Moving Architectural Description from Under the Technology Lamppost

Nenad Medvidovic

University of Southern California
Los Angeles, USA
neno@usc.edu
http://sunset.usc.edu/~neno/

(joint work with Richard N. Taylor of the University of California, Irvine and Eric M. Dashofy of the Aerospace Corporation)

A Brief History of ADLs

- Software architecture emerged as a research discipline in the early 1990s
- Soon thereafter, many notations were either invented, recast, and/or argued for as architecture description languages
  - Wright, UniCon, Aesop, Acme, Rapide, Darwin, SADL, C2, Weaves, CHAM, LILEAnna, MetaH, Demeter, UML 1.x, …
  - It seemed very important to have, or at least know, one
- Each provided modeling capabilities geared at software design
  - Though not necessarily architecture!
- They saw varying degrees of adoption and use

Enter the “Funny” Questions

- Is UML really an ADL?
- Is Statecharts an ADL?
- What makes LILEAnna an ADL?
- Is Demeter a software design philosophy or a language? And why is it an ADL?
- Is Aesop an environment or an ADL?
- Why is Rapide an ADL but its close cousin VHDL is not?
- Aren’t C2 and Weaves architectural styles?
- Why isn’t Java na ADL?

And the Most Important Question

- *What is an ADL?*

Trying to Answer the Question

- Conducted a study of ADLs in the late-1990s
- Defined what an ADL is
  - Eliminated several candidate notations in the process
- Suggested multiple dimensions for ADL understanding and classification
- Provided a detailed comparison of ADLs
- Expanded and updated the study several times
- Two principal publications came out of this work
  - ESEC/FSE 1997
  - IEEE TSE 2000

So, What Was the Answer?

- An ADL is a language that provides features for modeling a software system’s *conceptual* architecture, distinguished from the system’s *implementation*.
- An ADL must support the building blocks of an architectural description
  - Components
    - Interfaces
  - Connectors
  - Configurations

The Study in Retrospect – Benefits

- Improved the understanding of ADLs
- The two papers became a commonly accepted references in the SA community
  - After some grumbling, even the ADLs' authors accepted that the study was ultimately unbiased
- The definition became a "litmus test" for determining whether a particular notation is an ADL

The Study in Retrospect – Shortcomings

- The "litmus test" was not always effective
  - It took a 3-year study and a 60-page paper to "prove" that UML 1.x is not an ADL
  - It took another 2-year study to demonstrate that, e.g., Darwin does, in fact, support (limited) connector modeling
- Still did not answer the question of what "conceptual architecture" means
- Did not provide any help with understanding deeper questions
  - What is a model?
  - What is architecture?
  - What are differences among styles, domain-specific architectures, application families, product lines, product populations… ?

Wanted

answers

Once and for all

No Monetary Reward

Why Bother?

- These questions have been personally “bugging” me
- The discipline has matured enough to require them
  - Research
  - Practice
  - Pedagogy
- One added, specific impetus

Why Bother?

- These questions have been personally “bugging” me
- The discipline has matured enough to require them
  - Research
  - Practice
  - Pedagogy
- One added, specific impetus

Software Architecture:
Foundations, Theory, and Practice

Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy

(To appear, 2008)

What Happened to the ADLs?

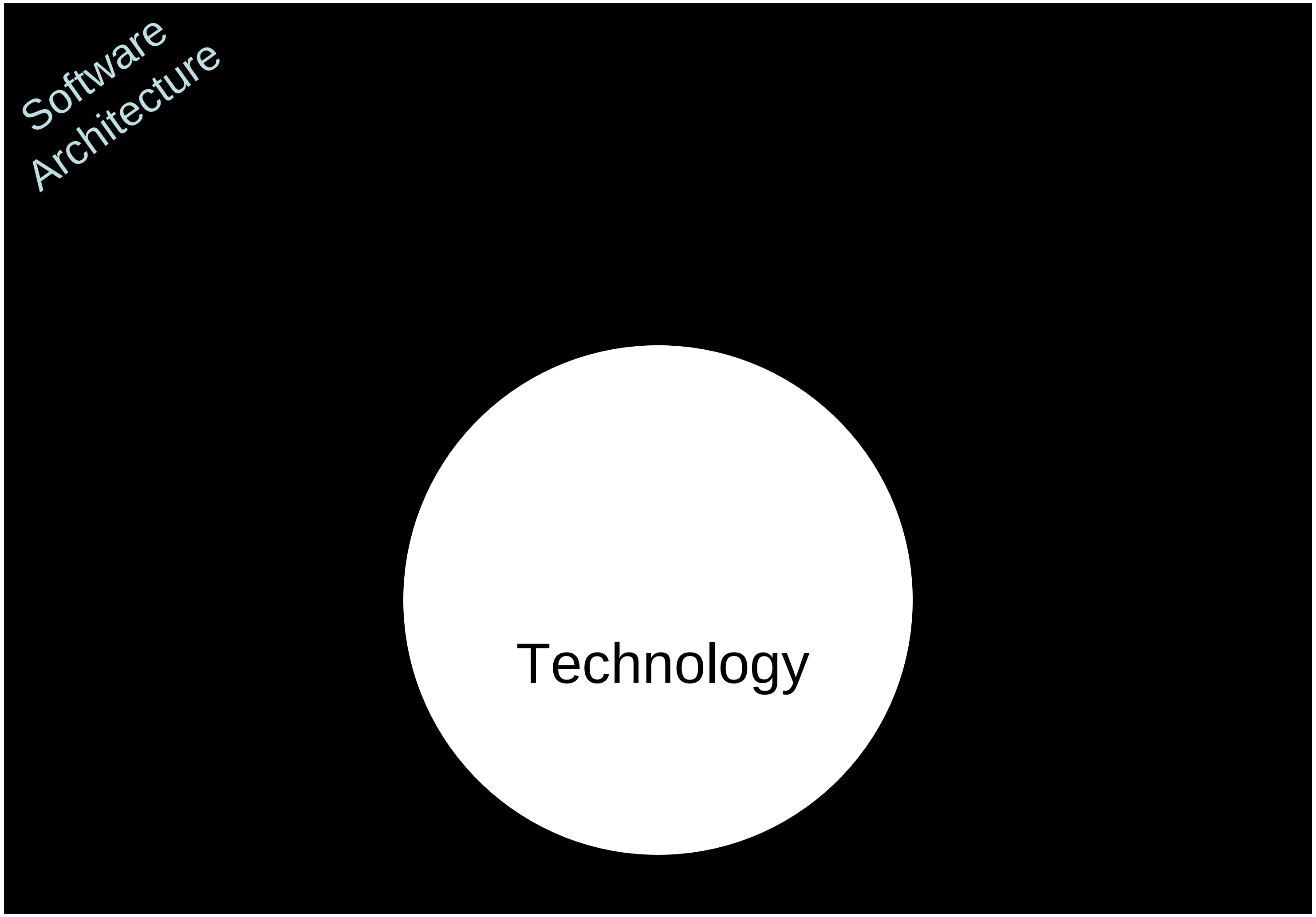
- The 1st generation ("1G") did not catch on
  - Although there are some 2G ADLs in use
- Almost no broader adoption
  - (slight) Exceptions are MetaH, Weaves, and Rapide
- What are some of the obvious reasons?
  - Often targeted at research environments
  - Awkward syntax and/or semantics
  - Modeling rigidity
  - Limited and idiosyncratic analysis support
  - Inadequate tool support
  - UML
    - Video killed the radio star…

A Deeper Reason

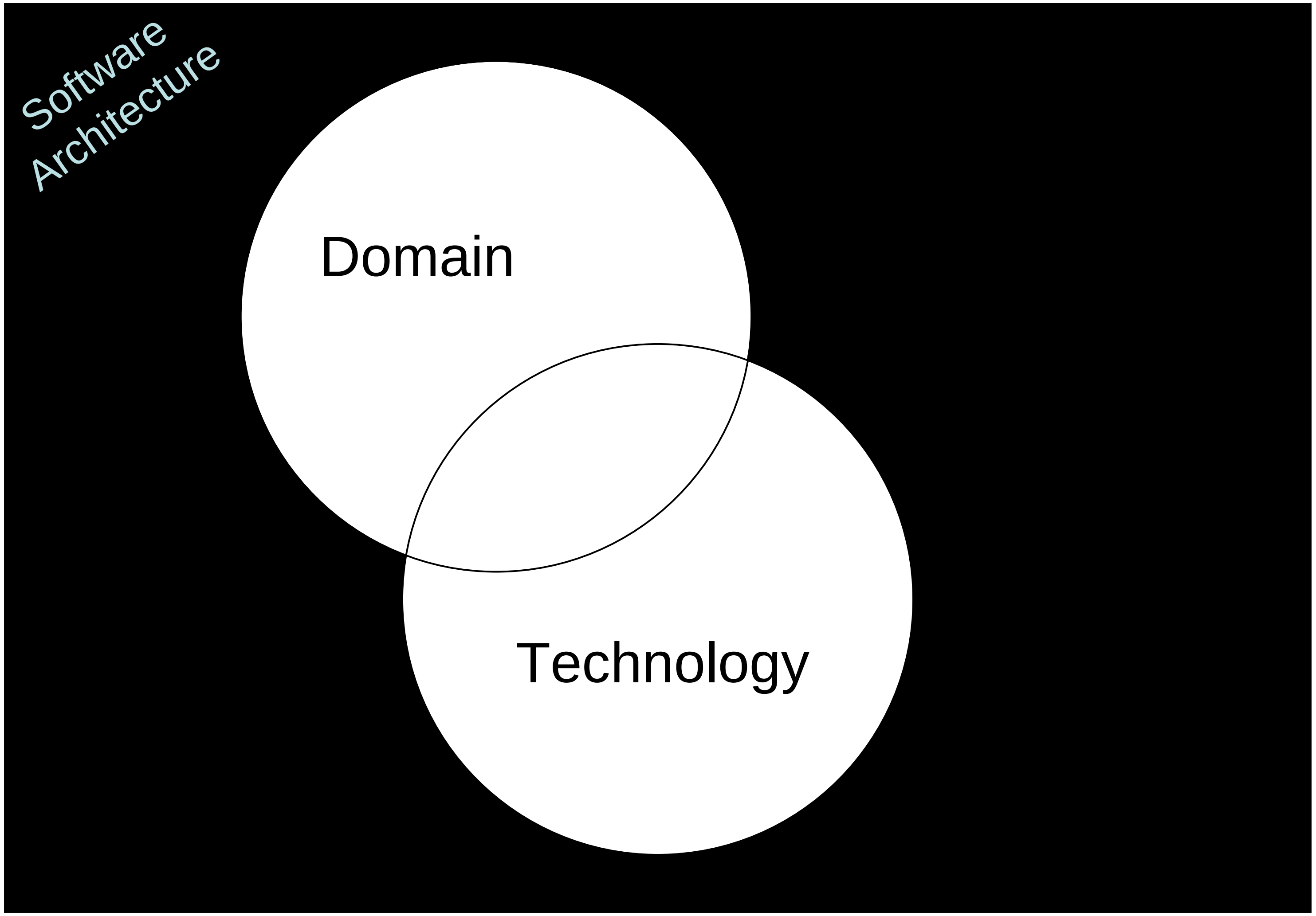
- 1G ADLs focused exclusively on technology
  - So did our study
- The broader context was completely missing
  - Relation to system requirements
  - Constraints imposed by implementation platforms
  - Characteristics of application domains
  - Organizational structure and politics
  - Business model
  - Position in the marketplace
  - …

What’s Out There?

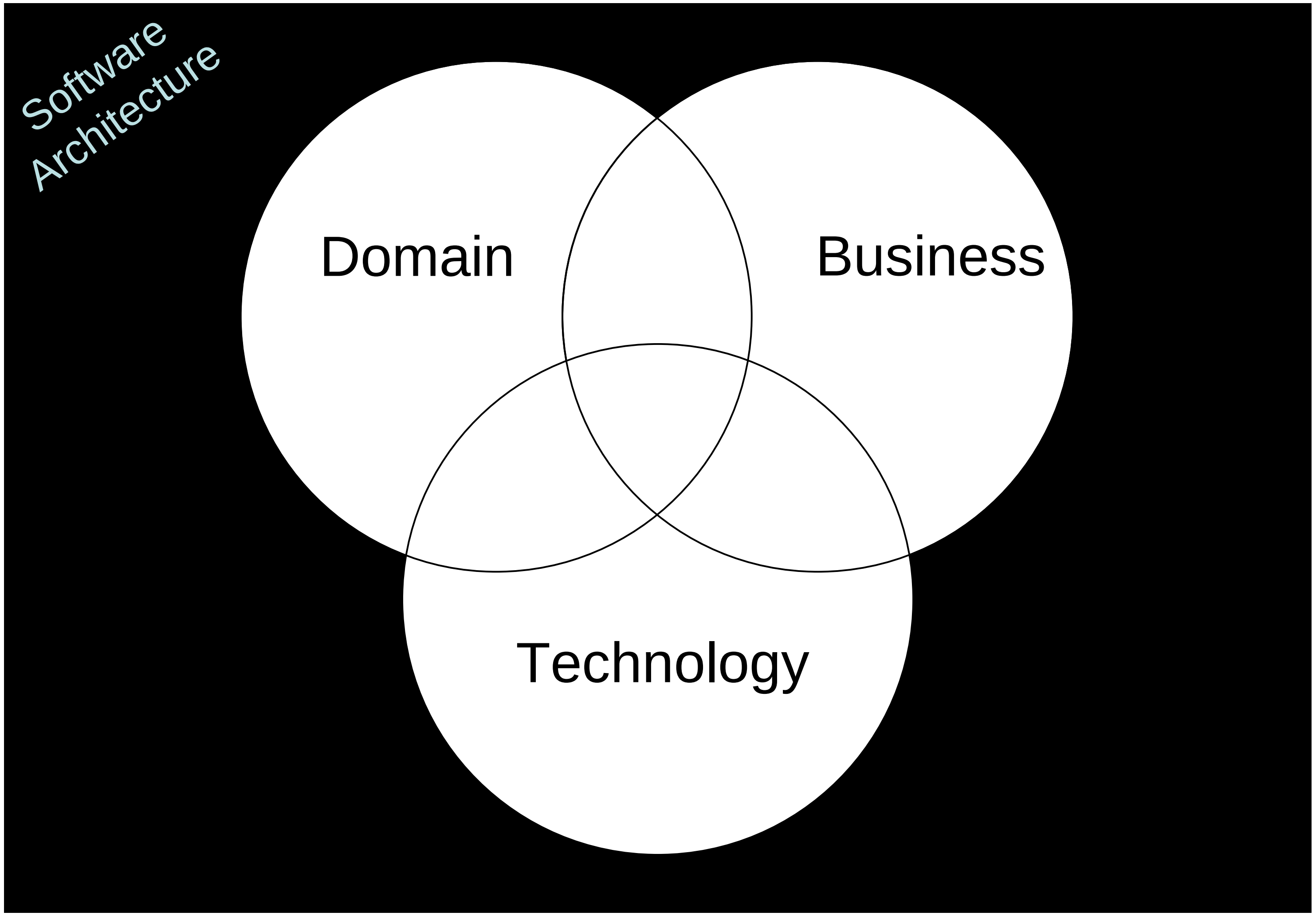
What’s Out There?



