MISSION**CRITICAL**

ICLP'97

The challenges of software engineering

Logic Programming in the Real World

Some thoughts

Leuven - July 10, 1997

Michel Vanden Bossche-Marquette
Chief Executive

Tel +32 2 759 95 60
Fax +32 2 759 27 60
E-mail mvb@miscrit.be

MISSION**CRITICAL**

Background

MISSION**CRITICAL**

The changing environment

Society ⇔ ICT

- ✔ **Complexity**
- ✔ **Post-industrial ➜ InfoAge**

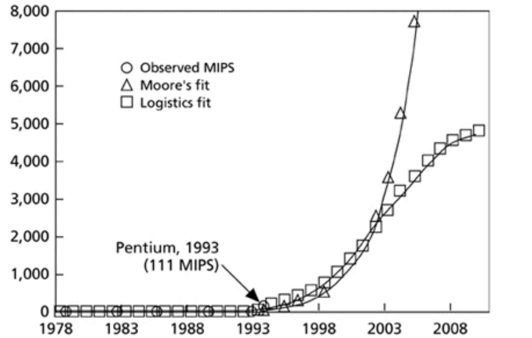
- ✔ **Computers**
- ✔ **Communications**
- ✔ **Software**

- **ITC (Information & Communications Technologies)**
  - ✔ Computers — technological explosion (mastered)
  - ✔ Communications — technological explosion (mastered)
  - ✔ Softwarei — Instability, confusion, crisis

MISSION**CRITICAL**

Computers

- **Technological explosion (microelectronics)**



source: IEEE computer

Asymptotic (= major technological factor)

Communications

- **Digitalisation**
  - ✔ Fiber Optic — SDH (2.4 Gbps), WDM (50+ Gbps)
  - ✔ Switching — Low cost cell switching fabrics
  - ✔ Access — Impact of DSP (xDSL, CATV)
- **Internet**
  - ✔ Mass market — x 2 (hosts/year), x 5 (volume/year)

    more PCs than TV sets this year

Software

- **Contradiction**
  - ✔ Enormous progress with <u>commodity</u> software (OA, Web, ...)
  - ✔ Major problems with software in general

    Development time and cost are increasing

    Quality is very poor

    Maintenance is very expensive (70%)

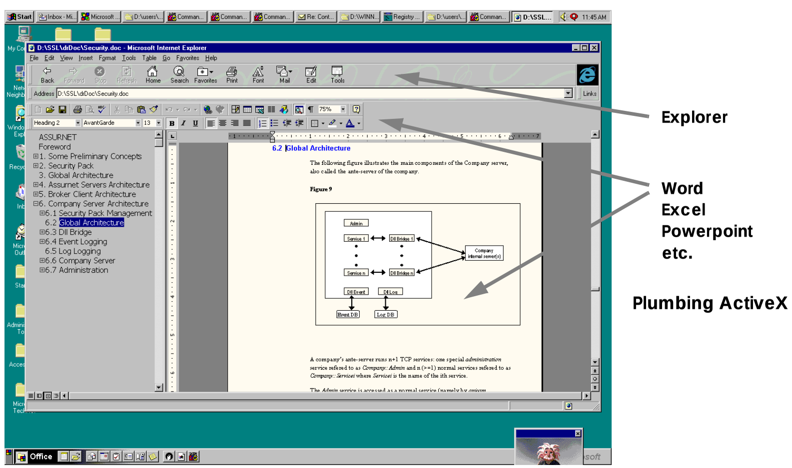
    Rigid

    Additional problems (event-driven, client/server, Web, etc.)

