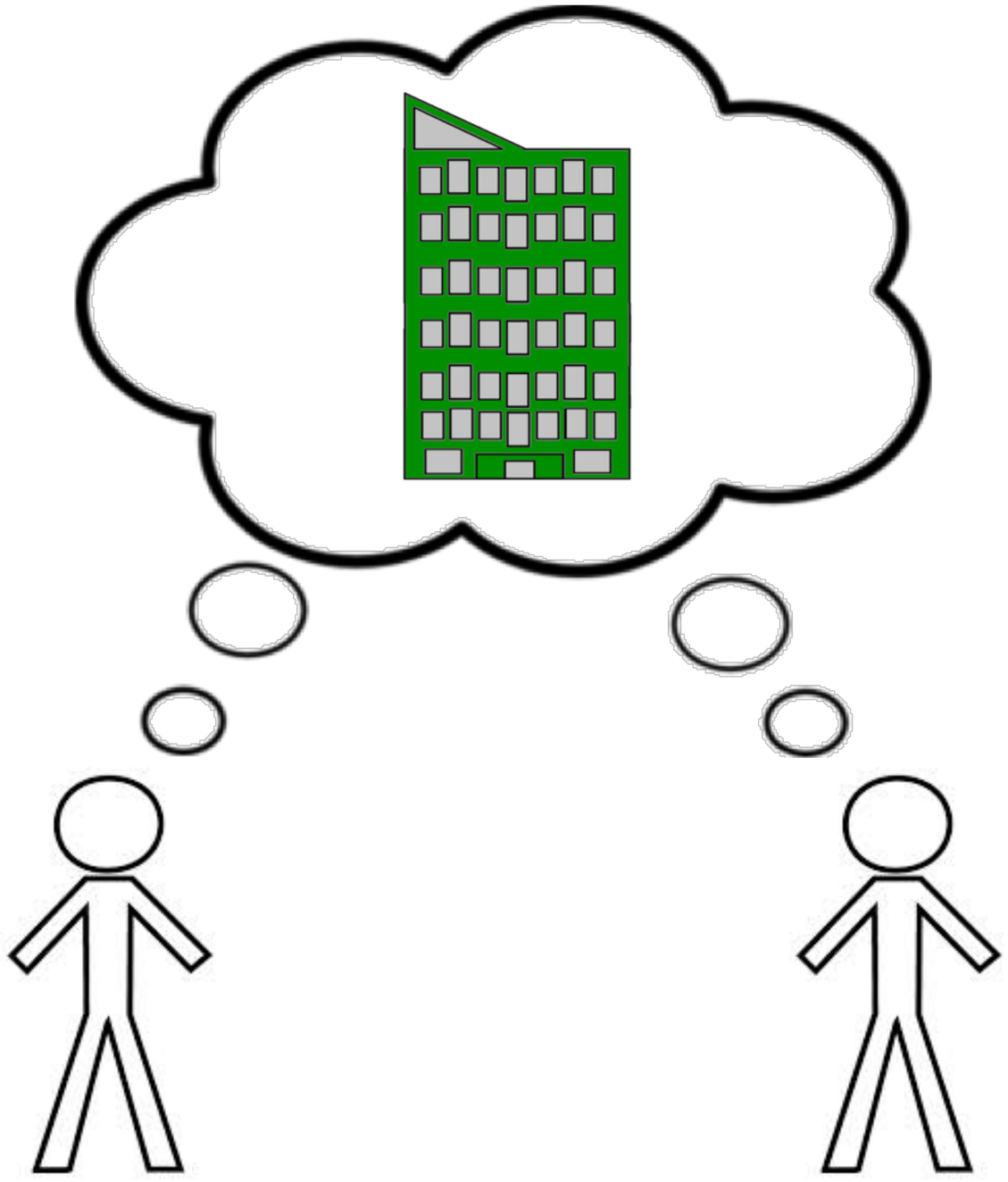
Retaking Rules for Developers

Ryan Brush
@ryanbrush
Strangeloop 2014

1977

Intelligent systems derive their power from the **knowledge they possess** rather than the specific formalisms used.

-Edward Feigenbaum



Domain knowledge

What the code looks like

Composable units of knowledge

Hypertension (severity == moderate)

⇨ recommend-diet-changes

Hypertension (severity == moderate)
Demographics (age > 60)

⇨ prescribe-bp-meds

1979


digital
DECSYSTEM 2020
digital
DECSYSTEM 2020
MEMBER
DEC PDP-10: KS-10 (DECSYSTEM-2020)
XKL
TOAD-1

On the Efficient Implementation of Production Systems

Charles L. Forgy

February, 1979

Carnegie-Mellon University
Department of Computer Science

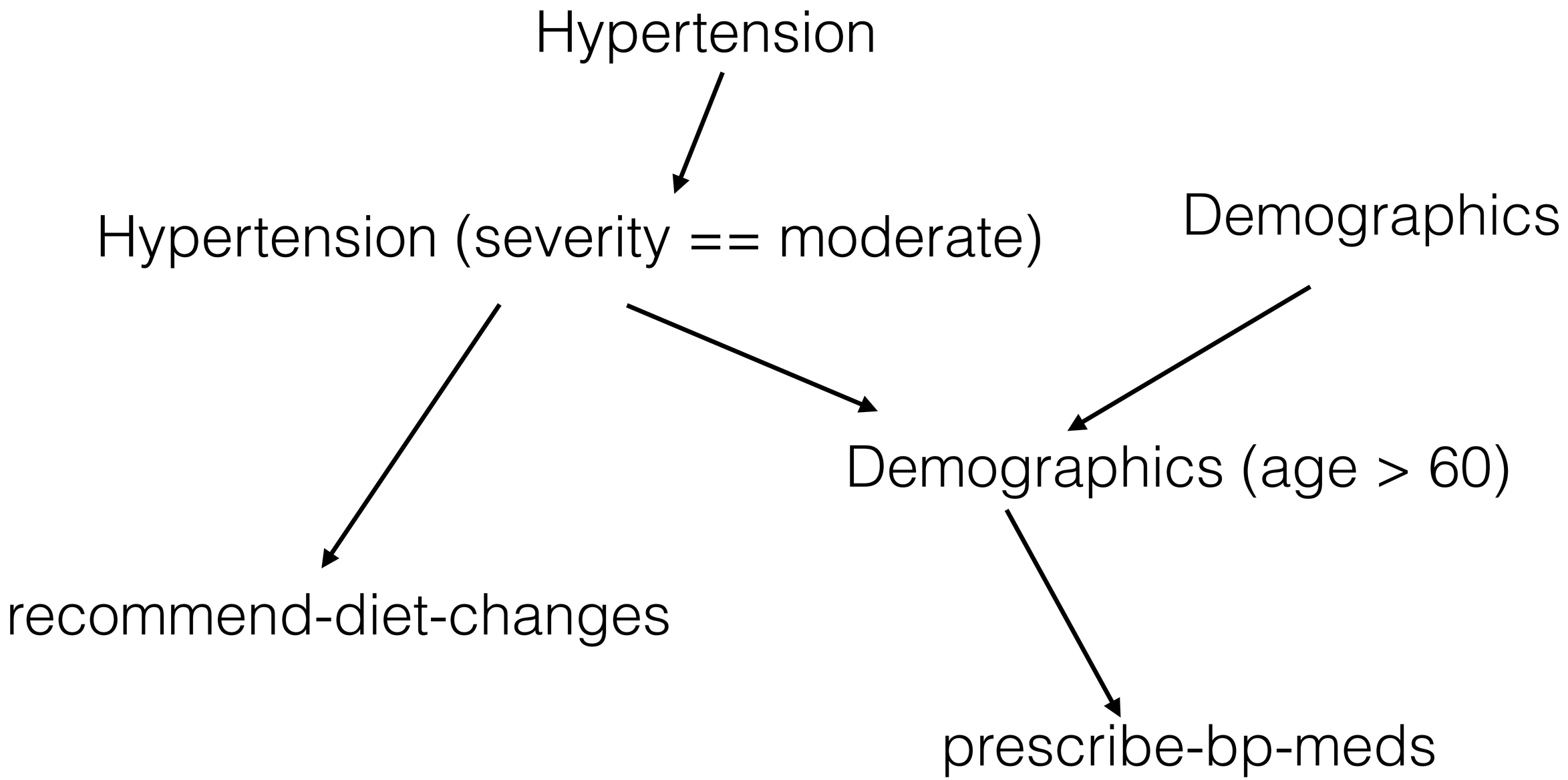
http://reports-archive.adm.cs.cmu.edu/anon/scan/CMU-CS-79-forgy.pdf

Hypertension (severity == moderate)

⇨ recommend-diet-changes

Hypertension (severity == moderate)
Demographics (age > 60)

⇨ prescribe-bp-meds


Hypertension
Hypertension (severity == moderate)
Demographics
Demographics (age > 60)
recommend-diet-changes
prescribe-bp-meds

