

**[Session S1]  
RATIONAL UNIFIED PROCESS AND ITERATIVE  
DEVELOPMENT METHODOLOGIES**

Philippe Kruchten  
Professor of Software Engineering at the University of  
British Columbia



THE UNIVERSITY OF BRITISH COLUMBIA

# The “Making Of” the RUP® A learning experience

Philippe Kruchten

SD Process, Canada  
Toronto, December 6<sup>th</sup>, 2004

# Presenter

Philippe Kruchten, Ph.D., P.Eng.

*Professor*

Department of Electrical and Computer Engineering

University of British Columbia

Vancouver, BC Canada

[pbk@ece.ubc.ca](mailto:pbk@ece.ubc.ca)

[kruchten@ieee.org](mailto:kruchten@ieee.org)



# Outline

---

- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

Rational Unified Process and RUP are registered trademarks of IBM.



# Rational Software Corporation

---

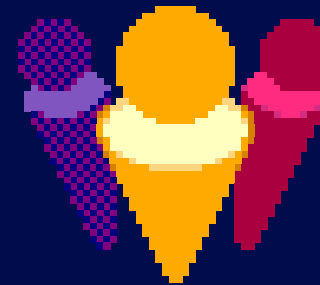
- Founded 1980 – M. Devlin, P. Levy
- Ada programming environment (1985)
  - Object-oriented, in Ada, Hardware+Software
- APEX (2<sup>nd</sup> generation Ada Programming Environment)
- Rational Rose (1992) – Booch method
- Rational Suite (1997)
- 2000: 4000 persons, US\$600 millions revenue
- Acquired by IBM, February 2003

# Outline

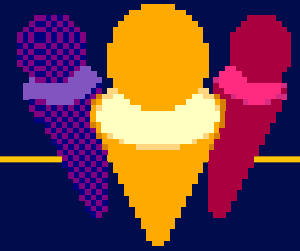
- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

# What's RUP?

- A software development process platform
- An extensible knowledge based
  - Iterative
  - Architecture-centric
  - Use case driven
  - Configurable
  - Supported by a palette of tools
- Sold by IBM
- Educational variant: UPEDU



# What's RUP ?

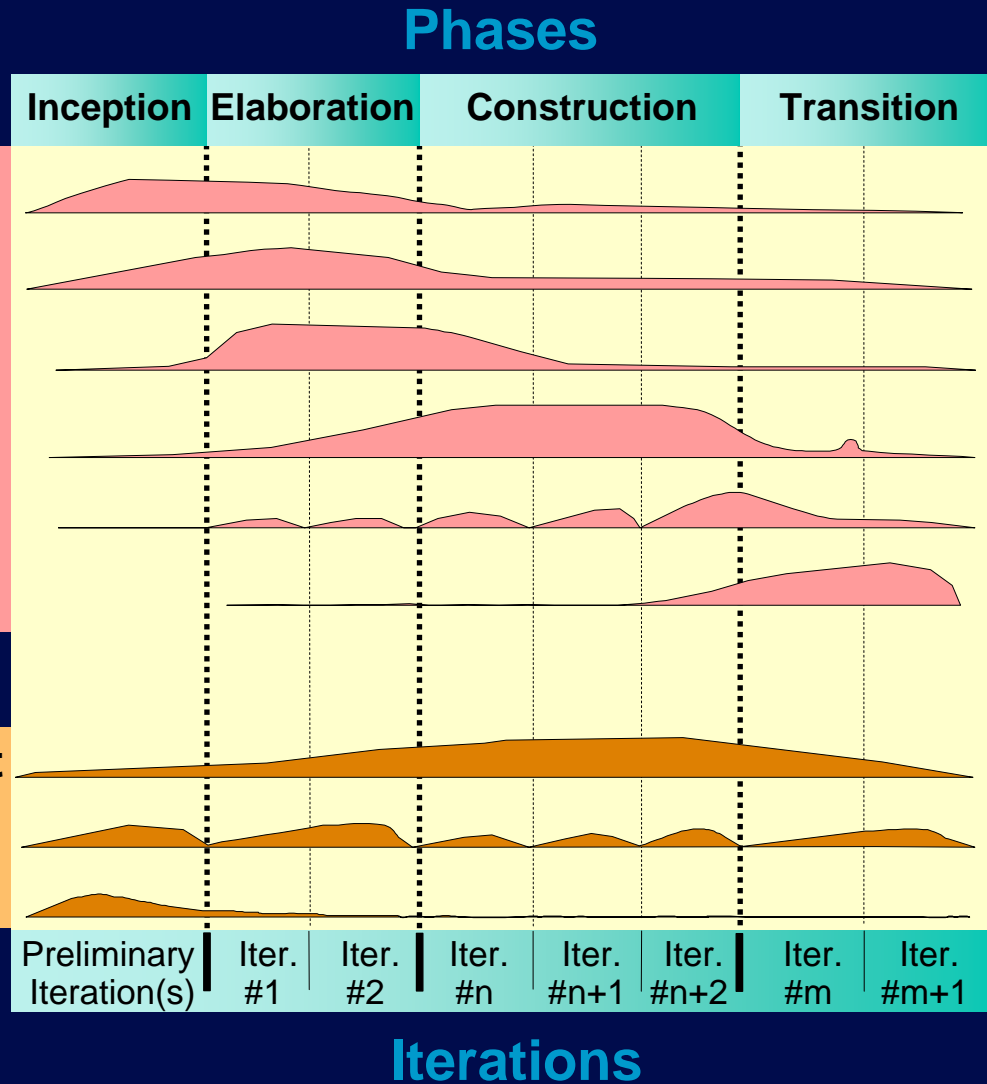


## Core Disciplines

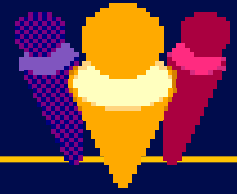
- Business Modeling
- Requirements Analysis & Design
- Implementation
- Test & Assessment
- Deployment

