporting/enforcing a process?
- Where are the use cases that elaborate configurability?
- Is there a clear distinction between using and producing applications?
- Are humans really working by following sequential processes?
- What is a suitable notation for describing work products (artefacts)?

The process-driven organization



workflow steps = the ritual that works, more or less ...

Process outputs

B A L L

B L U E F A S T

H A L O C A R I O N

T E V A C T

artefact components

Local semantics of process output

B A L L

B L U E

F A S T

H A L O

C A R

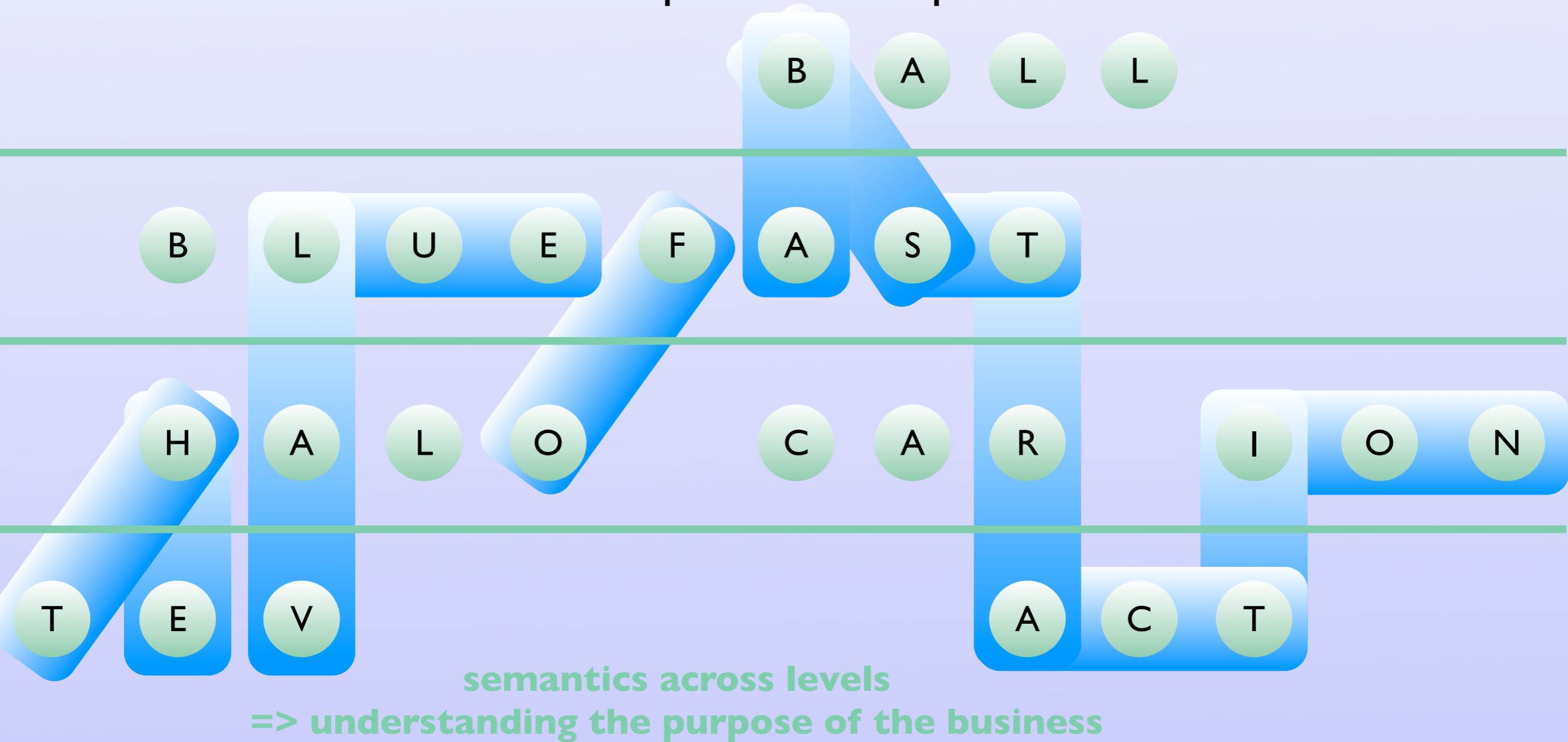
I O N

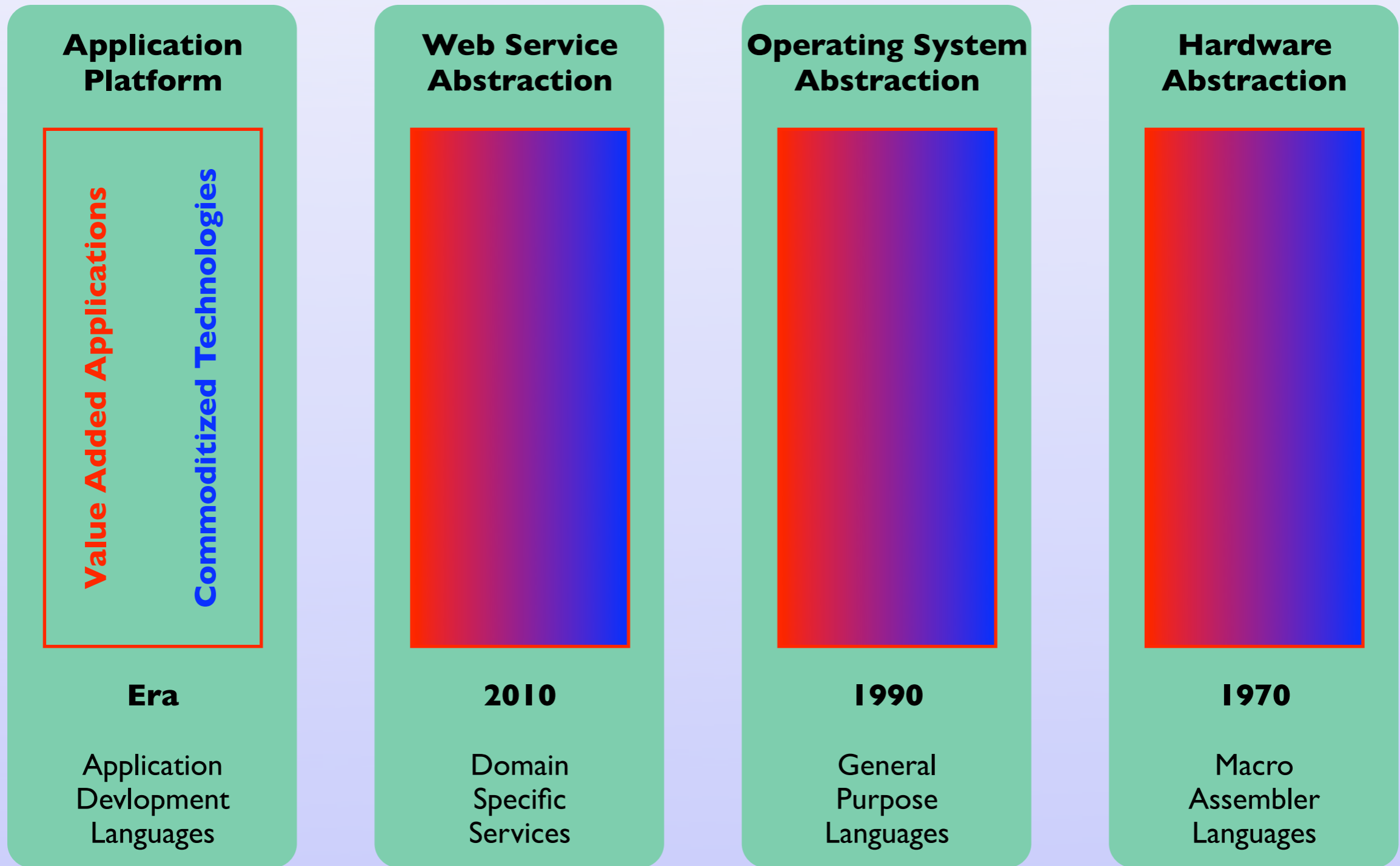
T E V

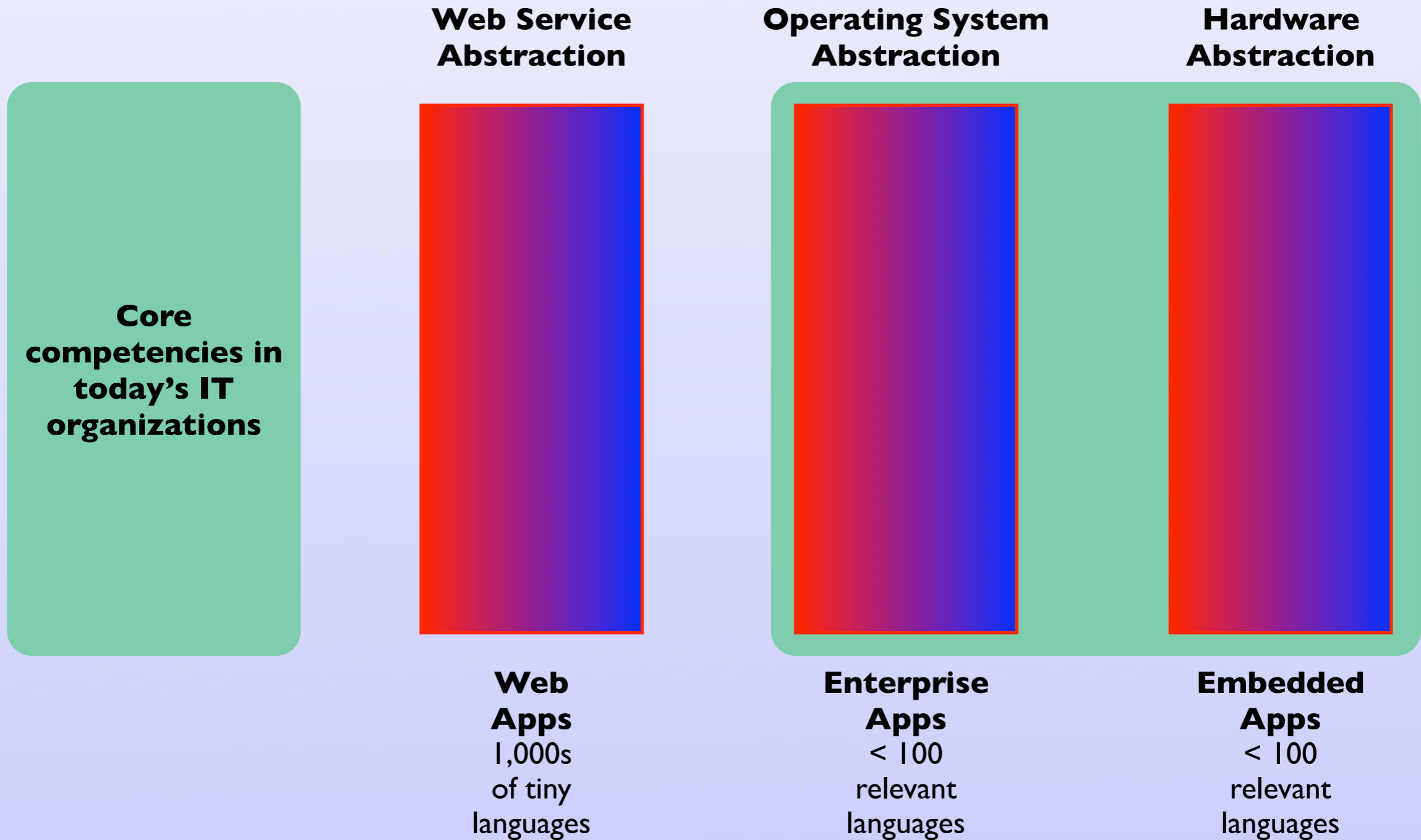
A C T

artefacts at different levels

Global semantics of process output







Web Service Abstraction



**Web
Apps**
1,000s
of relevant
frameworks

Operating System Abstraction



**Enterprise
Apps**
< 10
relevant
operating systems

Hardware Abstraction



**Embedded
Apps**
< 50
relevant
processor families

Paradigm shift:

One person's application
is the next person's platform!