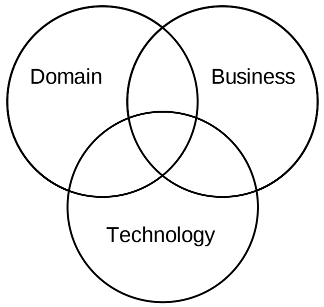
What’s Out There?



What’s Out There?

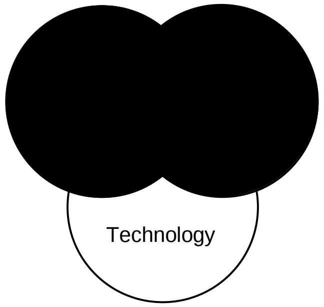


The Three Lampposts ("3L")



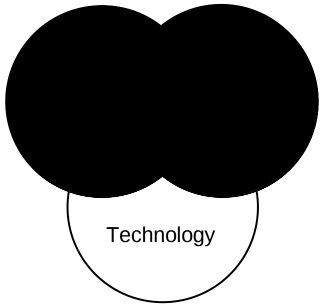
- Excessive or exclusive focus on technology is a critical failing of early ADLs
- 3L provides the needed answer
  - Illuminates the space of ADLs appropriately
  - Provides the necessary broad perspective on ADLs and their role in product development
  - Helps to classify and evaluate ADLs
  - Explains ADLs' successes and failures
  - Provides guidance for ADL developers
- ➢ Different lamps can still "shine" at different intensities

Technology



- Concerned with
  - Recurring technical challenges of engineering systems
  - Means for representing and reasoning about architectures
  - Critical abstractions and conceptual foundations of SA
- Results in
  - Most all 1G ADLs
  - Focus on analysis
    - Often using pre-existing analytical formalisms
  - Esoteric discussions
    - Relative merits of declarative vs. imperative ADLs
    - ADL interoperability
  - And some important ones
    - How do we transform architectures into implementations

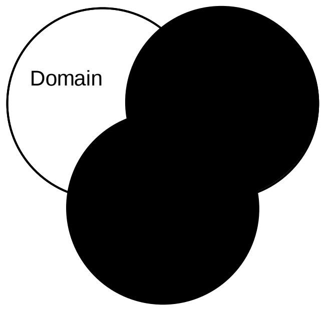
A Technology-Driven ADL



```
connector Pipe =
  role W = write → W ⊓ close → √
  role R =
     let Exit = close → √
     in let DoR = (read → R
                       ▯ read-eof → Exit)
     in DoR ⊓ Exit
```

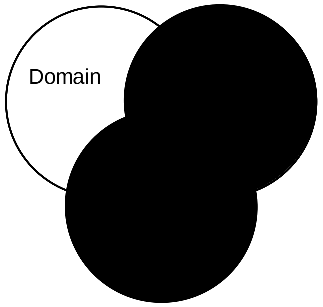
```
glue = let ROnly = R.read → ROnly
              ▯ R.read-eof → R.close → √
              ▯ R.close → √
     in let WOnly = W.write → WOnly
              ▯ W.close → √
     in W.write → glue
      ▯ R.read → glue
      ▯ W.close → ROnly
      ▯ Reader.close → WriteOnly
```

Domain

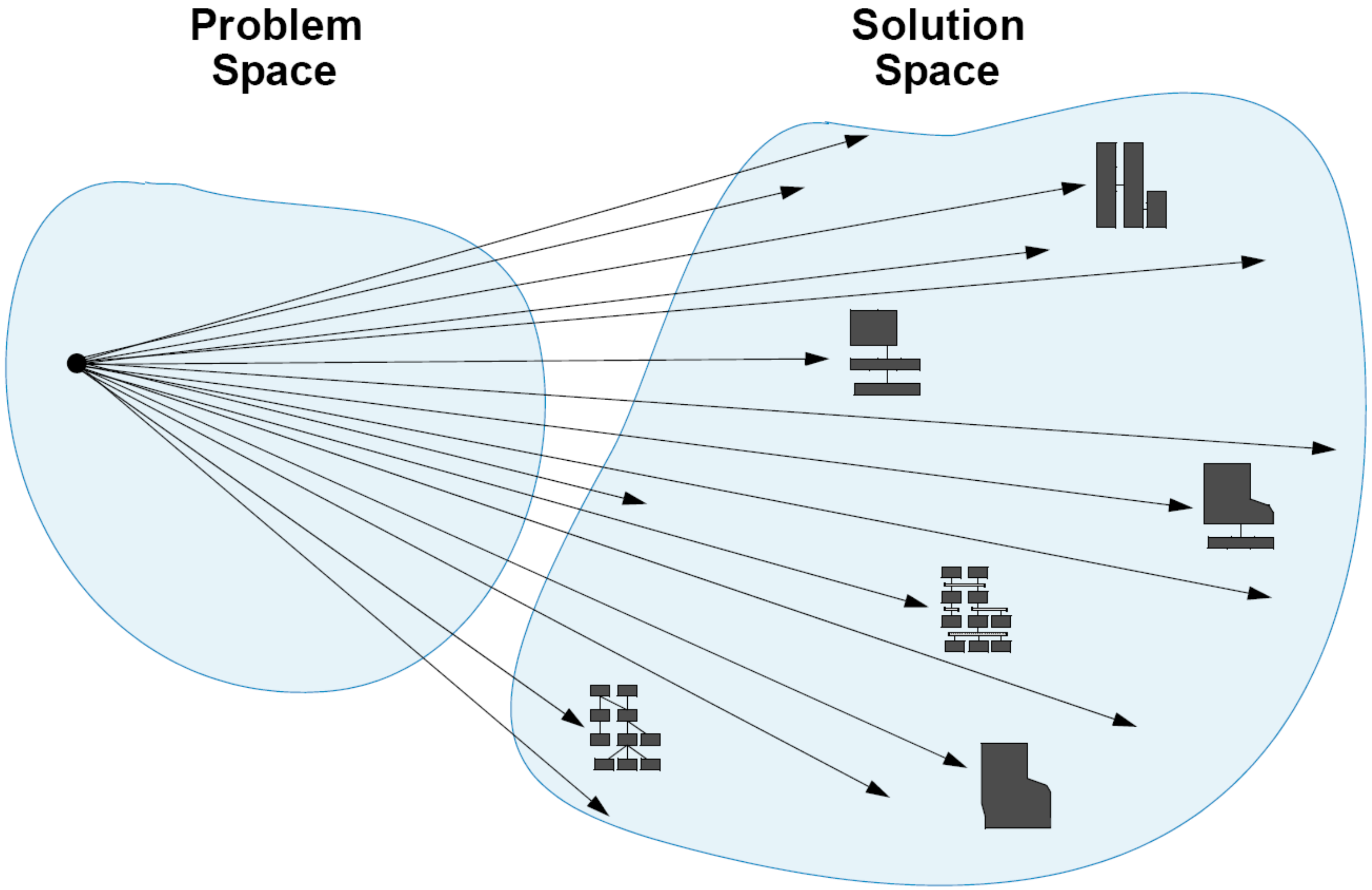


- Concerned with
  - Exploiting domain characteristics to aid system development
  - Means for representing and reasoning about problems in a given domain
- Results in
  - Successful 1G ADLs
    - MetaH, Weaves, GenVoca
  - Specialized, deeper solutions
  - Reusable assets
    - Including the architecture!
  - Engineers speaking the language of the users

