Domain Modeling and Software Process Management

Model-Based Software Management

Topics

- **Historical Trends in Software**
  - What is software? Where's it been? Where's it going?
- **Domain Modeling and Active Models**
  - What's been missing
- **Software Process Domains**
  - What we know, what we need to know
- **Honeywell's DOME Tool**
  - Not your regular CASE in point… and it's free!
- **The Future**

What is "Software"?

Software is not a *"Product"*...
...it is a *Medium in which we store knowledge*

What is "Software"?

- *A simple demonstration using a Hacking Model shows that, certainly "code" is not the product*
- *Interestingly, there are two distinct outputs from this process*
- *The nominal "product" (the system) is almost invariably corrupted*

Historical Trends in Knowledge

- ***Knowledge Storage Media:***
  - –4 Billion Years Ago: DNA
  - –2 Million Years Ago: Brains
  - –1 Million Years Ago: Hardware
  - –600 Years Ago: Books
  - –50 Years Ago: Software



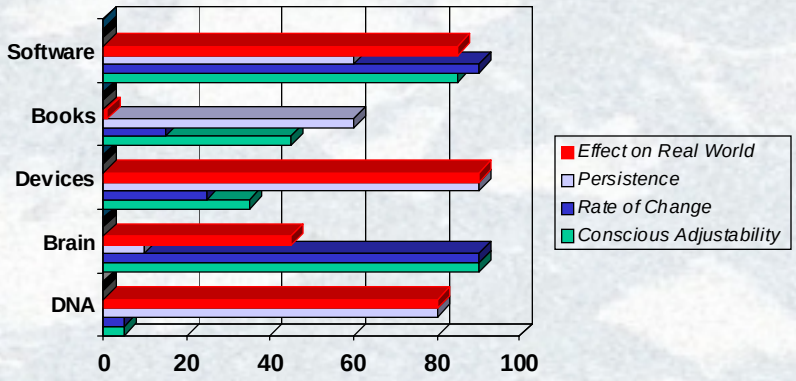


Historical Trends in Knowledge

If we look at the characteristics of each knowledge storage medium, we can see a logical progression. We can also see why software is becoming so ubiquitous.

Storage Medium	*Medium Persistency*	*Update Frequency*	*Ability to affect the outside world*
DNA	Very persistent	Very slow	Through physical medium, limited
Brain	Volatile	Quite fast	Through physical medium, not limited
Hardware Design	Very Persistent	Slow	Limited to specific design
Books	Persistent	Quite slow	None
Software	Persistent	Fast	Through machines

Historical Trends in Knowledge



Historical Trends in Knowledge

These stages of knowledge storage evolution can be characterized as follows:

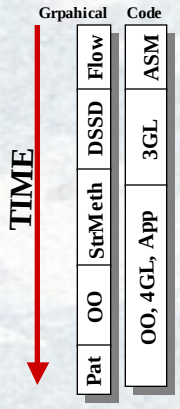
- ***DNA:*** ***unintentional knowledge storage, and application***
- ***Brain:*** ***intentional knowledge storage***
- ***Devices:*** ***intentional knowledge application***
- ***Books:*** ***portable intentional knowledge***
- ***Software:*** ***intentional knowledge development, storage, transport and application***

Historical Trends in Knowledge

- ***Basically, if we know something, there are three places we can put the knowledge…***
  - **Brains**
  - **Books**
  - **Software**

Historical Trends in Software

- ***Persistent trend toward abstraction***



- **Simple linear function**
- **Functional structure**
- **Modularity of function**
- **Data and data structure**
- **Encapsulation of function, data, and interface**
- **Architecture**
- **Event and state modeling**
- **Association of data, function, and state**
- **Abstract attributes**