## Supporting Disciplines

- Configur. & Change Mgmt
- Project Management
- Environment

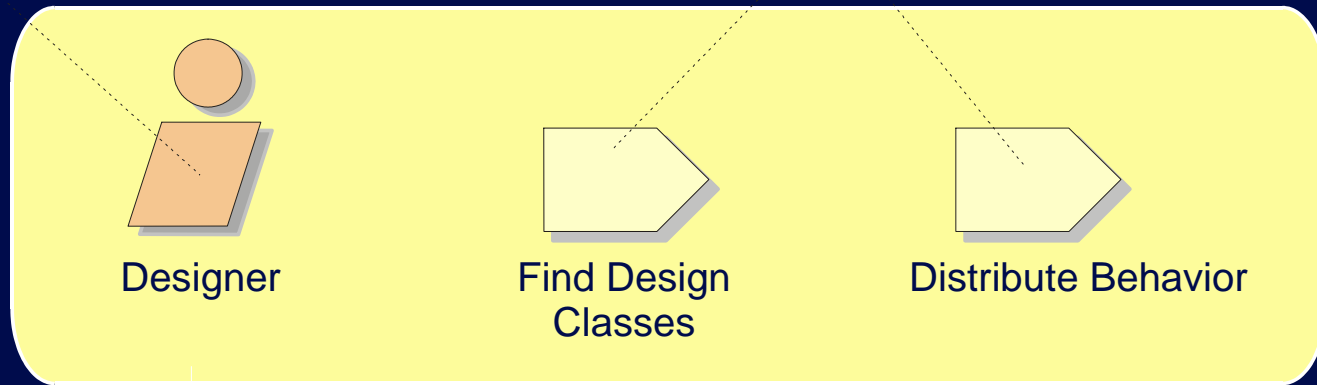


# Role, artifact, activity



Role

Activities

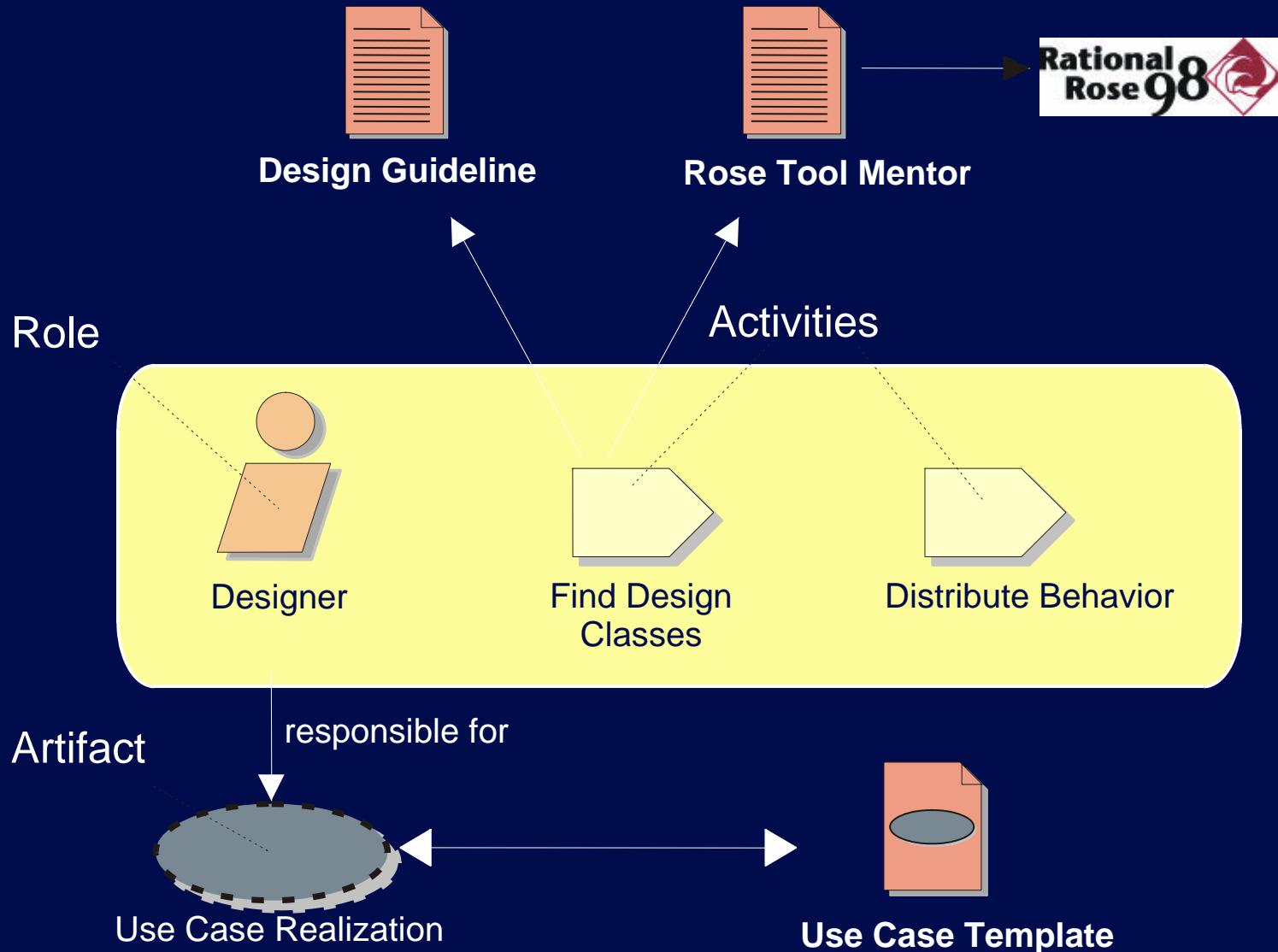
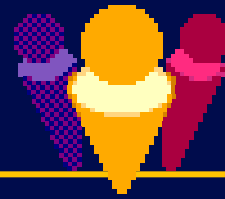