1985

1985

1988

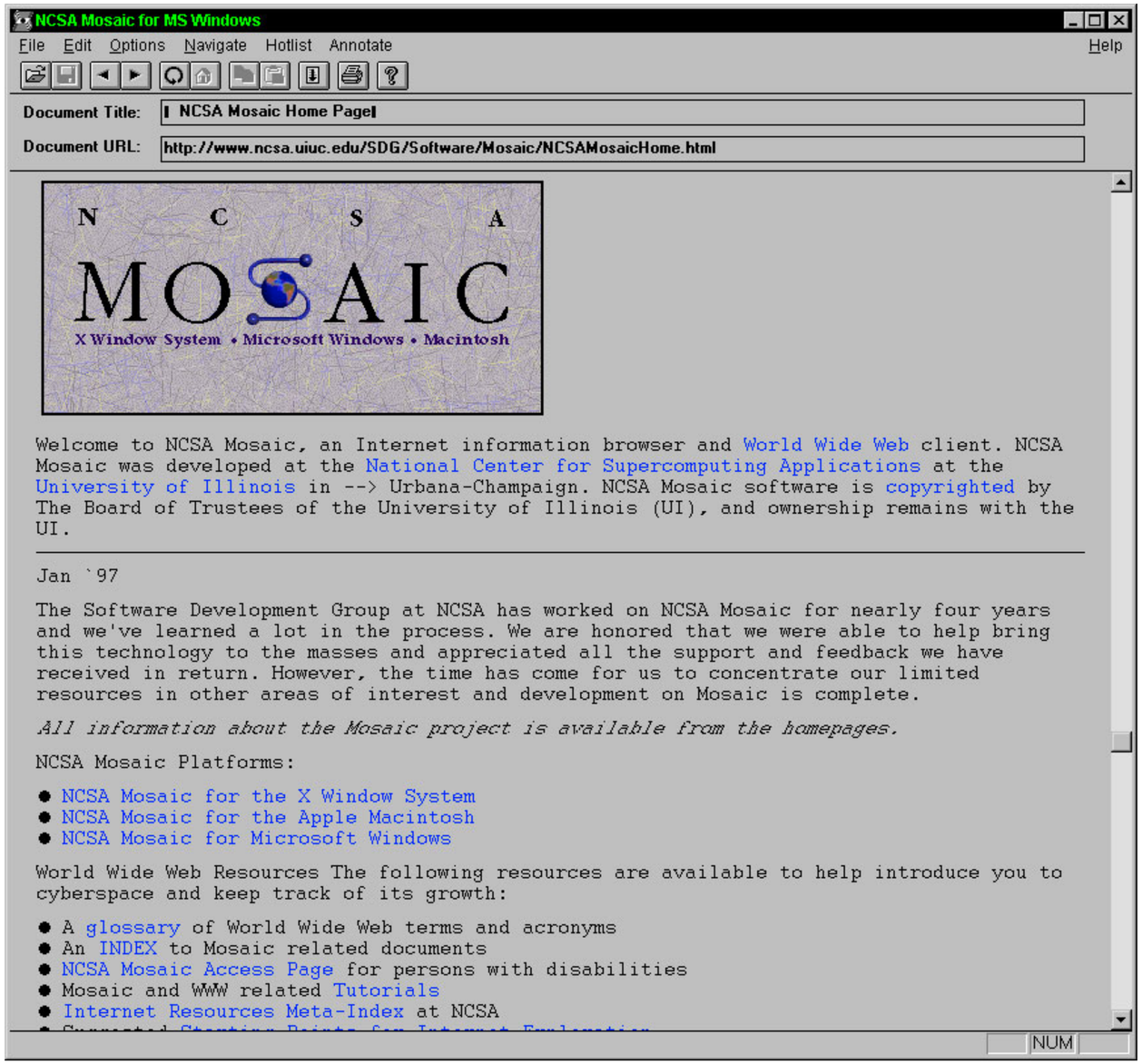
1988

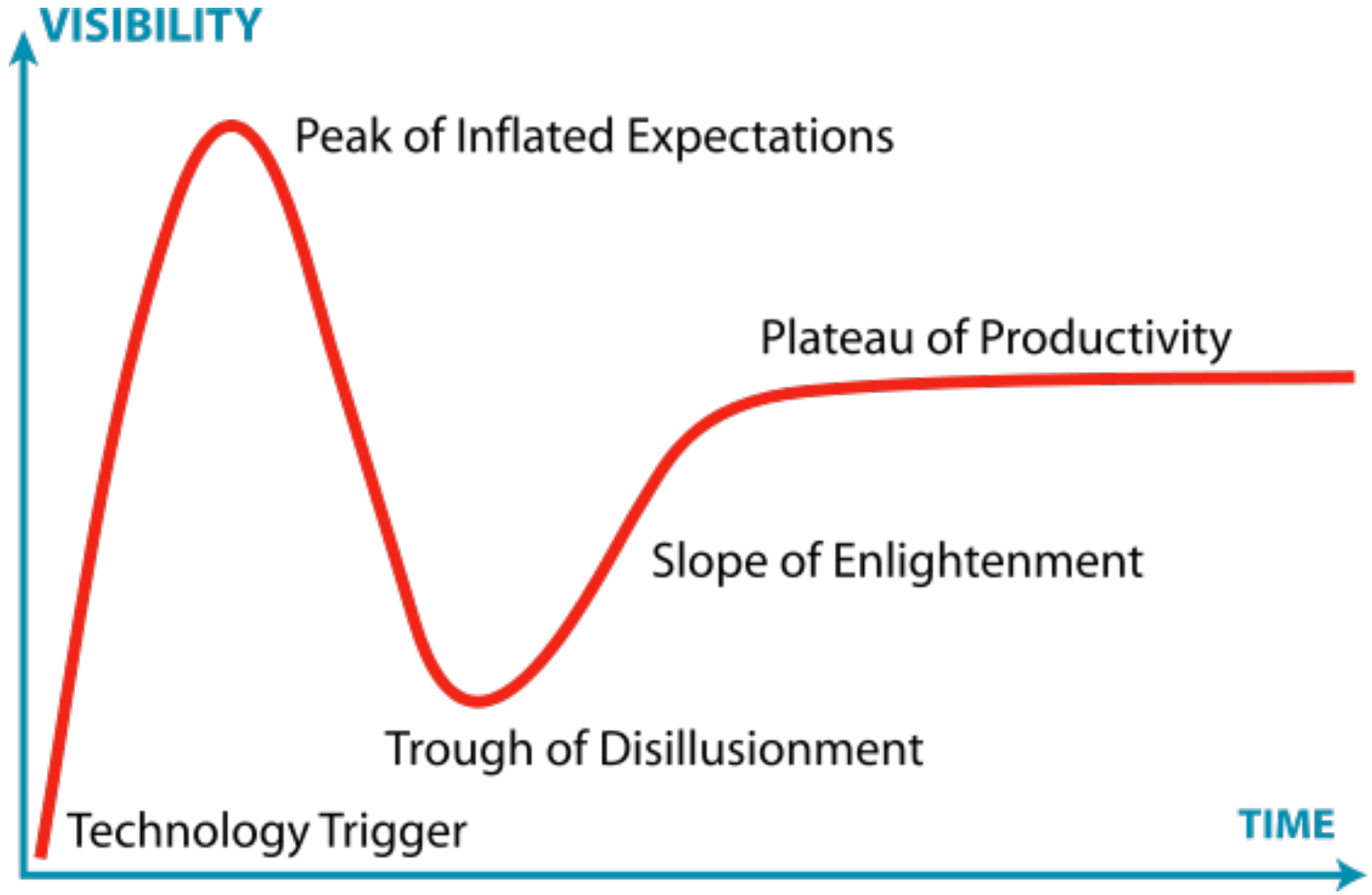
“Frankly, there were a lot of intellectually dishonest promises made.”