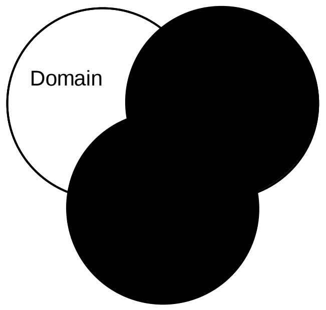
How Domains Help



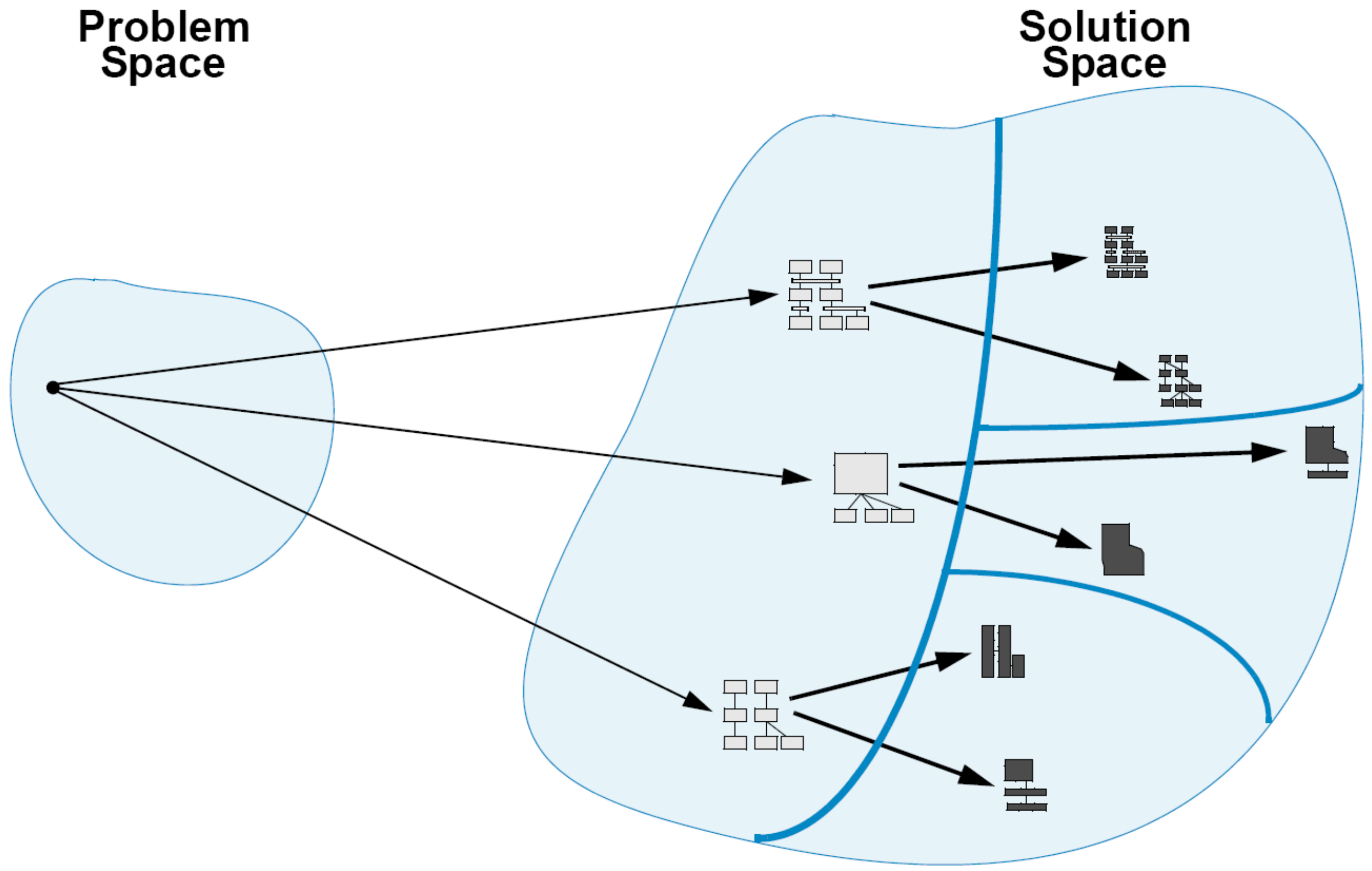
- Traditional software development



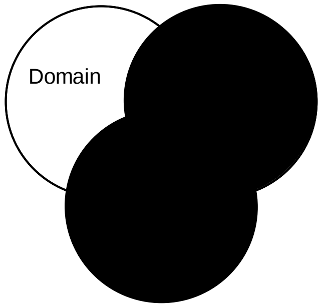
How Domains Help

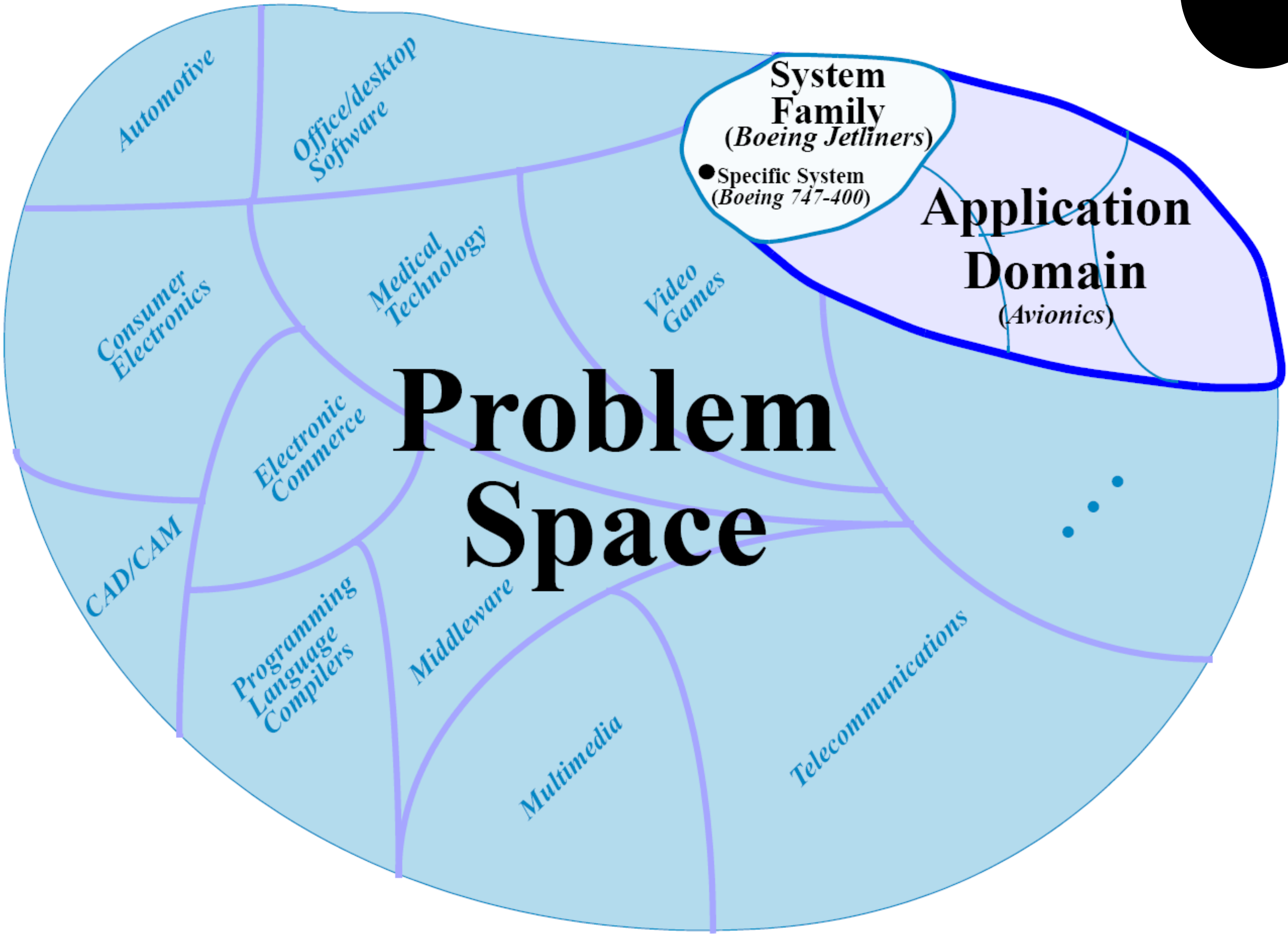


- Architecture-based software development

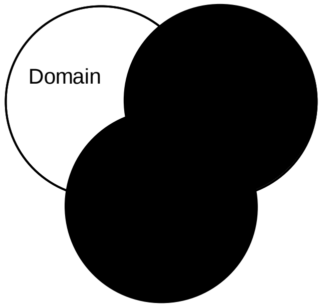


SE Problem Space

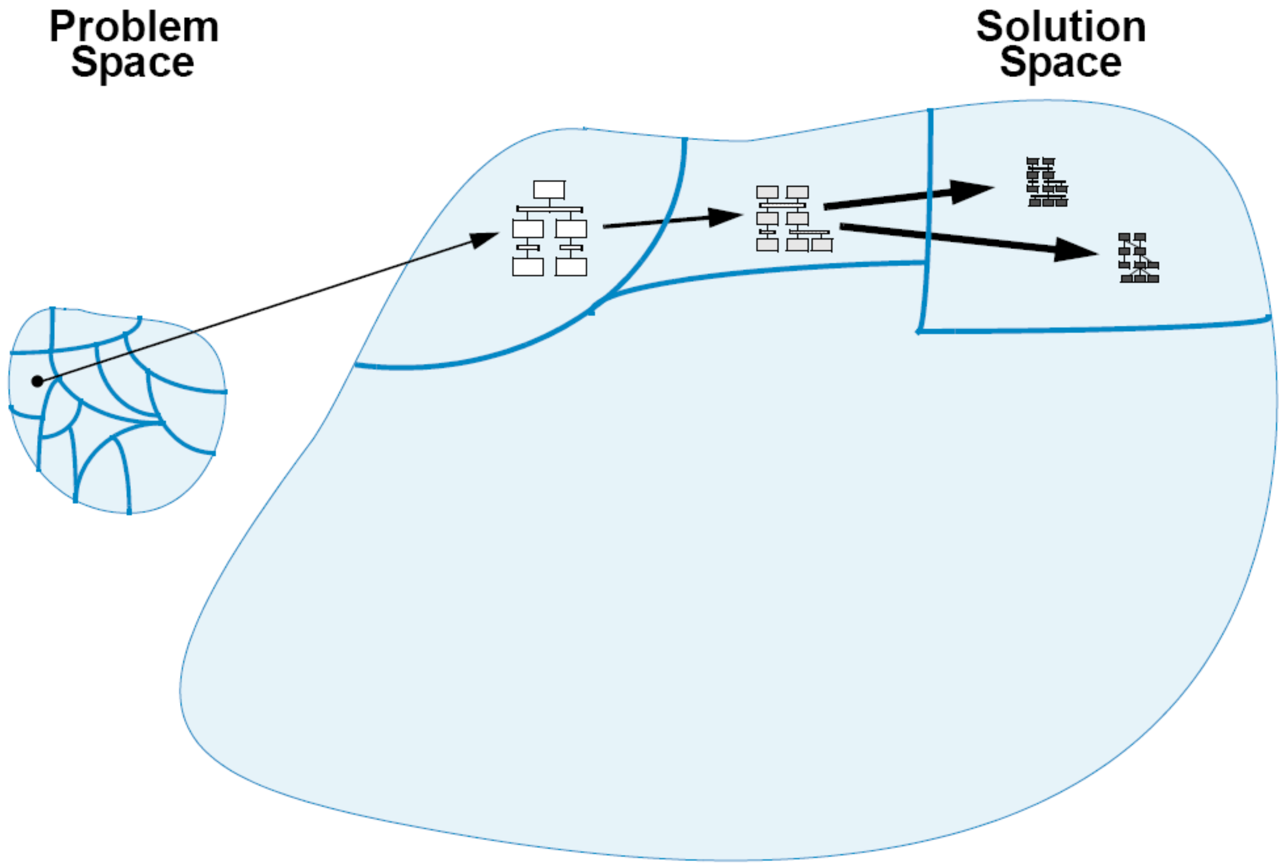




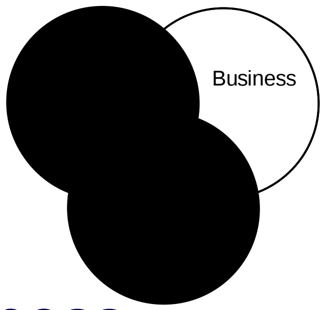
How Domains *Really* Help



- Domain-specific architecture-based software development

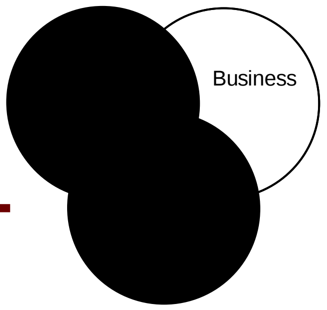


Business

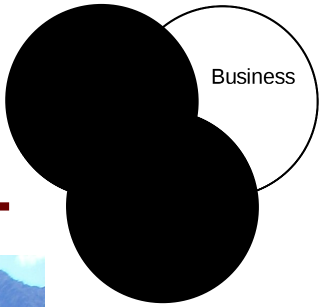


- Concerned with
  - Capturing and exploiting knowledge of the business context
  - Core competencies
  - Processes
  - Costs
    - Includes valuation of assets
- Results in
  - *No* 1G ADLs
  - Product strategy
  - Means for capturing multiple stakeholder perspectives
  - Characterization of desired product qualities
    - Tied to marketplace performance
  - What specifically, in an ADL?
    - Product relationships within a product line
    - Cost data per component

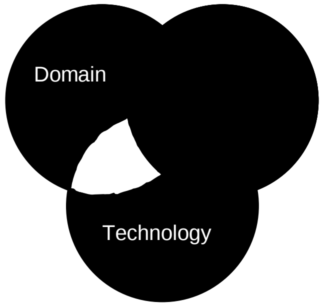
Example of Business Concerns Modeled in a 1G ADL



Example of Business Concerns Modeled in a 1G ADL

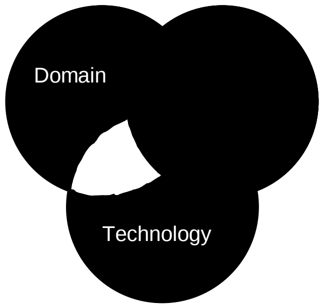


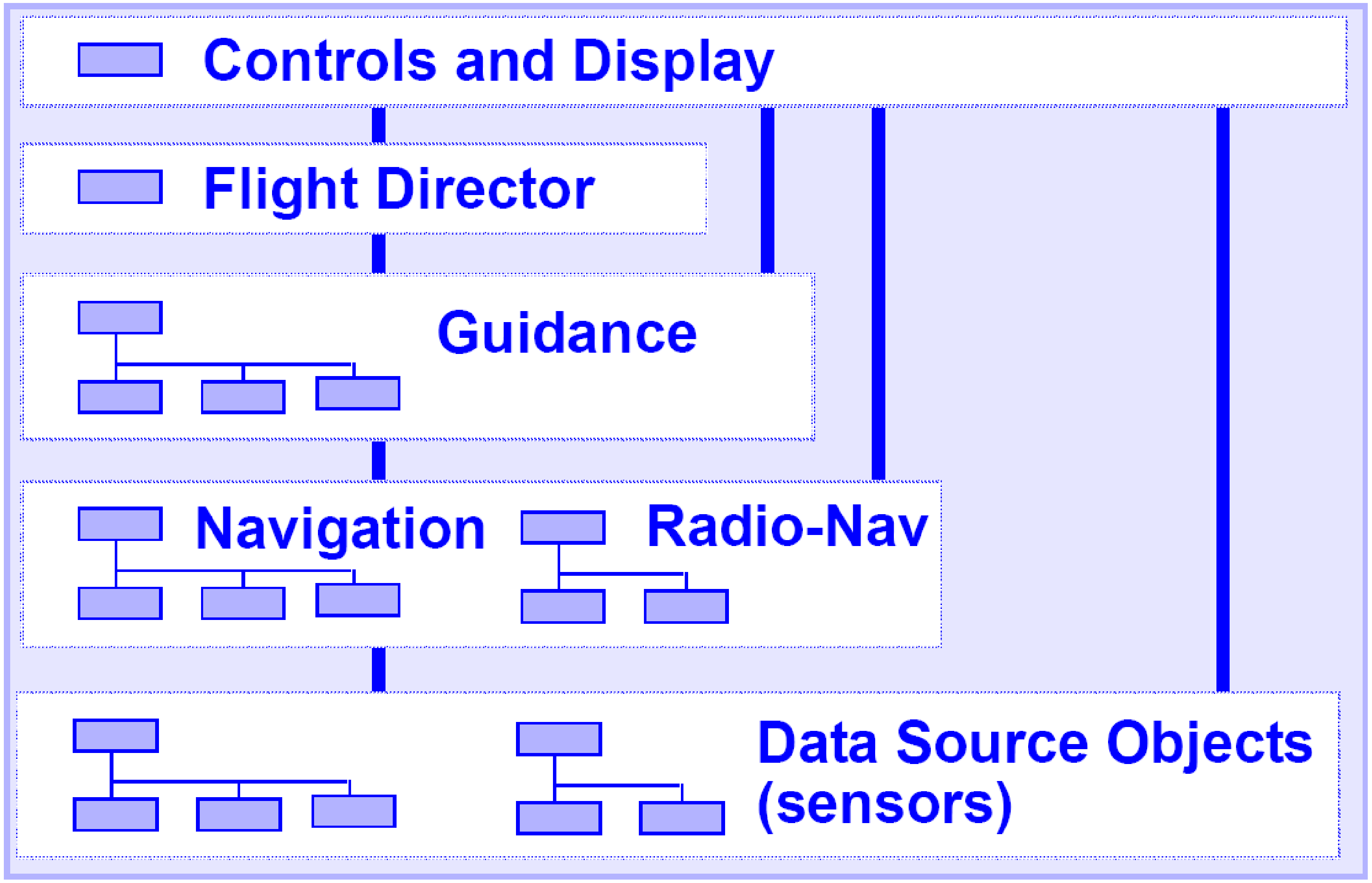
Technology + Domain



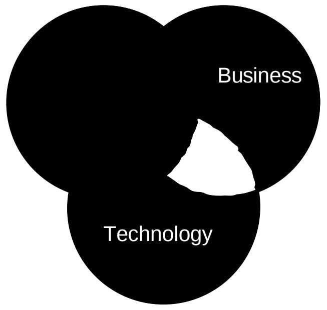
- Concerned with
  - Technological concerns specific to a domain
  - System generation from models
- Results in
  - Application-family architectures
  - Domain-specific languages

A 1G DSSA


Domain
Technology

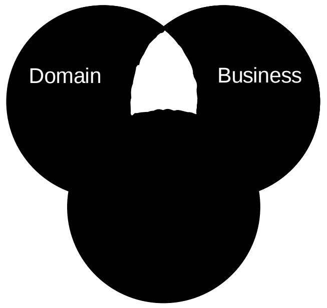

Controls and Display
Flight Director
Guidance
Navigation
Radio-Nav
Data Source Objects (sensors)

Technology + Business



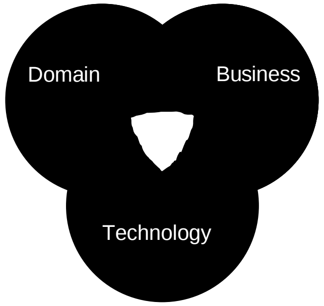
- Concerned with
  - Linking business issues with system construction
  - Investment in infrastructure
    - Winning “technology wars”
- Results in
  - Relationship of process steps to software elements
  - CM systems
  - Architecture-centric cost estimation tools
    - COCOMO, COSYSMO, COCOTS