Historical Trends in Software

- *Software and Systems Development is a system to which systems principles can be applied.*
- *We can use the same*
  - **architecture concepts,**
  - **abstraction,**
  - **modularity, and**
  - **encapsulation ideas**

  in the **process** *of building systems*

Historical Trends in Software

- *Here's a thought…*
  - **If software is not a product, then software development is not a *"product producing"* activity**
  - **If software is just a place we store knowledge, then software development is a *knowledge acquisition* activity**
  - **There are only three places to store knowledge: brains, books, and software**
  - **So the software developers' job is to take knowledge from brains and books (and sometimes software) and make it executable**

Historical Trends in Software

- *So where do we put the knowledge of how to do that?*
- *In what media do we put the knowledge of how to put knowledge into executable form?*

Domain Modeling and Active Models

The Systematization of Systematization

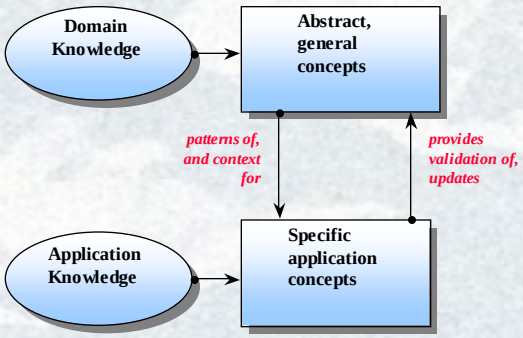
Domain Modeling

- *Also known as "Model-Based Software Engineering" (MBSE)*
- *Areas for development are "knowledge domains"*
- *The knowledge in domains has characteristic structure*
- *The domains can't be understood without understanding the structure*

Active Models

- *Given the right environment, domain models can be "active" allowing*
  - Interaction with the environment
    - Data collection into the model
    - Control of the environment
  - Interaction with other domain models
  - Production of artifacts
    - Containing the domain knowledge
    - Applying the domain knowledge

Domain Modeling



The abstract level provides general patterns and the context for interpreting the detailed level

The detailed level provides validation of the general level, and a way of "refreshing" the model knowledge at the higher level

Active Models

Levels of abstraction·

Abstract/Meta Models: Structure/Rules of Domain Knowledge	Highest level is the *kind* of knowledge needed.
Specific Model: What knowledge for Domain	What the system of interest needs.
Translation Rules: How to build a system in the Domain	How to produce a system that implements the knowledge
Tool: Apply knowledge of Domain	Use the knowledge in a tool

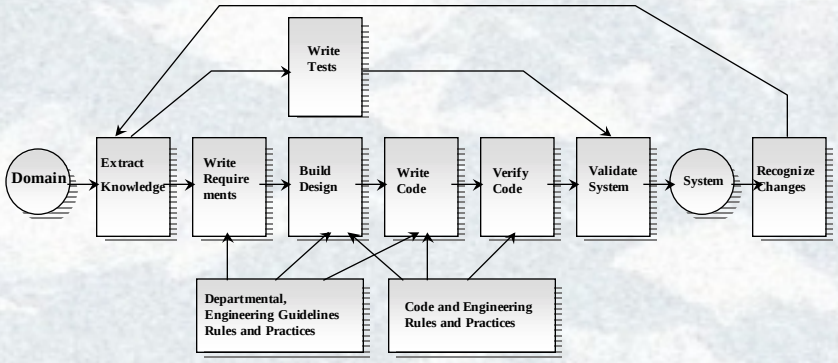
Software Process Domains

Software Process Domains

- *The software development arena is just full of knowledge domains*
- *The application of this knowledge has historically had some shortcomings*