➜ Software Crisis

While software is supposed to be <u>malleable</u>
it has become the <u>most rigid</u> part of an information system

Progress "commodity" - Microsoft 97



The Software Crisisl

PAUL STRASSMANN

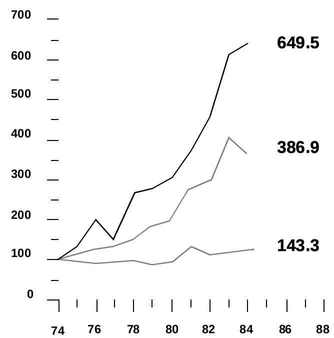
SOFTWARE: SO BAD, IT CAN ONLY GET BETTER

Software easily rates among the most poorly constructed, unreliable and least maintainable technological artifacts ever invented — with the exception, perhaps, of Icarus' wings.

The Software Crisis

- **Large Projects**
  - ✔ TAURUS London Stock Exchange : cancelled after 400 M GBP
  - ✔ SOCRATE SNCF, investment > 2.000 M FRF ⇒ Impact on sales
  - ✔ SABRE 92 Extension Marriott, Hilton, Budget : *write-off* $160 M
  - ✔ DENVER *Bagage Handling* : late by 18 months, loss $400 M
  - ✔ France Projet "Ministère de la Justice" : cancelled, 2.000 M FRF
  - ✔ DBB No payroll during 10 months
- **Not forgetting**
  - ✔ Many smaller project
  - ✔ Remanent quality problems (OS/2 Warp, Windows 95, C/S tools, etc.)

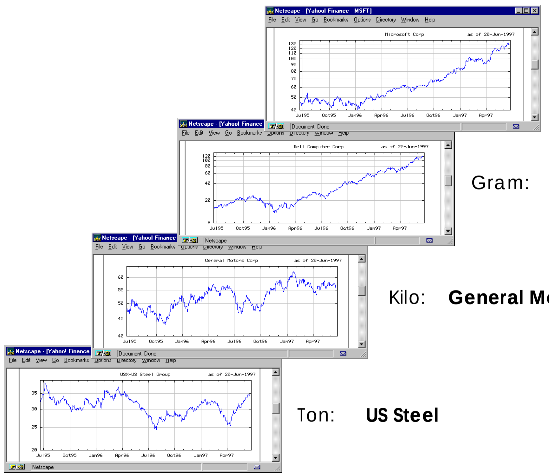
At the core of the *Information Society*



?	"0 Gram" age	software
649.5	"Gram" age	computers (Hw) electronics pharmaceutical
386.9	"Kilo" age	machine tools automobile aerospace
143.3	"Ton" age	shipyards steel civil engineering

STOCK PRICES INDICES (*Source : Nikko Securities*)

Examples



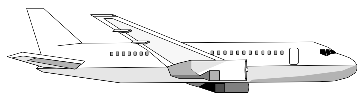
0 gr: **Microsoft**

Gram: **Dell**

Kilo: **General Motors**

Ton: **US Steel**

Fundamental Problem



Complex systems
Engineering is mastered

- conforms to specifications
- predicted performance
- within budget
- very high reliability

```
Part::TraverseForward( int hops )
{
   register Part* p;
   --hops;
   for( register int i = 0; i < MAX_CONNECTIONS; i++)
   {
      p = ConnectedTo(i)->ToPart();
      nullproc(p->X(), p->Y(), p->PartType());
      if( hops > 0 )
          p->TraverseForward( hops );
   }
}
```

Always problems
even for moderately complex systems

- in line with requirements !
- performance !
- budget !
- reliability !

"Classical" Products

Conceived and built using
models based on <u>continuous mathematic</u>

Always a combination of

Theory	**Practice**
✓ models based on physics, ...	✓ applicability of the models
✓ mathematical expression	✓ tools with required precision
✓ proof and stability	✓ expertise (design, tools, ...)
✓ components (theories)	✓ components (sub-assemblies)

<u>Precise Corpus of Knowledge</u>

ICT Products

The mathematic on which these products
are based is <u>discrete</u> (combinatorial explosion)

- **Hardware**

  Hardware is based on simple repetitive structures that can be tested exhaustively (but Pentium flaw shows the limit) . Mass production is mastered.

- **Software**

  No repetitive structure and enormous nomber of states: impossible to do exhaustive testing. Production is much more craft than industry.