Domain + Business



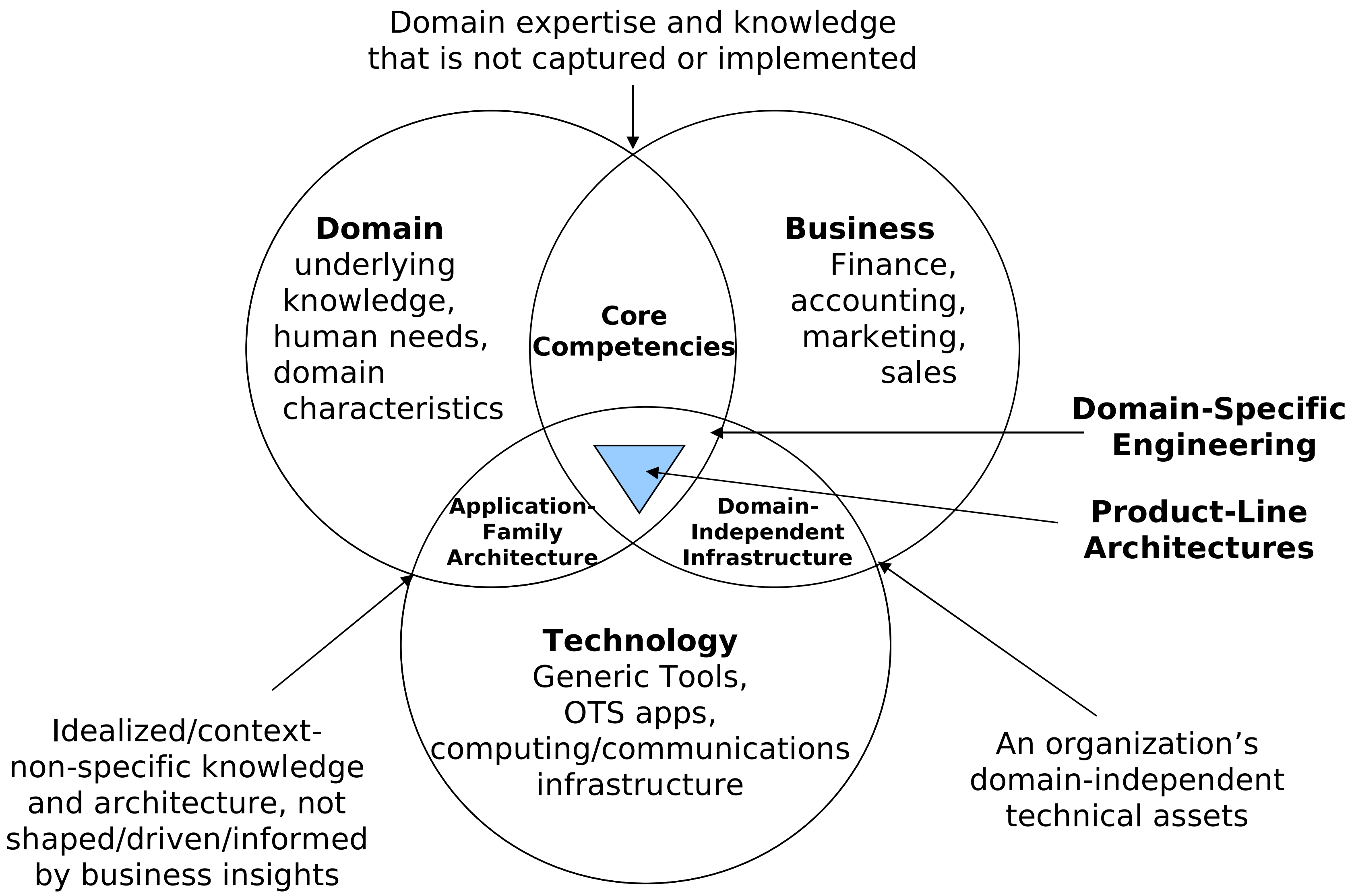
- Concerned with
  - Core competencies
    - What you know how to do well and profitably
- Results in
  - Domain models
  - Business models
  - Processes
  - Customer profiles and requirements
  - *No* technology!

Technology + Domain + Business

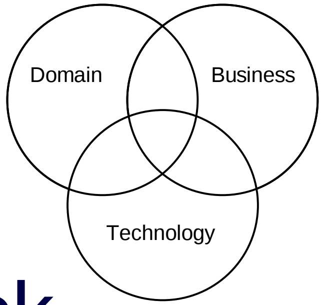


- Concerned with
  - Being a successful software development outfit
- Results in
  - Software product lines

Putting It All Together


Domain expertise and knowledge that is not captured or implemented
Domain underlying knowledge, human needs, domain characteristics
Core Competencies
Business Finance, accounting, marketing, sales
Domain-Specific Engineering
Product-Line Architectures
Application-Family Architecture
Domain-Independent Infrastructure
Technology Generic Tools, OTS apps, computing/communications infrastructure
Idealized/context-non-specific knowledge and architecture, not shaped/driven/informed by business insights
An organization's domain-independent technical assets

2G ADLs



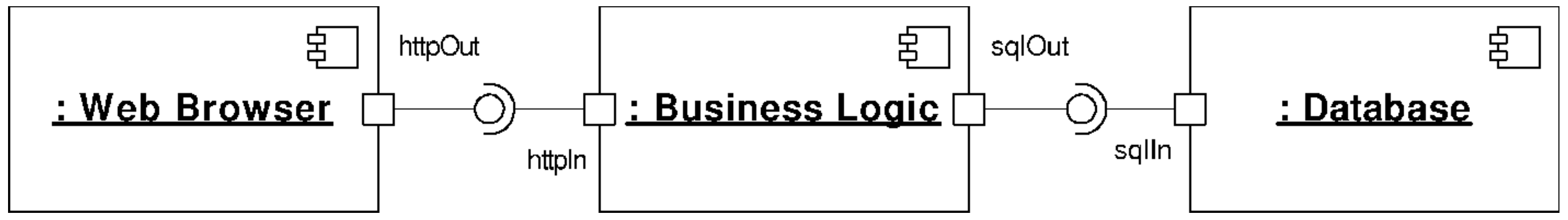
- Only a handful of 1G ADLs have “stuck around”…
  - …but, boy, have they changed
- They evolved into 2G ADLs
  - UML 2.0 ← UML 1.x
  - AADL ← MetaH
  - Koala ← Darwin ← Conic
  - xADL 2.0 ← xADL 1.0 ← C2
- All have strong technological foci
  - Yet they are very different from each other

UML 2.0

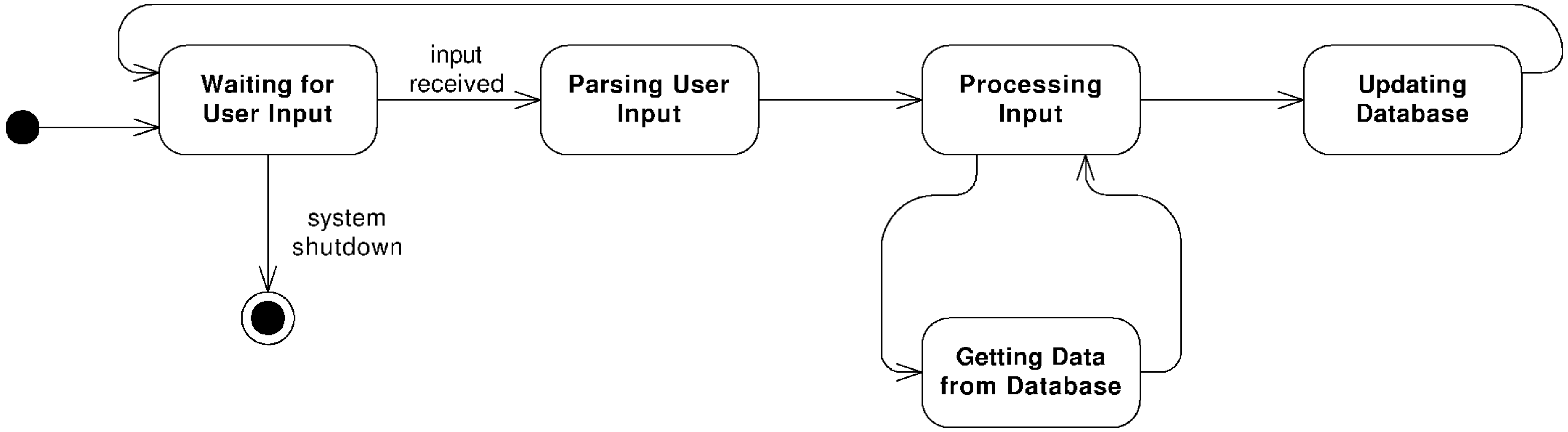
- De facto standard software design language
  - – Developed by OMG
- A “Swiss Army Knife” of notations
- Has a number of architectural constructs
- Ubiquitous