Software Process Domains

- *The "Traditional" View of Process*



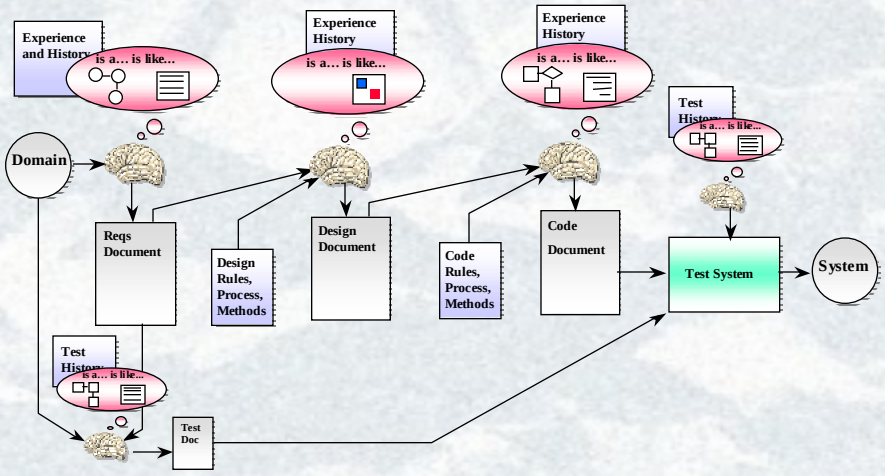
Software Process Domains

- *The "Traditional" View of Process*
  - This model is classic Taylor
  - The "product" is progressively manipulated on an "assembly line"
  - Requirements, Design, Coding and Implementation knowledge are folded into the product until it's "complete"

Software Process Domains

- *This is a stateless "flowchart"*
- *It doesn't show:*
  - the "data", only the "actions"
  - the "processor"
  - the "processor" knowledge basis
  - any domain rules; it is non-specific
  - process control criteria, etc., etc.

Software Process Domains

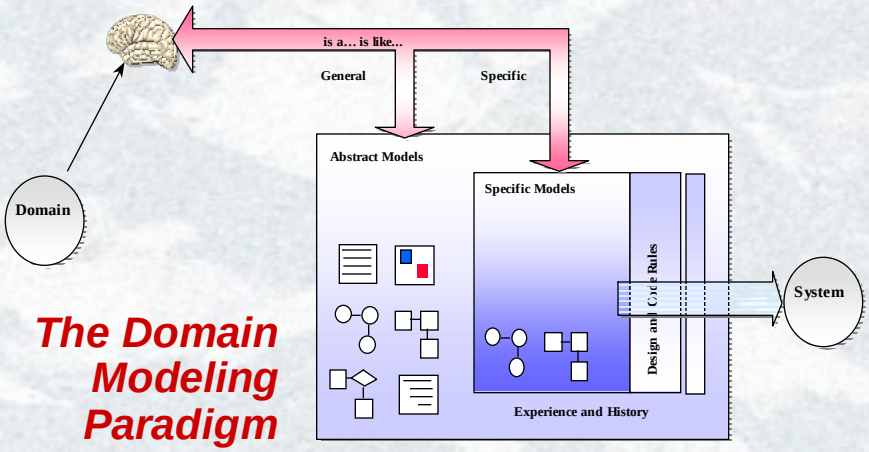


Software Process Domains

The processes, and "deliverables" of the development lifecycle are just places where *people* perform analytical and design functions

- ***Question:***
  - **where are the "rules", experience, history, heuristics, metaphors, and models that allow this work to be done?**
- ***Answer:***
  - **in peoples' brains, and (if you're lucky) in documentation**

Software Process Domains



The Domain Modeling Paradigm

Software Process Domains

- *In the domain modeling concept:*
  - **the models, rules, procedures, processes, and even management activities, are resident in the tool set *software*.**
- *Traditional CASE tools are* **passive**
- *Domain modeling tool are* **active**

Software Process Domains

- *Domain Modeling Tools have been pitched at the same target as classical modeling tools*
- *The concept of Active Domain Modeling applies to any domain*
- *There are lots of domains in software development*