Software

Conceived and built using mainly an
<u>ad hoc pragmatic approach</u>

The theoretical "corpus" is extermely limited

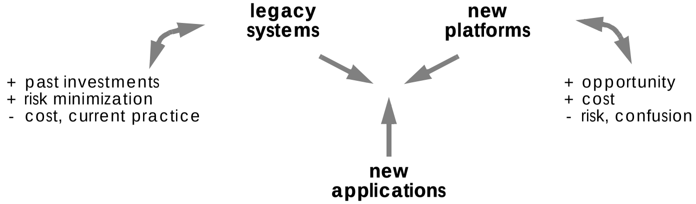
Theory : very weak with Cobol, C, C++, Java	**Practice = craft**
✓ variable = position in memory	✓ language, editor, compiler
✓ array = contiguous positions	✓ trial and error
✓ proof ?	✓ methodology (?)
	✓ metric (?)

Examples

- **Legal status of software engineering**
  - ✔ Not one of the 36 engineering profession in USA
  - ✔ Tennessee forbids the use of the term "Software Engineering"
  - ✔ Essential elements of the engineering profession that are missing:
    - well defined corpus of knowledge
    - formal control of qualification
    - defined set of standard practices
    - formal process in case of malpractice
    - liability insurances
    - etc.
  - ✔ Explict <u>warranty</u> ("products") ⇔ Explicit <u>no-warranty</u> ("software")

Software Challenges

- **Strategic role**
  - ✔ Very low cost of hardware ⇒ Demand increases
  - ✔ Supposed to be very flexible ⇒ proliferation of new requirements
  - ✔ Software added-value >> Hardware added value
- **Multiple tensions**



Current Trends

- **Hardware - Software**
  - ✔ Client/Server: distributed, relational DB, 4 GLs
  - ✔ Objects:, components, Java
  - ✔ Web: the "new" Client/Server
- **Impact**

Theory

- ✓ **no changes**

Practice

- ✓ **RAD (no methodology)**
- ✓ **Objetcs (better abstraction)**
- ✓ **Components (reuse)**
- ✓ **Turbulence (tools, appoaches, ...)**

Consequences

- **New problems**
  - ✓ Much more complex
  - ✓ Event-driven systems (instabilities)
  - ✓ Strong coupling (inheritance)
  - ✓ Impedance mismatch (object - relational)
  - ✓ Instable and immature tools
  - ✓ In 5 years: new legacy systems!
- **Packages (SAP, ...)**
  - ✓ Solution, but where is the differentiation
  - ✓ Software crisis is transferred to the vendor
  - ✓ New risk: *Commercial Of the Self Legacy System*

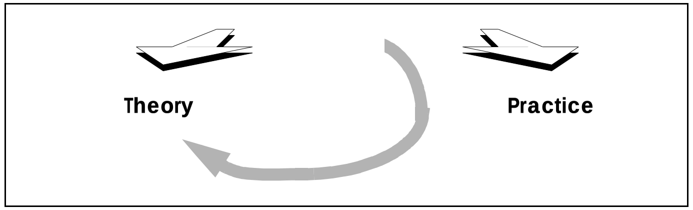
Quality

- **Current approach**
  - ✓ ISO 9003 : software development, delivery and management
  - ✓ Focus is the development **process**
  - ✓ Nothing said about the intrinsic quality of the **product**
- **Pratically**
  - ✓ Quality management addresses mainly the practical aspects
  - ✓ Approach is "Best Practice"
  - ✓ Not bad, but insufficient
- **Example : Windows 95**

  Beta test with 400.000 users

What is required

- **Better balance between practice and theory**



- **Key: respective roles**

Theory	for what, until where?
Practice	for what, until where?

