Inventing the future Business Programming Language

Inventing the Future Business Programming Language

- Inventing a new business programming language may be an act of insanity.
- But someone will eventually do it.

The future is now.
The nature of the thing we are programming is changing.

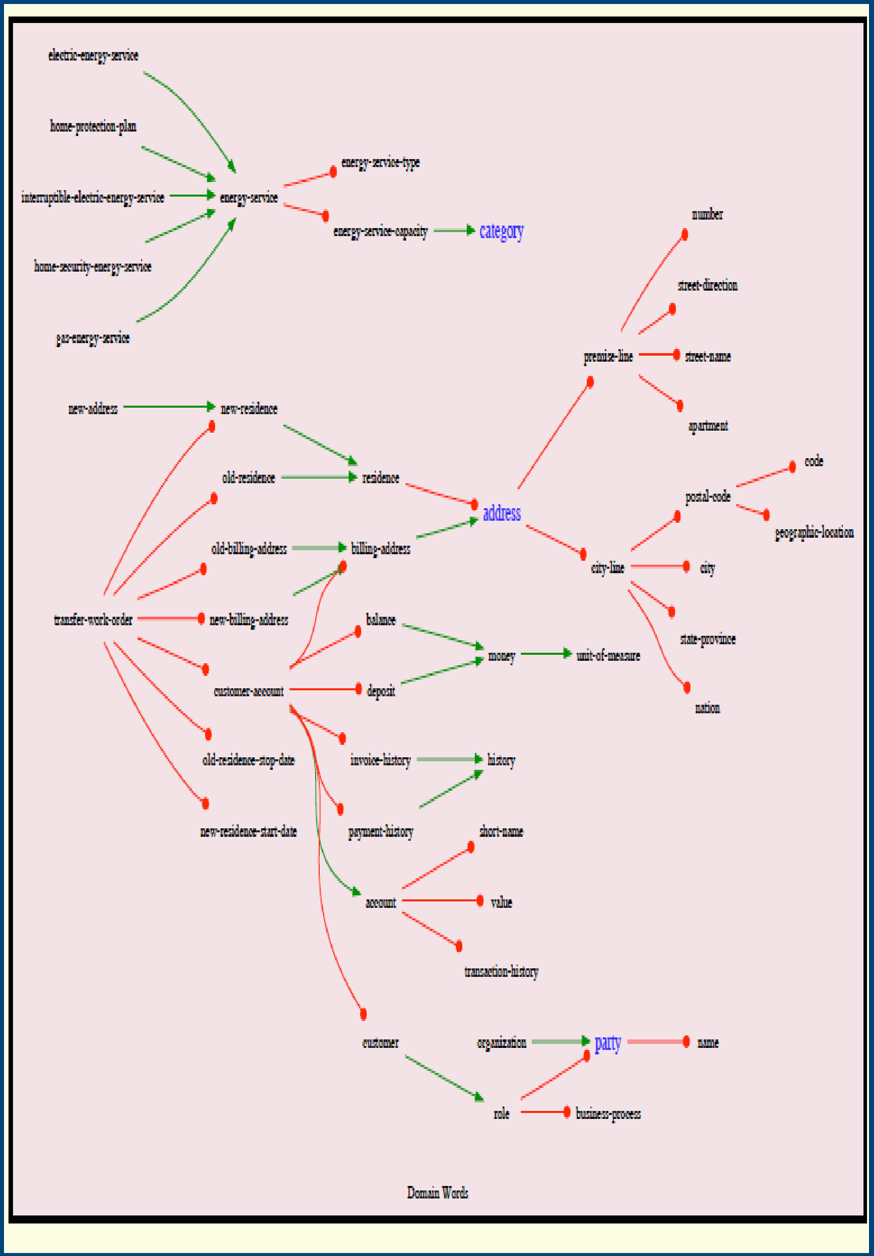
- Traditional programming languages are designed around the concept of a programmable calculator.
- But this model has always been largely irrelevant for business automation.
- Object-oriented languages extend the calculator model the statement syntax is still bound to an algebraic pattern.

Any major business process involves a network of collaborating agents

Instead of Algebra
– use natural language syntax.

- Follow the conventions of written natural language.
- Use a syntax that is convenient to the user. Avoid syntax that is not.
- A good notation should be readable and easily entered.