➢ Primary focus – to conquer the world

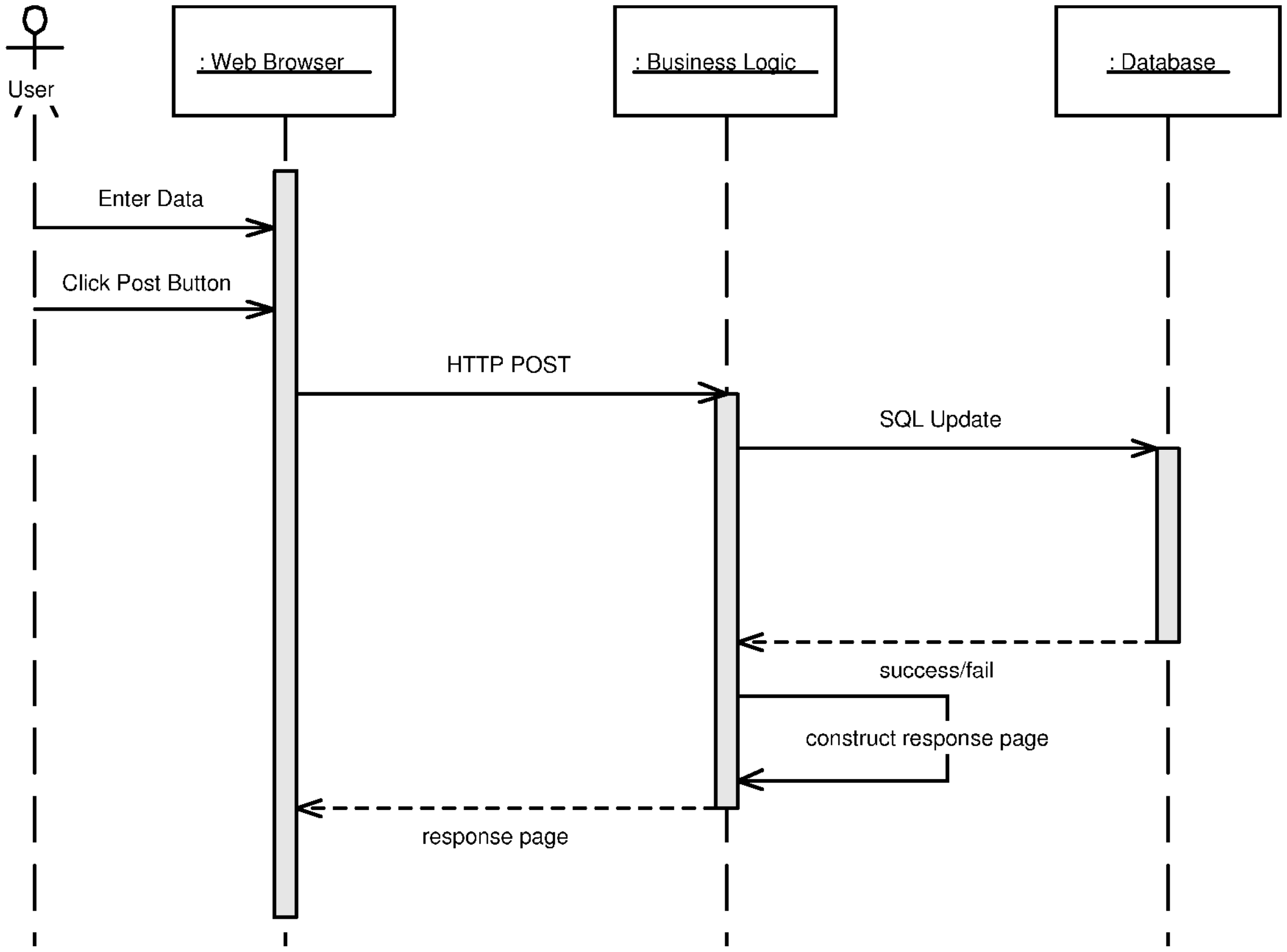
UML 2.0 in Action



UML 2.0 in Action

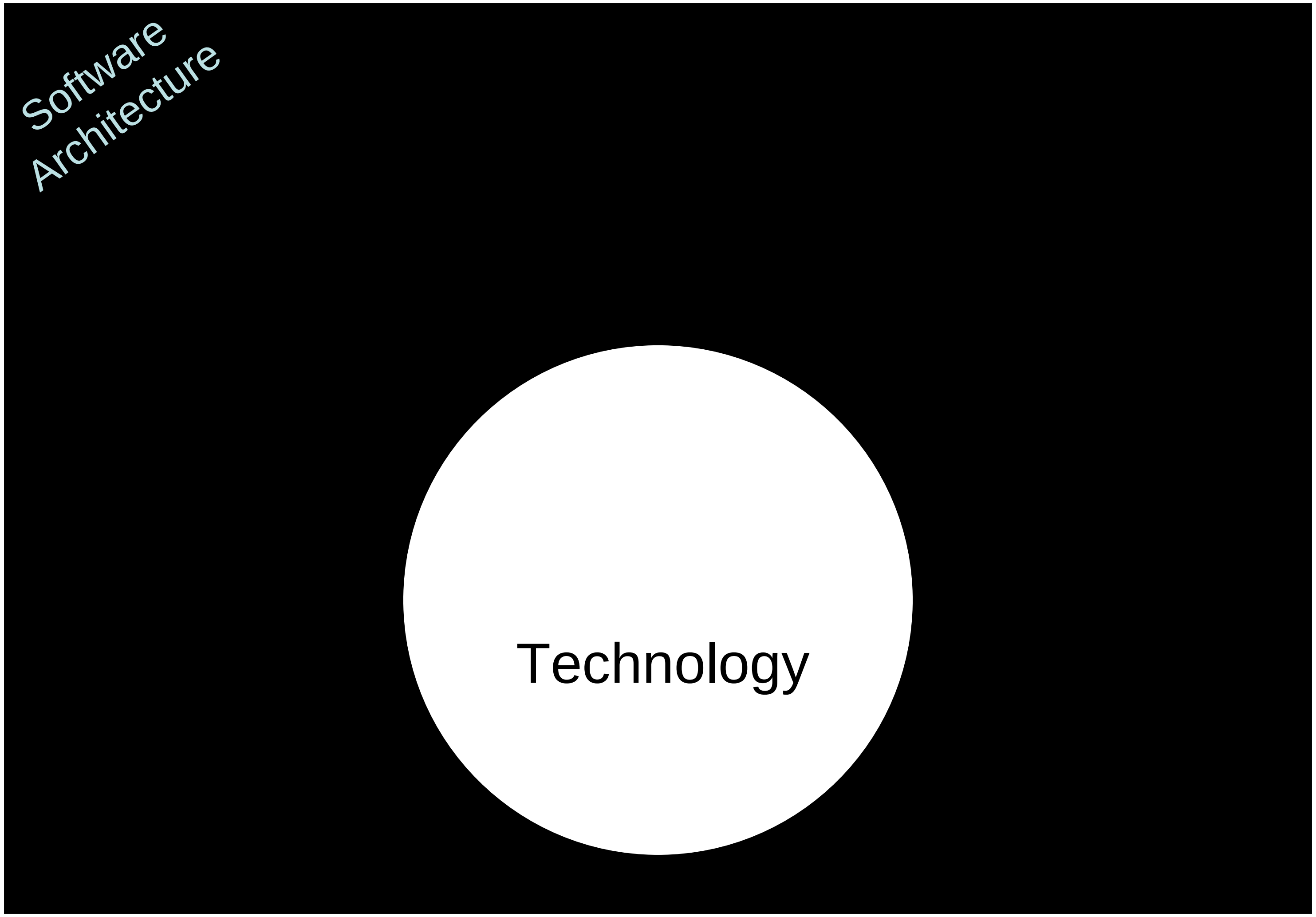


UML 2.0 in Action

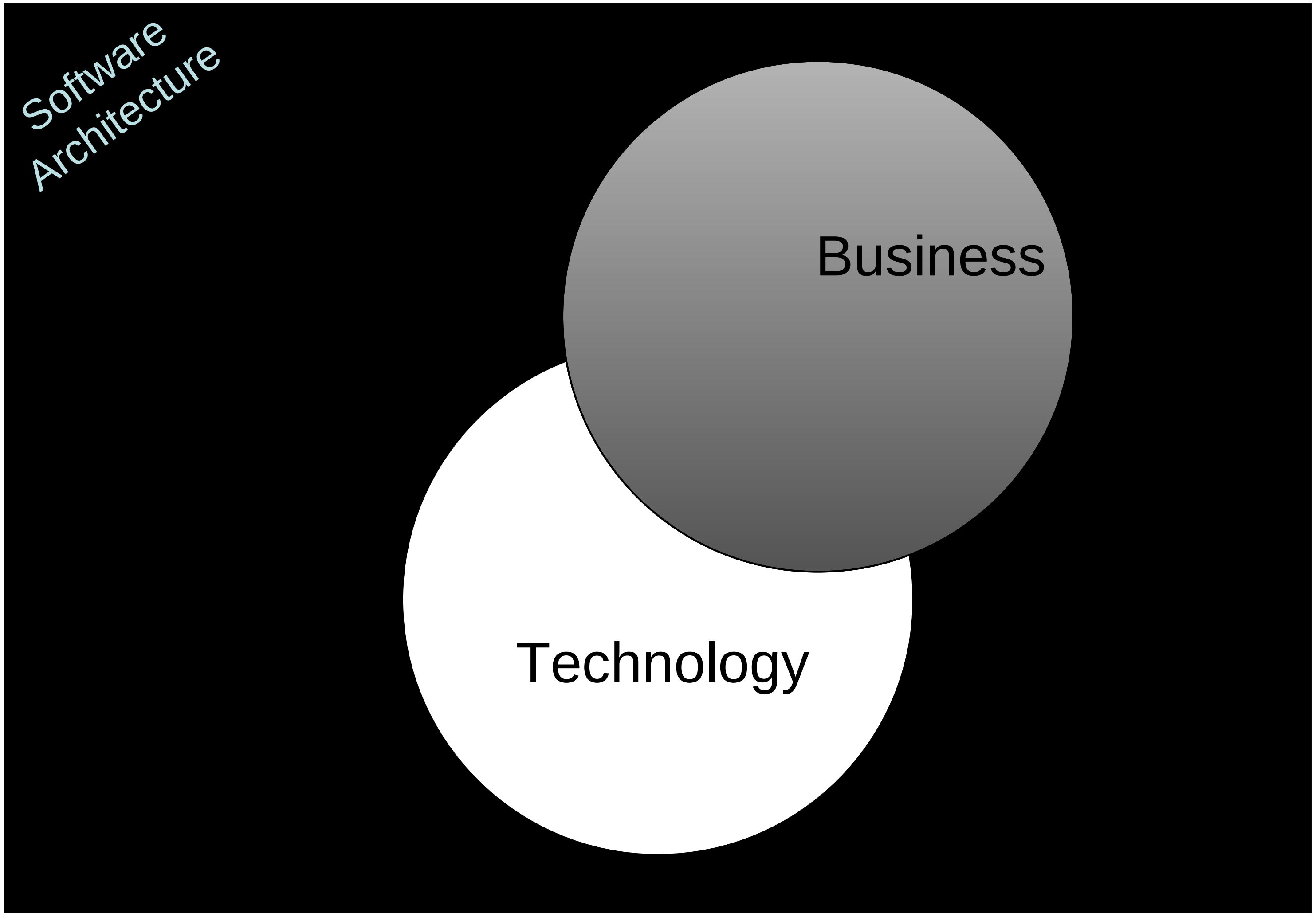


UML 2.0 Under the Lampposts

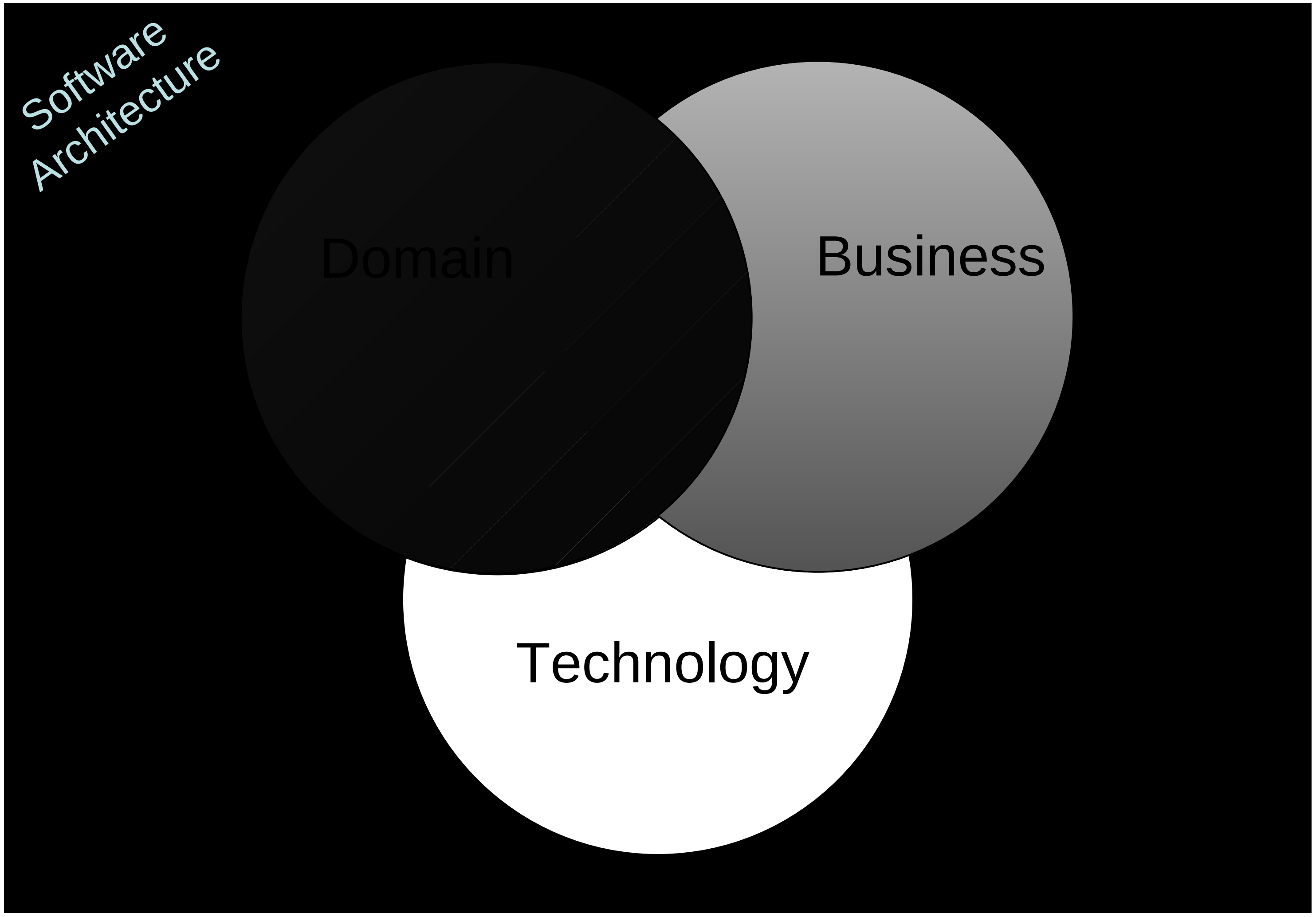
UML 2.0 Under the Lampposts



UML 2.0 Under the Lampposts



UML 2.0 Under the Lampposts



AADL

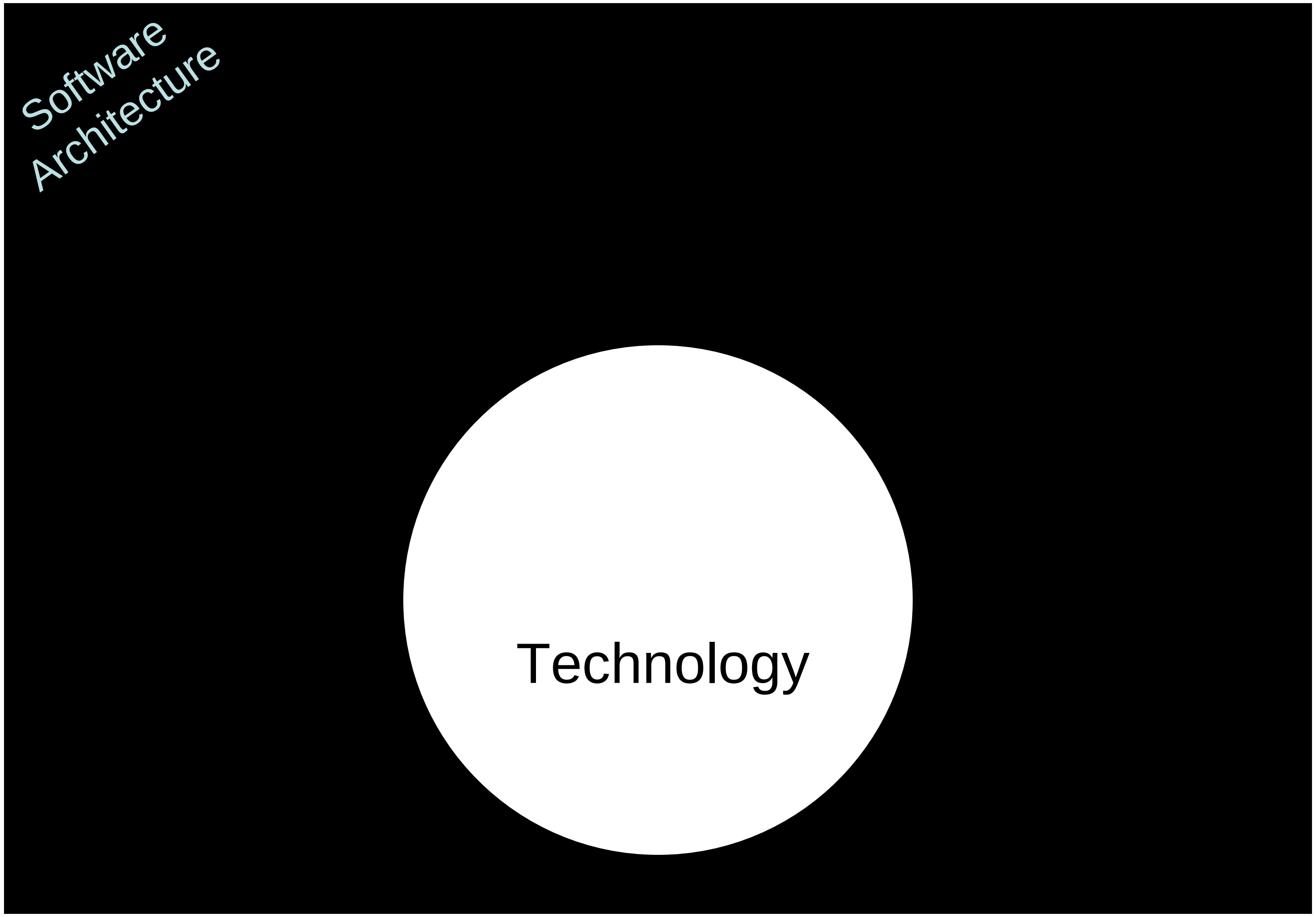
- Architecture Analysis and Design Language
  - Initially stood for “Avionics ADL”
- Primarily textual
- Very detailed
  - An AADL *component* runs on a *processor*, which runs one or more *process*es, each of which contains one or more *threads* of control, all of which can receive instructions through *in* ports and send data through *out* ports over a *bus*…
- ➢ Primary focus – embedded, real-time, hybrid systems