Not mandatory if problem is "simple"

- **1st industrial revolution**

1707	"Root" innovation for the steam engine (Papin)
1712	1st practical application: pumping (Newcomen)
1765	Basic technology is ready (Watt : condensor, double effect)
1775	1st mill with adequate precision (Wilkinson)
1854	Pudling iron by steam (steel)

Importance of components and tools

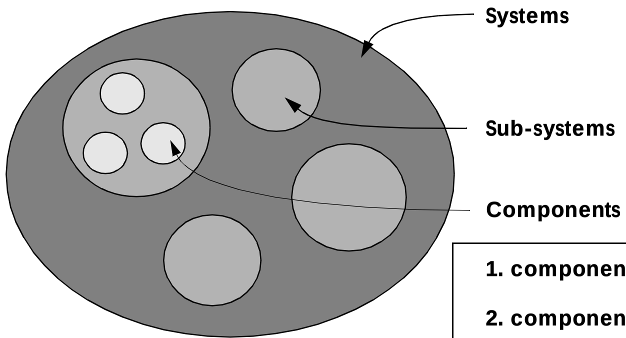
Key when "complex"

- **2d industrial revolution**

1831	Electro-magnetic induction (Faraday)
1864	Electro-magnetic field theory (Maxwell)
1869	Dynamo (Gramme)
1876	Telephone (Bell)
1886	Electro-magnetic waves (Hertz)
1896	Radio (Marconi)

Importance of components, tools and theory

Theory - Role #1



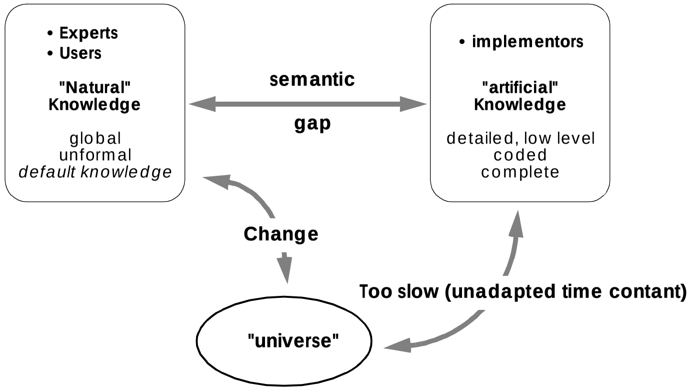
1. components properties

2. components properties

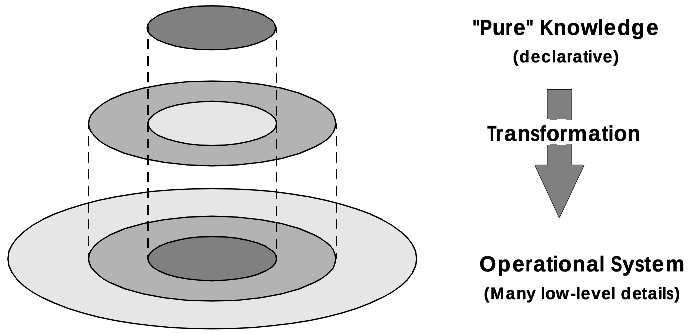
⬇

system properties

Theory - Role #2



Theory - Role #2



Research

- **Computer Science**
  - Program = theory in an adequate logic
  - Computing = deduction from that theory
  - ✓ Definition of the problem
  - ✓ Interpretation of the "intentions" of the programmer
  - ✓ Logical part (a restricted theory)
    - Logic programming (theory, axioms, 1st order logic)
    - Functional Programming (equations, theory of types)
  - ✓ Control part (efficiency of the implementation)
- **Real-life**
  - Nearly never (Z, B excepted for *safety critical systems*)

