PRISMTECH
Productivity Tools & Middleware

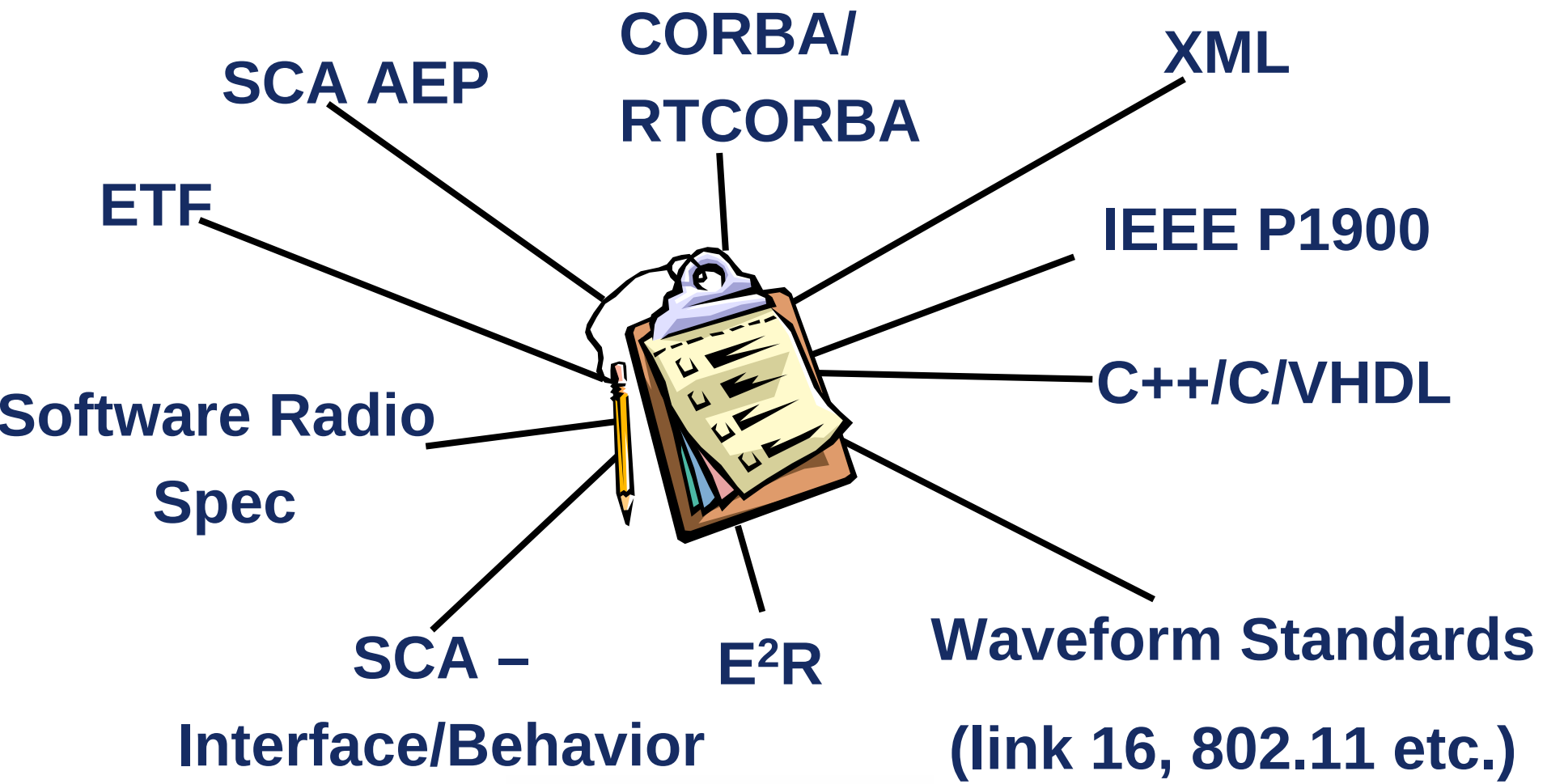
The Future of Software Radio MDD Tools

Dom Paniscotti
Bruce Trask

Current Challenges facing SDR Development

- Heterogeneous Processing Elements (GPP/DSP/FPGA)
- Achieving Portability
- Systematic Reuse - What is the correct granularity of reuse and standardization?
- Porting Legacy Systems
- Size Weight and Power
- Quality of Service/Real-Time
- Security Concerns
- Standardized Deployment, Configuration and Execution
- Long Lifetimes
- Repository Management/Maintenance & Versioning
- Lack of Productivity Tools