How do we make natural language work?



- The whole statement is the "method signature."
- The nouns in the statement identify variables.
- Variables are assigned values from a white-board. (context relevant data store)_

If we are programming a network of collaborating agents, what kind of notations do we need?

- **Plan**: For each business goal, we need a plan.
- **Procedure**: For each action, we need a procedure.
- **Dictionary**: Identify nouns and their relationships.
- **Data**: Transport request/response messages.

- **Dialog**: Provide the client / user interface.
  - Respond to client questions and commands.
  - Client invokes goals / actions.
  - Dialog may also be informational.
- **View**: Provide document metaphor.
  - Fill-in-the-blank forms. (Render as HTML form ...)_

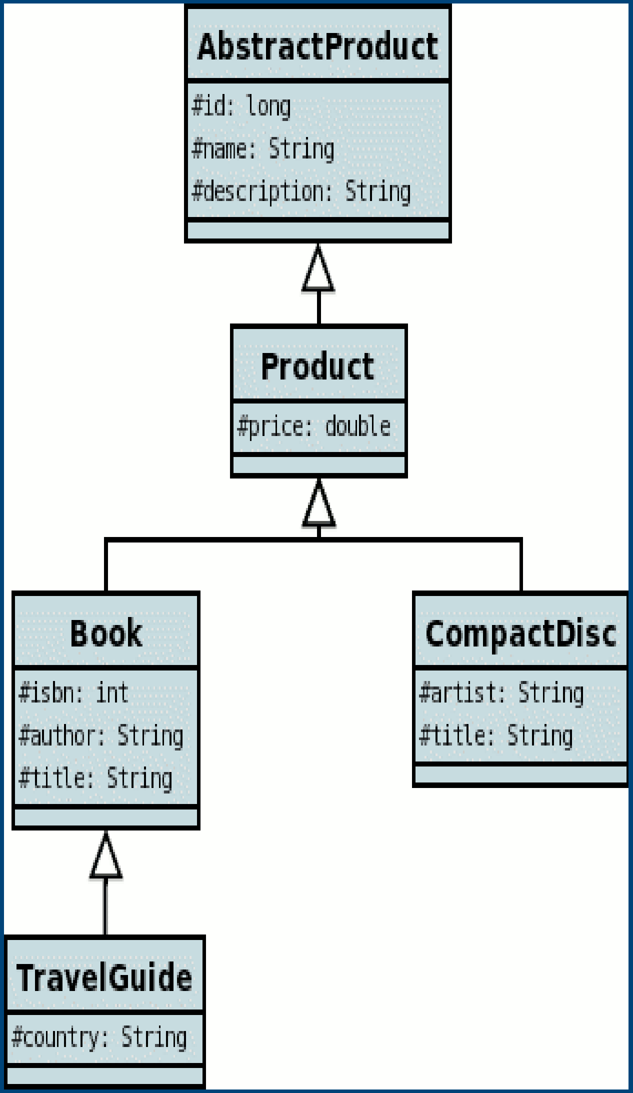
Nouns identify variables.
Nouns identify entities, attributes.

Dictionary: Energy service vocabulary.

- Site attributes include type, address, route.
- A site has a collection of services.
- Service attributes include type, meter, remote-switch.
- A service is a product.
- Meter attributes include type, configuration, GPS.
- A meter is a distribution-asset.
- A distribution-asset is an asset.

This is a dictionary frame. Each dictionary frame identifies entities, attributes, super-types, collections, and categories (enumerations).

How do we provide inheritance?



- Dictionary frames include "... is a ..." statements.
- These statements identify super-types.
- The statement matching process enables subtype nouns to be used in the slot for a super-type.

How do we enable specialization?

- Two different action statements
  may have a signature that only differs by the subtype / super-type in the same word slot.

- When matching a reference to a definition,
  the interpreter prefers the subtype version
  to the super-type version.

Naturally, natural language notations enable internationalization.

- There is a small number of keywords and a small number of domain-specific statement patterns. These can be replaced with equivalents from most European languages.
- I do not have the expertise to address the question of other languages.

Work-flow is the essence of business process automation

Task: Start electrical service for customer at site.

Post-Condition: Electrical service at site billed to customer's account.