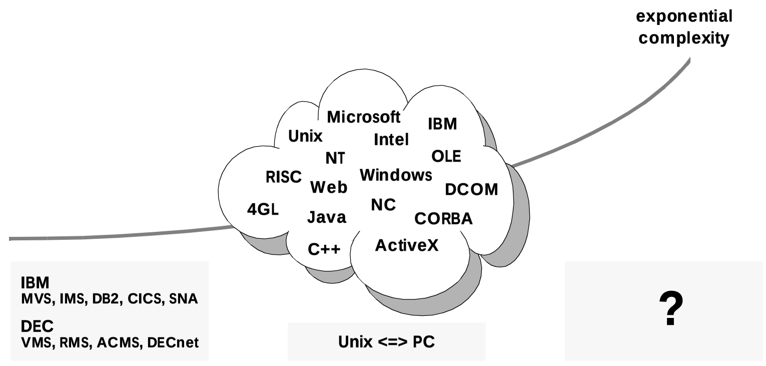
Software Industry

- **In general**
  - Yawning chasm between Practice and Computer Science
  - Even for advanced projects
  - Percolation very slow
- **For hardware**
  - Fundamentally different
  - Very fast transfer from the lab to the industry
- **Why?**
  - Culture and Education (in general, but some exceptions)
  - Added complexity from legacy
  - Intrinsic complexity
  - Not a closed world: interaction with humans

MISSION**CRITICAL**

Additional Problem

- **Instability and confusion**



MISSION**CRITICAL**

Information & Communication Society

Society	Duation (years)	Interaction (km/h)	Power	Technology
Agrarian	3000-5000	5-15	Land	Tradition
Industrial	300-500	50 - 150	Capital	Science
Post-Industrial	30-50	300 - 900	Structure, coverage	Inverse economics
Information	3-5	3.000.000+	Communication	Inference, creativity

- ✓ The speed of interaction in the society is continuoulsy increasing
- ✓ The industries of the first 3 "ages" are stabilized (oligopoles)
- ✓ The speed of interaction in the information society is "terminal"
- ✓ The information society is entering into turbulence and start oscillating
- ✓ Turbulence result from the continuous creation of immaterial artifacts
- ✓ Innovation (mainly "immaterial") = major opportunity-risk

MISSION**CRITICAL**

The good news

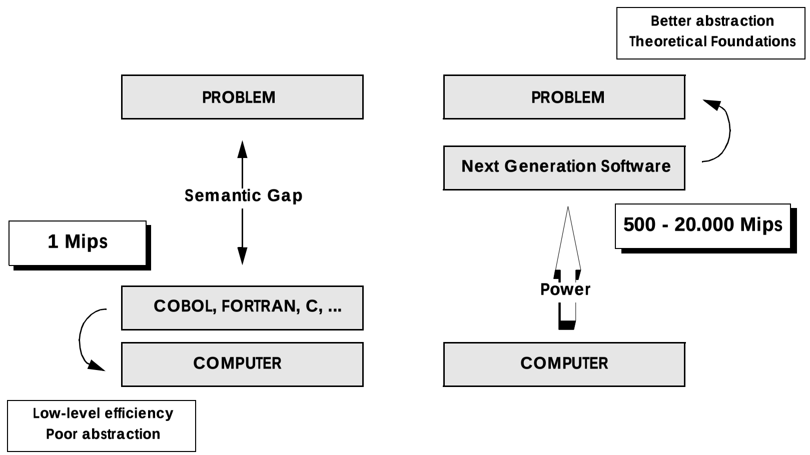
MISSION**CRITICAL**

Exponential Power at declining costs

	Mips	# users	Mips/ user	Cost	Cost/ user	Cost/ Mips
70s - 370	**2**	**100**	**0.02**	**3,000**	**30**	**1,500**
80s - VAX	**1**	**50**	**0.02**	**750**	**15**	**750**
1991	**30**	**1**	**30**	**15**	**15**	**0.50**
1994	**150**	**1**	**150**	**5**	**5**	**0.03**
1997 mono	**500**	**1**	**500**	**5**	**5**	**0.01**
2000 clusters	**20.000**					

× **250** × **25000**

The Possibility to leverage that Power



Emergence of some stability

- **Past**

Mainframe	Stability, but monopoly prices (IBM, etc.)