And Let's Not Forget Standardization



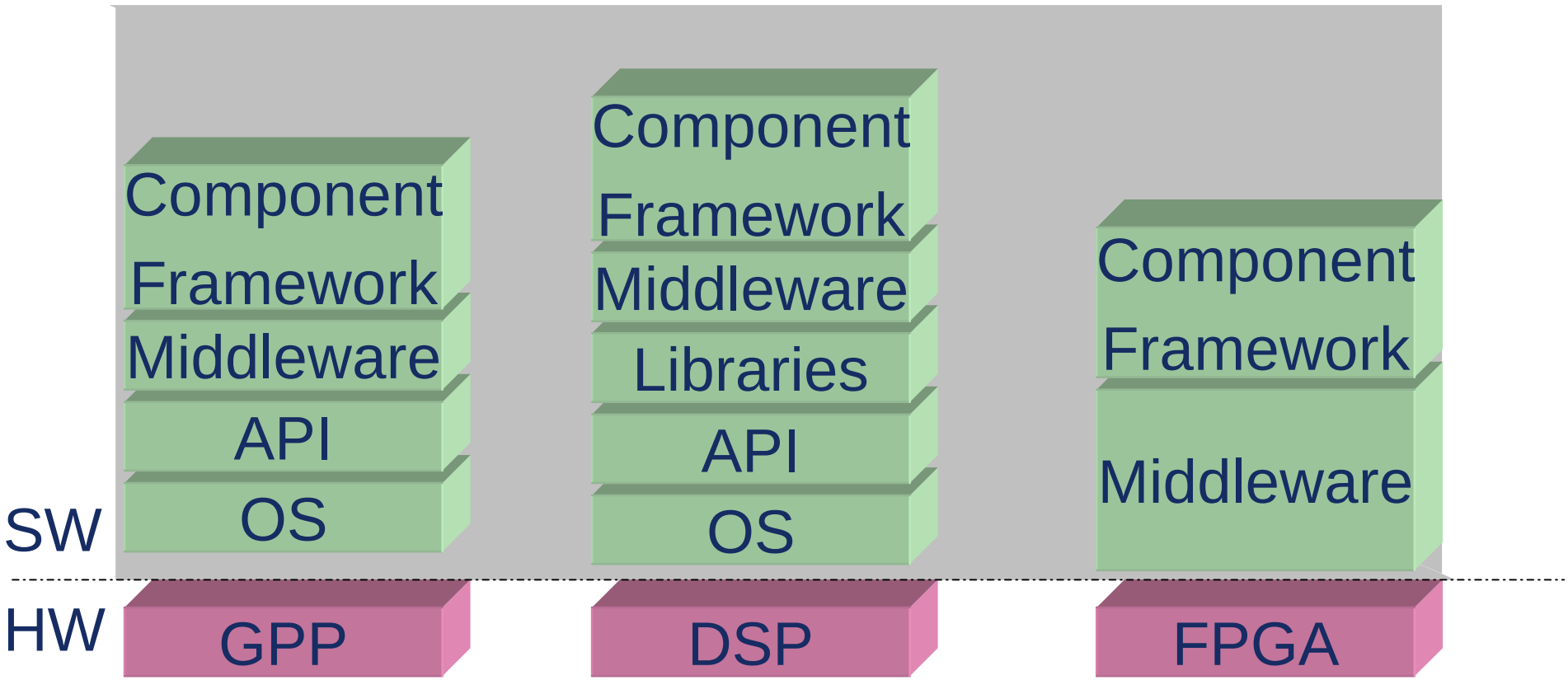
What's Needed… An Integrated View

- Once the core technologies are available to support the architecture…
- You've got to make it all work together seamlessly

"The lack of an integrated view—coupled with the danger of unforeseen side effects—often forces developers to implement suboptimal solutions that unnecessarily duplicate code, violate key architectural principles, and complicate system evolution and quality assurance."