AADL in Action

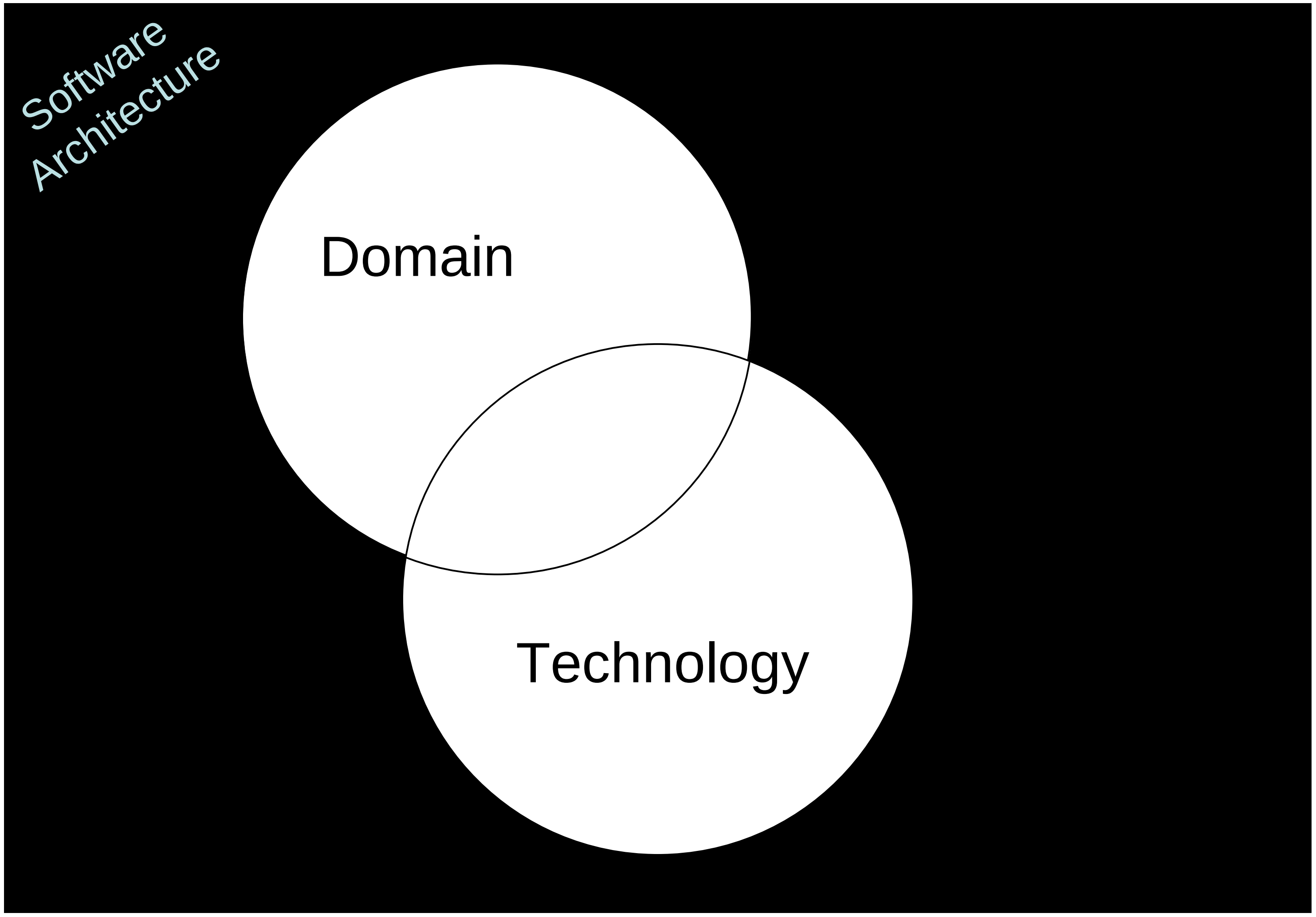
```
system implementation sensor_type.temperature
subcomponents
	the_sensor_processor :
		processor sensor_processor_type;
	the_sensor_process : process
		sensor_process_type.one_thread;
connections
	bus access network -> the_sensor_processor.network;
	event data port sensed ->
		the_sensor_process.sensed;
	event data port control ->
		the_sensor_process.control;
properties
	Actual_Processor_Binding => reference
		the_sensor_processor applies to
		the_sensor_process;
end sensor_type.temperature;
```

AADL Under the Lampposts

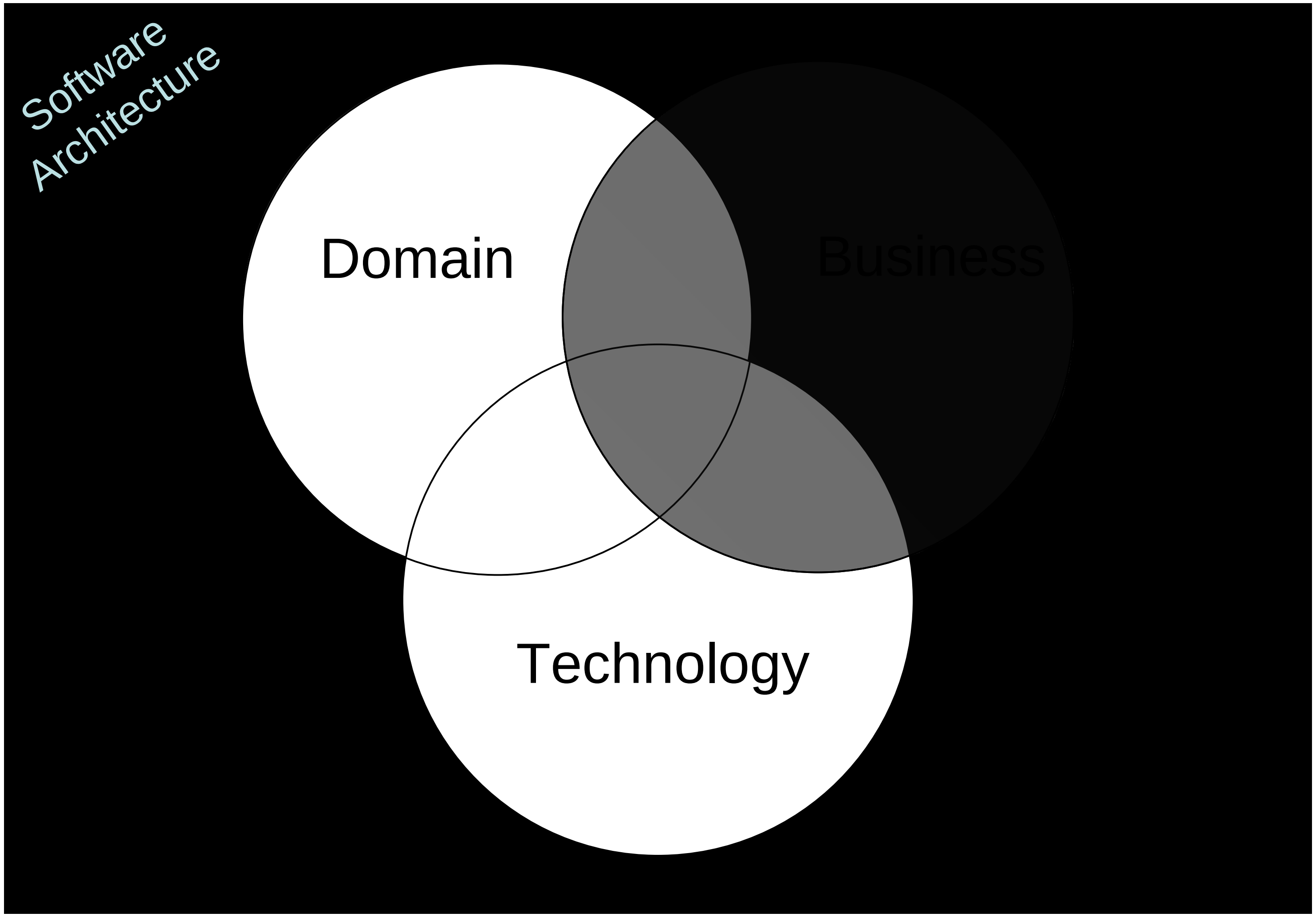
AADL Under the Lampposts



AADL Under the Lampposts



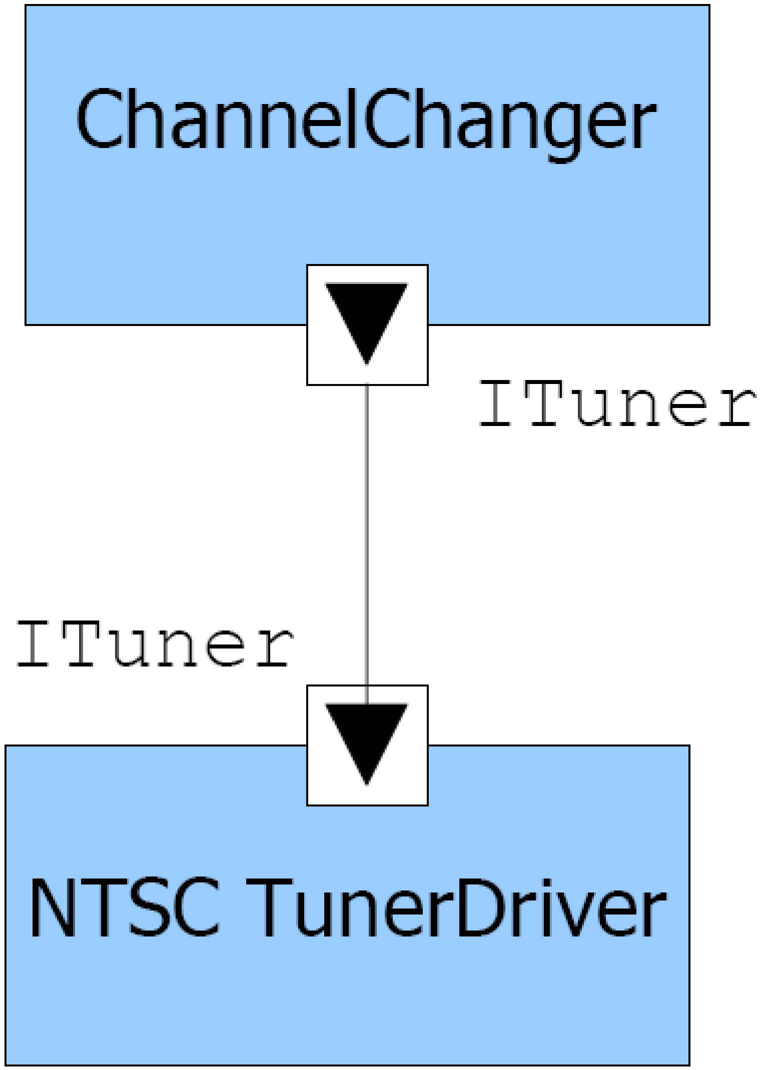
AADL Under the Lampposts



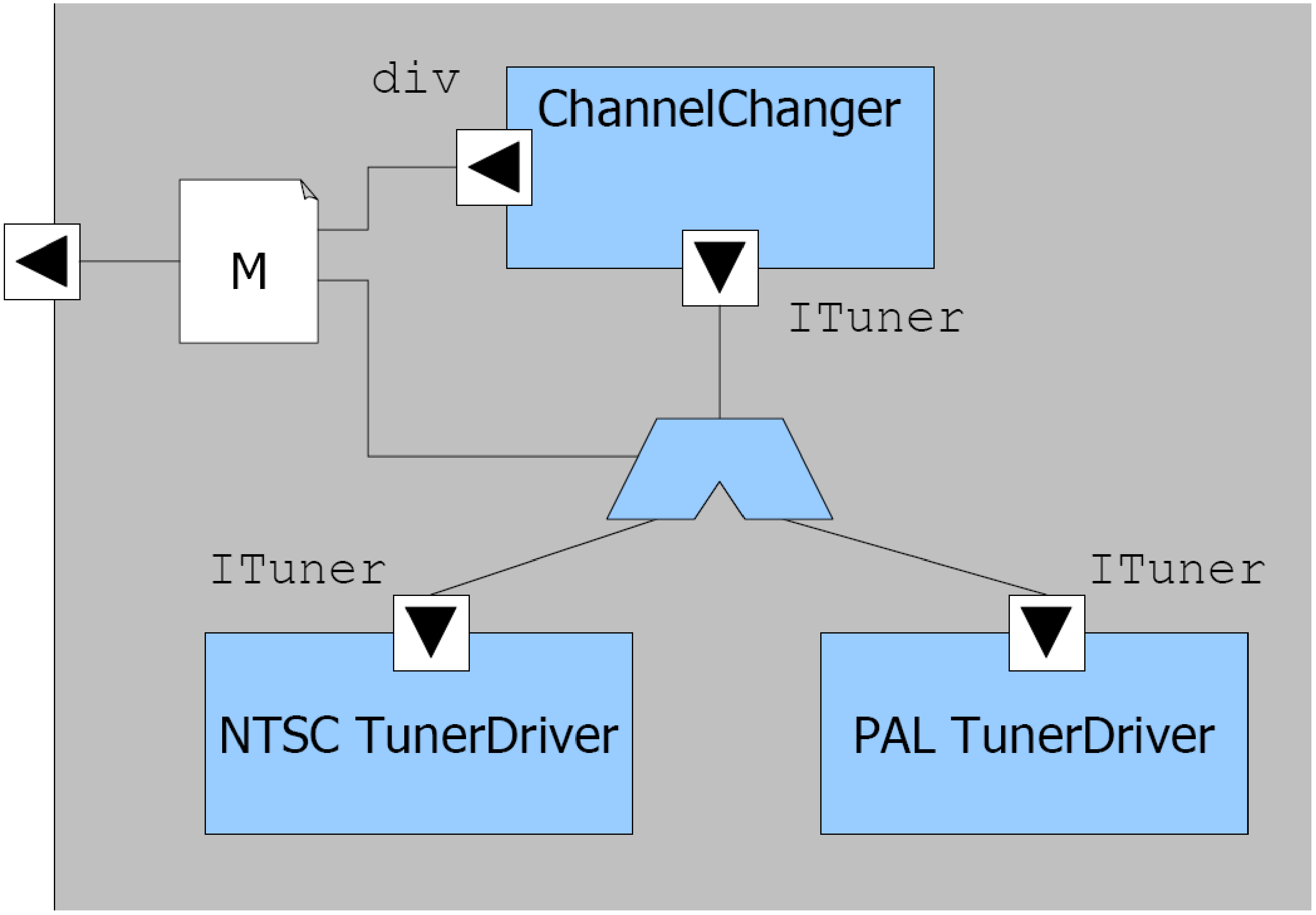
Koala

- Developed at Philips
  - In collaboration with Imperial College London
- Used in the consumer electronics domain
- Both graphical and textual
- ➢ Primary focus – management of product populations
  - Modeling
  - Analysis
  - Implementation generation
  - Deployment