- **Unix**

Openess	Lower costs, but ...
Balkanisation	Sun/Solaris, HP/UX, SCO, etc

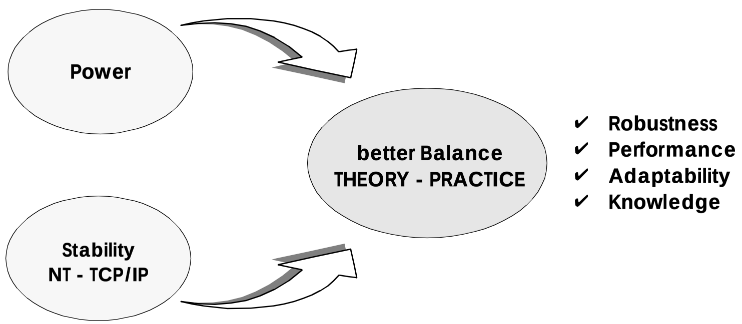
- **Microsoft - Windows NT**

Proprietary	But commodity pricing (volume)
Stability	Competition but on standardized equipment

- **Network**

TCP/IP	The winner: global connectivity and services
Transport	Various (Gigabit Ethernet, ATM, WDM, ...)

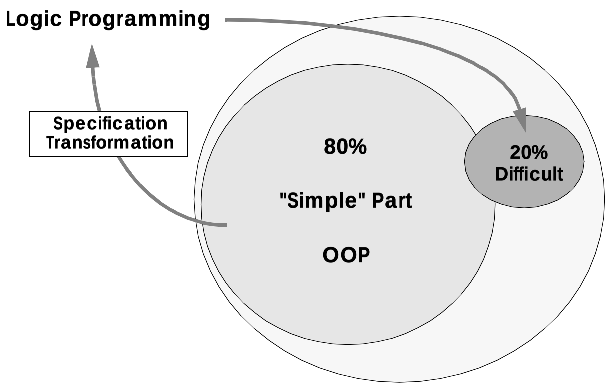
Action



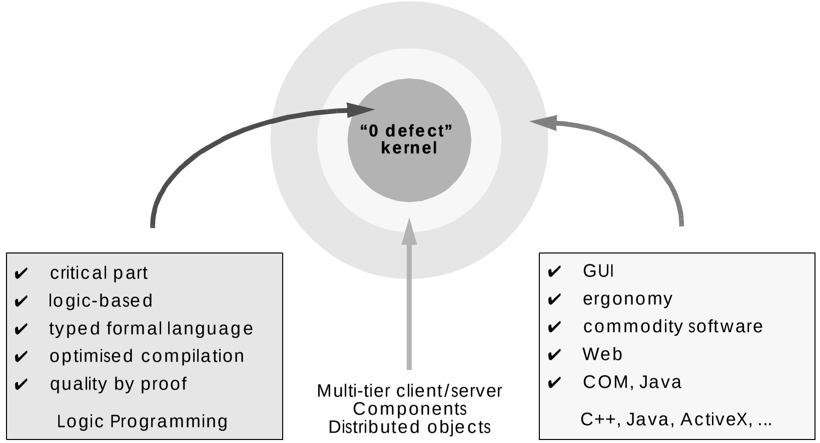
Balance

- **Theory**
  - Declarative (Logic) Programming
  - Discrete Mathematic
  - Formalization of complex and evolving knowledge
  - Theory to deduce and prove
- **Pragmatism**
  - Object Oriented Programming
  - Better abstraction
  - Adequate for GUI, Network, Security, etc.
  - Possibility to leverage components (ActiveX, JavaBeans)