-Doug Schmidt

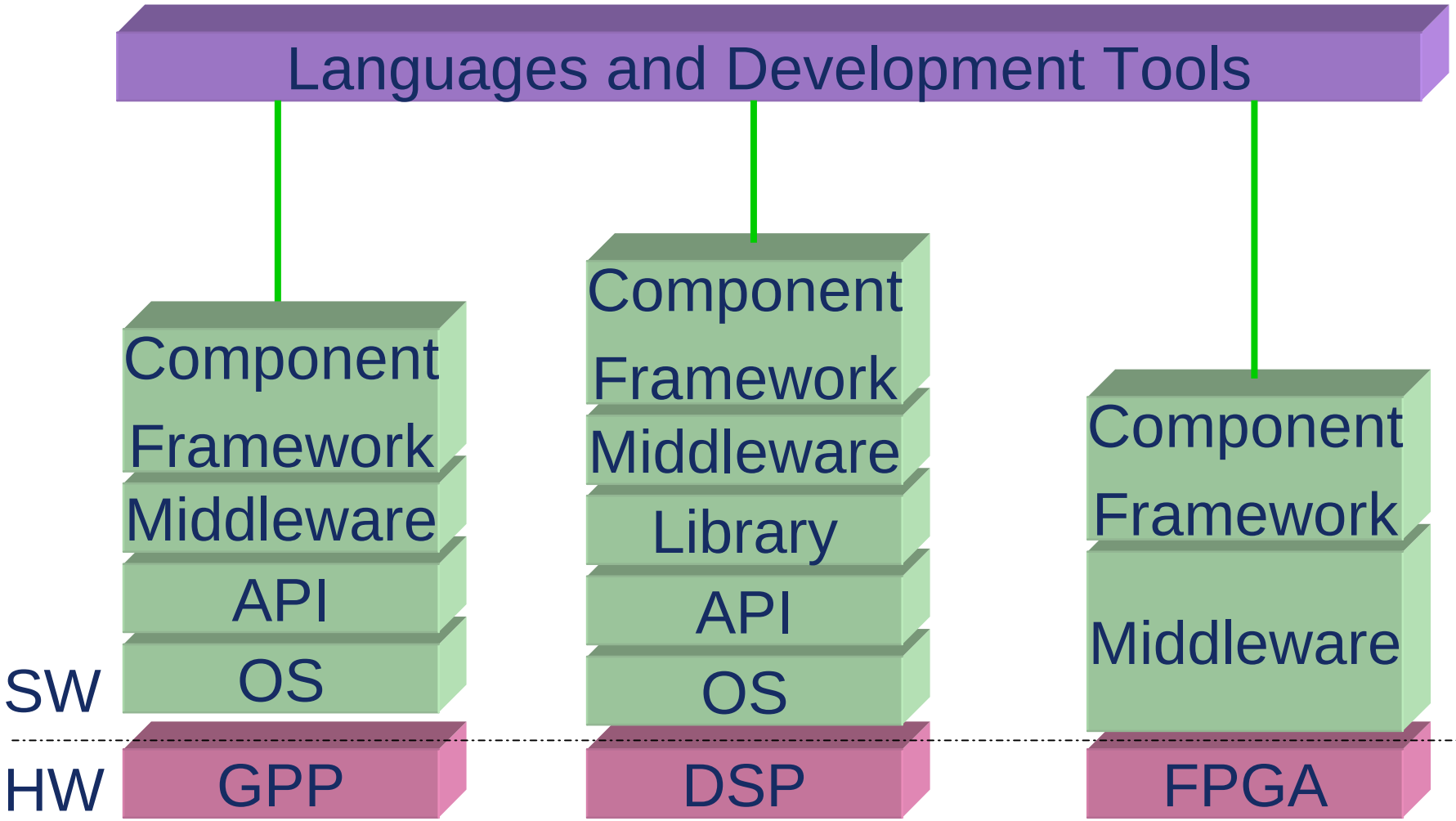
An Integrated View… Closing The Abstraction Gap



A way forward…

- Two things are accomplished by supporting a consistent architecture on all processing elements
  - All the benefits of the SBC architecture and vision (reuse, portability etc.) are enabled
  - The essential enabling technologies for development using higher level abstractions (MDA) are delivered
- Allowing Model Driven Tools and Languages to deliver significant increases in productivity and quality

Having Raised The Platform Abstractions Sufficiently… Languages and Tools Are Brought To Bear



Development Tools

- Deal effectively with mundane development tasks and rote/repetitive source code development
  - Allowing developers to tackle the truly difficult application issues
- Define Standardizations Points
  - Languages, Design Patterns
- Eliminate Ambiguity
  - By Providing Languages That Naturally Map to the Problem Space
- Support Information Interchange
  - Shorter ramp-up times
  - Facilitates Knowledge Transfer During Staff Transition