Preconditions:

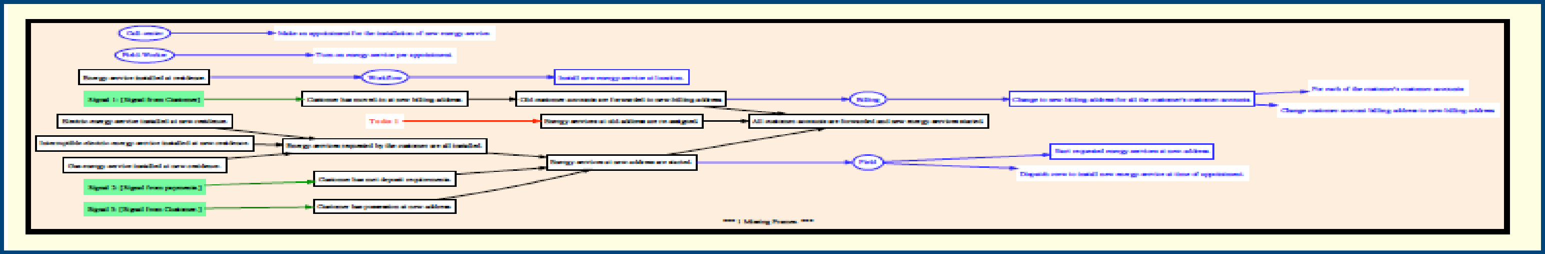
- Customer's account approved for credit.
- Electrical service installed at site.
- Electric meter installed on service.

Action:

- Field Service: Turn-on the electric service.

This is a task frame. Each task frame describes a goal, a set of precedence relations, and a transitional action.

Chain task frames into plan trees.


Call center
Make an appointment for the installation of new energy service.
Field Worker
Turn on energy service per appointment.
Energy service installed at residence.
Workflow
Install new energy service at location.
Signal 1: [Signal from Customer]
Customer has moved-in at new billing address.
Old customer accounts are forwarded to new billing address.
Billing
Change to new billing address for all the customer's customer accounts.
For each of the customer's customer accounts
Change customer account billing address to new billing address.
Electric energy service installed at new residence.
Tasks 1
Energy services at old address are re-assigned.
All customer accounts are forwarded and new energy services started.
Interruptible electric energy service installed at new residence.
Energy services requested by the customer are all installed.
Gas energy service installed at new residence.
Energy services at new address are started.
Field
Start requested energy services at new address.
Dispatch crew to install new energy service at time of appointment.
Signal 2: [Signal from payments]
Customer has met deposit requirements.
Signal 3: [Signal from Customer]
Customer has possession at new address.
*** 1 Missing Frames ***

- A plan tree is formed by back chaining from a goal through the task-frame preconditions.

- The action statement in a task frame may be executed when the preconditions are met.

- Enabled tasks on the “branches” of the tree may be processed concurrently.

- Parallel processing is implicit in the notation.

Agents are assigned actions.

Role: Field Service.

Action: Turn-on the electric <u>service</u>.