Tacit knowledge is converted into explicit knowledge

- Tailored specification languages directly tap into domain specific terminology
- Domain concepts become first class language elements
- Subject matter experts immediately feel at home
- No need for manual translation between domain & implementation technologies

Separation of concerns in software

- Is poor when using general purpose languages
- A change in requirements leads to code changes in many places
- Hence the need for intelligent diff/merge tools
- Objects are too small for reuse
- Frameworks break the clean separation of the problem space from the solution space
- The larger the system, the more problematic
- It gets worse over time

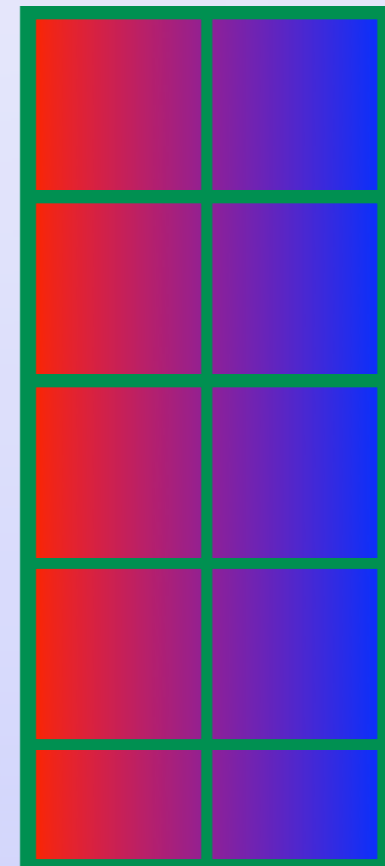
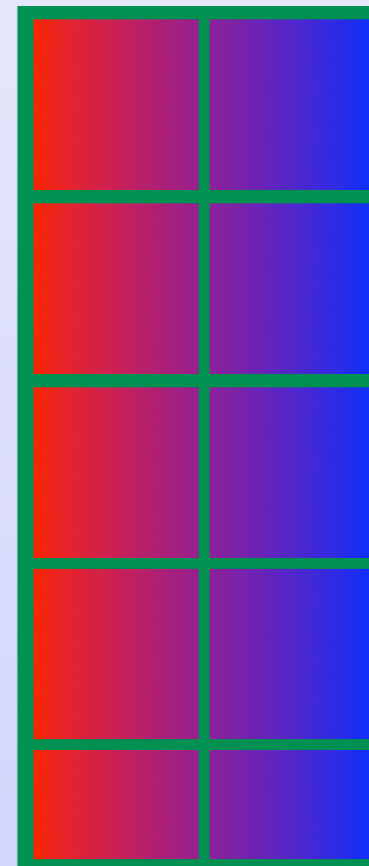
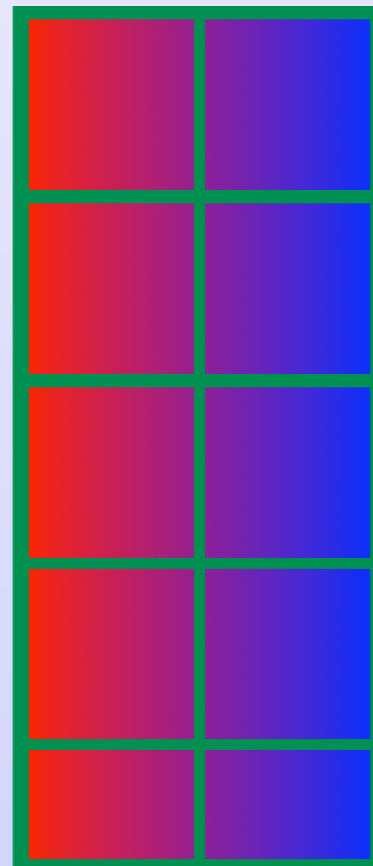
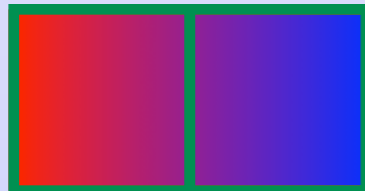
Web Service Abstraction

