

PROMOTER/RENOIR Summer School
on Software Process Technology

30 June – 4 July 1997, Grenoble, France

Process Modelling Languages:
Needs, coverage,
alternatives, evaluation.

Revised for IFIP WG2.4 meeting 22.-26.9.97, Estes Park,
Colorado.

Reidar Conradi,
Dept. of Computer and Information Science (IDI),
Norwegian University of Science and Technology (NTNU)

Phone: +47 73.593444, Fax: +47 73.594466, Email: conradi@idi.ntnu.no

Contents

| | | |
|----------|--|-----------|
| 1 | Outline of Talk | 5 |
| 2 | Introduction on PMLs | 6 |
| 2.1 | Initial Remarks | 6 |
| 2.2 | Some Experiences with PMLs and PSEEs | 7 |
| 3 | Process Parts and their Modelling | 8 |
| 3.1 | Simple process support scenario | 8 |
| 3.2 | Overall Taxonomy of Process Parts | 9 |
| 3.3 | More detailed Taxonomy of Process Elements | 10 |
| 3.4 | General characteristics of PMLs | 15 |
| 3.5 | Special characteristics of PMLs | 16 |
| 4 | Process Modelling Languages: One or many? | 17 |
| 4.1 | The right PML? | 17 |
| 4.2 | Possible Groups of PMLs | 18 |
| 4.3 | One or several PMLs? | 19 |
| 4.4 | Process Modeling Scenarios | 24 |

| | | |
|----------|---|-----------|
| 5 | Possible submodels/sub-PMLs for the different process elements | 26 |
| 5.1 | BA.ACT: Activity or Task Submodel | 26 |
| 5.2 | BA.PROD: Product submodel | 28 |
| 5.3 | BA.HUM: Human/role submodel | 29 |
| 5.4 | PR.ORG: Organization/project submodel | 30 |
| 5.5 | US.WS: Workspace | 31 |
| 5.6 | US.TOOL: Tool submodel | 32 |
| 5.7 | US.INTE: Tool Integration Model | 33 |
| 5.8 | US.VIEW: User-view | 34 |
| 5.9 | VA.EVOL: Meta-process submodel | 35 |
| 6 | Assessing PMLs/PSEEs against Typical Software Process Categories | 37 |
| 6.1 | Cross-match of 9 process elements and 10 process categories | 41 |
| 6.2 | Evaluation of some PMLs / PSEEs | 42 |
| 6.3 | Assessment of 5 PMLs / PSEEs against 9 process elements | 43 |

| | | |
|----------|--|-----------|
| 6.4 | Assessment of 5 PMLs/PSEEs against 10 process categories | 44 |
| 6.5 | Some more detailed comments | 45 |
| 6.6 | Future Work on PML/PSEE Assessments | 46 |
| 7 | Proposals for future PSEEs and PMLs | 47 |
| 8 | Conclusion on PMLs | 49 |

1 Outline of Talk

- What is a PML, some experiences.
- Process parts and concepts: the need for modelling.
- General PML characteristics.
- Special PML characteristics.
- Core vs. pre-defined vs. own vs. surrounding PMLs?
- A new support architecture: web, experience base, ... The role of the PML.
- Conclusion and future research. Interoperability and standardization.

2 Introduction on PMLs

2.1 Initial Remarks

- What is purpose of a PML:
medium for communication, analysis, enactment, and/or improvement?
- Software people love new concepts, languages, and models!
Seek user requirements and assessment criteria.
- And what is so special about software process?
Look at conceptual modelling, ...
- One or several PMLs?
Standardization and interoperability: UML, etc.
- Relation to PSEE tools?
- Relationship between researchers, vendors, and developers?

The starting point for industrial SPI is:

- rather informal process modeling,
- almost no computer-assisted process enactment,
- problems to follow-up real process evolution,
- few on-line and shared experience databases,
- coarse-level and project-oriented metrics,
- a need to cover all SPI meta-phases in PDCA [Dem84],
- a need to disseminate SPI results to all personnel.

On the other hand, there is:

- increased use of web-based project handbooks,
- a growing use of distributed technologies,
- a growing process awareness.

2.2 Some Experiences with PMLs and PSEEs

PML experiences:

- Flow formalisms OK for modelling and analysis at AT&T [Vot93].
Finding: 30% of task inputs/outputs are not connected.
- Process templates at NASA-SEL [BG94] and SEI [PWCC95] are described by informal flow notations.
- Four enterprise modelling groups at Statoil in Norway in 1994–96 just “invented” informal flow notations [Tot97].
- At Siemens Telecom in Norway: neither APEL [EDA97] or Process Weaver [Fer93] not cost-effective in process modelling.
Also did not want too detailed process modelling.

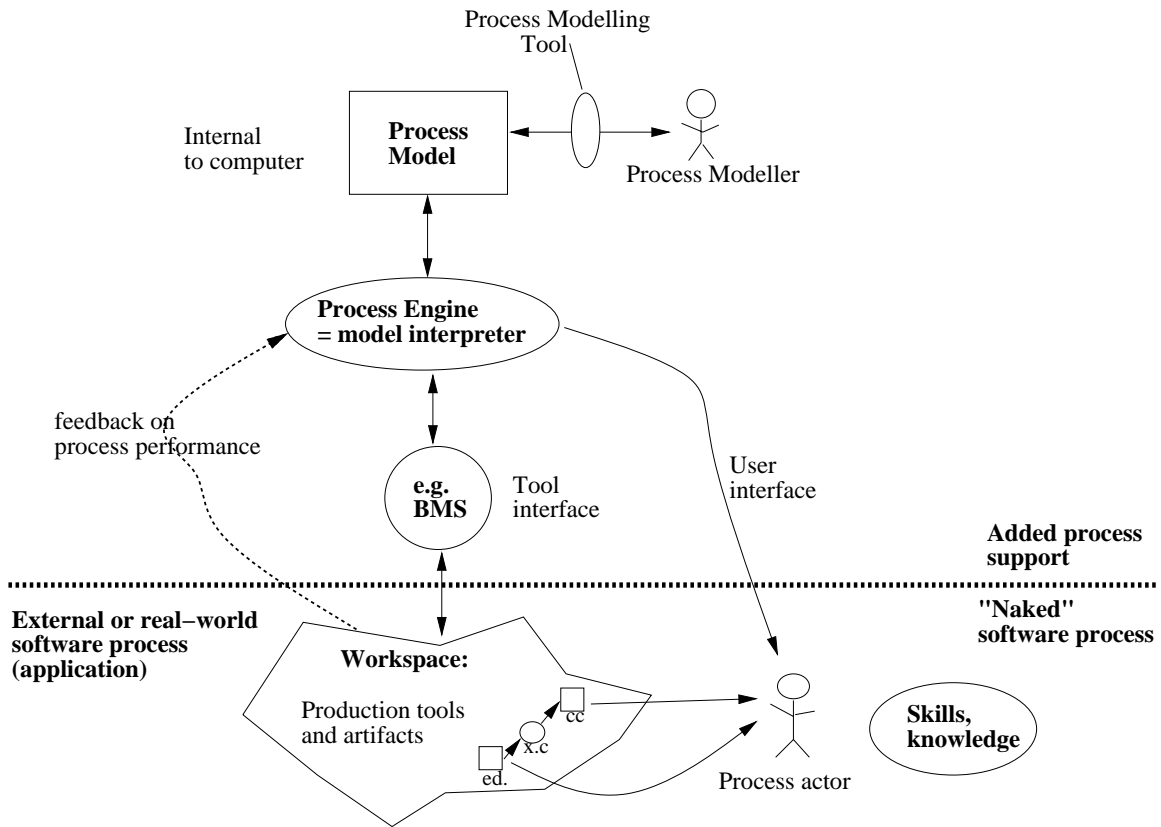
Simpler notations were preferred, present PMLs too “weird”.