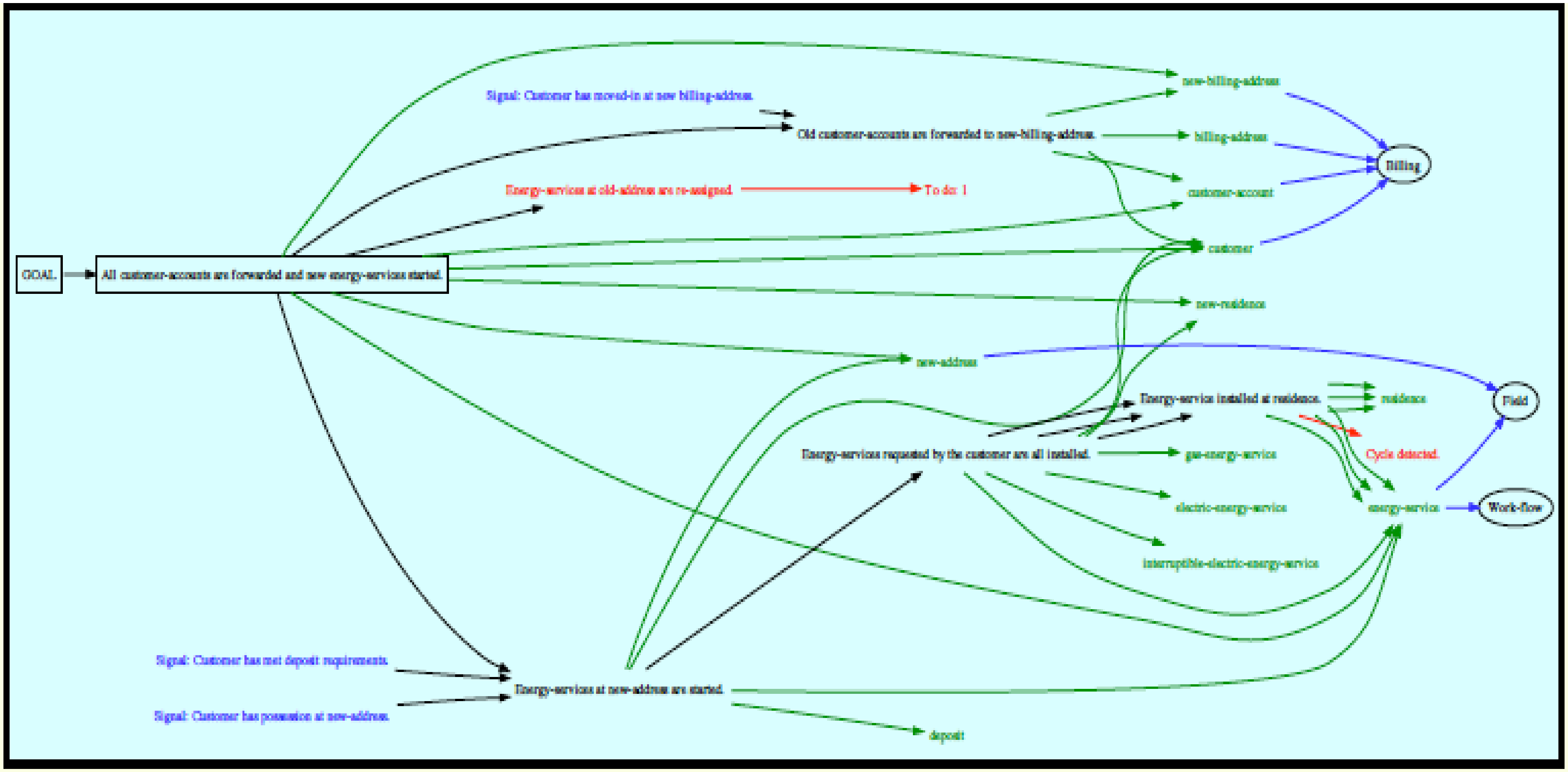
- If <u>service</u> <u>meter</u> has remote-power-switch:
  - AMI: Signal the <u>meter</u> to turn-on the <u>service</u>.
- Else:
  - Dispatcher: Send a one-man <u>crew</u> to turn-on the <u>service</u>.

This is a role-action frame.

Each action frame describes a set actions that may be assigned to a role.

An role may delegate some steps to other roles.

Task frame ==> parallel process. Action frame ==> sequence.


GOAL.
All customer-accounts are forwarded and new energy-services started.
Signal: Customer has moved-in at new billing-address.
Old customer-accounts are forwarded to new-billing-address.
Energy-services at old-address are re-assigned.
To do: 1
new-billing-address
billing-address
customer-account
customer
Billing
new-residence
new-address
Field
Energy-service installed at residence.
residence
Energy-services requested by the customer are all installed.
gas-energy-service
Cycle detected.
electric-energy-service
energy-service
Work-flow
interruptible-electric-energy-service
Signal: Customer has met deposit requirements.
Energy-services at new-address are started.
Signal: Customer has possession at new-address.
deposit

The future is now ...

- In the future ...
  - The reception desk may be in a virtual world.
  - The user interacts with an avatar.