Anatomy of Model Driven Design

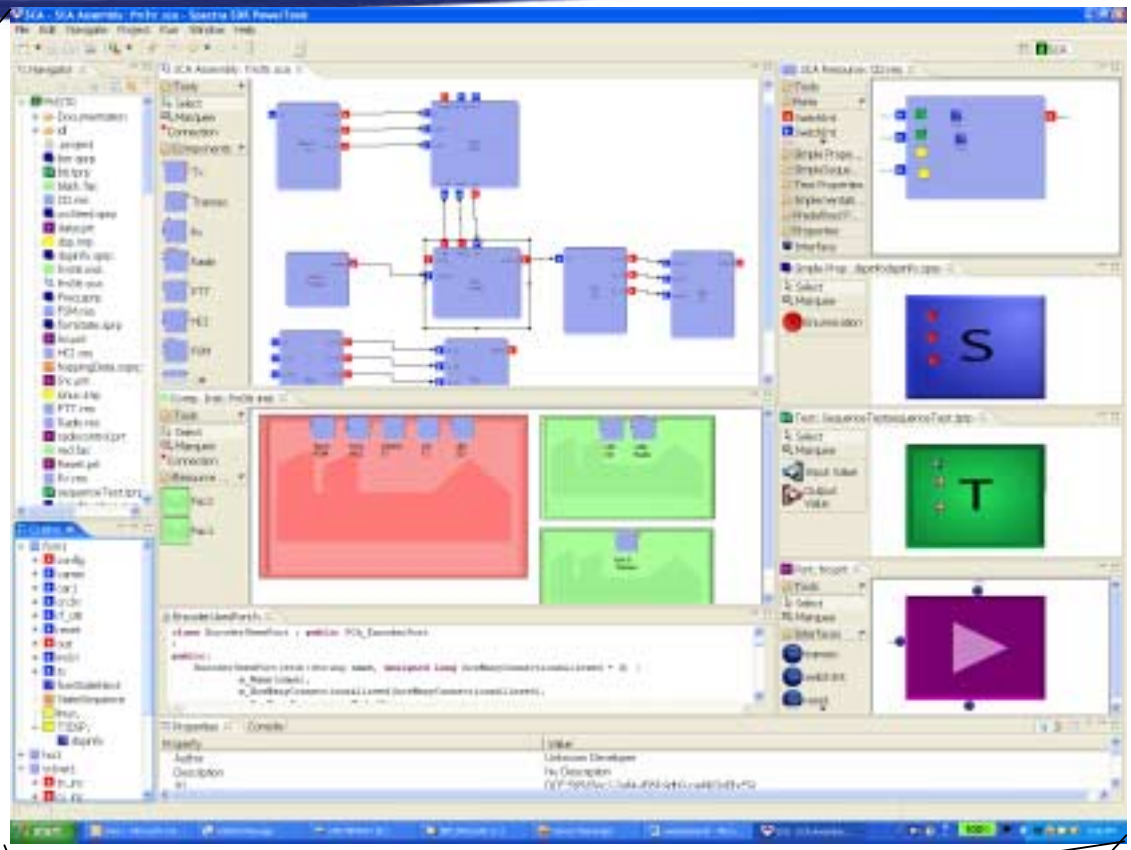
- Language - Editor – Generator (LEG)
  - ***Language*** - Correctness/Completeness
    - A language in the fullest sense of word
    - Supporting Debugging, Testing, Optimization, etc.
  - ***Editor*** - Expressing Design Intent
    - Easing Manipulation of the Language by Developers
  - ***Generator*** – Supporting the architecture and tapping into the execution environment
    - Providing consistency by directly transforming the language into the required source code and executable artifacts.