PSEE experiences:

- Process Weaver saved 40% time for certain CM-related QA tasks at Siemens Telecom, yet not cost-effective [Høy97].
- Sysdeco A.S., Oslo experienced 100 changes on user requirements in one project (1994), all demanded and paid by the customer!
- Actual process evolution at NOVIT software house at task network level, not template level [NWC97].
- Conventional project management tools, such as MS/Project, cannot cope with process evolution during execution [Håk94].

3 Process Parts and their Modelling

3.1 Simple process support scenario



Questions:

- How to evolve the Process Model during interpretation?
- How does Process Engine communicate with Process?
- External Process in “synch” with internal Process Model?
Conformance: descriptive, prescriptive, proscriptive?

3.2 Overall Taxonomy of Process Parts

Three parts in a (whole) software process:

- **Production process:** normal software production, employing computerized tools and humans.
- **Meta-process:** evolves the entire process.
- **Process support:** computerized tools (PSEEs) that manipulate a process model expressed in some PMLs, plus process guidelines/methods.

All these process parts are partly in the computer, partly outside.

Process model = Product process model + meta-process model.
Evolve process on-the-fly, incremental and dynamic bindings.

3.3 More detailed Taxonomy of Process Elements

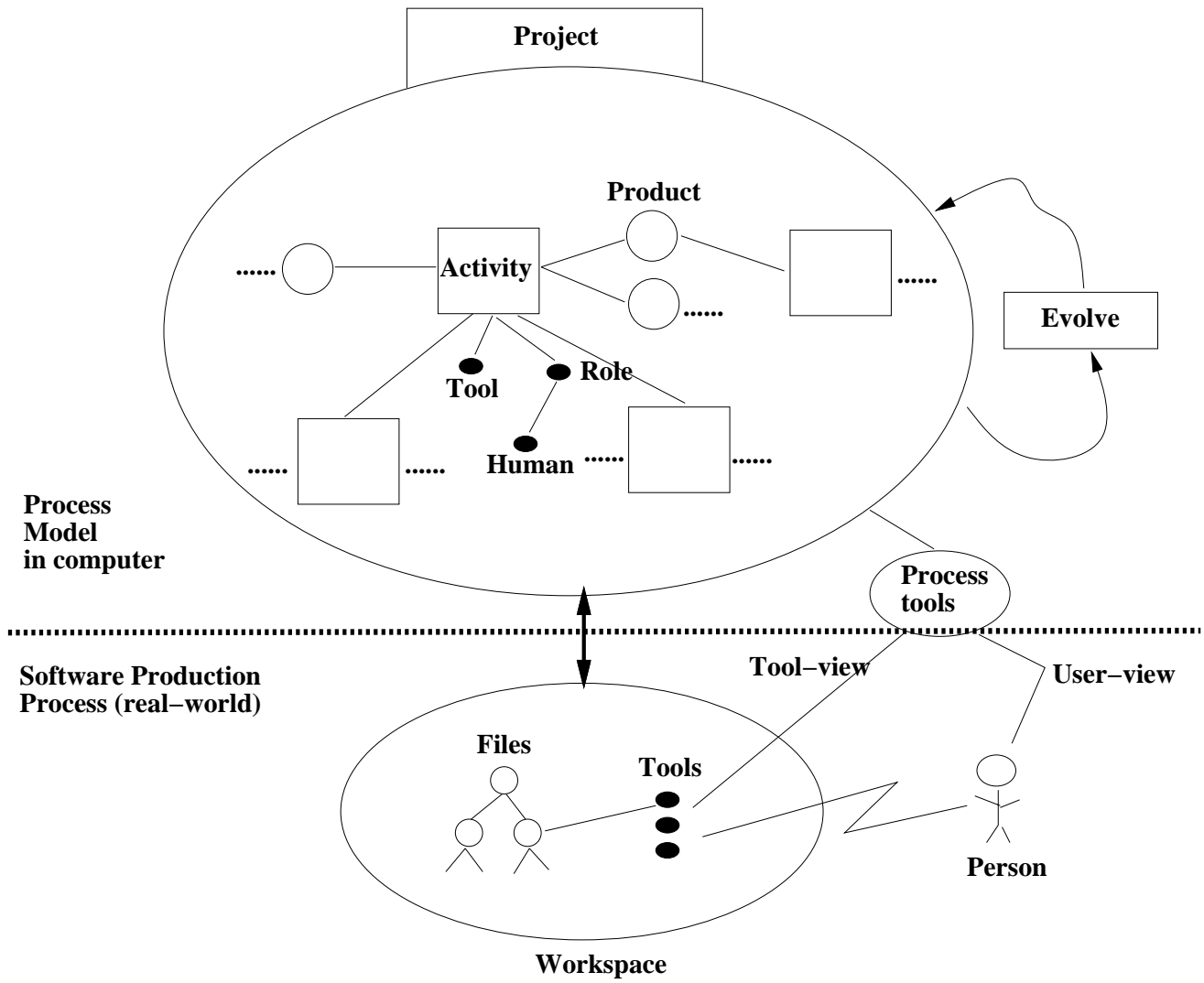


Figure 1: Example of process elements.

Taxonomy of Process Elements [CE94]:

Three **basic, BA** process elements:

- 1. Activity, BA.ACT:** A concurrent process step, working on software artifacts and coupled to a human role and to external production tool(s).
Can be at different abstraction levels, i.e. decomposed activities.
- 2. Product, BA.PROD:** A software artifact, persistent and versioned, simple or composite, with mutual relationships.
- 3. Human/role, BA.HUM:** A role describes the rights and responsibilities of an activity/tool.
The role can be filled by humans with that role capability.