Koala in Action

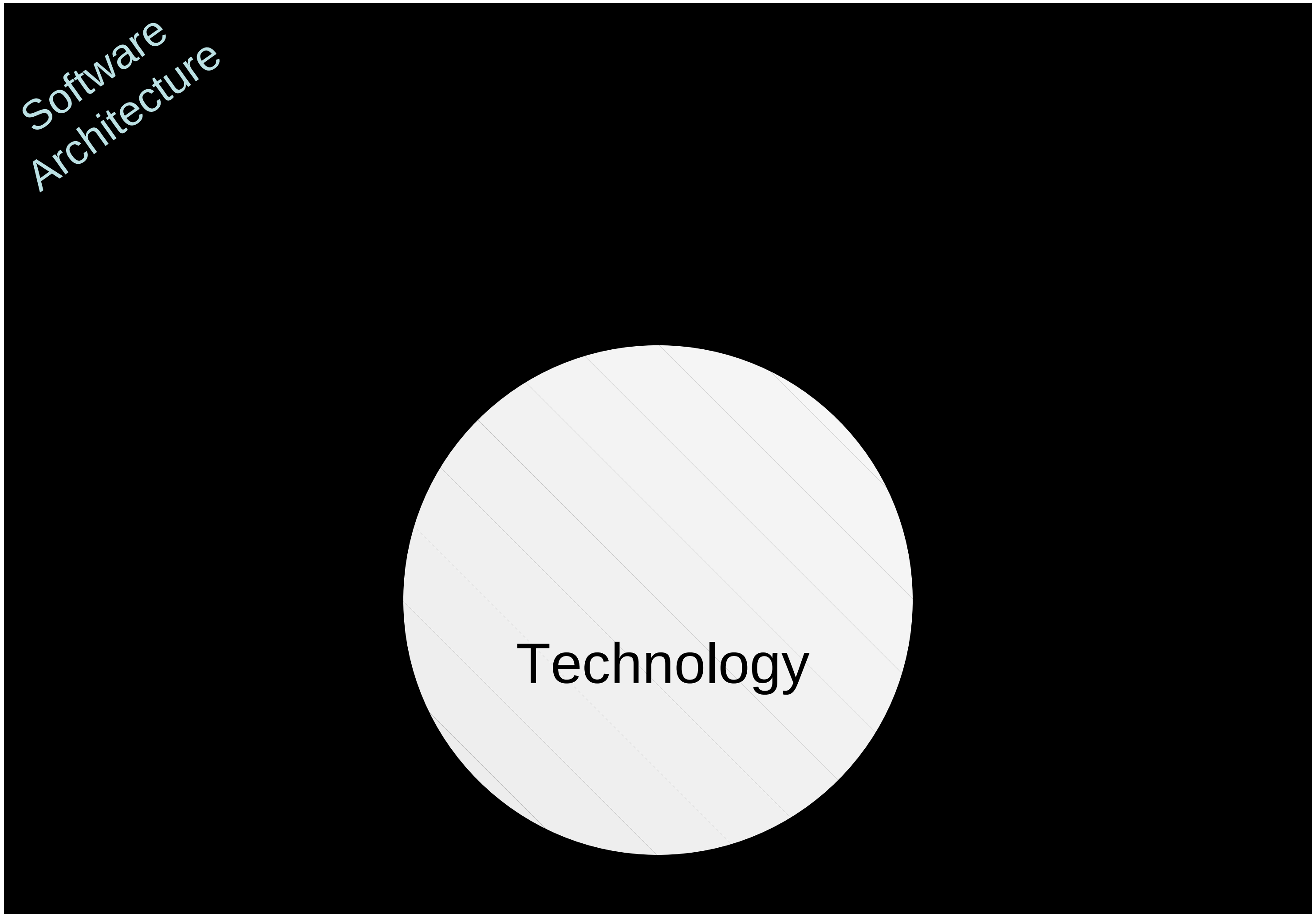


Koala in Action

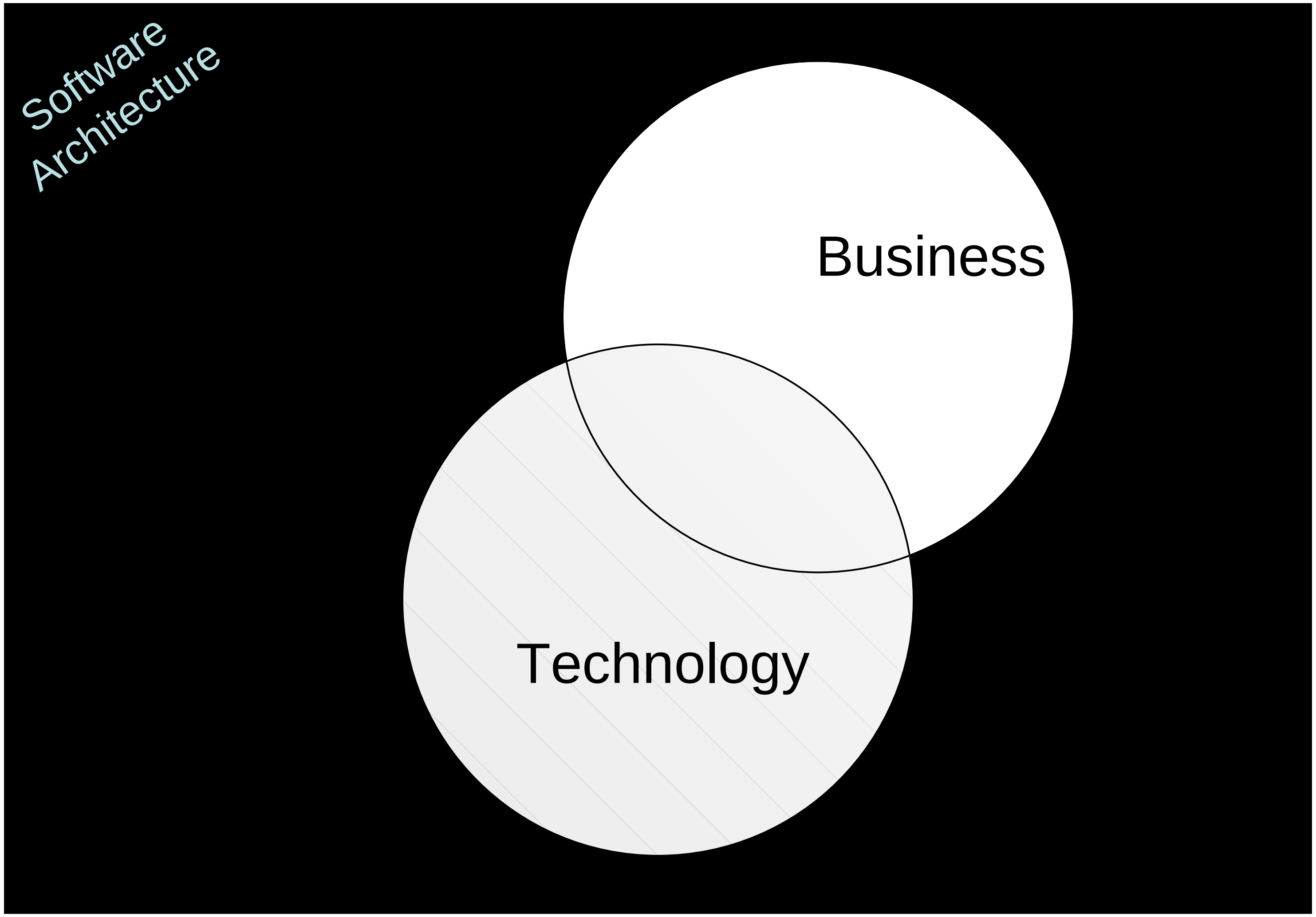


Koala Under the Lampposts

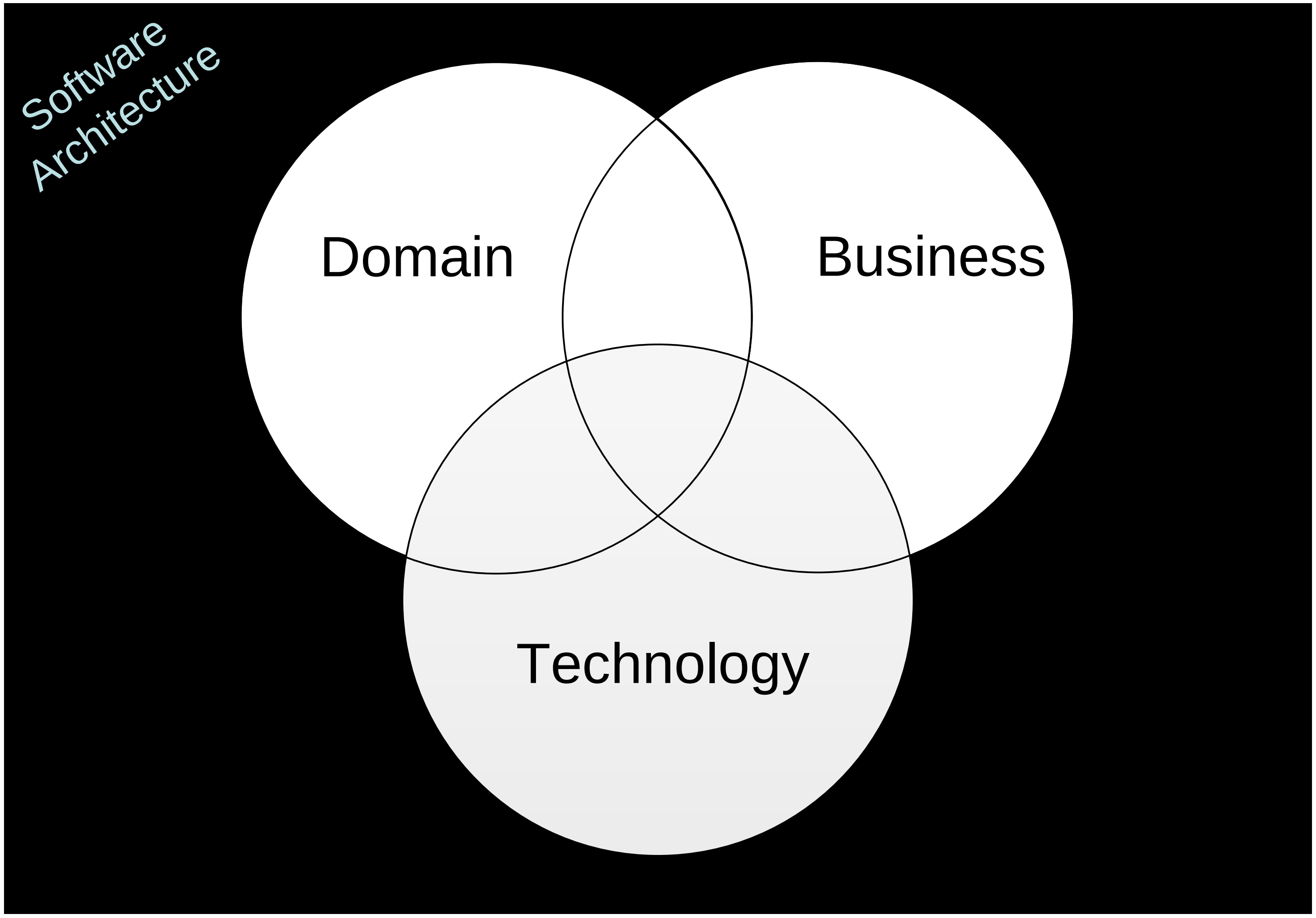
Koala Under the Lampposts



Koala Under the Lampposts



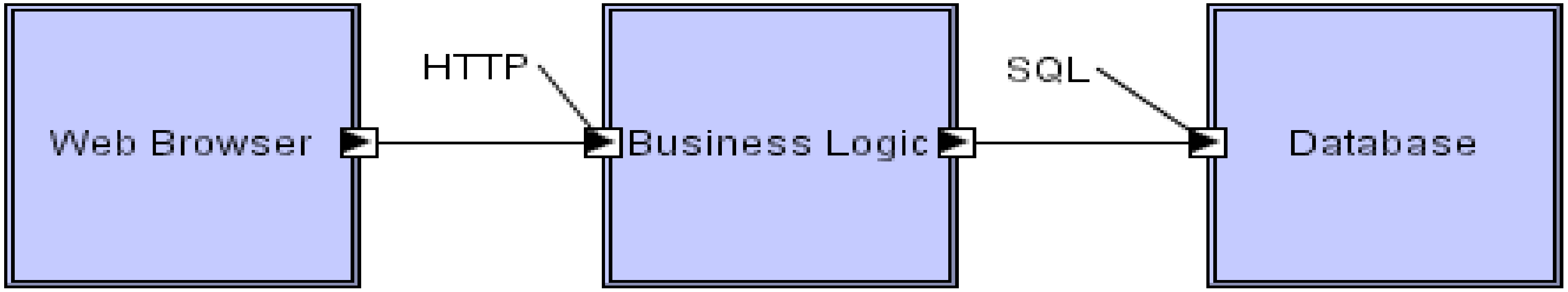
Koala Under the Lampposts



xADL 2.0

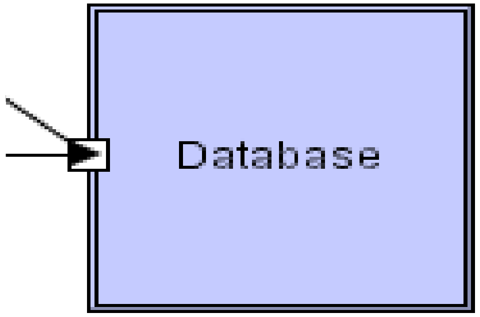
- Developed at UC Irvine
  - In use at Boeing
- XML substrate
- Both graphical and textual
- Primary focus – extensibility