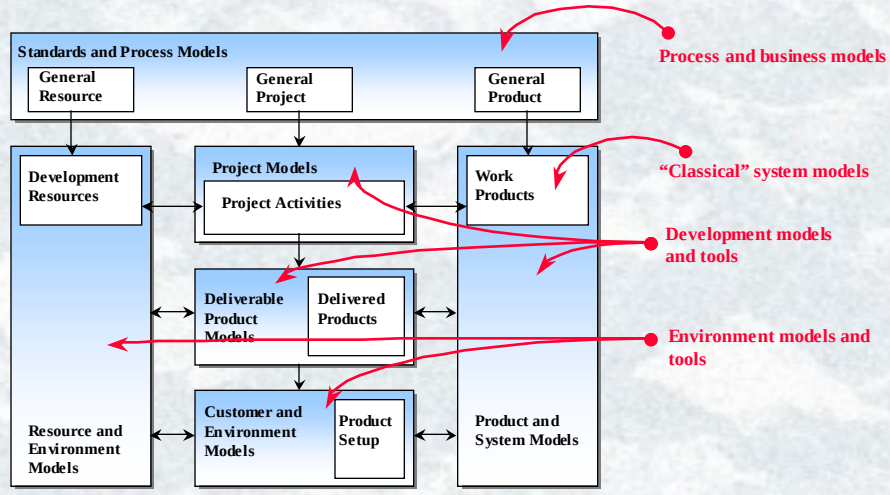
Software Process Domains

Problem.....	*what about the problem, the need, the customer?*
Product.....	*what about the product, how to design it, build it, extend it?*
Process.....	*what about building things in general, and this thing in particular?*
Development....	*what tools, libraries, methodologies, architectures, standards, are suitable?*
Implementation.	*what does it take to actually make it work in the real world?*

Software Process Domains

- *The Software Process Domains look like they could be a fertile area in which to research*
- *There is much competition in classical "software engineering" modeling, much less in process management.*

Software Process Domains

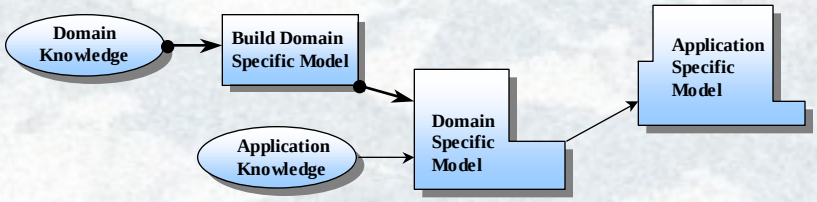


Software Process Domains

- *A "project" is the conjunction of:*
  - a set of process standards (= how we work)
  - a collection of resources (= what does the work)
  - a set of work products (= where the work is done)
- *"Deliverable products" are*
  - a subset of work products
  - additional bundled products and services
- *Some implementation knowledge exists in the customer domain*