One **project-level, PR** process element:

- 4. Organization/project, PR.ORG:** Organizations consist of humans, and have relationships to each other and to other process elements. Normally project organizations.

Four process elements on **user support, US**:

5. **Workspace, US.WS**: A workspace (WS) contains and controls the artifacts for the actual (sub)process. Often as **files** checked out from a repository.
6. **Tool, US.TOOL**: A “enveloped” production tool (external procedure), with possibilities for automatic and monitored execution.
7. **Tool-view, US.INTE**: I.e. PSEE integration with foreign tools and workspaces.
8. **User-view, US.VIEW**: General user interface to help comprehend the (enacting) process model.

One process element on process model **variability, VA**:

9. **Evolution Support, VA.EVOL**: general meta-process support for static or dynamic variability of the process model (i.e. by reflection).

“Software artifacts have longer life-time and are more stable than their models!”, Jacky Estublier at EWSPT’92 [Der92].

Plus associated metrics on all above.

Relations between the process elements:

- a. BA.ACT is the most fundamental.
- b. BA.PROD assumes BA.ACT – its “siamese twin”.
US.TOOL and BA.HUM is also on this level.
- c. US.WS and US.INTE assume all above.
- d. PR.ORG assumes all above.
- e. VA.EVOL and US.VIEW apply to all above.

Illustration of these relations:

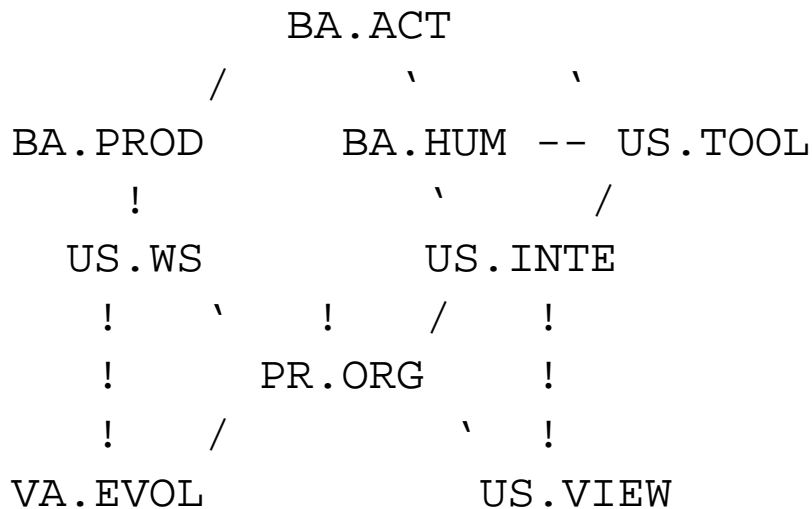


Figure 2: Modelling dependencies between process elements.

Some auxiliary process elements:

1. **Versioning/transaction model:** Versioning of process model repository, associated transactions.
2. **Quality/performance model:** Overall quality model (product/process).
3. **Cooperation model:** Cooperation protocols, sharing / locking etc.
4. Etc.

NB: Taxonomy is not exhaustive or final!

3.4 General characteristics of PMLs

- **Understandability**: diagrammatic notation?
- **Simplicity** and orthogonality: apply “Occam’s razor”!
- **Formalization** and conciseness: e.g. process program vs. process rules, ...
- **Expressiveness**: domain and m-p coverage.
- **Multiple perspectives/views**.
- **Abstraction**: hierarchical decomposition, gradual refinement.
- **Modularization** (“vertical”): encapsulation, OO subtyping, subprojects, reusable submodels / architectures, ...
- **Customization** and **evolution** (“horizontal”): reflection, parameterization, dynamic binding, versioning, copy-overwrite, ...
- **Monitoring and feedback** mechanisms.
- Etc. etc. ...

3.5 Special characteristics of PMLs

Four primary goals for **process models** [Kel89]:

- Enable effective communication regarding the process.
- Facilitate reuse of the process.
- Facilitate management of the process (enactment support).
- Support evolution of the process.

A set of **PML requirements** (same ref.):

- Use a highly visual approach to information representation, such as diagrams.
- Support multiple, complementary viewpoints of the process.
- Support multiple levels of abstraction.
- Enable compendious descriptions.

Summary:

- Core PML cover most of 9 above process elements, with BA.ACT, BA.PROD, US.TOOL, BA.HUM, and VA.EVOL – with BA.ACT as most crucial.
More high-level PMLs should express the other elements.
- Human-oriented activities:
partial models, evolution, flexible enactment, good UI.
- Respect audience: intended usage, existing tool platform.

4 Process Modelling Languages: One or many?

The million dollar question!

4.1 The right PML?

Much discussion of the “right” PML, or several PMLs.
Decide along many design axes:

- **Technical issues:**
 - application domain (technical/managerial, single/team),
 - expressive power,
 - meta-process focus (understand, analyze, enact),
 - structuring/evolution properties,
 - automation level (control vs. assistance),
 - implementational feasibility, ...
- **Non-technical issues:**
 - market availability and price,
 - standardization,
 - interoperability, ...

Ex. In first meta-process phase: flow diagrams and prose OK.

Ex. Semantic information models and conceptual models:

Product + process modeling by OO/ER + Petri nets.

Ex. Product modeling by placeholders in Process Weaver.

Ex. Federated models/DBs.

Need **process support** for all this: process concepts, formalisms, methods, tools, entire environments.

4.2 Possible Groups of PMLs

Four groups of PSEEs / PMLs:

- **Product-centered** on software engineering databases:
EPOS, Adele etc.
Often object-oriented.
- **Activity-centered:** MARVEL, MERLIN, OIKOS, Adele–Tempo (rule- or trigger-based, implicit), Process Weaver, SPADE, MELMAC, EPOS (network/Petrinet-based, explicit), Arcadia (imperative).
Tasks: as proc calls, concurrent objects, or with internal threads?
But process models are more than concurrent programs!
- **Project-centered:** CADES, ISTAR, the Virtual Design Team. MicroSoft/Project.
- **Role-based:** like PWI and many groupware systems, speech-act.

Alternative PML sub-categories:

- **Functional:** what activities are performed (BA.ACT).
- **Behaviour:** when and how they are performed (BA.ACT, US.TOOL).
- **Organizational:** where and by whom they are performed (PR.ORG, BA.HUM).
- **Informational:** artifacts, processes and their relationships (BA.PROD).