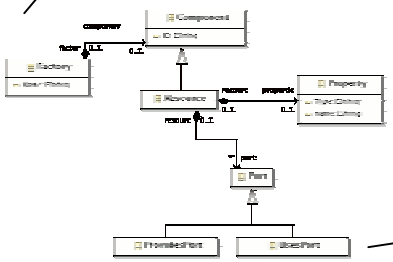
Applying MDD to the SDR Domain

- Languages
  - Component and Assembly Definition
  - Deployment
  - Quality of Service
  - Security
- Graphical Editors and Viewer that map to Language concepts
- Generators
  - 3GLs (C, C++, VHDL, etc) - Designed with portability, re-use and architectural consistency in mind
  - Tests – functionality and standards compliance
  - Infrastructure Descriptors
  - Supporting Different Aspects of a System
    - Exception/Non Exception, Logging/Non Logging, Debug/Non Debug, Tracing/Non Tracing, Thread Models, etc.
    - Optimizations for size, speed, power

A Graphical Domain-Specific Language

Images, layout, organization based on meta-model

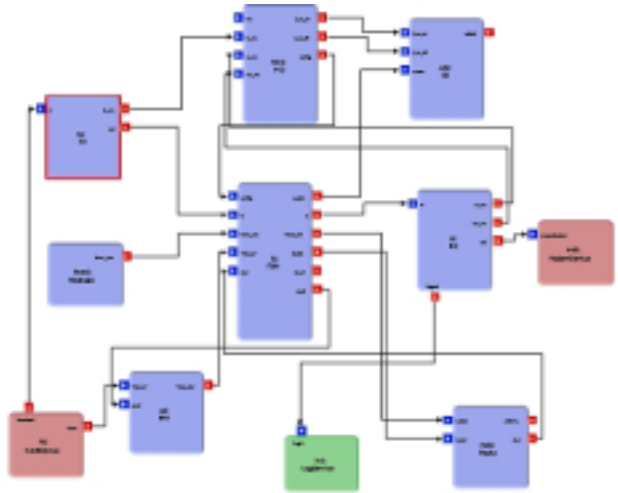




```
  <components Name="BitFlipper" organization="PrismTech"
id="DCE:8f647411-91a1-4295-bbc6-6d3eff4982f7">
<ports xsi:type="com.prismtech.spectra.sdr.sca2_2.models:UsesPort"
    instanceName="TX" name="Data"/>
<ports xsi:type="com.prismtech.spectra.sdr.sca2_2.models:ProvidesPort"
instanceName="RX" name="Data"/>
</components>
</com.prismtech.spectra.sdr.sca2_2.models:Assembly>
```