-Peter Gabel
Bring in the Expert
InfoWorld, October 1990

The AI Winter

1990s

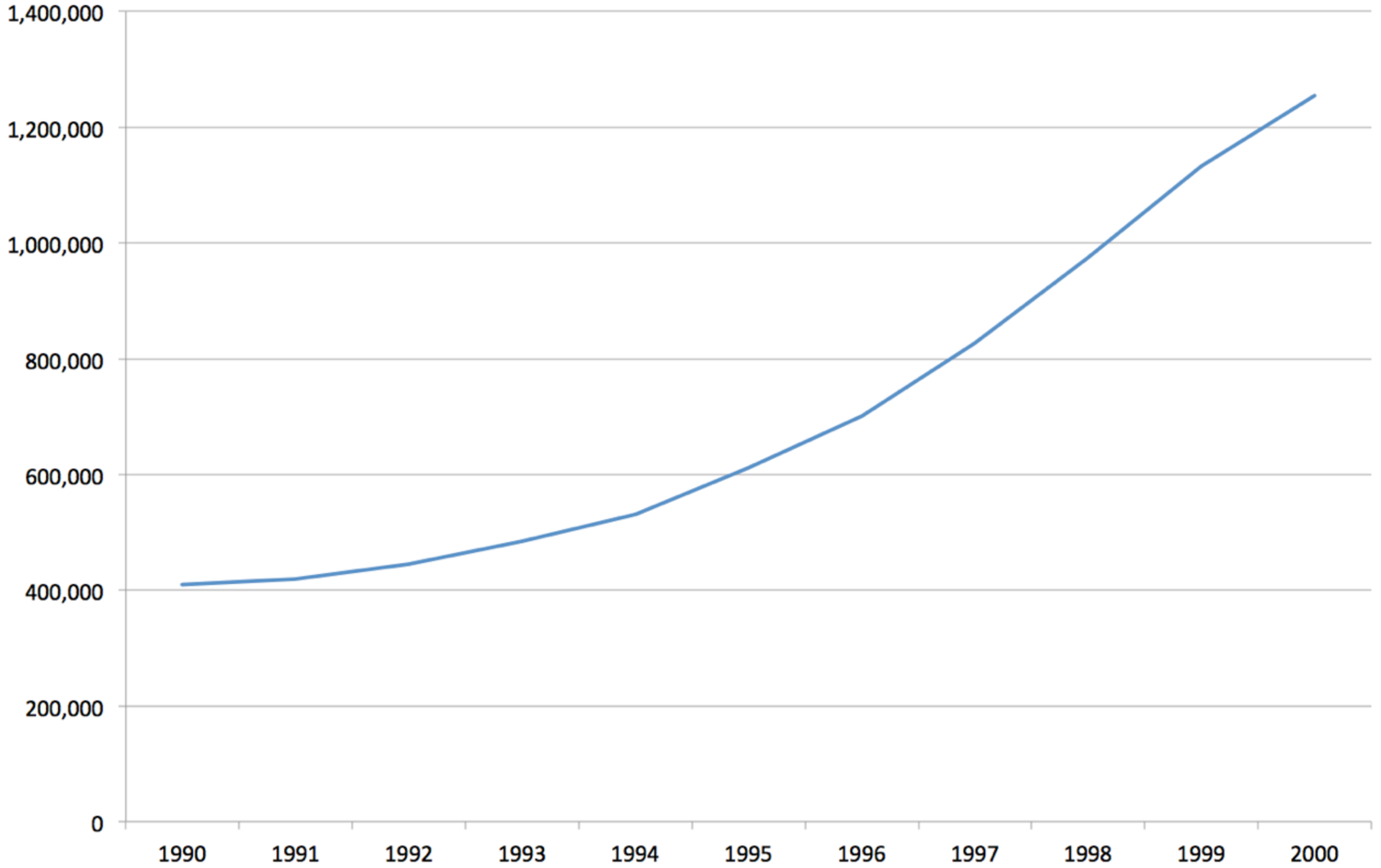

NCSA Mosaic for MS Windows
File Edit Options Navigate Hotlist Annotate Help
Document Title: NCSA Mosaic Home Page
Document URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
N C S A
MOSAIC
X Window System • Microsoft Windows • Macintosh
Welcome to NCSA Mosaic, an Internet information browser and World Wide Web client. NCSA Mosaic was developed at the National Center for Supercomputing Applications at the University of Illinois in --> Urbana-Champaign. NCSA Mosaic software is copyrighted by The Board of Trustees of the University of Illinois (UI), and ownership remains with the UI.
Jan `97
The Software Development Group at NCSA has worked on NCSA Mosaic for nearly four years and we've learned a lot in the process. We are honored that we were able to help bring this technology to the masses and appreciated all the support and feedback we have received in return. However, the time has come for us to concentrate our limited resources in other areas of interest and development on Mosaic is complete.
All information about the Mosaic project is available from the homepages.
NCSA Mosaic Platforms:
● NCSA Mosaic for the X Window System
● NCSA Mosaic for the Apple Macintosh
● NCSA Mosaic for Microsoft Windows
World Wide Web Resources The following resources are available to help introduce you to cyberspace and keep track of its growth:
● A glossary of World Wide Web terms and acronyms
● An INDEX to Mosaic related documents
● NCSA Mosaic Access Page for persons with disabilities
● Mosaic and WWW related Tutorials
● Internet Resources Meta-Index at NCSA
NUM



http://en.wikipedia.org/wiki/Hype_cycle

A Wrong Turn

1990s IT Employment Level



http://www.bls.gov/opub/btn/volume-2/careers-in-growing-field-of-information-technology-services.htm

Wanted:

A system that can produce code without requiring coders

rule “My Rule”
when: This
then: That

Often the central pitch for a rules engine is that it will allow the **business people to specify the rules themselves**, so they can build the rules **without involving programmers**.

-Martin Fowler, 2009

Tools Target Business Users

Limited mini languages

Decision tables in spreadsheets

Tools Target Business Users

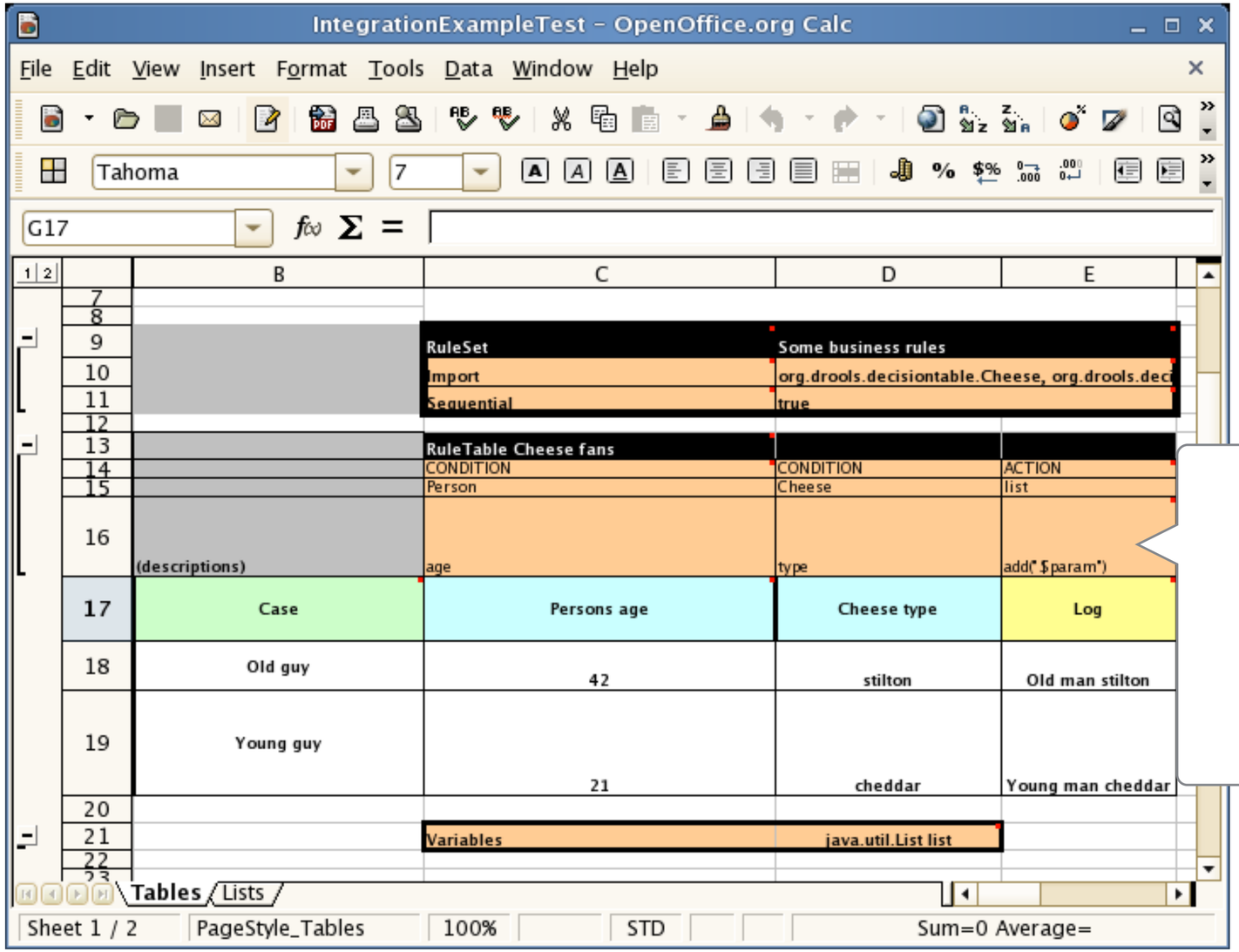
Great for simple problems

But problems don't stay simple

Tools Target Business Users

```
rule “My Rule”
  when:
    // Pile of Java code
  then:
    // Another pile of Java code
```

Tools Target Business Users



Write Code Here

http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch06.html

Rule authoring is development, whether we call it that or not

So let’s approach it as code

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

A = Travel Expenses
B = Contract Budget
C = Over Budget Expenses
D = Required justifications
E = Expense report summary

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

A = Customer Demographics
B = Order History
C = VIP status
D = Current Offerings
E = Sales Package

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

A = Diagnosis history
B = Lab test results
C = Complication risks
D = Interventions
E = Plan of Care