Artifact

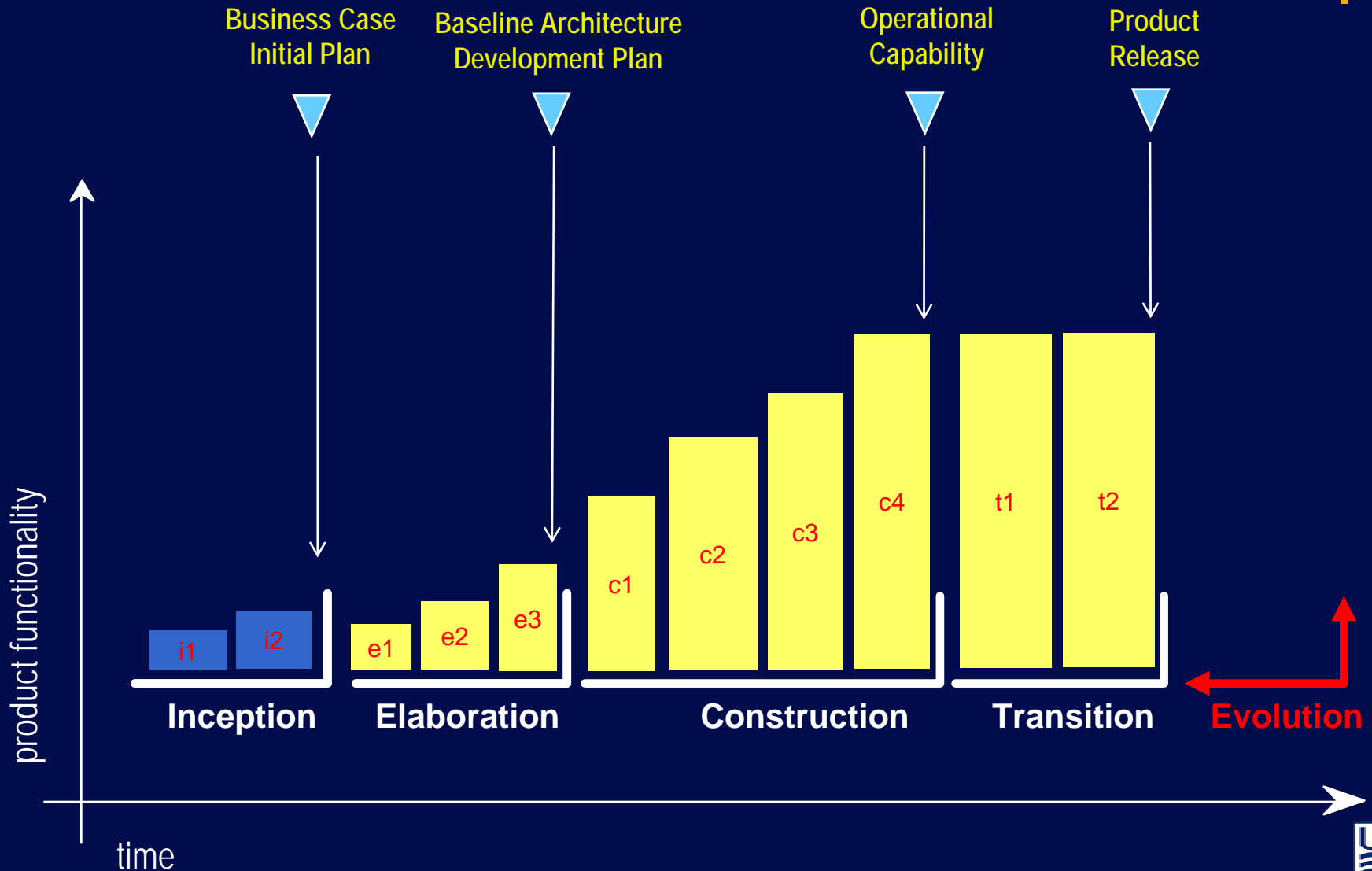
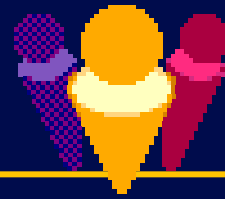
responsible for



# Guidelines, tools, templates, etc.



# Phases, Iterations and Milestones



# Outline

---

- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

Rational Unified Process and RUP are registered trademarks of IBM.