4.3 One or several PMLs?

Four approaches **L1–L4** for PML design:

- **L1: One fixed and large core PML.**
The PML primitives cover all relevant process elements.
E.g. **BA.ACT, VA.EVOL.**
Ex. “Hard” primitives for concurrent activities.
- **L2: One extensible and smaller core PML.**
The PML primitives allow declarative declarative constructs.
E.g. **BA.PROD, BA.HUM, US.TOOL.**
Ex. “Soft” primitives, such as extensible types.
- **L3: one core and several compatible sub-PMLs.**
More high-level sub-PMLs (APEL or statecharts), can be down-translatable to a core PML.
E.g. **PR.ORG, US.WS, US.TOOL, US.INTE.**
Ex. Check-out/in of WS, or user interface paradigm.
- **L4: one core and several incompatible sub-PMLs.**
The other sub-PMLs are outside and not translatable to a core PML.
E.g. **Cooperation, versioning / transactions, quality / performance.**
Ex. Transaction protocols, or quality models (metrics).

But may not control design space.

And lift perspective up from low-level activity formalisms!

Coexistence of possibly **inhomogeneous** and partial process models/tools.

Standardization efforts, cf. IWFC/PIF, STEP/CDIF, OMG, and UML.

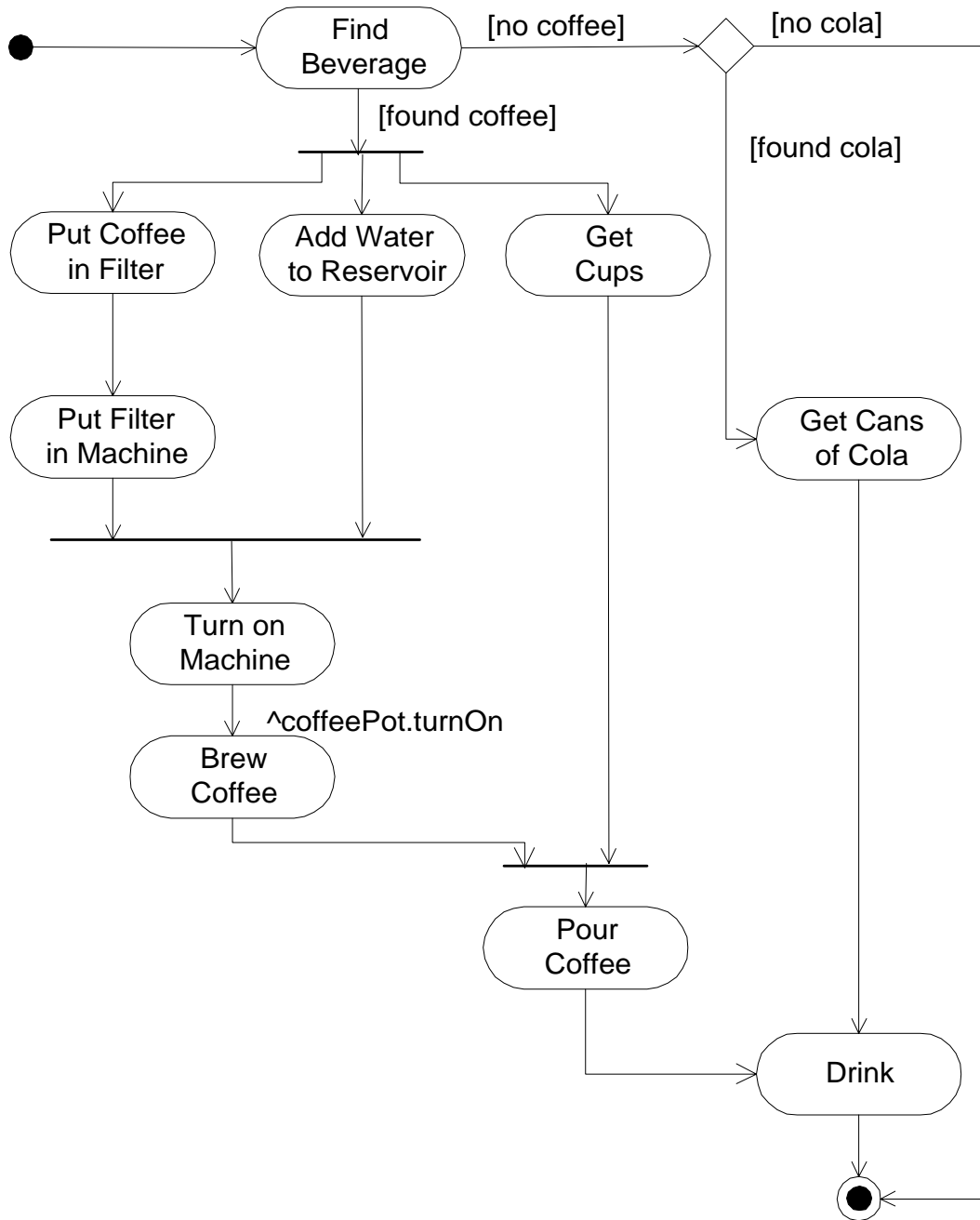
Ex. **UML coverage**, from OO product models to enterprise models:

- **Sterotypes**: meta-types (Event, Exception, Metaclass, Utility),
- **Static structure diagrams**: class and object diagrams,
- **Use case diagrams**: stimuli-response,
- **Sequence diagrams**: cf. CRCs,
- **Collaboration schemas**: role diagrams,
- **State diagrams**: cf. statecharts,
- **Activity diagrams**: cf. DFDs,
- **Implementation diagrams**: component and deployment diagrams.