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

…or many other domains

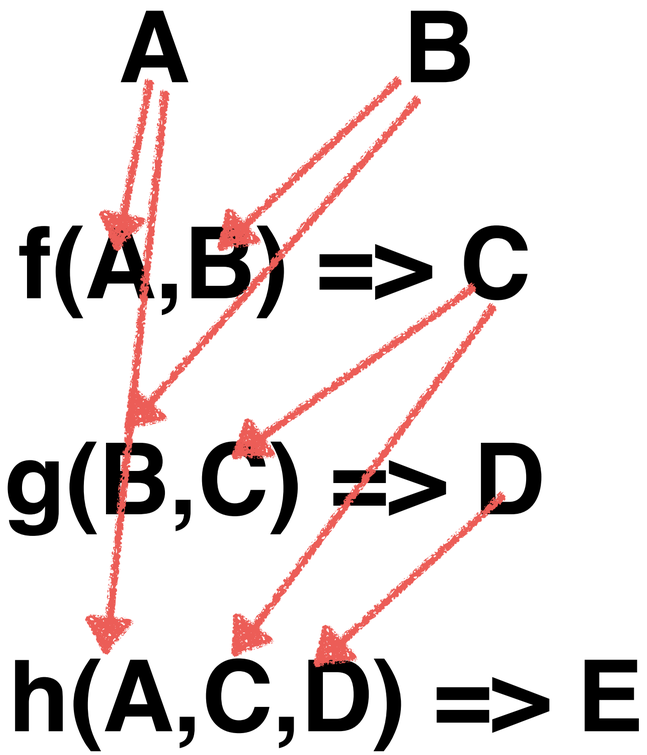
A B

f(A,B) => C

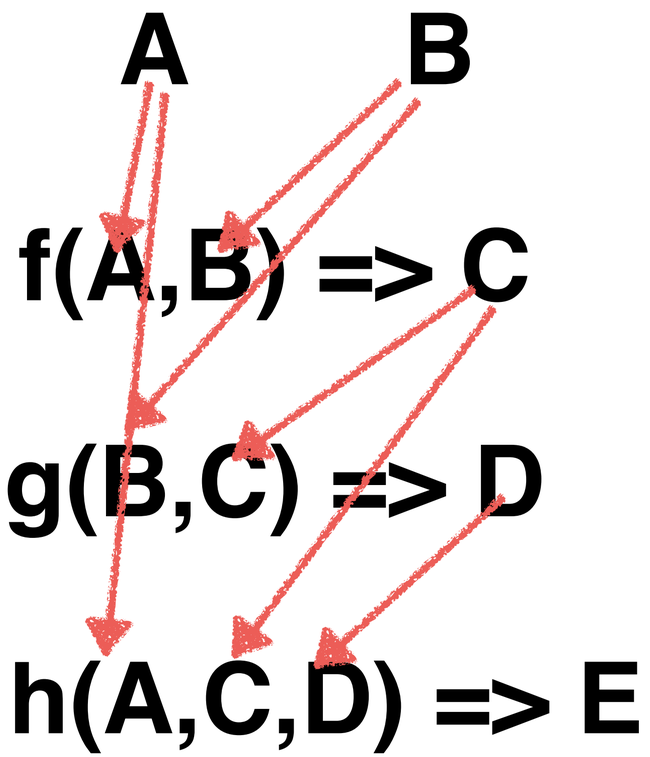
g(B,C) => D

h(A,C,D) => E

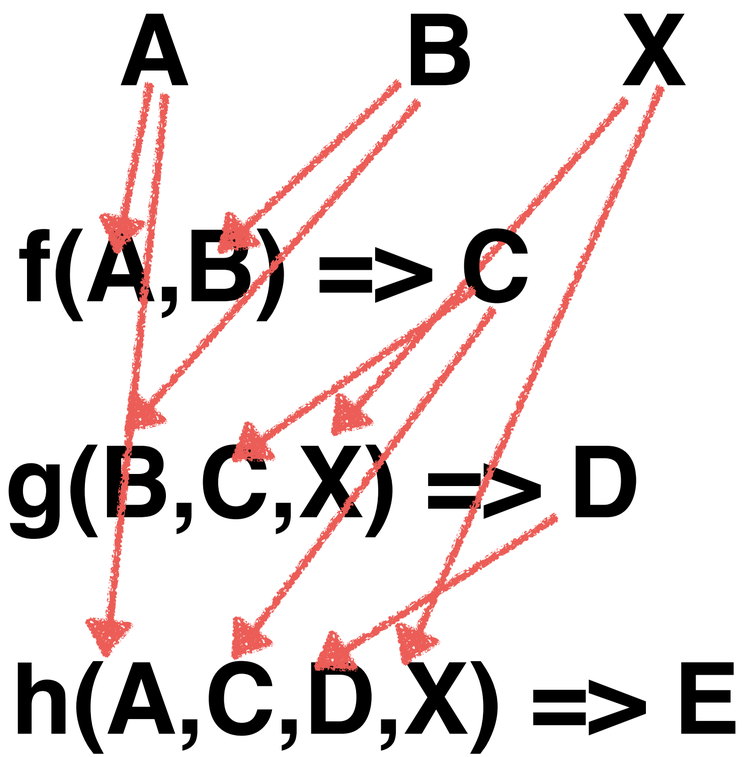
But we must wire things together!



But we must wire things together!



A = Diagnosis history
B = Lab test results
C = Complication risks
D = Interventions
E = Plan of Care



A = Diagnosis history
B = Lab test results
C = Complication risks
D = Interventions
E = Plan of Care
X = Family History

Now try this with thousands of requirements from different sources

Explicit wiring is complex and error prone

So let’s wire them automatically

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

How do we automatically wire this?

A B

f(A,B) => C

g(B,C) => D

h(A,C,D) => E

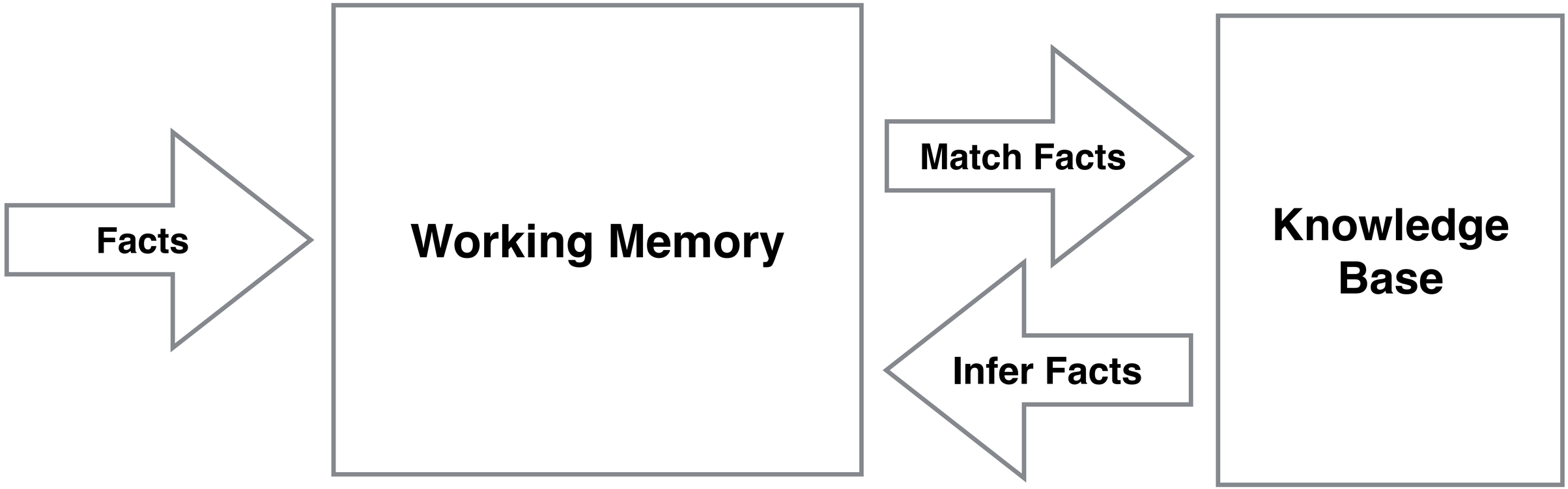
unit-of-logic
Type[constraints]
Type[constraints]
=>
action

Rules as a Control Structure

Rules as a Control Structure

A Principled Design

Start with a proven model



Simple structure

unit-of-logic

Type[constraints]

Type[constraints]

=>

action

Type system of the host

unit-of-logic

Type[constraints]

Type[constraints]

=>

action

Simple expressions

unit-of-logic

Type[**constraints**]

Type[**constraints**]

=>

action

Unification

```
unit-of-logic
   Type[?a == field1]
   Type[?a == field2]
  =>
   action
```

Arbitrary actions

unit-of-logic
  Type[constraints]
  Type[constraints]
 =>
 action

Truth Maintenance

if A assert B

if ~~A~~ retract B

Rules as a first-class participant in development

Nools (JavaScript)

Wongi (Ruby)

Clara (Clojure)

Nools (JavaScript)

Wongi (Ruby)

Clara (Clojure)

Why Clojure?