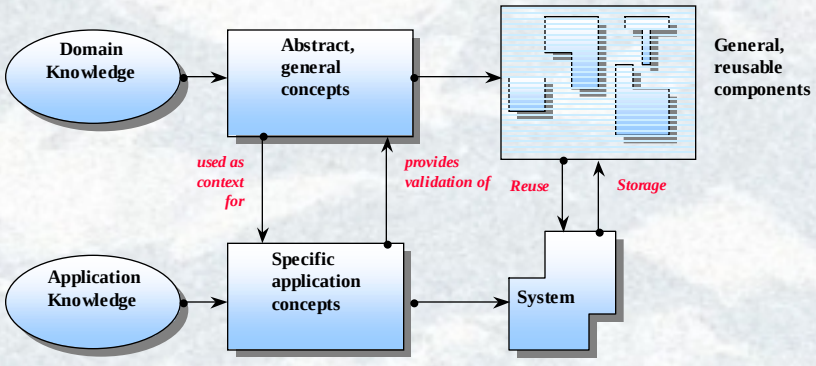
Software Process Domains

- **Building Domain Models**



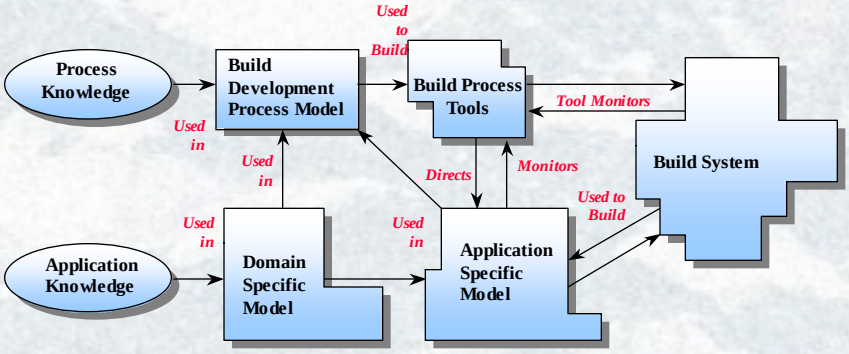
Software Process Domains

- **Using Domain Models**



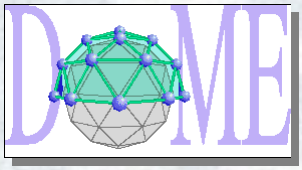
Software Process Domains

- *Building Process Models*



Honeywell's DOME Tool

An Example of a Domain Modeling Tool



Honeywell's DOME Tool

- *Developed at Honeywell Technology Center (HTC)*
- *HTC developed many tools over many years and noticed similarities*
- *Asked "what if we considered tool building to be a domain?"*
- *Asked "what if we built a tool that builds tools?"*

Honeywell's DOME Tool

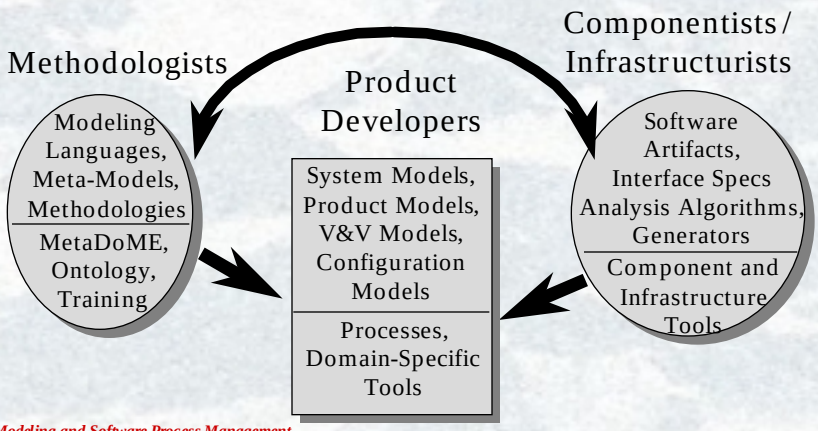
- *Used extensively by*
  - US Armed Forces (eg. MICOM)
  - Aircraft Primes
- *HTC had no charter to market tool*
- *Early 1999 decided to "do a GNU"*
- *Now free and Open Source*

Honeywell's DOME Tool

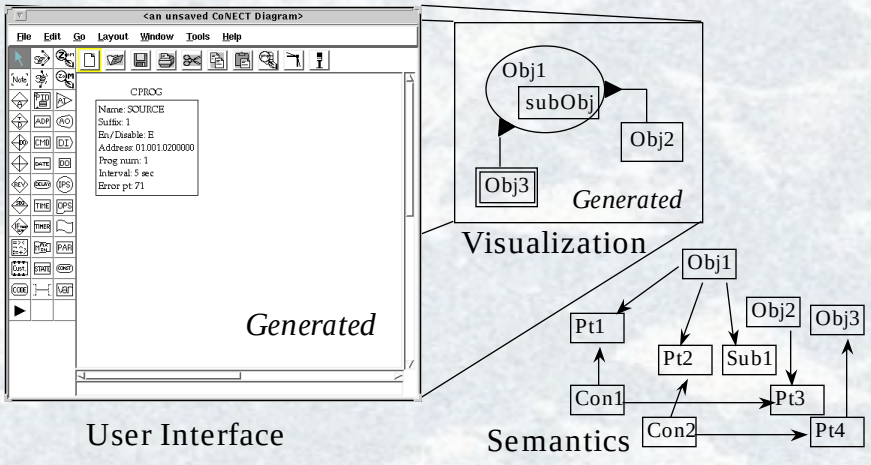
- *A Design Processor*
  - Supports complex graphical, table, or code syntax
  - Dozens of notations currently supported, infinite number possible
  - Standard properties on all elements defined in the *metamodel*
  - Component shelf (for reuse)