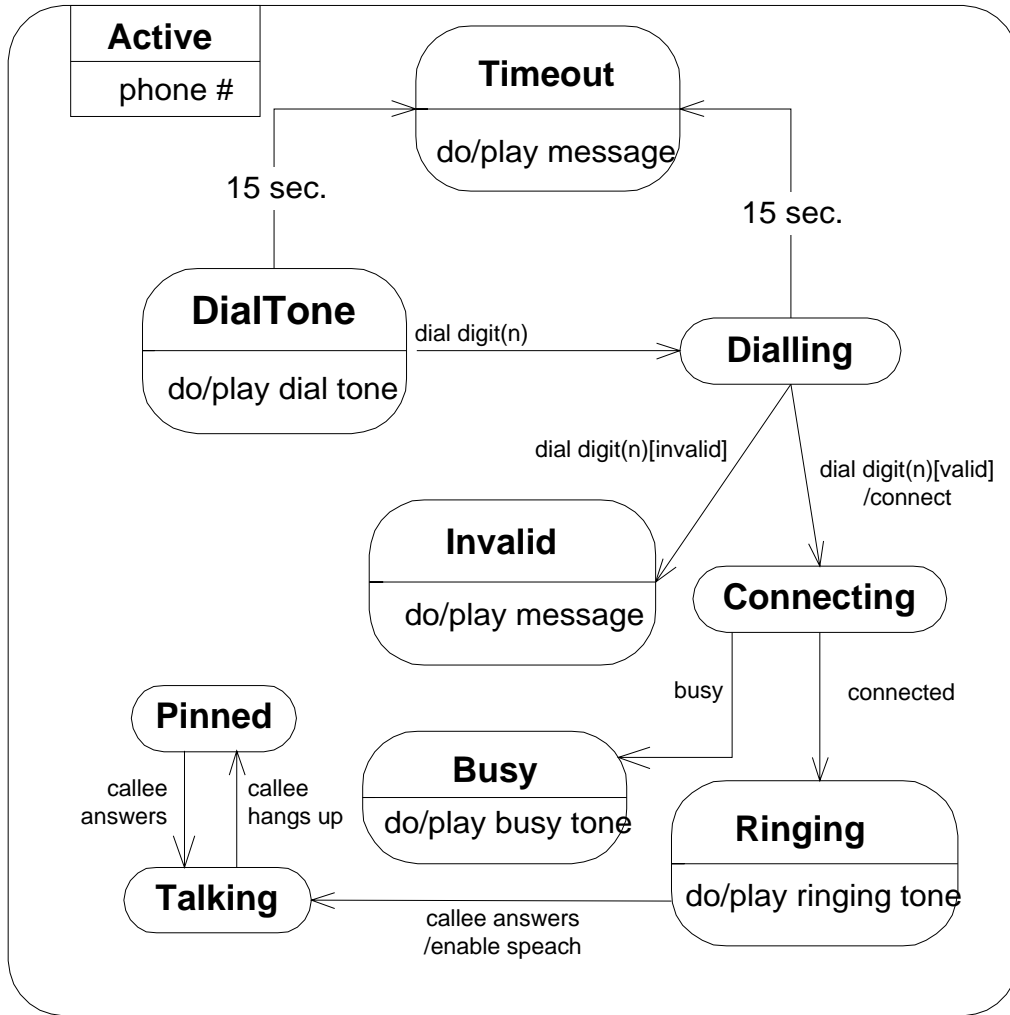
Ex. **Process Interchange Format, PIF**, – initiated by MIT, ARPA, IWFC, ... – with the following subtypes of Entity:

- Activity, subtype: Decision
- Object, subtype: Agent
- TimePoint
- Relation, subtypes: Creates, Modifies, Performs, Uses, Before, Successor, Activity-Status

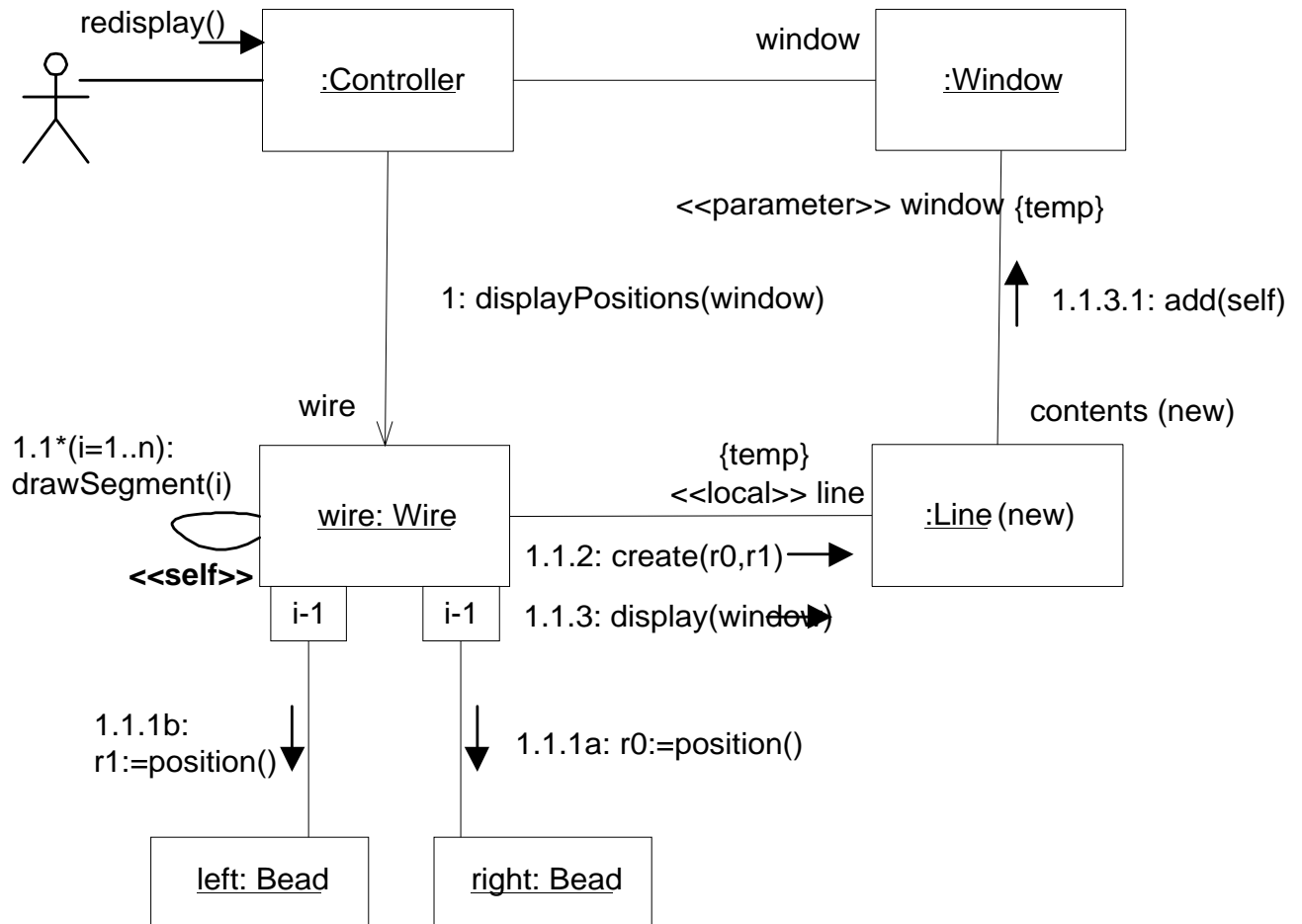
UML activity diagrams:



UML state diagrams:

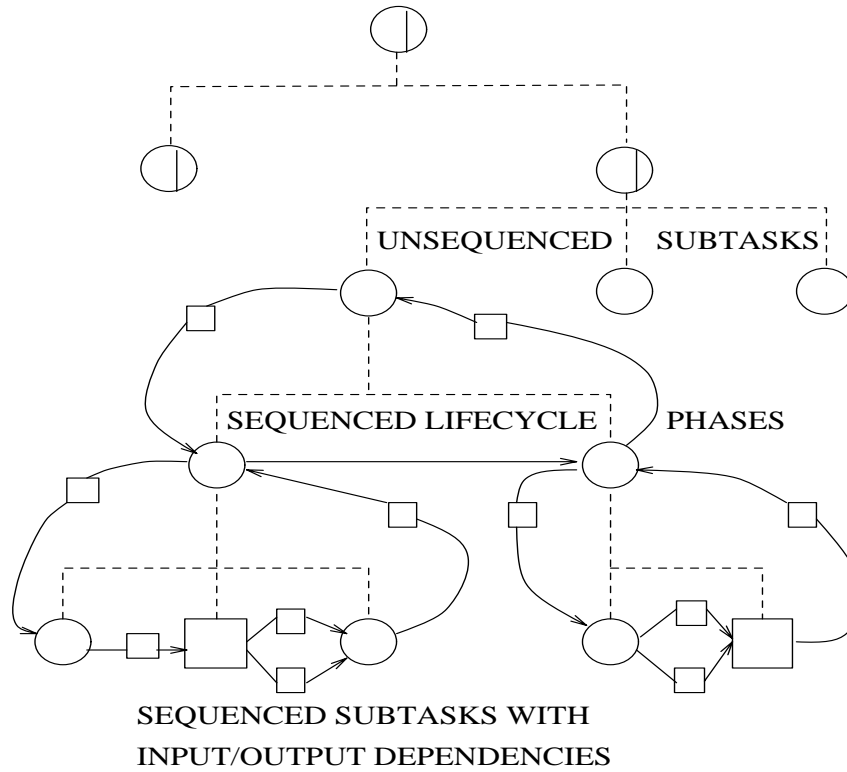


UML collaboration diagrams:



4.4 Process Modeling Scenarios

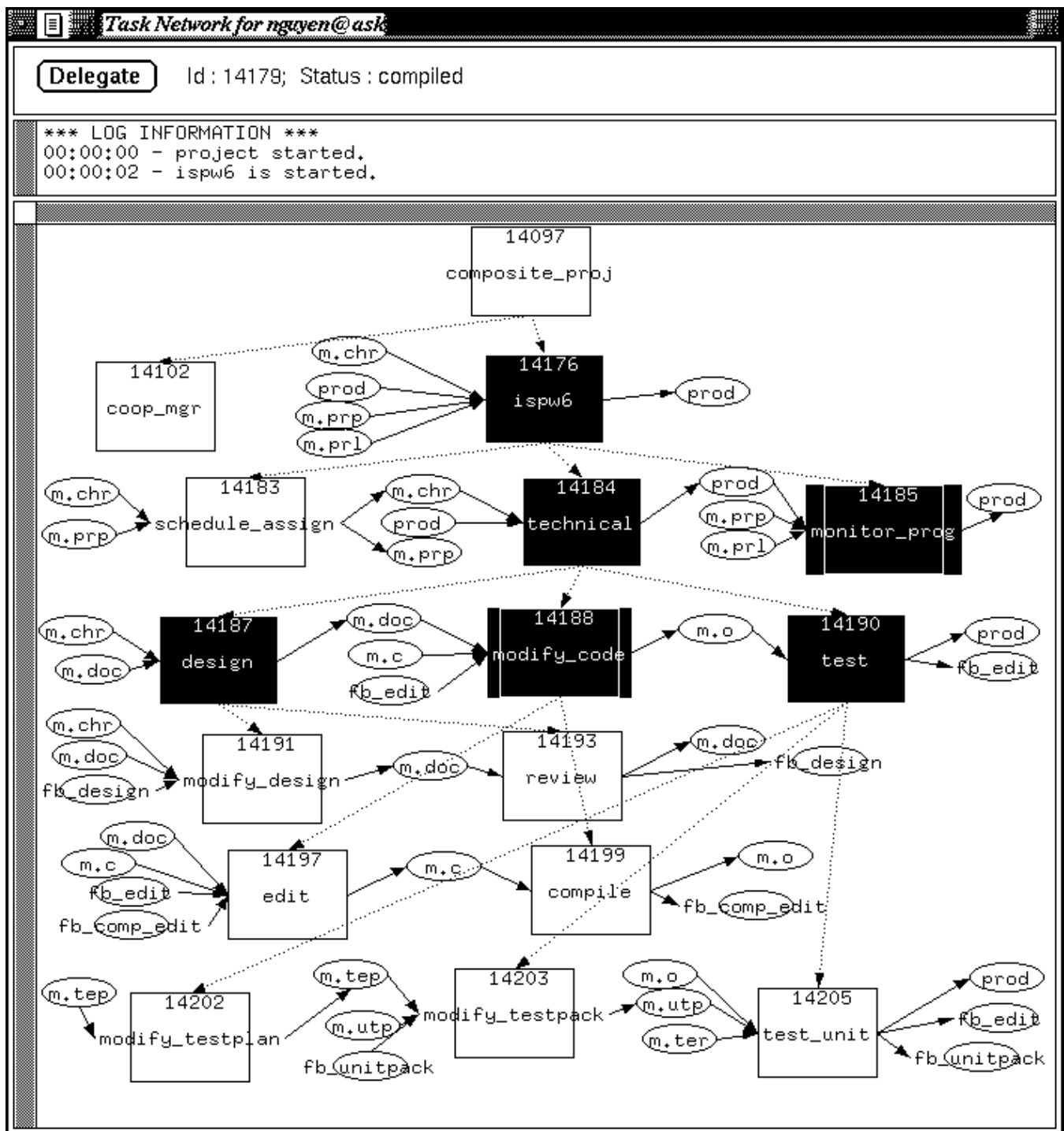
Ex. Conventional activity (task) network, with vertical breakdown and horizontal chaining



LEGEND:

- ⊖ PROJECT
- TASK PERFORMED BY HUMAN
- TASK PERFORMED BY DERIVER
- ◻ INPUT/OUTPUT OBJECT
- TASK DECOMPOSITION
- (SUB)TASK SEQUENCING

Ex. EPOS task network to model the ISPW6 reference example



5 Possible submodels/sub-PMLs for the different process elements

5.1 BA.ACT: Activity or Task Submodel

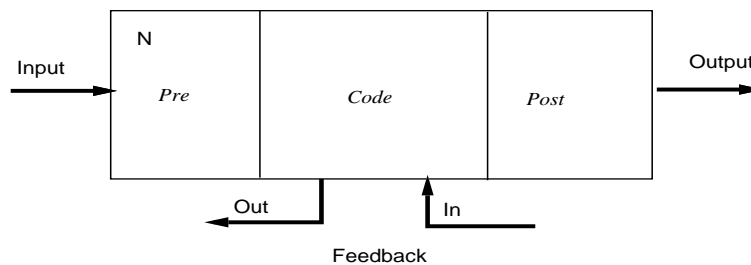
Activity = task = free-standing “program” part.