Data over objects

Equality means something

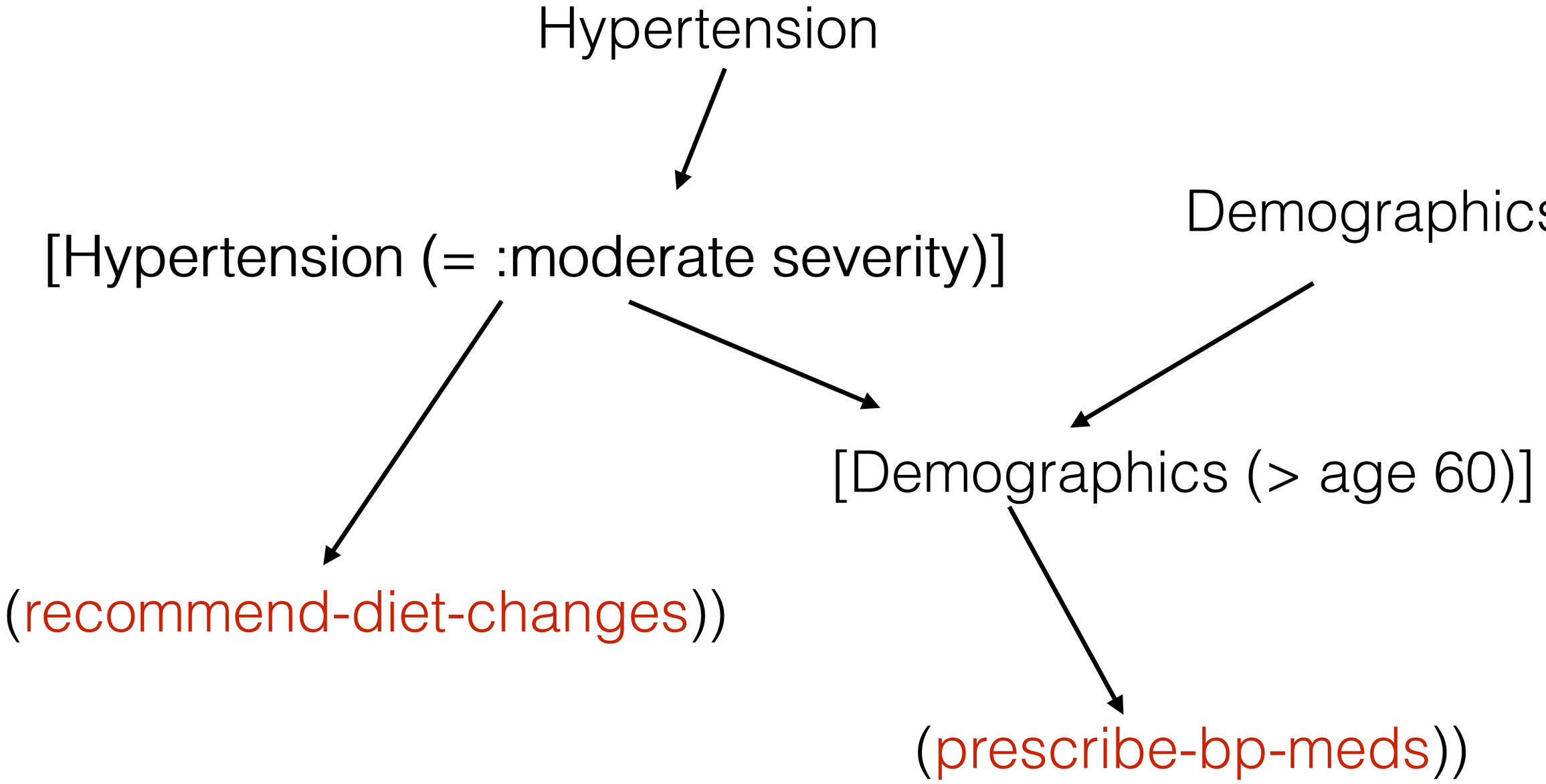
Reach

Can grow the language

(Hello, macros!)

```
(defrule diet-recommendation
   [Hypertension (= :moderate severity)]
   =>
   (recommend-diet-changes))
```

```
(defrule senior-bp-management
   [Hypertension (= :moderate severity)]
   [Demographics (> age 60)]
   =>
   (prescribe-bp-meds))
```


Hypertension
[Hypertension (= :moderate severity)]
Demographics
[Demographics (> age 60)]
(recommend-diet-changes))
(prescribe-bp-meds))

Rules Demo

And the usuals..