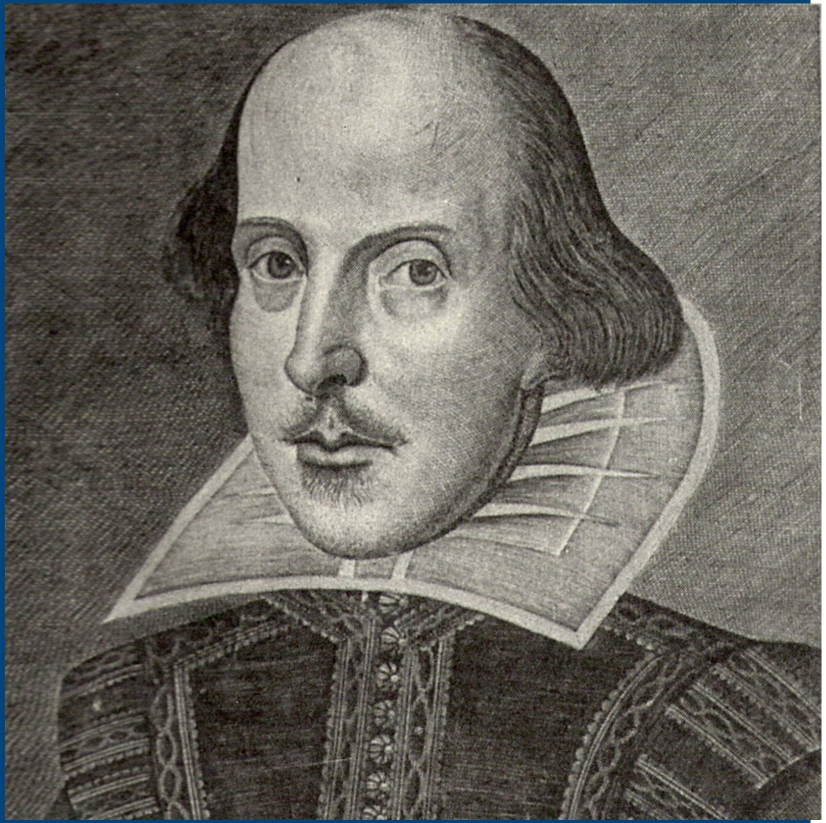
- Sensors are evolving.
  - Dialog logic should not be tied to specific sensors.
- The user interface may be rendered in multiple ways using multiple media.
  - Dialog logic should not depend on the rendering.

Programming the avatar - speaking part

Dialog: Hello yourself.
Context: Greeting.

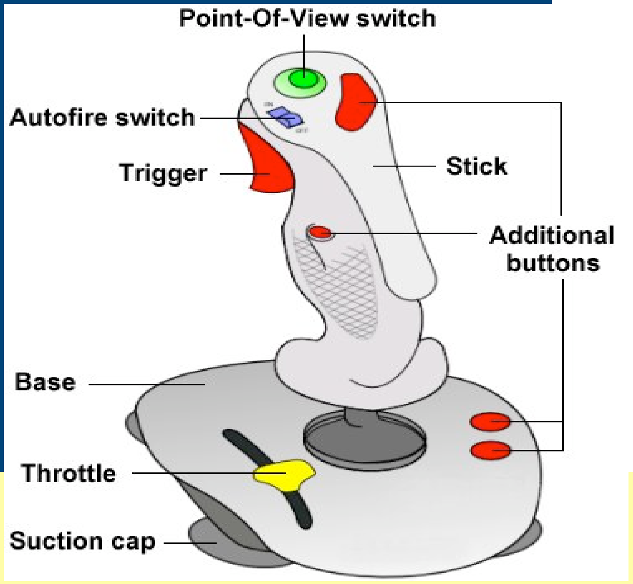
User: Knock, knock.
System: Who's there?

User: *.
System: * who?



This is a dialog frame. The notation is similar to a stage play, but pattern matching enables flexible dialogs (protocols) with variable input and output.

Responding to sensors ...



Dialog: Space flight simulator.

Context: Pitch and roll rate. (Translate from control stick.)_

U: Make the pitch and roll rates P and R degrees per second.

S: (Don't talk. Just do it.)_

. Pitch-Thruster: Produce P degrees/second change in pitch.

. Roll-Thruster: Produce R degrees/second change in roll rate.

Sensor inputs are mediated by a scribe (dialog agent) to isolate technology dependence.

Fill-in-the-blank ... (Electronic Forms)

View: User.

!!! User Preferences

E-Mail: [e-mail]   Name: [name]

Digest frequency: [digest-frequency]

Self description: [description 5]

This is a view frame defining a fill-in-the-blank electronic form.

The notation is similar to wiki mark-up.

Fields are rendered according to type.