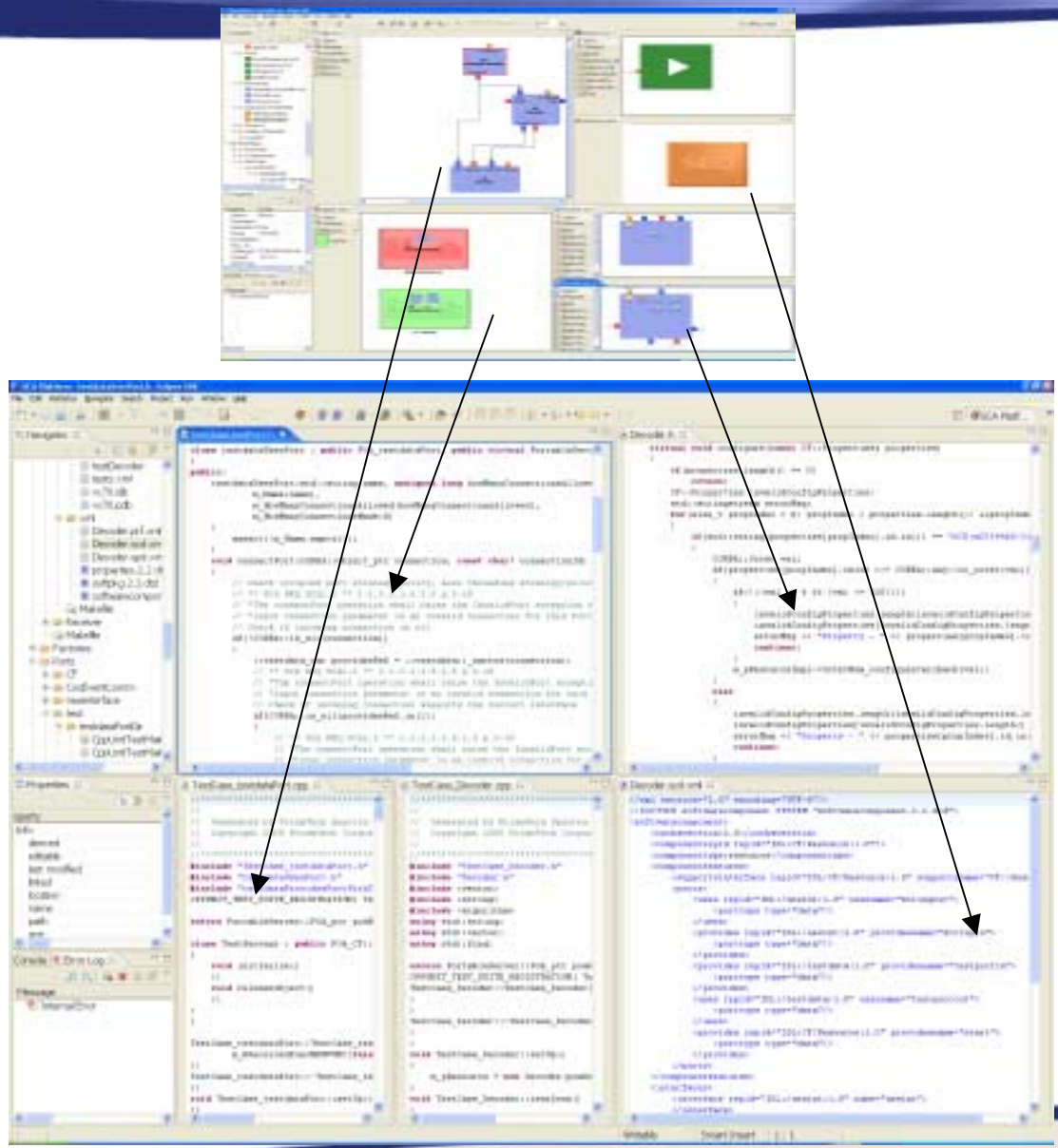
In Practice…

- Architectures/Frameworks tend to create a demand for the development of rote, duplicate, domain independent source code to support them
  - SDR Architectures Are Not Immune
- This Code Lends Itself Very Well To The Application of Code Generation Techniques
  - Design Patterns Are Typically Applied By Experts In The Given Architecture/Framework To Increase Productivity
  - Yielding Generated Code Which Is Correct By Construction

Replaces  with 

- Enables developers to more naturally express design intent while ensuring SCA compliance **by design**
- Enables reviewers and testers to more easily understand the design
- Ensures SCA-compliant architecture by enforcing constraints (i.e. correct-by-construction design process)

The Editors and The Generators



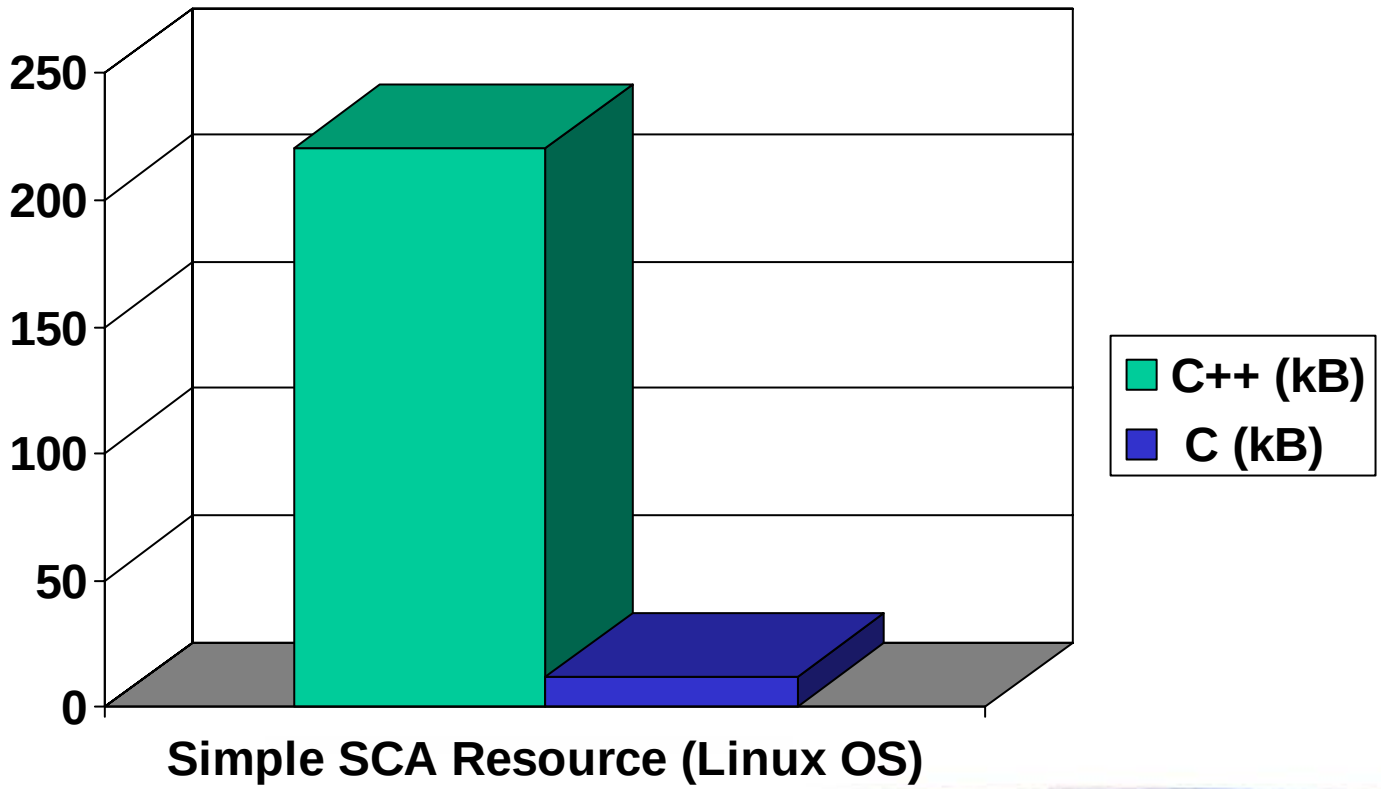
Fully functional and correct infrastructure source code, tests, descriptors, and build artifacts are automatically generated