Arbitrary expressions

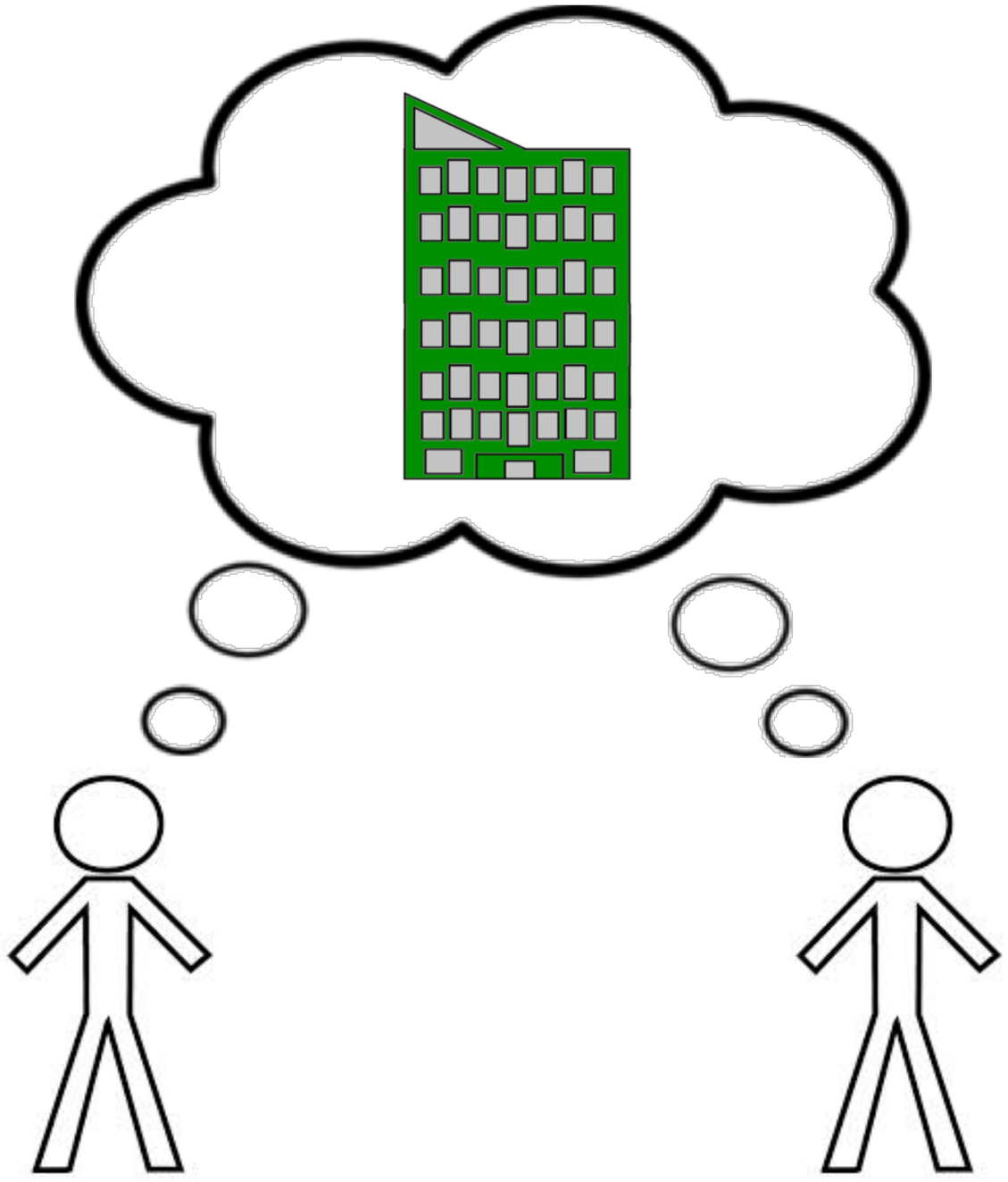
Accumulators

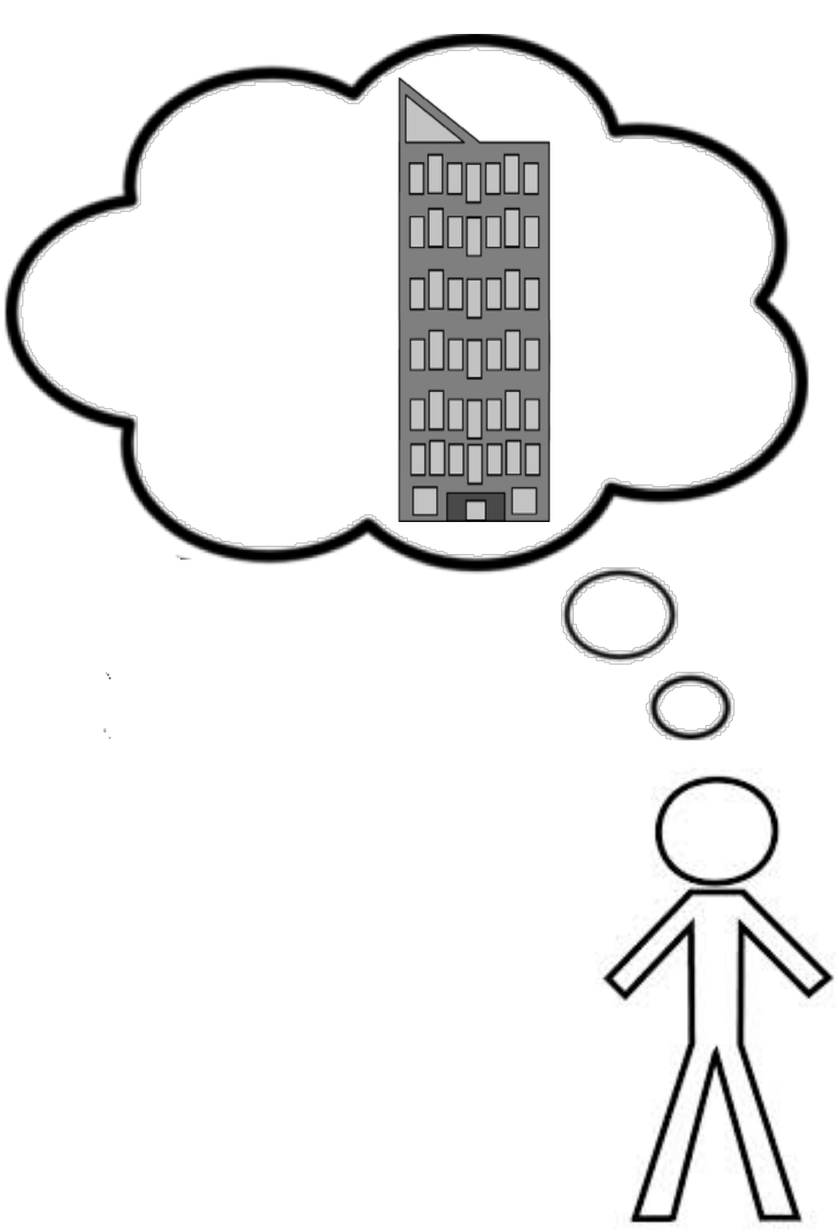
Session durability

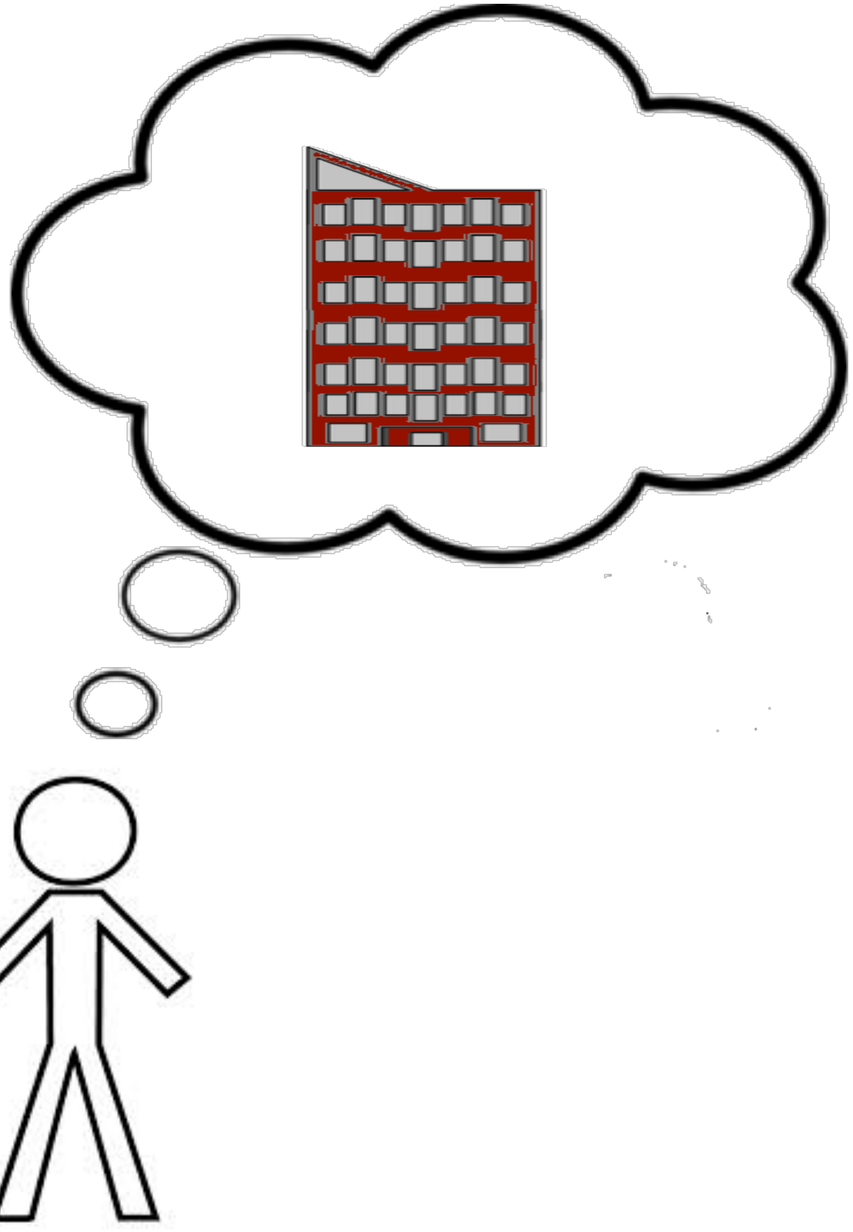
Tracing

Intelligent systems derive their power from the **knowledge they possess** rather than the specific formalisms used.