Operating System Abstraction

Hardware Abstraction

Qualities that are only scalable when using models

I. Modularity



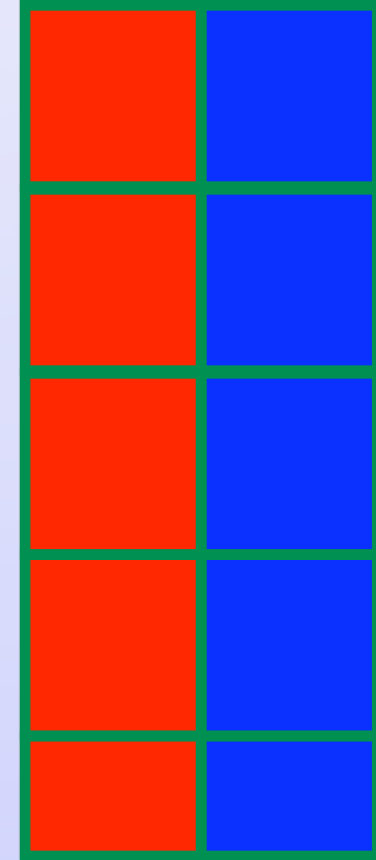
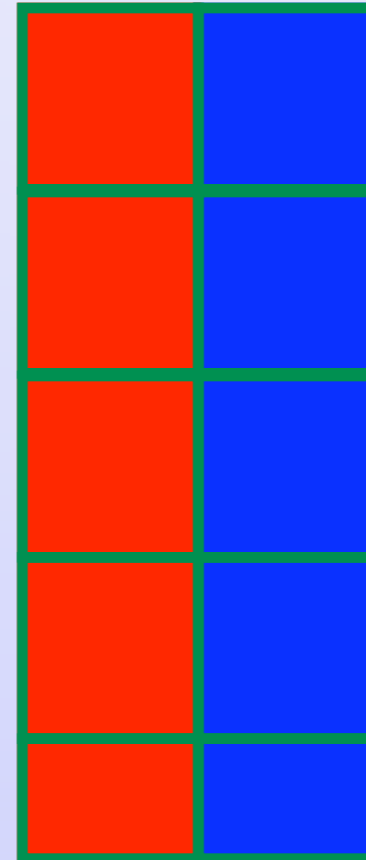
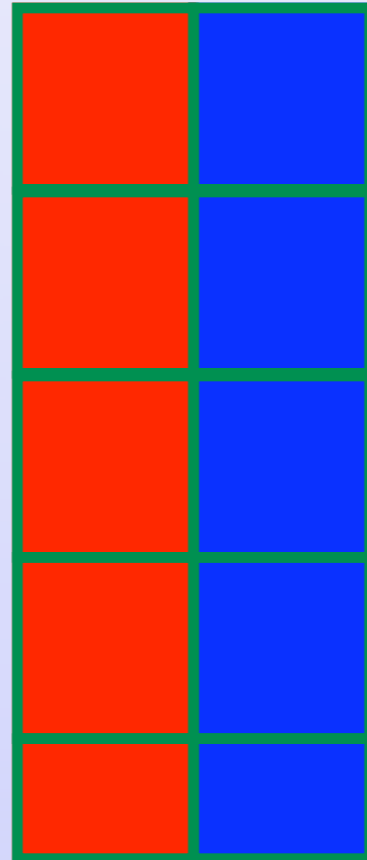
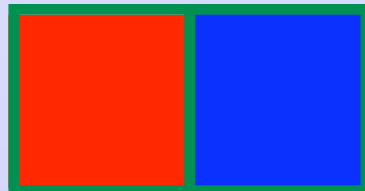
Web Service Abstraction