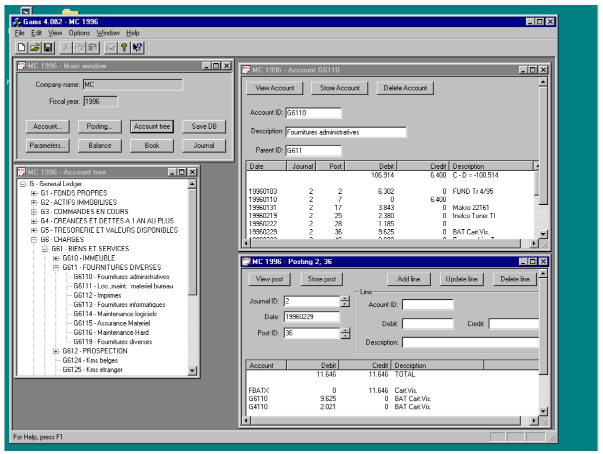
Respective Roles



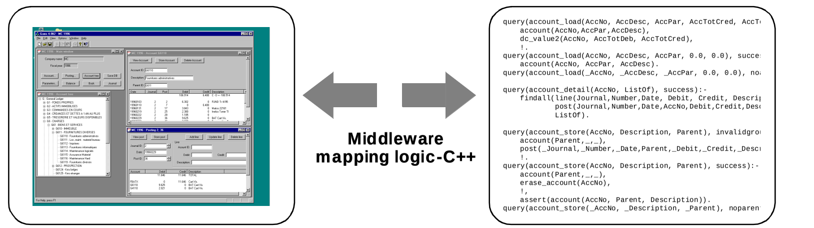
How to Implement



Pedagogical Example



Pedagogical Example



Surface

- ✔ Microsoft plumbing
- ✔ VC++, MFC, ActiveX

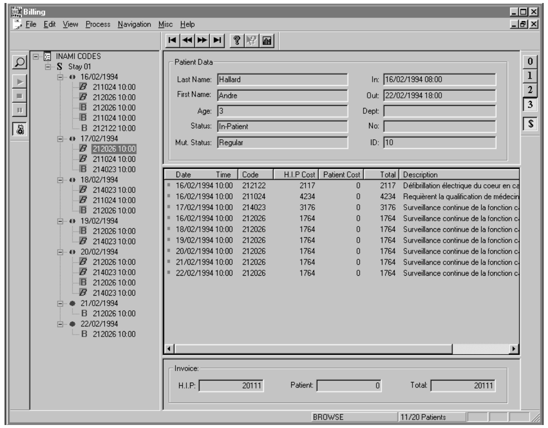
Fundamental aspects

- ✔ Accounting Ontology
- ✔ Logic (Prolog)

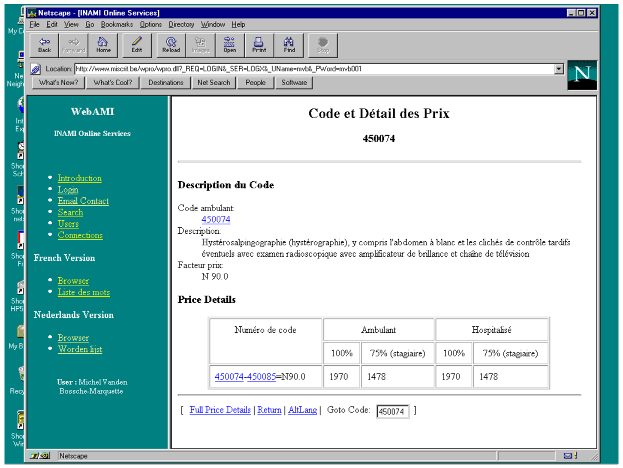
Example - LogIHCS

- **Tarification and Billing (Hospitals)**
  - ✔ Complex regulation (social security)
  - ✔ 4000+ “codes” and 1000+ “rules”
  - ✔ Very complex inter-relations
  - ✔ Continuously changing
  - ✔ Very difficult to implement and maintain in Cobol, C, 4GL, etc.
- **Mission Critical’s approach**
  - ✔ Codes and rules in a dedicated formal logic language
  - ✔ OOP where adequate
  - ✔ Billing = inference engine
  - ✔ Tools : browsers, local heuristics, explanation, etc.
  - ✔ Scientific partnership (B. Le Charlier, Y. Deville)

Win32 Client



Electronic Documentation ...