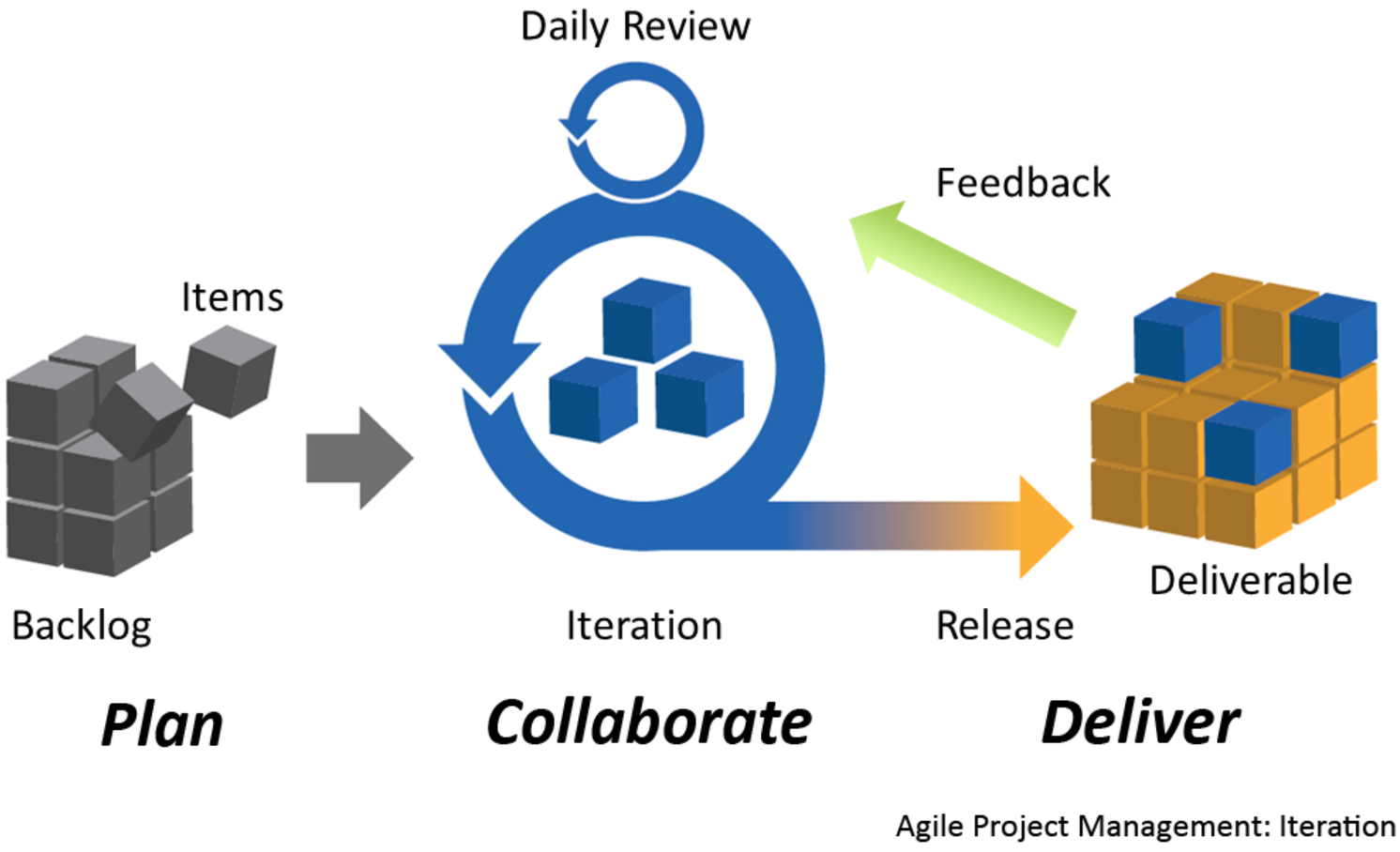
-Edward Feigenbaum





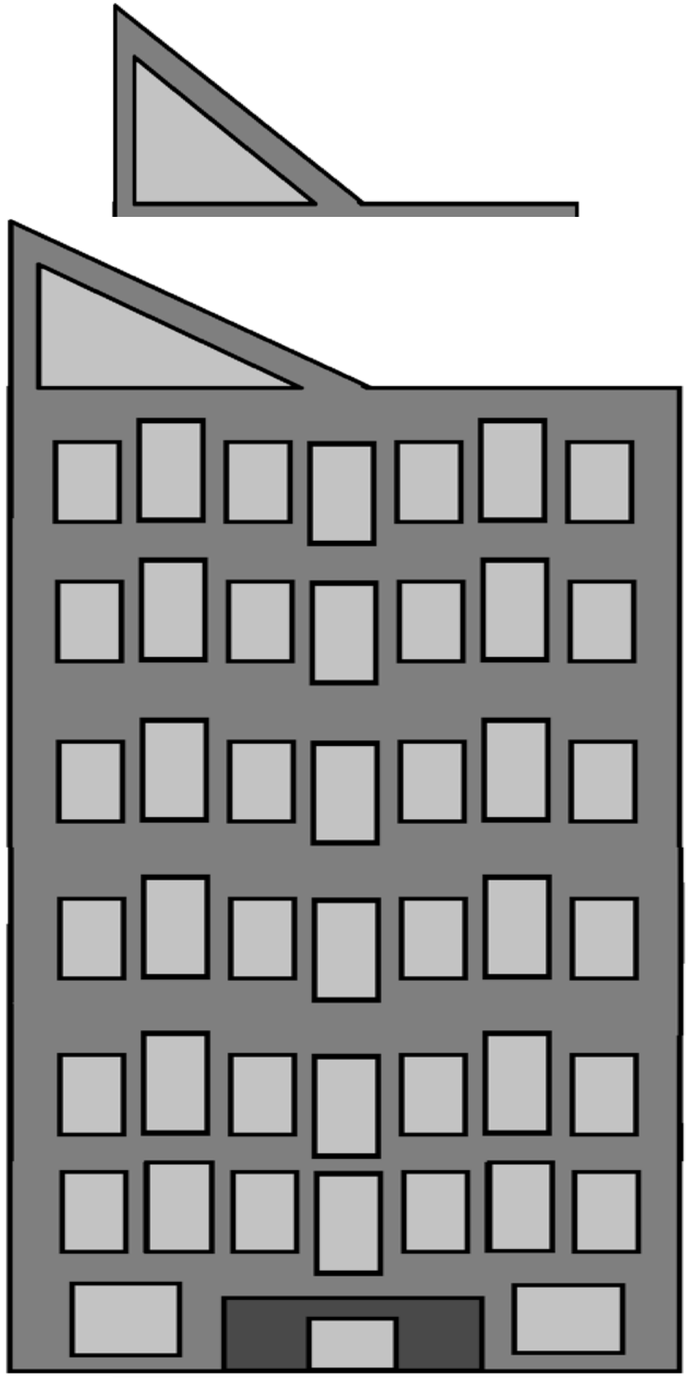


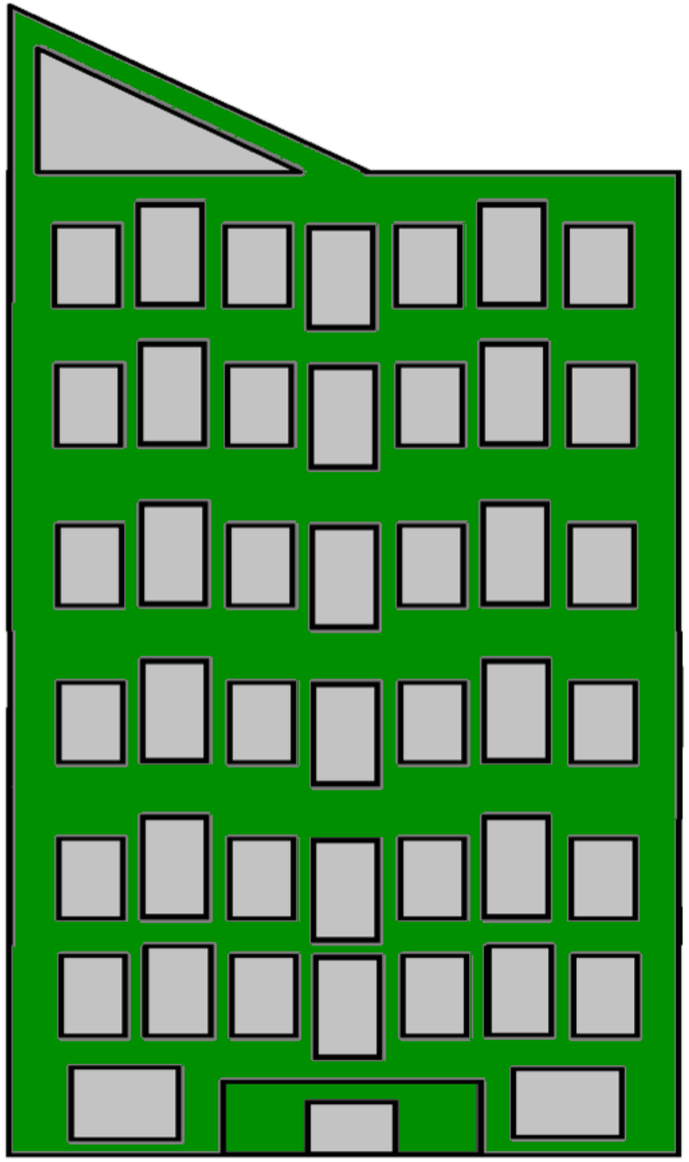
Agile Development!



http://en.wikipedia.org/wiki/Iterative_and_incremental_development

Imagine if other industries used agile development….



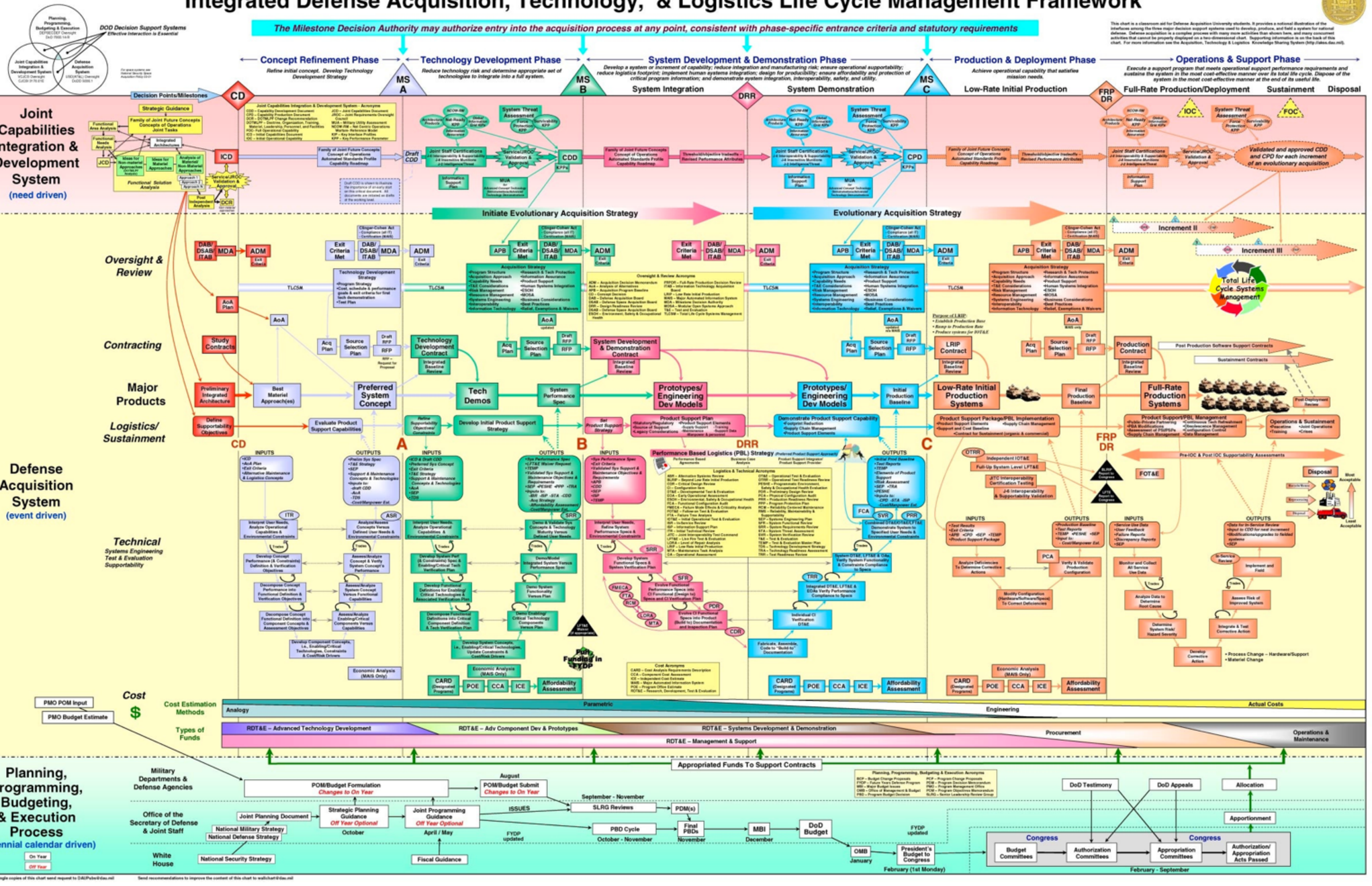


Agile works when iteration is cheap,

but iteration isn’t always cheap.