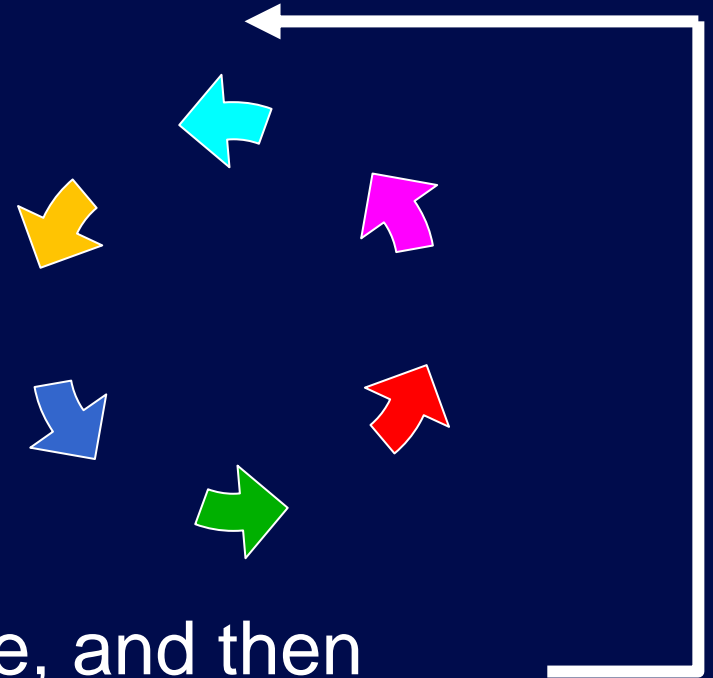
# Side note: Rational Consultants

- Our imperative in 1985:  
make our customers succeed;  
to do so, get involved
- Backlash on CASE tools with  
limited support
  - => meager features, buggy, can't  
scale up
  - => “shelfware”, no repeat sale
- Consultants may become  
deeply involved in projects, up  
to several years.

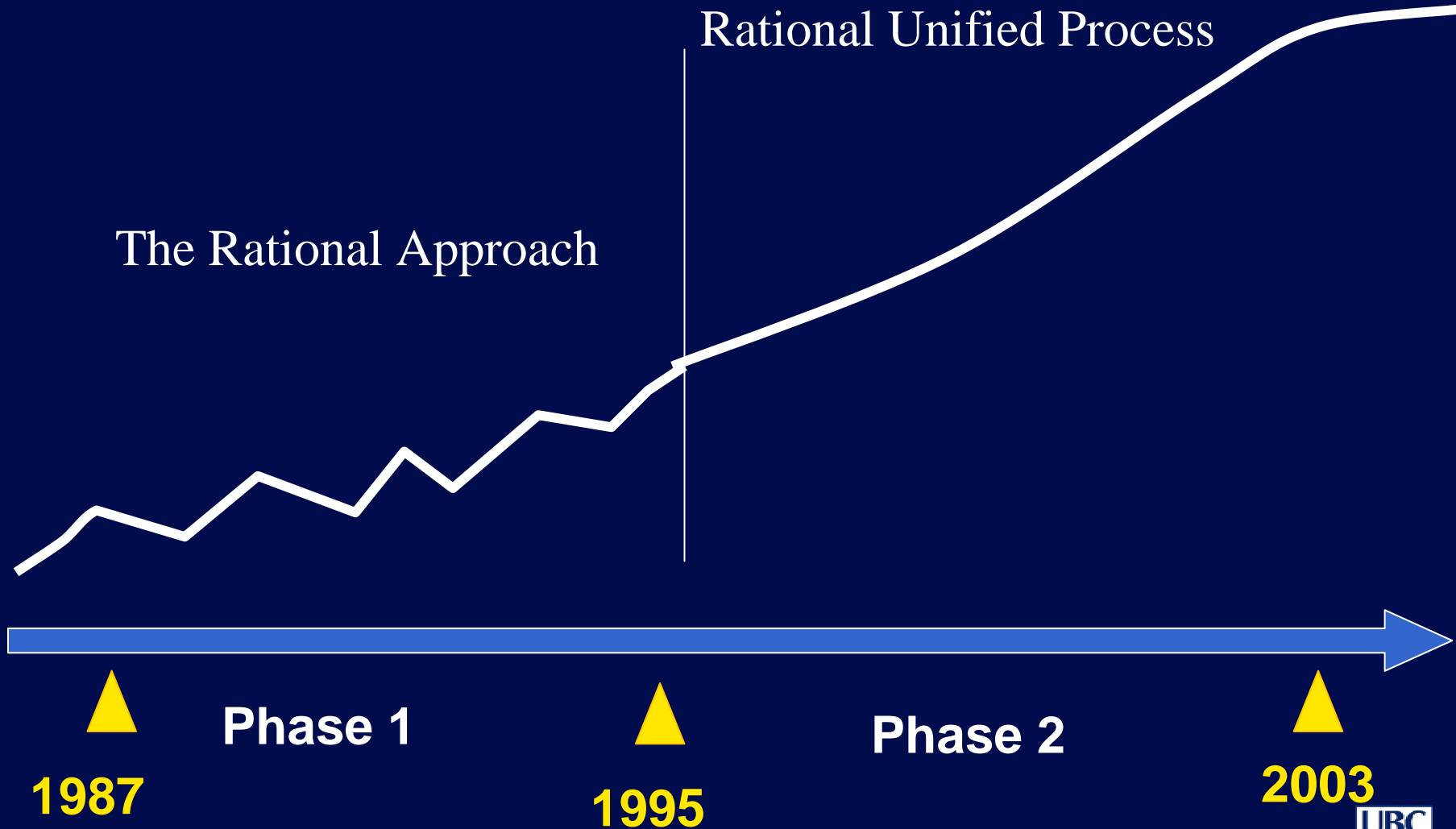


# Learning

1. Accumulating knowledge
2. Integrating knowledge
3. Packaging knowledge
4. Distributing knowledge
5. Using knowledge
6. Reflecting on the knowledge, and then



# Two phases



# Phase 1: The Rational Approach 1987-95

---

- Ad hoc effort, driven by motivated individuals trying to help each other
- Doing a better job as consultants, reuse solutions
- No specific organization; volunteering
- Knowledge with little structure:
  - Iterative development
  - Software architecture
  - Design method (Booch method)