Operating System Abstraction

Hardware Abstraction

Qualities that are only scalable when using models

2. Separation of Apps & Platform



Open Source

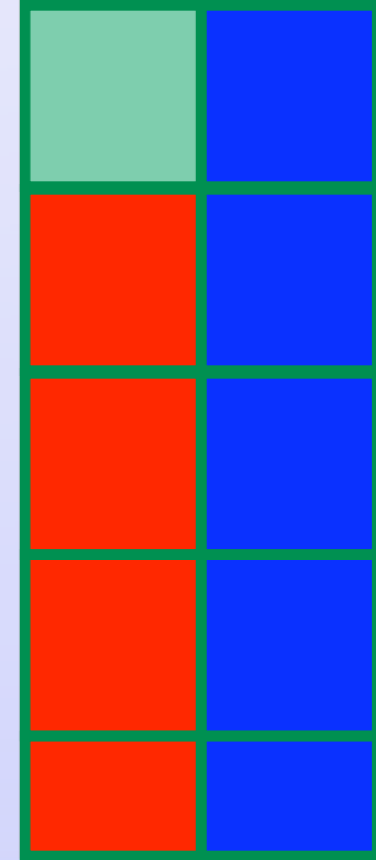
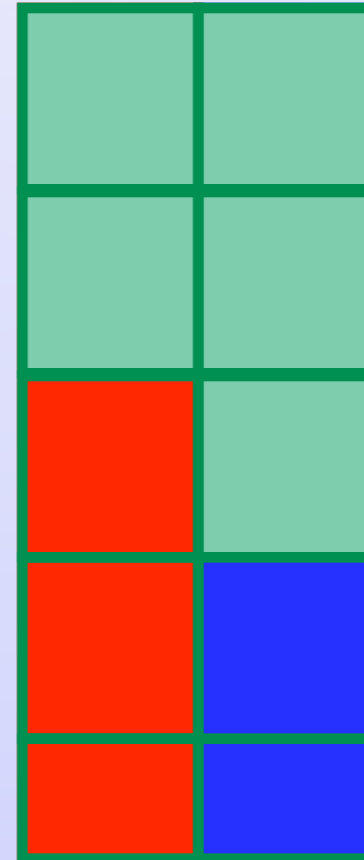
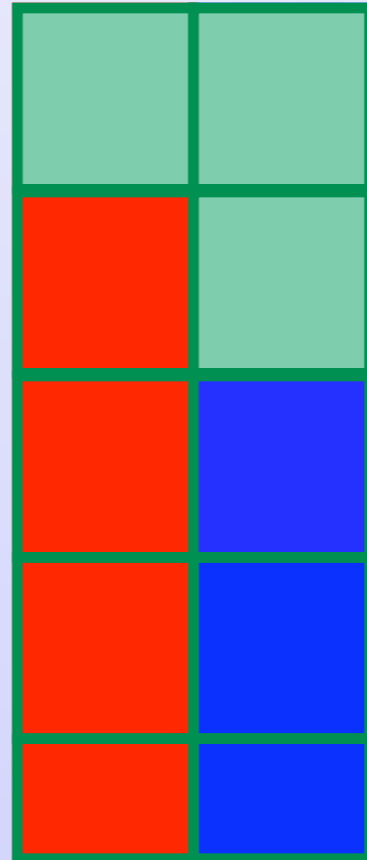
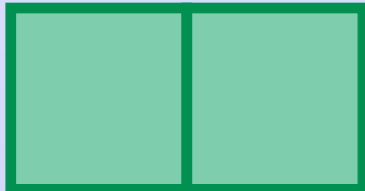
Web Service Abstraction

Operating System Abstraction

Hardware Abstraction

Qualities that are only scalable when using models

3. Strategic use of Open Source



The challenges

- Those who have been disillusioned by UML
- Those who are not yet disillusioned by UML
- Those who have discovered the power of model driven generation in the last five years
 - ★ focus on amount of code generated, not quality
 - ★ low quality input specifications (models, config files, ...)
 - ★ ignorance of best practices for modelling language designs
 - ★ some of the same people sold objects as a silver-bullet in the 90s ...
- Love affair of software professionals with complexity & technologies
- Underestimating the value of true domain expertise



Thank you!

Jorn Bettin
jbe @ sofismo.ch
Skype jorn_bettin
+41 62 891 0987