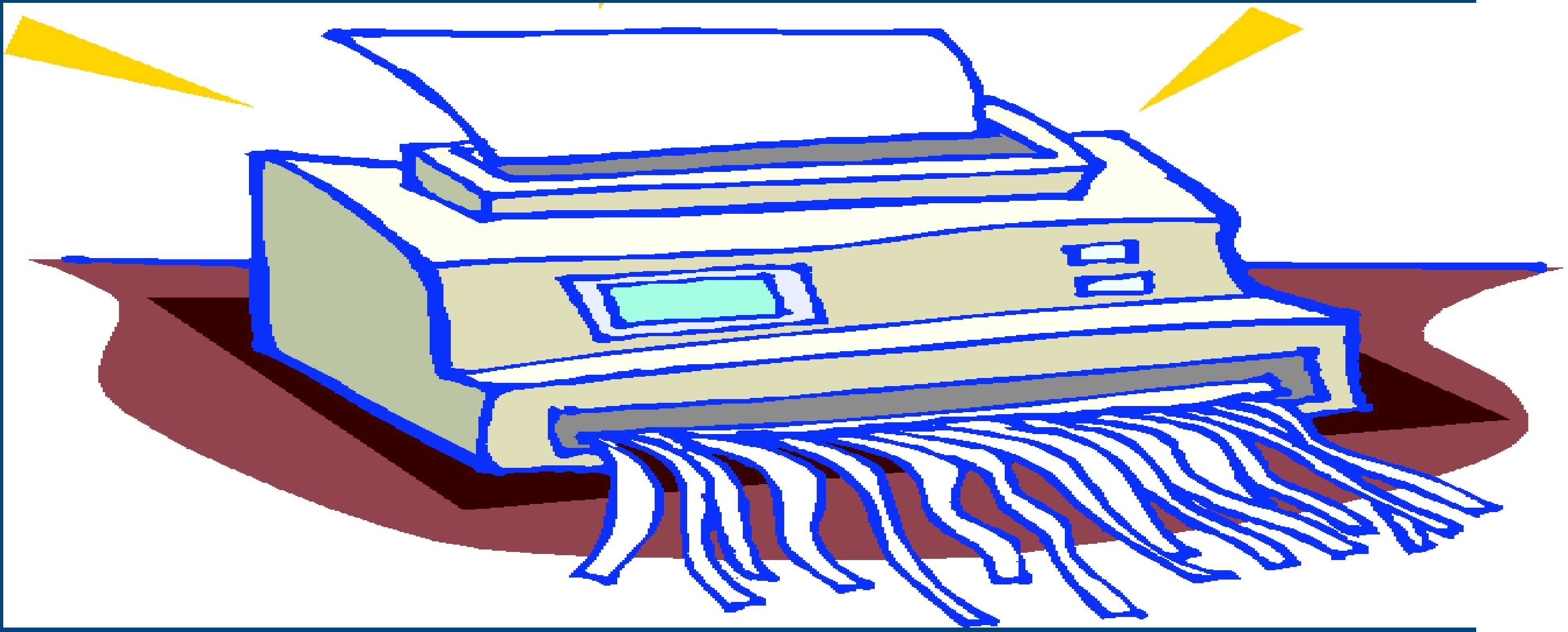
Example: The digest-frequency field is a category (enum) typically rendered as a drop-down list.

Don't waste time on data processing. Use built-in standard solutions.

- **Persistence**: Programmers spend too much time deciding what to store, when, and how.
- **Messaging**: Programmers spend too much time deciding what to send, when, and how.
- **Conversion**: Programmers spend too much time writing code that converts measurements between currencies and units of measurement.
- **Metrics**: Programmers spend too much time writing code for standard accounting and metrics.

Persist everything.
Purge per records retention policy.

- Don't waste time designing persistence.
- Change the problem from persistence to retention.