# Accumulating, Phase 1

---

- Series of workshops
- Twice a year, 2 days to a week long
- Off-site, closed
- 12 to 40 participants
- On invitation only (on submission of a contribution)
  
- A first community of practice (1995):
  - Software architecture

# Integrating, Phase 1

---

- Very little integration
  - Definition of key terms and concepts
  - Architecture centric process (1991)
  - Architecture (4+1 views, 1992)
  - Lifecycle, Phases IECT (1995)
- No central repository (besides Newsletter archives)
- Terminology remains major stumbling block

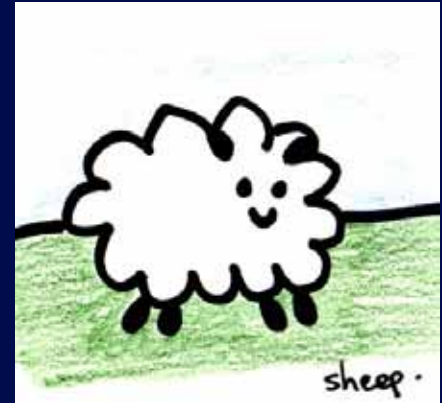
# Packaging, Phase 1

- Collections of papers and slides
  - Available, if you knew whom to ask
- Internal Technical Newsletter: The Sheep
  - Ad hoc, irreverent, lots of humour in between very serious stuff, strictly internal
- A few books:
  - e.g., G. Booch: Object solutions



# Distributing, Phase 1

- Emails (Bulletin board)
  - “The Sheep” (monthly)
  - Whitepapers, articles
  - Book sales
- 
- Training material (Slides on acetate)
    - many consultants manage their own variants



# Using and reflecting, phase 1

---

- Take it and run
- Use the newsletter, email distribution list and occasionally the workshops to bring feedback to the community
- Strong sense of community: people helping each other directly
- Little internal use, beyond general principles (iteration, architecture)

# Outline

---

- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

## Phase 2: Rational Unified Process 1995-03

---

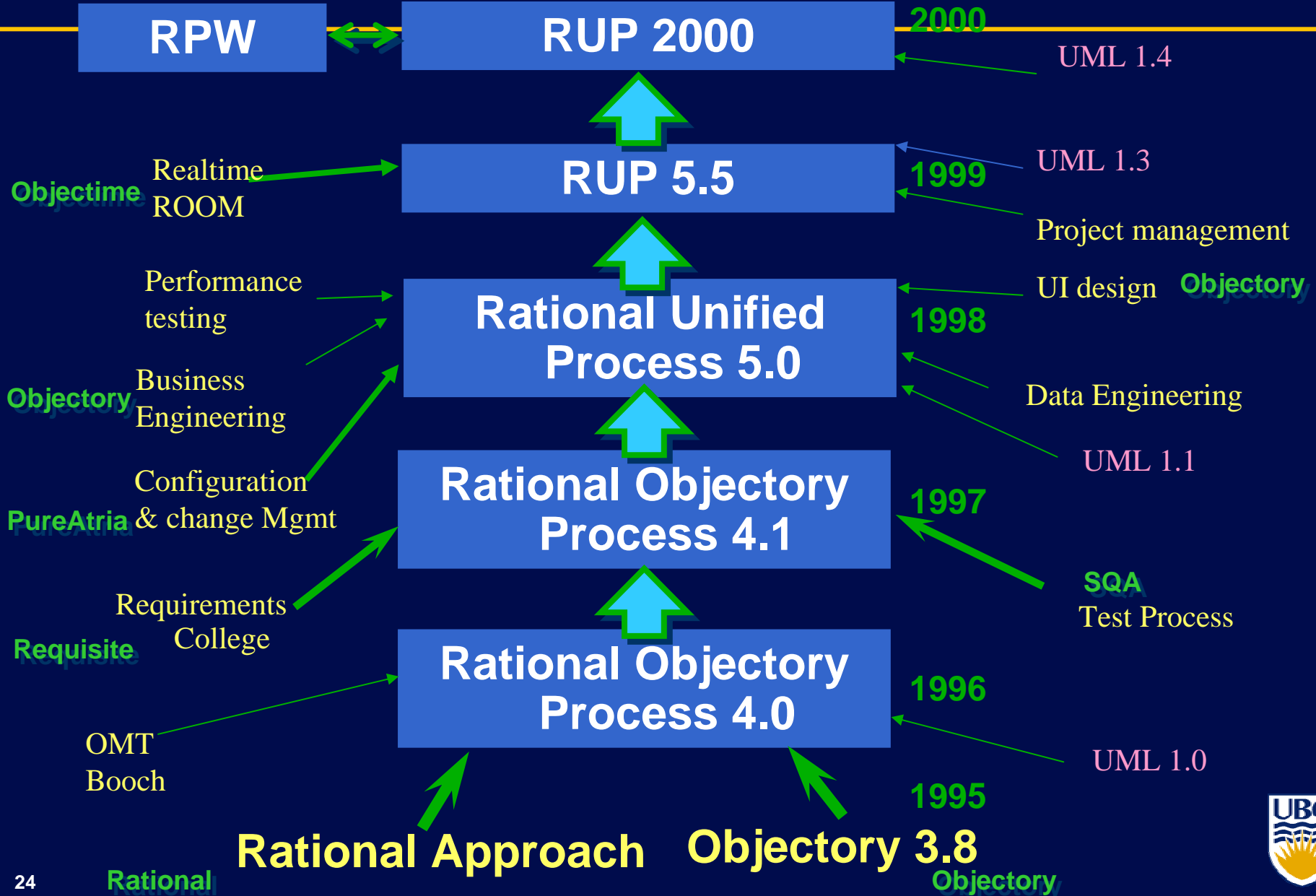
- Product building
- Organized phase: process team, communities of practice, tools, 'method'
- Dedicated staff
- Knowledge becomes more structured
  - ROP then RUP
  - Coverage of all lifecycle: test, requirement, deployment, management
  - Alignment with tool set