Some of the Benefits

- The Automatic Generation Of These Artifacts Literally Isolates The Artifacts Required By The SDR Architecture From Those Needed To Deliver Radio Functionality
- In Doing So
  - Increases Portability and Reuse
  - Drives Architectural Consistency Across the Family of Systems
  - Lowers The Barriers To SDR Market Entry
  - Lowers Time To Market
  - Increases Quality
- Furthermore This Isolation Protects Investment
  - Modifications To The SDR Architecture Itself Can Be Dealt With By Simply Adjusting The Generators As Required

Some of the Benefits (continued)

- And Allows Developers to make the proper platform-specific choices
- Comparison of SCA components using C and C++ code generators



New Problems and Complexities To Deal With

- Generating Source Code, Descriptors, and Build Environments Is Simply The *Tip of the Iceberg*
- In The Near Future, Model Driven Approaches Will Be Used To:
  - Analyze Deployments
    - Allowing Middleware and Operating System Overhead To Be Isolated and Eliminated
    - Generating Source Code In Optimal Language To Support Processor Type
  - Provide Higher Level Debugging
    - Debugging Data and Control Flow At The Modeling Level
    - Completely Integrated with Operating System Debugging Tools
  - Security
    - Modeling The Security Aspects Of A System and Generating Security Policies and Signed Executables

New Problems and Complexities To Deal With

- Support Timing and Throughput Analysis
  - Static and Runtime Profiling to isolate inefficiencies in processing and data flow
- Perform Trade Off Analysis
  - RTOS, Processing Environment, Processor Type, Middleware, Transport, Deployment/Partitioning, Component Granularity
- Refactor Designs
  - Modifications automatically permeated though entire design
- Configuration Management
  - Not Just Source Code
  - Support For Model Versioning With Differencing, etc.

New Problems and Complexities To Deal With

- Simulation
  - Integrating Simulation Environments
    - For Digital Signal Processing, Networking (both wired and wireless), RF, etc.
- Requirements Management
  - Tying Together System/Architecture requirements with models and generated artifacts
  - Automatically Producing Traceability Reports
- Documentation
  - Automatically Producing High Level Design Documentation Directly From Requirements Information, Models, Source Code etc.
  - With Tailoring Support – Allowing End Users to Create Required Styles

Conclusion

- Future SDR Tools Will Deal With all These Complexities and More
  - Allowing Developers To Focus On Delivering Radio Functionality
  - Producing High Quality Products Both Faster And At Lower Cost
- If Industry…
  - Supports The Standardization Of SDR Architectures
  - Demands Architectural Consistency Across Their Product Lines
  - Doesn't Flinch When Presented With The Demands Of Resource Constrained Systems
  - Carefully Considers The Business Case For These New Standards-Based COTS Solutions

PRISMTECH
Productivity Tools & Middleware

Questions?