World Base … Temporal Database

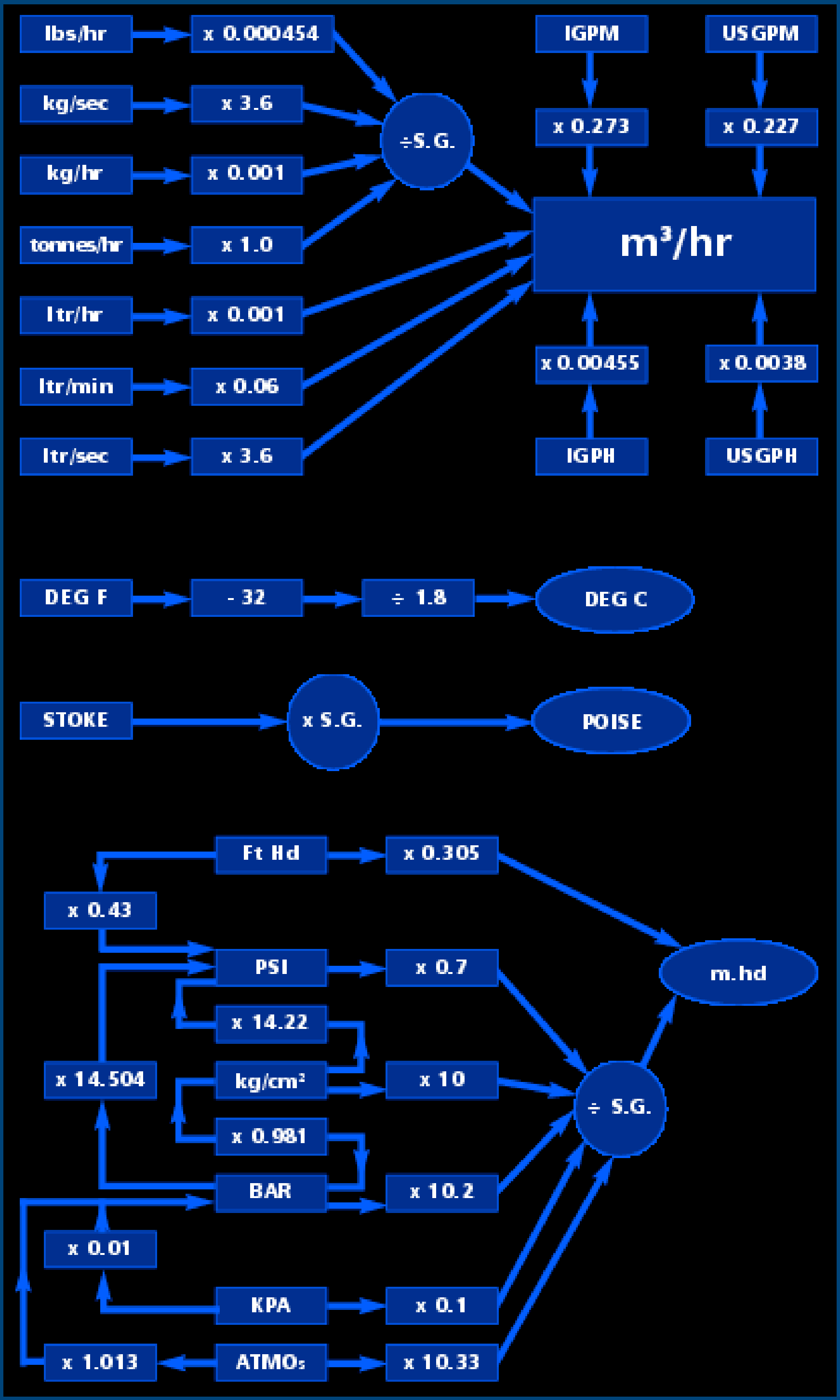
- Micro-fact: (entity, attribute: value, w5).
  - Smalltalk: (entity, attribute) -> (value, w5).

- Event (w5): (who, what, when, where, why).
  - Who: data source (user or client identifier)
  - What: statement causing the change.
  - When: timestamp (UTC)
  - Where: agent mediating the change
  - Why: job ticket

- History: (entity, attribute, time-period: value, w5).
  - time-period: (when start, when end).

Messenger
Do it once, one way.

- Actions are work-orders.
- Agents check-in / out with Resource Manager.
- Messenger provides store-and-forward asynchronous network.



In business data, numbers are measurements, not mathematical abstractions.

- Spurious precision should be obsolete.
- Track error propagation.
- Track unit of measure.
- Provide automatic conversion to preferred unit of measure.

… after the horse has left the barn

Resource Accounting

Job Ticket

- Assign a job ticket to each client request.
- Track resources consumed by the job.
- Sub-job (delegated task) rolls up to parent job.