xADL 2.0 in Action



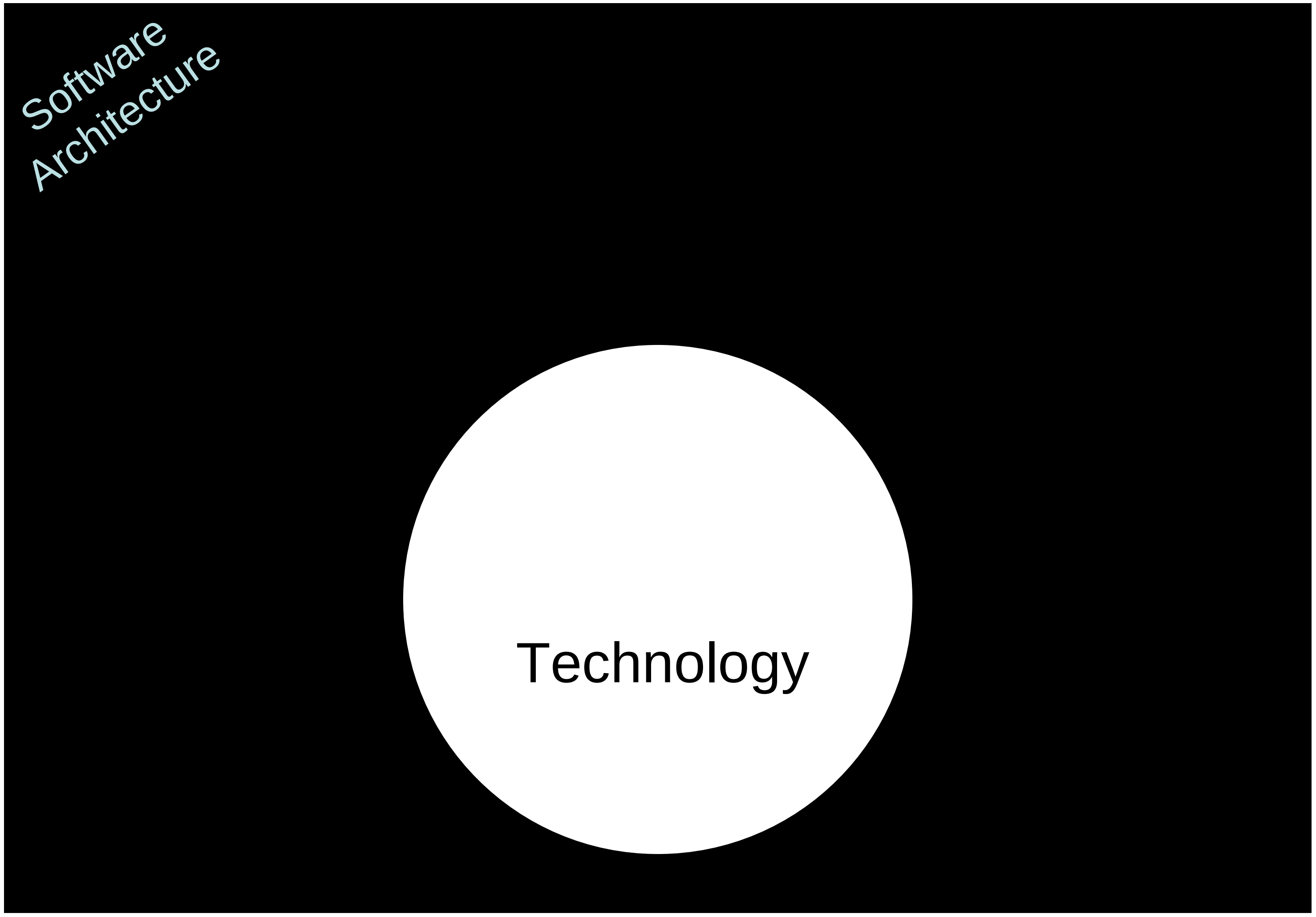
xADL 2.0 in Action

```
<component id="dbComp">
    <description>Database</description>
    <interface id="sql-in">
        <description>SQL</description>
        <direction>in</direction>
    </interface>
    <datasource>
        <vendor>Oracle Corp.</vendor>
        <location>db.example.com:1234/db1</location>
        <username>webUser</username>
        <password>secret</password>
    </datasource>
</component>
```

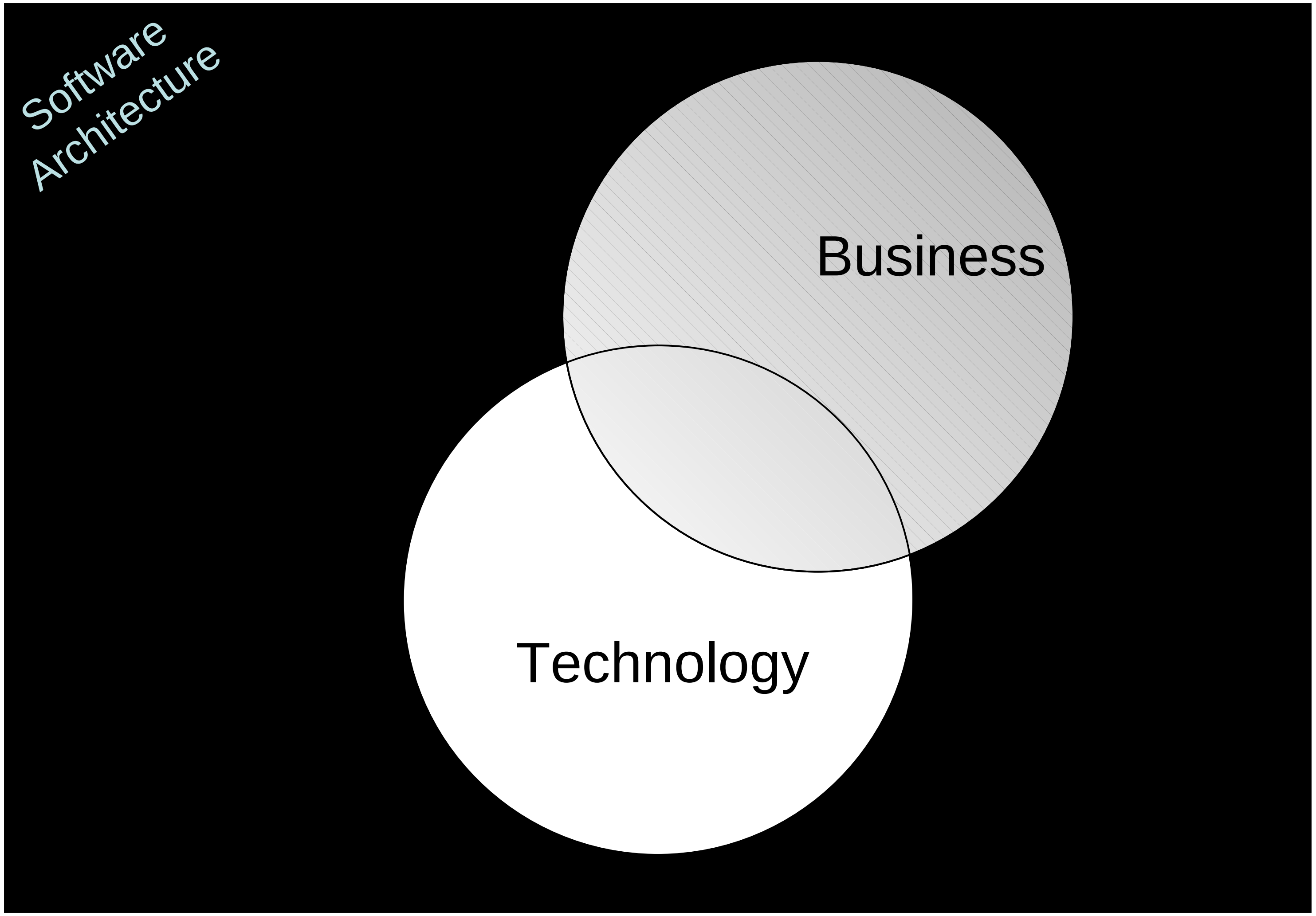


xADL 2.0 Under the Lampposts

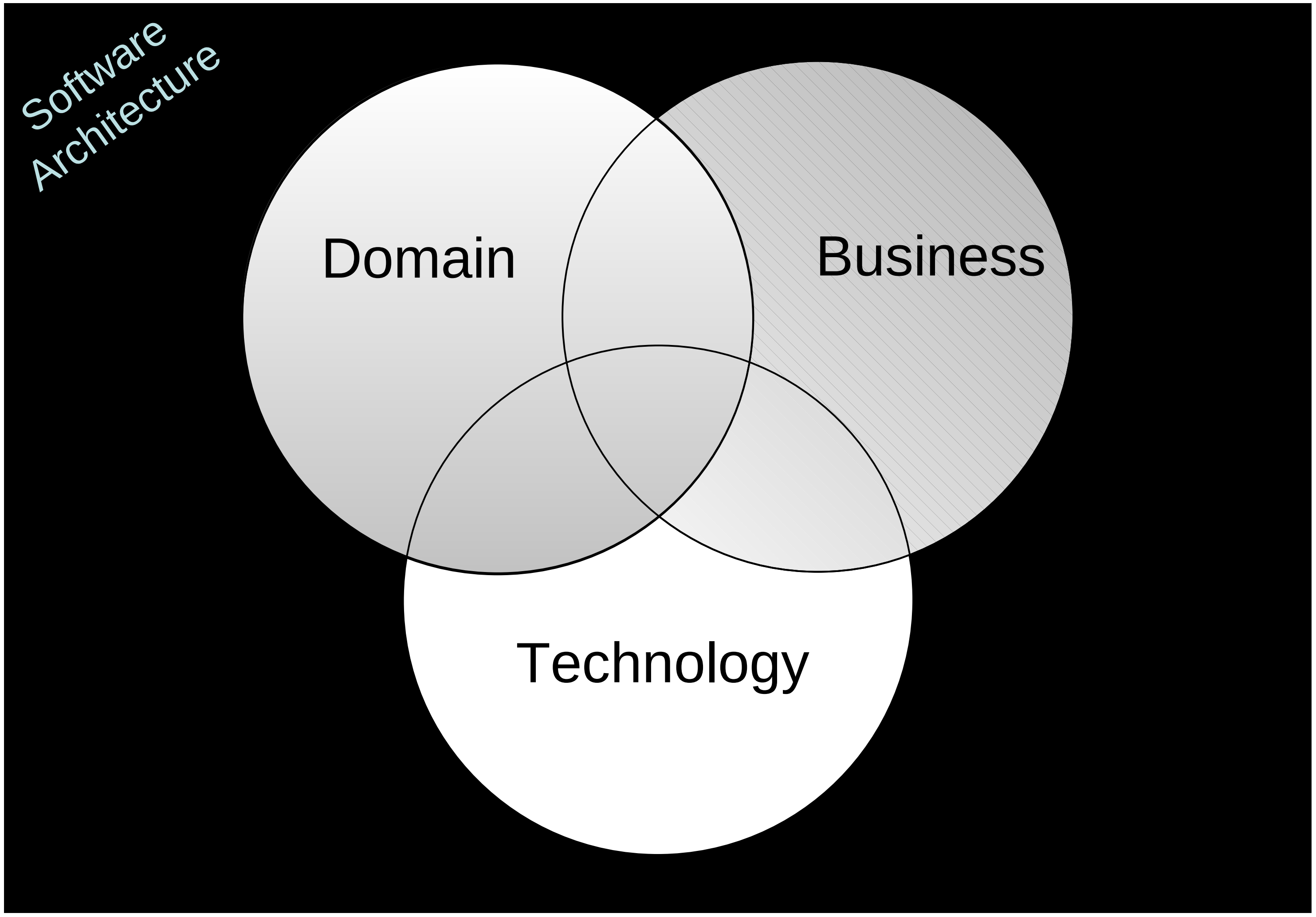
xADL 2.0 Under the Lampposts



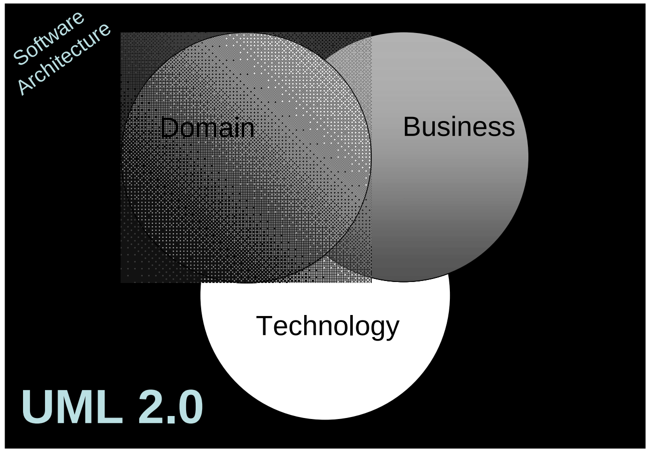
xADL 2.0 Under the Lampposts

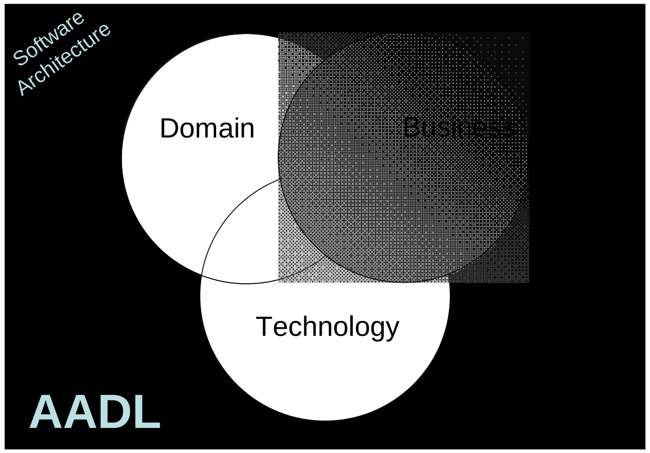


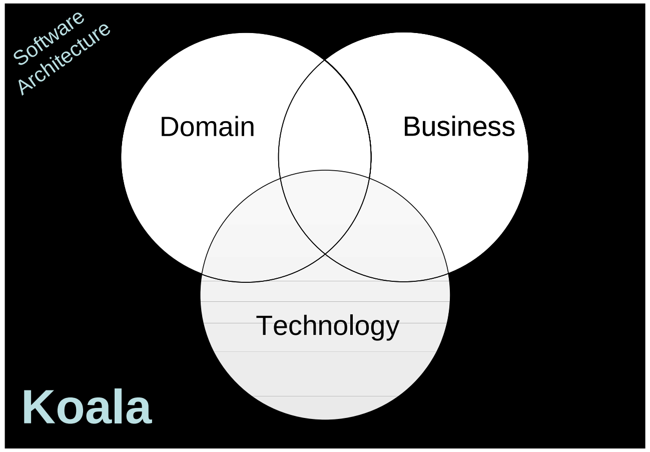
xADL 2.0 Under the Lampposts

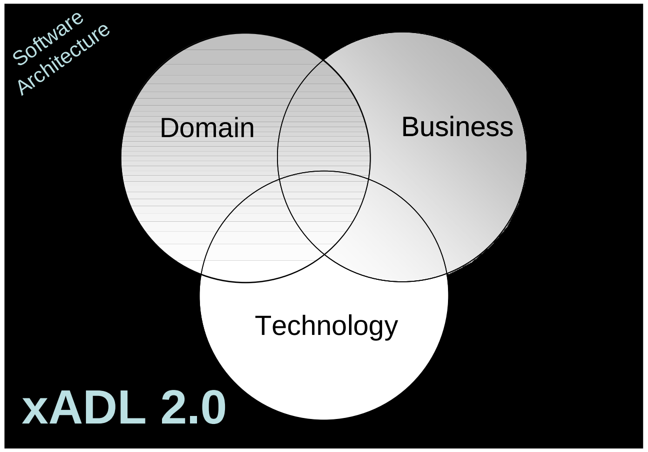


2G ADLs Side-by-Side









Some Observations

- Architecture embraces many concerns
- More mature and successful ADLs incorporate concerns from 3L
- Multiple views are a must
- No single set of modeling features is sufficient for every project
- Extensibility is a key property of ADLs
- Tools are often as important as notations

Questions