Honeywell's DOME Tool

- *General Paradigm*



Honeywell's DOME Tool

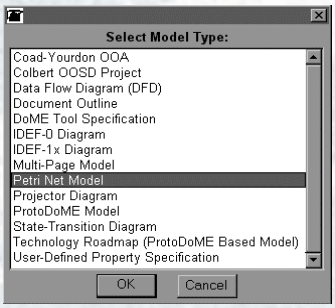


Honeywell's DOME Tool

- ***Methodologists can build language models***
- ***Domain experts can use language models to build domain models***
- ***Domain models can be used to build domain-specific tools***
  - **Powerful notations can be developed for a model in minutes to days.**
  - **Executable analysis functions, simulation, environment execution and model transforms increase the model's value**

Honeywell's DOME Tool

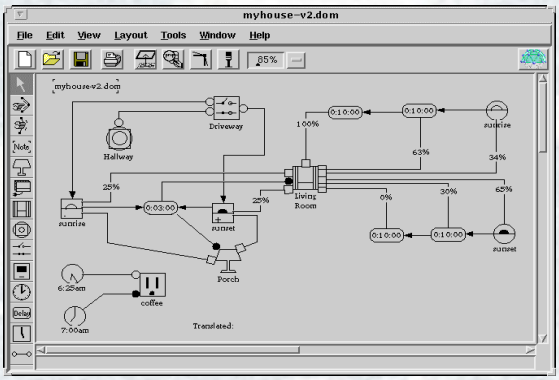
- *"Classic" Tools in DOME's toolset*



- DFDs
- State Models
  - STDs
  - Petri Nets
- Document Models
  - Outliner
- Code Generators
  - Projector
  - Alter
- Object Models
  - Coad-Yourdon
  - Colbert
- Process Models
  - Technology Roadmap
- Model Management
  - Multi-Page Model
- Military Models
  - IDEF
  - IDEFx

Honeywell's DOME Tool

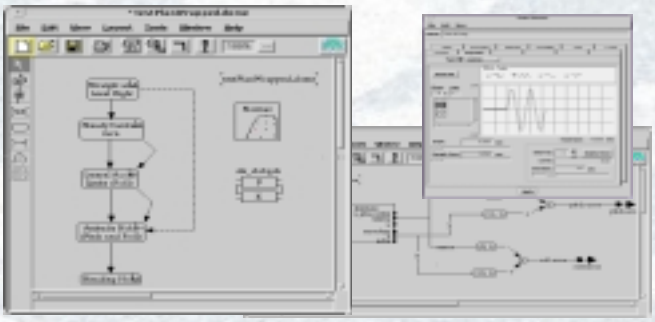
- *Domain-specific tools*



- X-10 Home Automation Model
- Written by one of DOME's authors
- Can control a house using X-10 protocols

Honeywell's DOME Tool

- *Proprietary Domain Tools*

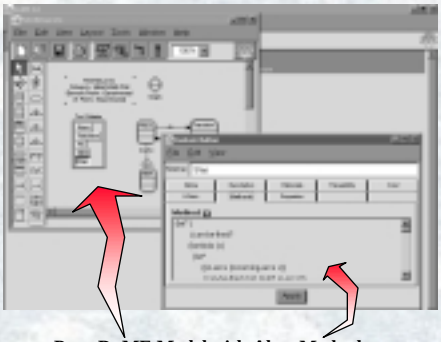


Flight Control test tool set developed by McDonnell-Douglas

Honeywell's DOME Tool

Code Development

- Code can be written in several ways using the *Alter* programming language
- Alter can be used for
  - –programming I/O functions such as displaying, printing
  - –animation
  - –constraint checking, analysis