ver. 5.2. August 2005

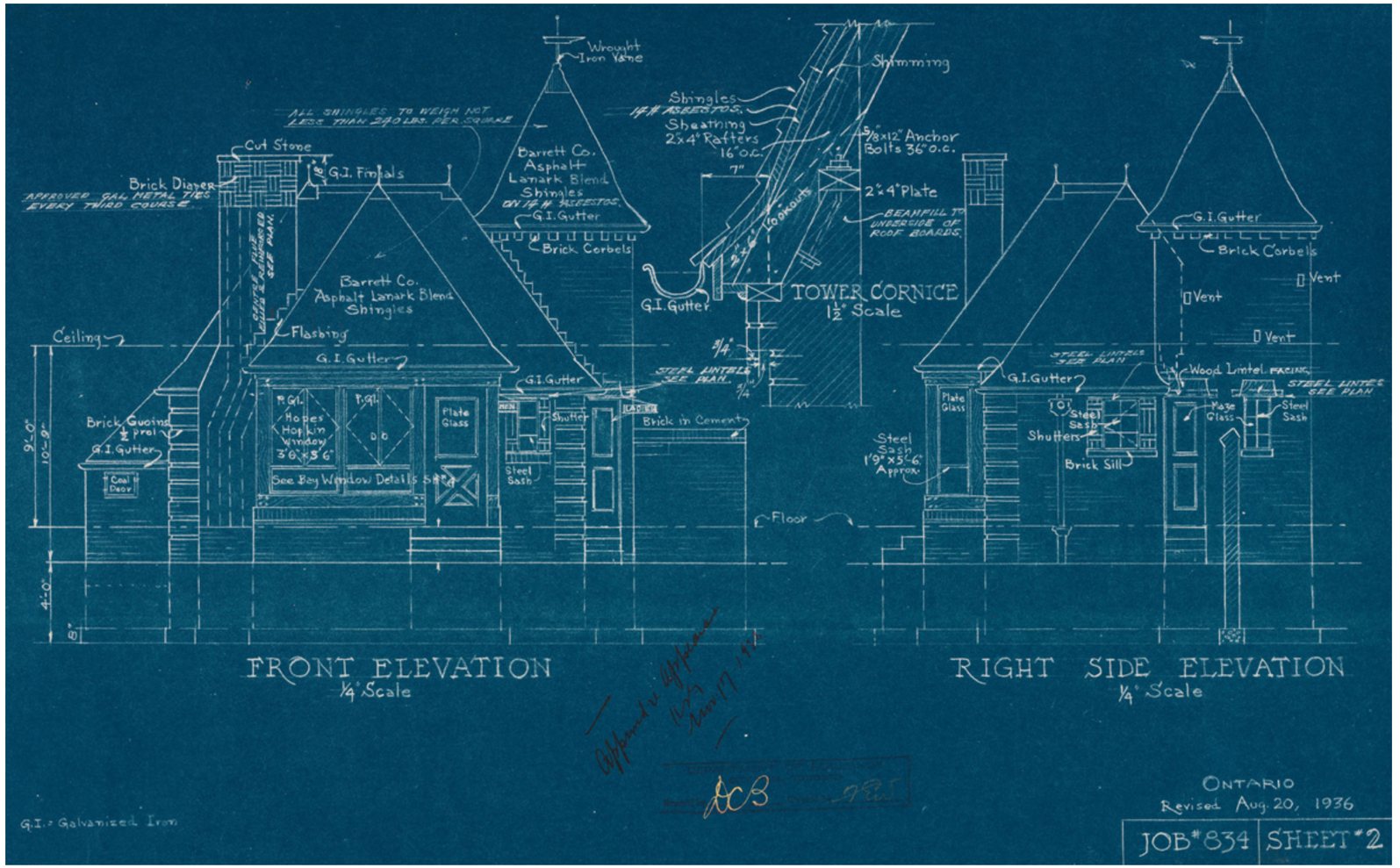
Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework



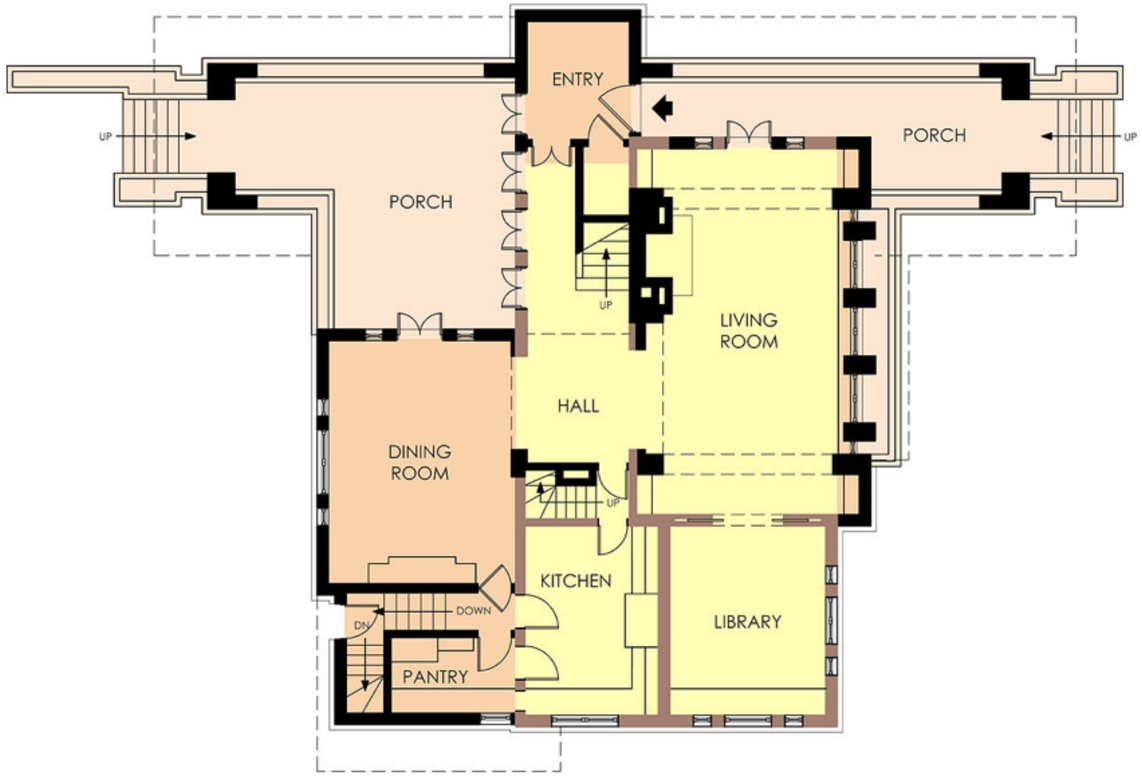
This chart is a classroom aid for Defense Acquisition University students. It provides a notional illustration of the interfaces among the three major decision support systems used to develop, produce, and field a system for national defense. Defense acquisition is a complex process with many more activities than shown here, and many concurrent activities that cannot be properly displayed on a two-dimensional chart. Supporting information is on the back of this chart. For more information see the Acquisition, Technology & Logistics Knowledge Sharing System (http://akss.dau.mil).

What can we learn from industries where agile doesn’t work?

Blueprints


Wrought Iron Yoke
Shimming
Shingles
Sheathing
2"x4" Rafters 16" o.c.
7"
5/8"x12" Anchor Bolts 36" o.c.
2"x4" Plate
Beamfill to underside of roof boards.
Cut Stone
Brick Diaper
G.I. Finials
Barrett Co. Asphalt Lamark Blend Shingles
G.I. Gutter
Brick Corbels
TOWER CORNICE
1½" Scale
Ceiling
Flashing
Brick Quoins
G.I. Gutter
Coal Door
Plate Glass
Shutter
Steel Sash
Brick in Cement
See Bay Window Details
Floor
Steel Sash 1'9"x5'-6" Approx.
Vent
Wood Lintel Facing
Shutters
Brick Sill
Mage Glass
FRONT ELEVATION
¼" Scale
RIGHT SIDE ELEVATION
¼" Scale
G.I. = Galvanized Iron
ONTARIO
Revised Aug. 20, 1936
JOB # 834
SHEET # 2

Floor plans


ENTRY
PORCH
PORCH
UP
UP
UP
LIVING ROOM
HALL
DINING ROOM
UP
KITCHEN
DOWN
DN
PANTRY
LIBRARY
NORTH
0 1 2 4 8 16 FEET
0 1 2 4 8 METERS
1884
1906

Floor Plan Demo

A New Role for Rules

A tool for **developers**

A means to minimize the **semantic gap**

Leverage for a **shared understanding**

(Just don’t write Java code in Excel.)

Questions?

@ryanbrush

http://engineering.cerner.com

http://github.com/rbrush/clara-rules