Trend

(www.computerworld.com) December 9, 1996 Computerworld

VIEWPOINT

Start your business rules engine

Patricia B. Seybold

Are you still embedding the rules of your business in application logic or database triggers? You’re in big trouble. There’s no way you can change those systems into flexible applications that meet the rapidly changing needs of your business.

Instead, you should separate the business rules from the rest of your application code. All businesses must be able to change business rules on the fly as they strive to be nimble and competitive. It must be easy for business managers and even customers to make those changes, without help from programmers.

If you’re not tuned in to the concept of business rules, you’re not tuned in to your business.

Eavesdrop on almost any conversation among business managers, and you’ll hear talk about business rules: Why are we refusing credit to Hispanics under 30? What’s our policy for educational discounts? Does it apply to private colleges? When does this new pricing plan take effect? Does it supersede prices in effect for existing customers?

Separate business rules from applications so your company can be more nimble and competitive

Those are the kinds of policies that should be stored in a rules engine, in such a way that they easily can be changed by the business owner.

Business rules previously were associated with niche technologies such as rules-based expert systems and workflow software.

But the concept of a separate rules engine is on the verge of broader acceptance because of the World Wide Web and the compelling need to integrate disparate applications.

On the Web, companies are beginning to cater to customers’ individual preferences with one-to-one marketing and fulfillment. As a customer, I can configure and buy computers at Dell Computer’s site, subscribe to a personalized edition of *The Wall Street Journal* and have Amazon.com notify me whenever a new book is available in my field of interest.

Another way to serve customers is to integrate the applications — from ordering to shipment and billing — that affect the customer. Business rules should be a key part of that integration.

We need to accommodate the following business rules:

- The rules that govern policies, pricing, product configuration and the myriad of “if/then” statements in today’s application logic. These should be maintained by the business owners.
- The rules in the customer profile. That is where the customer specifies how he wants to do business with your firm.
- The rules embedded in your existing systems. Each legacy system has a set of assumptions about what it needs to do its job and what it expects from the applications or users that provide input. Those rules need to be documented as part of the middleware you use to glue applications together. That way, when you substitute one back-end application for another, you’ll know what rules need to be changed.

GROWTH IN ACCEPTANCE

My predictions: Business rules engines will become much more popular in the next few years. And the practice of separating and codifying business rules during the early phases of systems development will become common practice. Business users and customers will begin to take ownership of their rules.

So as you select off-the-shelf applications or development tools, look for the business rules engine. If it isn’t included, look elsewhere.

Seybold is president of Patricia Seybold Group in Boston. Her Internet address is pseybold@psgroup.com.

Theory - Opportunities

- ✔ The market begins to understand the software challenge
- ✔ Business Eengineering requires new Information Systems
- ✔ Quality concerns (TQM, ...)
- ✔ New platforms (Windows NT, Web, Java, ...)
- ✔ Exponential power (PentiumPro at commodity prices)
- ✔ Many *mission-critical* and *business-critical* problems
- ✔ *Global Information Society* : mainly a software problem
- ✔ R&D results
- ✔ Long term, as opposed to the short-term software fashion

Theory - Threats

- ✔ Legacy systems
- ✔ Conservatism
- ✔ Intellectually challenging, in opposition to mass market
- ✔ Complex problems $\Rightarrow$ difficult
- ✔ Necessity to understand the application domain $\Rightarrow$ difficult
- ✔ Very short life cycle of IT products
- ✔ Upstream dominance of the USA
- ✔ Logic = "academical" image
- ✔ Lack of clear demonstration of the relevance

Approach Mission Critical

- ✔ Clear demonstration of the relevance
- ✔ Mercury (Melbourne)
  - pure
  - *programming in the large*
  - declarations: polymorphic types, modes, determinism
  - high performance
- ✔ Industrialisation of Mercury (ARGo, within ESPRIT)
  - abstract interpretation (*destructive updates*)
  - environment
  - methodology
  - demonstration of the relevance

Engineers of the "0 gram" industry