# Accumulating, Phase 2

---

- Workshops & newsletter....
- Acquisition of companies
  - Temporary relocation of key individuals
- Acquisition of Missing know-how
- Hiring individuals for their knowledge
- Partnering companies (RUP plug ins)
- Later, systematic review of internal practices (800 developers on 10 sites)
- 5 Communities of Practice (as of 2002)

# Genesis of RUP



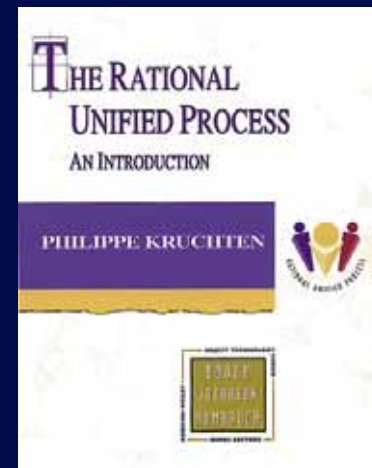
# Integrating, Phase 2

---

- From Objectory model, development of a consistent process knowledge metamodel:
- USPM: Unified Software Process Metamodel
  - an ontology of software process
  - will lead to the OMG specification of SPEM (Software Process Engineering metamodel) in 2000
- Integration done by a dedicated team of process engineers
- Systematic construction of training courses
- CoPs used for review and validation
  - Workshops by CoP

# Packaging, Phase 2

- Rational Unified Process
- Grew to a few million words (2.1M in 2002),
- Dedicated team: 18 people at peak
- First release: 9 books and a CD
- Then conversion to HTML
- Tool support: web site generated out of the process model, based on the metamodel
- Many books
- Sheep becomes web sheep, then a real web based newsletter



# RUP Evolution (before 1998)

---

A small process

Based on

The Rational Approach

&

Objectory



# RUP Evolution (2000)

---

A small process,  
growing bigger

As Rational acquires  
companies and know-  
how



# RUP Evolution (2001)

---

A small process,  
growing bigger.