Bookkeeper

- Accumulate resource usage by:
  - Job
  - Agent
  - Resource Pool
  - Action
  - Goal
  - Client
  - (custom dimensions)
- Track inventory.
- Pricing is a plug-in.

Resource Manager -- Plug In

For many enterprises, specialized resource scheduling is a **secret sauce**

How do you test a business process?

Simulations:

- Agents replaced by "sim-bots."

- Drive from dialog frames
  using standard regression testing tools.
- Drive from message history (requests to agents).

- Durations from history (bookkeeper).
- Durations from annotations.

- World-Base snapshot from real-world world-base.

The critical component is the open-source community process.

-
-

The critical component ***is the open-source community process.***

- The thing that distinguishes frameworks today is their **community process**.

- Think about how the community process makes these frameworks quite different in character:
  - CORBA ... UML ... MDA ... ebXML ... BPEL
  - Apache ... Mozilla ... JBoss
  - BSD Unix ... Linux ... Ubuntu
  - C++ ... C# ... Java ... Python ... Smalltalk
  - Second Life … Croquet Consortium

Extensions / Plug-Ins / Sub-projects

Infrastructure

- Smart IDE
  - Code Critic
  - Version Control
  - Ontology Merge
- Messenger
  - Hub
  - Gateway
  - Peer to Peer
- World Base
  - OQL
  - Extract (Simulation)_
  - Export (to RDBMS)_

Application

- Bookkeeper
  - Resource Pricing
  - Inventory Pricing
- Domain Dictionaries
  - Utility Vocabulary
  - Factory Vocabulary
  - Retailing Vocabulary
- Business Templates
  - Customer Relations
  - Order Fulfillment
  - Utility
  - School

Summary

- A programming environment is a set of notations and a run-time.
- Each notation presented here is derived from a well-known notation in another domain.
- The language design assembles the notations into a consistent and coordinated system.

- While the notations provide a better and more efficient expression of intent,
  the bigger break-through comes from eliminating repetitive data processing tasks
  from the programmer’s work-load.

How and Why

How

- Pick application area.
- Identify audience.
- Design notations.
- Try them out.
- Refine (iterate).

- Write the manual first.
- Write translator.
- Write the run-time.
- Iterate some more.

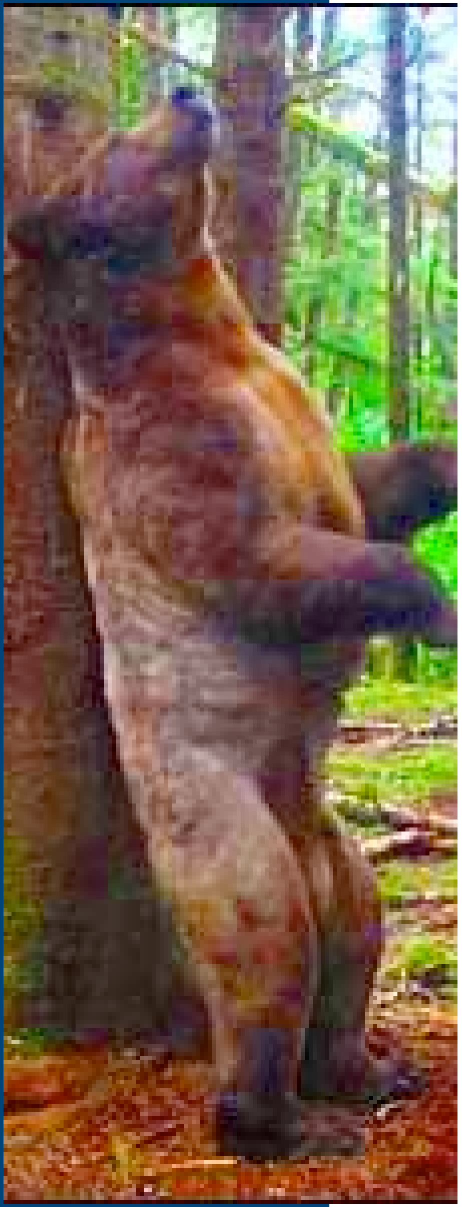
- Deploy a usable product

Why

- Scratch your itches
  - There ought to be a better way.
  - Tired of solving the same problem over and over

- New environments
  - new problems
  - new opportunities

- Advance the state of art
  - Identify problems
  - Find the root cause



Questions and Answers