Task **instance** = **active object** in **task network** (e.g. “Petri net”):

Inputs/Outputs (actuals), internal state, ...

Task **type** = **activity “rule”**: PRE/POST, CODE script with named tool, formals, ...

Ex. **Basic Task Cell**, performs a transformation:



Some low-level, activity formalisms:

1. **Active DB**: triggers as Adele2, deductive as EDBLOG.
2. **Rules**, as PRE–CODE–POST in MARVEL, OIKOS.
3. **Task Network**, extended Petri nets in MELMAC, SPADE.
Cf. statecharts at the type-level [Har87].
4. **Process Programming Languages**: APPL/A, PWI.
5. **Hybrids**: as in EPOS and most PMLs.

Merge formalisms 1 and 2?

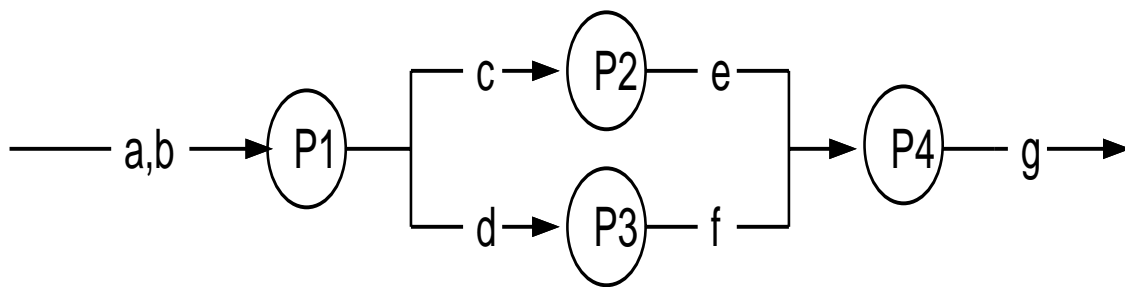
Ex. Process program vs. tasks vs. rules:

1. Process model, as a sequential/parallel program:

```
P1(a,b; c,d); P2(c;e) | P3(d; f); P4(e,f; g);
```

```
% a,b is read; c,d is written; etc.
```

2. Process model, as a more free-standing task network:



3. Process model, as facts and unbound rules:

Facts: a, b, c, d, e, f, g.

```

Rules:  IF Pred1(a,b) THEN P1(a,b; c,d);
        IF Pred2(c)    THEN P2(c; e);
        IF Pred3(d)    THEN P3(d; f);
        IF Pred4(e,f)  THEN P4(e,f; g);
  
```

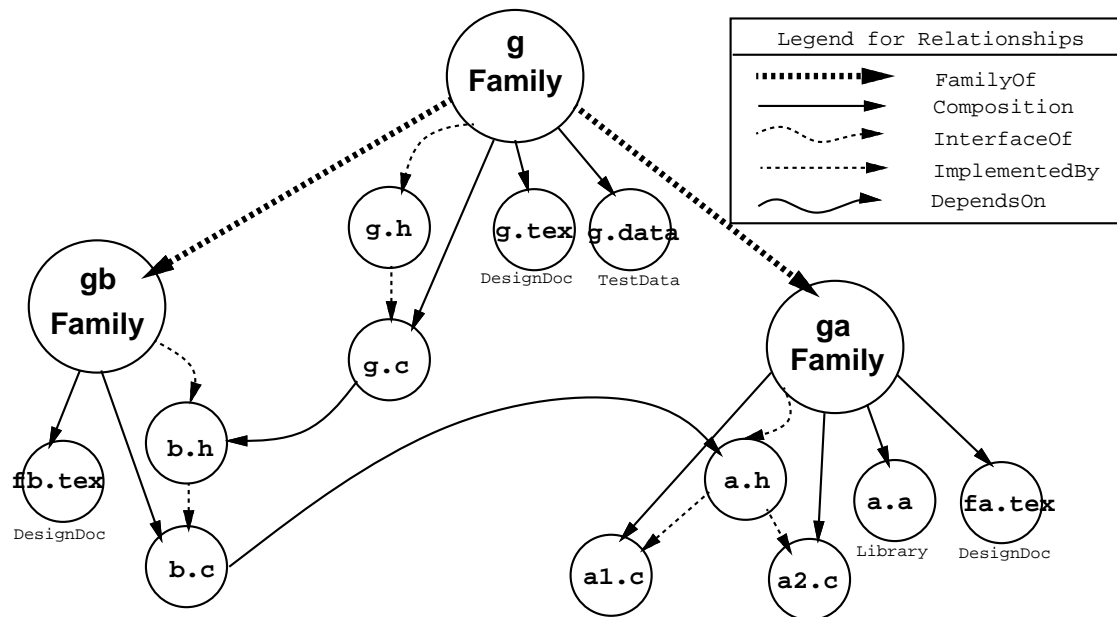
5.2 BA.PROD: Product submodel

Product: passive objects, connected by relationships
(task = active object).

Product type: often defined in an OO ERA formalism,
maybe with ADT-like procedures, triggers, and constraints.
I.e. in a database schema?

Pre-defined product models available for certain domains
(SEE, VLSI, CIM, CSCW), and supported by tools.

Ex. Product families with Interface/Body, and with Part_Of and Depends_On relationships.



5.3 BA.HUM: Human/role submodel

- **Role:** generic person (“placeholder” for an activity), with certain rights and plights.
E.g. a manager, tester, or programmer.
- **Agent:** concrete person filling a role, thus responsible for specific objects or operations.

Cf. many **speech-act** systems à la Winograd-Flores [Win88].

5.4 PR.ORG: Organization/project submodel

- **Project:** often the main “process”. Technically a high-level task to control sub-DBs/workspaces. Contains management data, subprojects, resources (time, HW/SW), quality model, ...
Unit of process customization / evolution.
- **Team:** set of agents
- **Organization:** set of persons and resources, goals, constraints, policies/methods.
Can be seen as an envelope around projects.

Different goals and technologies for:

- Enterprise level
- Overall project level
- Technical project level

Strong overlap with project management!