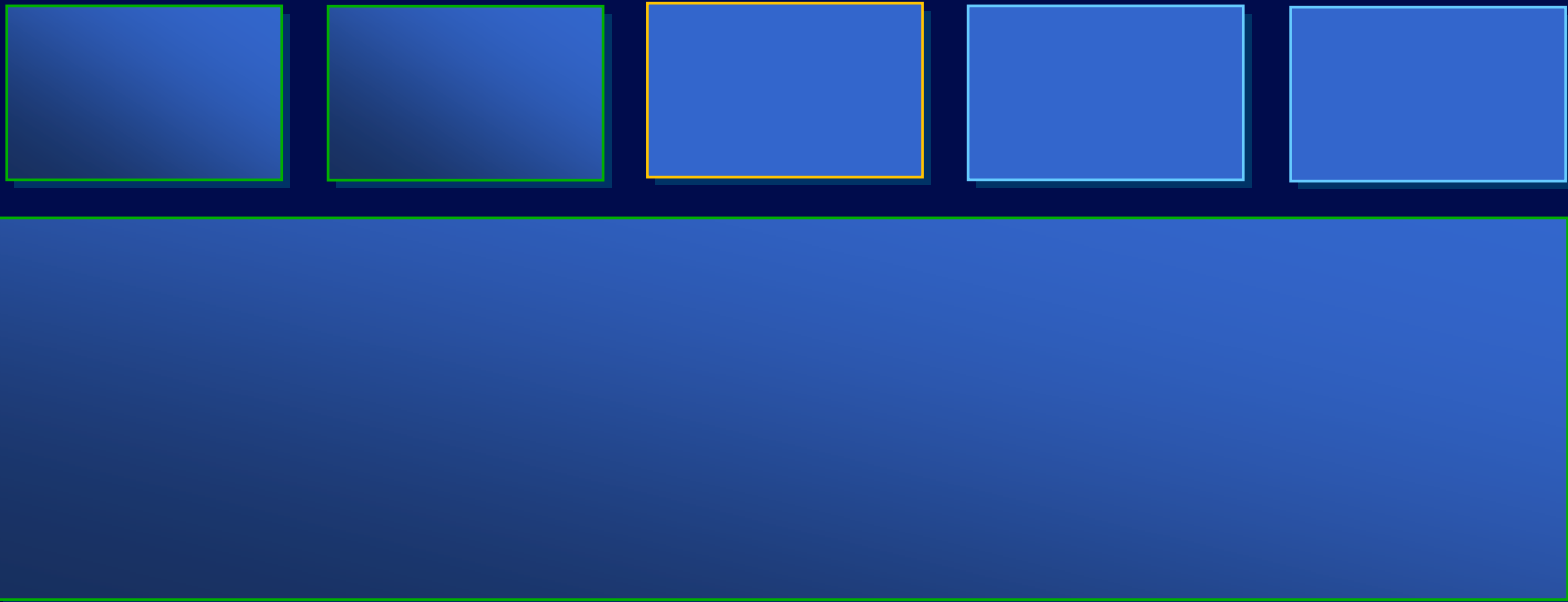
Multiple configurations



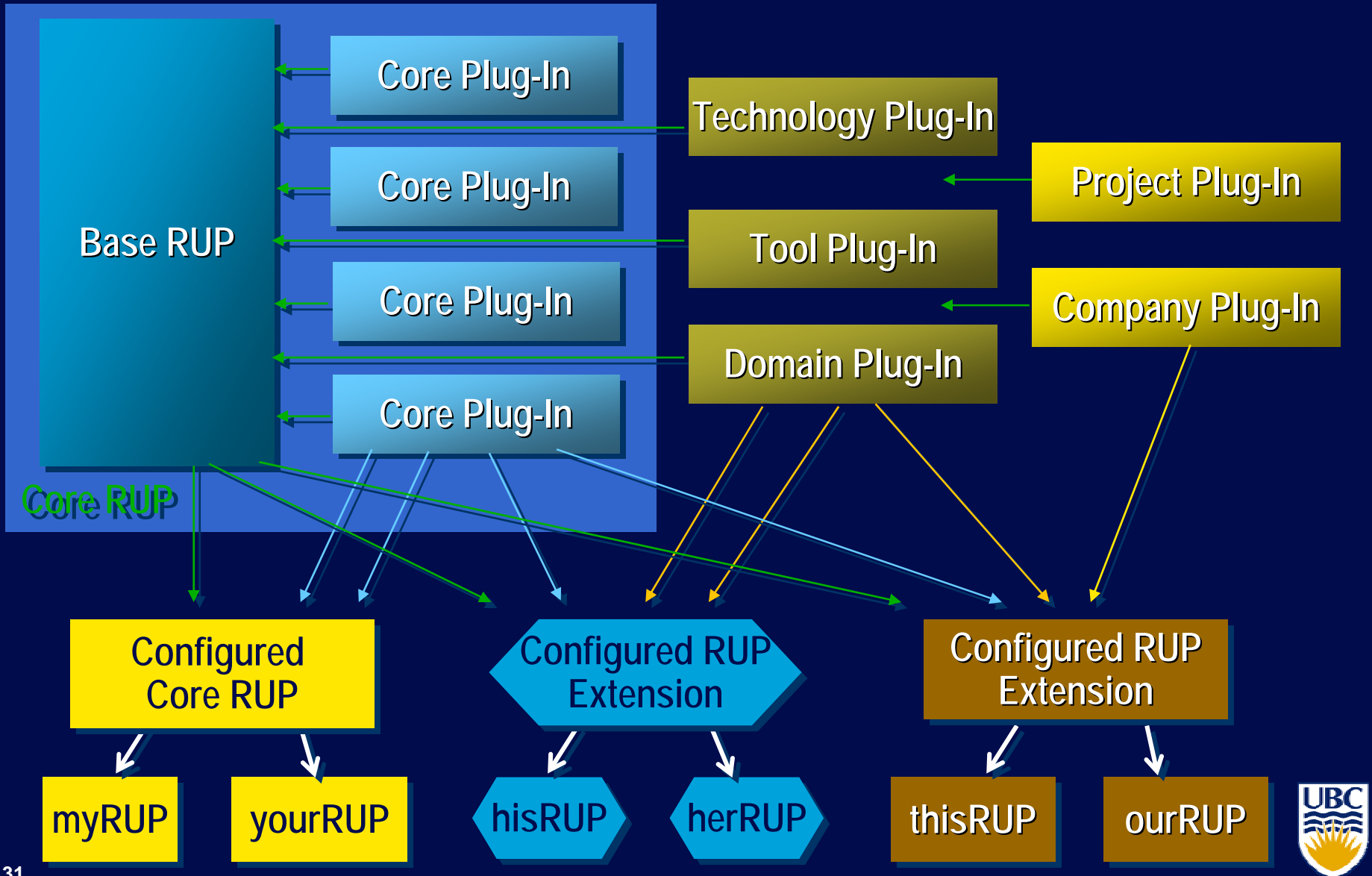
# RUP Evolution (2002)

---

A small set of plug-ins,  
on a big common base



# RUP as a set of components



# Distributing, Phase 2

---

More of the same, plus:

- Process as a product: RUP (\$12M/year)
- Many books (about 35)
- Adding external technical magazines
  - Rose Architect, The Rational Edge
  - Issues of politics, correctness, etc... “Rattle”
- Rational University
  - for training and certification of trainers
- Licensing of training material

# Using and reflecting, Phase 2

---

- Bitching, moaning
- Lesser sense of a company wide community
- Effect of multiple cultures
- Many newcomers just waiting to be “spoon-fed”
- Few active contributors (less than in phase 1 in absolute numbers)

# Outline

---

- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

# Lessons learnt



- Many lessons learnt on **software development**
- Not the topic here today, see RUP and associated literature

# Key lessons learned

---

1. Need a model to structure knowledge
2. Terminology always gets in the way
3. Need dedicated people to make progress
4. Size has a negative impact
5. Fancy technology does not help
6. You simplify, then users simplify more
7. Culture plays funny tricks

# Need a model to structure knowledge

---

- Developing a metamodel (an “ontology”) was a very important milestone
- An enabler:
  - Organized the content
  - Create tool support
- Avoid dispersion, scattering, waste, redundancy

# SPEM

- Software Process Engineering Metamodel
- Developed by a group of companies (IBM, Rational, Unisys, Alcatel, etc.) at the OMG
- A UML profile and metamodel of software engineering process description
- Used by tools such as Rational process Workbench, or Osellus IRIS
  
- Now aiming at a version 2.

# Terminology gets always in the way

- The hardest thing to achieve...Man-years spent!
  - Artifact, workproduct or deliverables, or items
  - Worker and role
  - Workflow and disciplines
  - Task and activities
  - Iteration or increment (or build, or release)
  - Use case and scenarios (or threads)
  - Milestone, gate, guideline, method, subsystem, model, use case, system, ....
- Too many “standards” ...