ProtoDoME Model with Alter Method Listing for Step Function

Honeywell's DOME Tool

- *The Challenge:*
  - Most of DOME's public tool set is *Domain Non-Specific (why?)*
  - *For this concept to take root, domain models must be built, shared, and built on*
  - *HTC has made DOME free to for this very purpose*
  - *How about Software Process?*

Honeywell's DOME Tool

- *Where to get it*

www.htc.honeywell.com/dome

Domain Modeling Tools

- *Other tools*
  - *Lincoln: Engineer/Hood/Toolbuilder™*
    www.ipsys.com
  - *Mark V Systems: ObjectMaker™*
    www.markv.com
  - *MetaCASE: MetaEdit+™*
    www.metacase.com
  - *Platinum: Paradigm Plus/ADvantage™*
    www.platinum.com
  - *Advanced Software Technologies: GDPro™*
    www.gdpro.com

The Future

The Future

- *Systems developers deal with models:*
  - **models in text,**
  - **models of requirements,**
  - **models of architecture,**
  - **models in code,**
  - **models in test sets**
- *Models, models, models...*

The Future

- *The content of the models represents our accumulated knowledge*
- *The results of all this modeling activity is translated, finally, into executable software, into code*
- *Finally, in executable software we have* **active knowledge**

The Future

- *But what if the models themselves were executable?*
- *What if we could translate our ideas and concepts into executables as quickly as we could think of the ideas?*

The Future

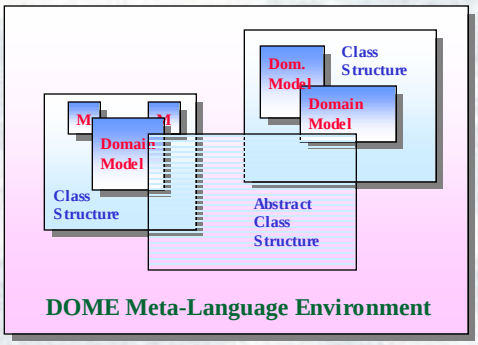
- *Process Tool Building*
  - **Active model requirements gathering**
  - **Environment control such as network management**
  - **Project management and management model interfacing to development**
  - **Tool interfacing either between DoME tools, or other proprietary or open products**
  - **Domain-specific code generation**

The Future

- **Meta-Languages**
  - A meta-language is a language that manages languages
  - With active Domain Modeling tools, we could develop domain-specific models and objects, which represent their area of knowledge
  - Using active meta-models to manage and integrate these domain-specific models could be one of the most effective ways of building systems in the future

The Future

- **Meta-meta Process Domain**



The Future

The basic economic resource—"the means of production," to use the economist's term—is no longer capital, nor natural resources (the economist's "land"), nor "labor."
It is and will be knowledge.

Peter F. Drucker
Post-Capitalist Society. P.8
(author's italics)

"Whereas at one time, the decisive factor of production was the land, and later capital…
...today the decisive factor is…
…*Knowledge*."

Pope John Paul II
Centesimus Annus 1991
(Pontiff's italics)

The Future

- *These different sources have arrived at the same place: knowledge is the asset of the future*
- *We can store knowledge in three places: brains, books, and software*
- *Our job is inescapably to put knowledge into its executable medium, into software*
- *The question is: where do we put our knowledge?*
- *We must find a way to put it into an active medium*

The Future

- *It is cliché to equate the "Information Revolution" with the "Industrial Revolution", and there are parallels*
- *However something is usually missing from the metaphor*
- *The Industrial Revolution did not occur when we built steam engines…*
- *…it occurred when we used steam engines to build steam engines*

Summary

- *Historical Trends in Software*
- *Domain Modeling and Active Models*
- *Software Process Domains*
- *Honeywell's DOME Tool*
- *The Future*

Phil's email: armour@corvusintl.com

Corvus International Inc is a consulting and educational company specializing in the space between technology and the people creating and using the technology. Our byline is:

Systems, Psychology and Software

http://www.corvusintl.com