# Need dedicated people to make progress

- Volunteer model has limited scope
- “Burn out” the real dedicated one
- Too much dependence on a few key individuals, who may depart
- What is the reward model?
- But: Dedicated people implies fixed expenses, implies necessity to justify revenue... ROI, etc.
  - How to put a value on what is learnt?



# Size has a negative impact

---

- The larger the company grew, the harder it was to really “learn”
- Teams do things, but share less and less
- Bureaucracy tends to replace real communication
- Some duplication of effort & some parallelism is probably OK, but integration is made much more difficult later.
- More people, but more consumers, not more producers.

# Fancy technology does not help

- Several times we went too fancy, too early, and had to backtrack

*Examples:*

- Server-based, not portable
- Java-based, not portable (and more)
- OO-based process model (and tools), cute but process engineers do not understand OO concepts
- Process ahead of tools.... ooch, this hurts
- Go to the latest version or tool

# On Simplification



- Simplify, or people won't get it
  - Most people out there are not that smart, and you won't be there to hold their hand
- But then, you are interpreted too literally
  - one size fits all
  - Context is not taken into account
  - Common sense is forgotten
- Lose – lose situation

# Outline

---

- Rational Software Corp., 1980-2003
- The Rational Unified Process<sup>®</sup> (RUP)<sup>®</sup>
- The Making of the RUP
  - Phase 1: the chaotic years 1987-1995
  - Phase 2: the systematic years 1996-2003
- Lessons learned, on the learning process
- Impact of culture of software development

# Culture plays funny tricks

- We think we operate with the same 'base', with the same system of values, but we don't.
- Small cultural misunderstandings
- Large cultural clashes, hard to 'debunk' and to 'debug'
- Ethnic culture and corporate culture



# Global Software Development



- Creating development teams across national borders
  - Mergers and acquisitions, partnerships
  - International projects “by design”: for example EC Esprit program
  - Multinational companies (e.g., IBM, Alcatel)
- More recently
  - Outsourcing (off-shoring?) of software development to India, Thailand, Hungary, Poland, ....
  - Rationale: diff. in manpower cost offsets communication and risks

# Virtual teams



- Half of software development is **communication** between humans
  - Requirements, design, management, reviews
- High bandwidth communication means
  - email, voicemail, teleconference, video, video conference
  - networks, hypermedia, web-based app.
  - collaboration tools: e.g., Groove

# Communication

is affected by the mix:

- Personality
  - Specific to one individual
  - Individual behaviour, attitude
- Culture
  - Shared by a group
  - **Group Values**, behaviours, attitudes



# Culture as an Iceberg

**Arts, literature, language, food, dress, games**

**time, beauty, privacy, values,  
role in society, education, behaviour,  
motivations, fears, etc...**



# Culture and software development?

- Conjecture\*

*A world-wide computer-literate culture, the internet, a programmer (hacker) culture largely dominate the dynamics of these global teams. As a result of the net culture, programmers behave the same in San Jose, Boston, Budapest or Bangalore.*

- I disagree. A blind conception.

- See also “How to behave in country X” books

# Case-lets

- Vancouver – Stockholm development
  - Morning meetings
  - Silence and disapprobation
  - Role in team
- Tokyo – Vancouver – Santa Clara
  - Negotiating a relocation
  - Hierarchy
- Paris – Santa Clara
  - Hugs and kisses
  - Lunch with the enemy



# Sociology and anthropology

---

- Models to reason about culture
- Edward Hall, 1975...
- Gert Hofstede, 1980...
- Alan Fiske, 1990
- Fons Trompenaars, 1995...

# Meeting other cultures

---

- Ethnocentric stage

- Denial (blame issues on personality or misbehaviours)
- Defense (and try to force things one way)
- Minimization (push it under the rug)

- Ethnorelativist stage

Not one culture is central and reference for judging others

- Acceptance
- Adaptation
- Integration
- *xenophilia* ?

# Cultural factors: Edward T. Hall

---

- Low context, high context
  - HC: unspoken meanings (jp, cn, fr)
  - LC: just what the words say (us, de)
- Time:
  - Polychronic
    - many things interleaved (Middle east, France)
  - Monochronic
    - one thing at a time, “time is money” (US, Scand.)

Source: E. Hall

# Cultural factors: G. Hofstede

---

IBM employees around the world

Multivariate analysis, lead to 5 dimensions:

- Power distance
- Collectivism versus individualism
- Femininity versus masculinity
- Uncertainty avoidance
- Long-term versus short-term orientation

Source: G. Hofstede

# Other factors: F. Trompenars

---

- Universalism vs. particularism
  - Judging on fixed rules, or based on circumstances ?
- Individualism vs. communitarianism
  - Self, or group?
- Neutral vs. emotional
  - showing emotions in business setting?
- Specific vs. diffuse
  - How far do we get involved?

Source: Trompenars



	Neutral	Emotional
Specific	USA (east coast), Scand. Approval/disapproval	USA West coast, Canada Sympathy/Outrage
Diffuse	Japan Esteem/Disrespect	South of Europe Love/Hate

# Other factors: F. Trompenars (cont.)

---

- Achievement vs. ascription
  - attitude toward titles, degrees,...

And a few secondary ones, such as:

- Attitude to time
- Attitude to the environment (i.e., nature)
- Gender, race, class, religion

Source: Trompenars



# Impact on software development

---

- Management
- Communication
- Meetings
- Task allocation
- Requirement
- Negotiation
- Bug reporting

# Wrap Up

---

- Building a **knowledge product** by drawing from one's own collective experience is a learning process.
- It goes beyond observing oneself to be better (like process improvement), as it needs to be integrated, packaged and distributed.
- We also learned a few things on the learning process itself: model, structure, organization, and **culture**.