For organizational modeling: lots of technology available.

5.5 US.WS: Workspace

As mentioned, a workspace contains and controls the artifacts for the actual (sub)process, often as check-out files.

The number of files may be big (1000s), and partly shared.

Also the tools may be configured here.

Look at middleware technologies here: CORBA, ...

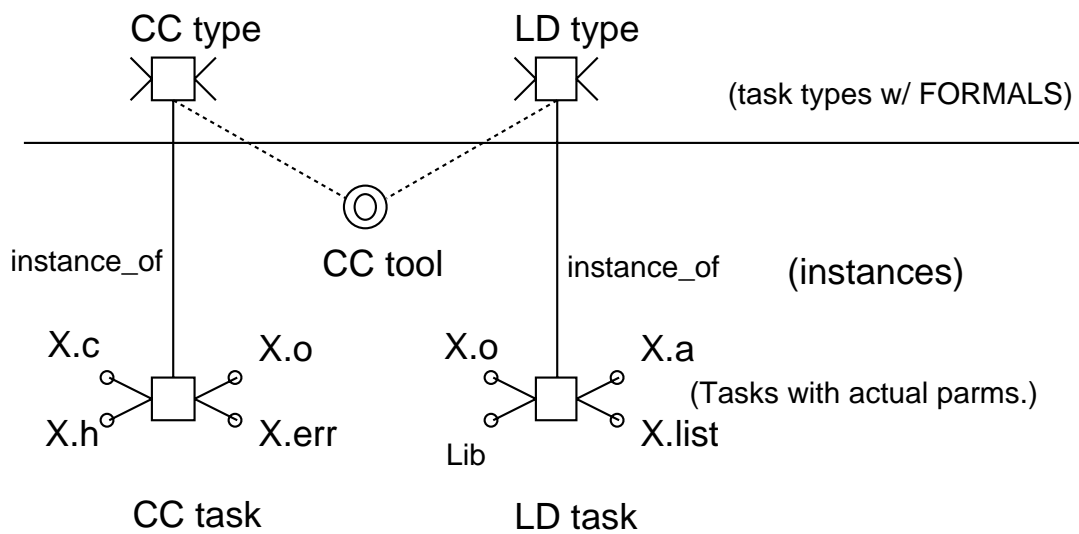
5.6 US.TOOL: Tool submodel

Tool: passive object = “external” PROC with (file) name, inputs/outputs (FORMALS), tool switches, ...

Tool invocation (an activity): often by a task, serving as a tool envelope.

Its type describes a tool, with PRE/POST and CODE (a script).

E.g. Tools coupled to tasks.



How to model interactive tools?

5.7 US.INTE: Tool Integration Model

Often using a **BMS** (Broadcast Message Server) for flexible tool communication in a computer system (Field, HP SoftBench, DEC FUSE) [Rei90].

Both used for PSEE-tool and tool-tool communication.

PSEEs applying BMS technology: Process Weaver, SPADE, ...

The simpler RPC can be used for point-to-point communication.

Also look at middleware technologies here.

5.8 US.VIEW: User-view

The user interface is very important [Mye89], but how to display complex information comprehensively?

Work agendas also play a part here.

Very important subject: for m-m interaction in general, and to receive an understandable **view** of a knowledge universe (a “lens” into this).

Large pool of technologies, including www client-interfaces against PSEE servers (PWI-web, EPOS-web).

5.9 VA.EVOL: Meta-process submodel

Process lifecycle, as meta-activities:

- **MP0. Technology Provision:** PMLs, predef. models, tools, ...
- **MP1. Process Specification:** produce a simplified process description. Requirement elicitation, human communication, sense making.
Informal.
- **MP2-3. Process Analysis and Process Design:** produce a high-level process architecture.
More formal (“dry runs”).
- **MP4-5. Process Implementation and Process Enactment of above design:**
 - **Making the process support:**
 - Process model:
 1. Make template process model (e.g. types).
 2. Make enactable model (e.g. activity network).
 3. Make enacting model (and interpret this model).
 - Process tools (PSEE): ...
 - **Shaping the real-world process:**
e.g. by migration tool.

Formal, many details (“wet-runs”).
- **MP6. Process Assessment:** monitoring, evaluation, evolution.
Partly detailed, partly overall.

No specific lifecycle model imposed: waterfall, evolutionary, ...
Incremental model creation and evolution
⇒ diffuse borderlines between model variations.

Many mechanisms to assist process model evolution:

- Flexible and reconfigurable architectures,
- Reflection and interpretation,
- Schema evolution and -versioning with subtyping etc.,
- Impact analysis,
- Restricting access rights, modularization etc.

Also some method support for evolution: Prism [Mad92], ...

6 Assessing PMLs/PSEEs against Typical Software Process Categories

The categories are at a very coarse granularity level; need much refinement to serve as standard, reference models.

Intra-phase work – simple categories for single person:

1. Clerical Work (Cler.Work): filling out forms, answering letters and emails, ... – often using computerized tools, and with notifications to others.

Main characteristics: BA.ACT, PR.ORG, US.INTE.

2. Technical Work (Tech.Work): developing, fixing, enhancing, and testing a program module. This involves many software artifacts in a workspace, with proper synchronization against others.

Main characteristics: BA.PROD, US.WS, US.TOOL, US.INTE.

3. Groupware-related: i.e. distributed meetings, involving interactive, concurrent modifications of fine-grained documents.

Main characteristics: BA.HUM, PR.ORG, partly VA.EVOL.

Inter-phase work – aggregate categories for several persons:

4. Business: can be described by a fixed workflow of clerical activities.

Main characteristics: BA.ACT, PR.ORG.

5. Cooperative work (Coop.Work): can be described by a non-fixed workflow between technical work in different lifecycle phases. May instrument groupware.

Main characteristics: BA.PROD, US.WS, US.INTE.

6. Software Production (SW.Prod): can be described by an overall workflow of the production phases, with **big** variation in technology (waterfall, Cleanroom, OO, ...). It is based on the previous categories.

Main characteristics: BA.ACT and BA.PROD, US.INTE, VA.EVOL.

7. Project Management (Proj.Mgmt): emphasizes planning, coordination and control – often at a coarse granularity – and is often coupled to the previous category.

Main characteristics: BA.ACT, PR.ORG, BA.HUM

Example of inter-phase work: Processing a MR:

For 4. Business view: An MR document is written for each bug report, then sent to the developer, forwarded to the validator, and finally submitted to the releaser.

For 5. Cooperative view: Ongoing work must be adjusted to reflect the new MR work, including a revised validation protocol.

For 6. Software Production view: The MR is a requirement specification that must be implemented all the way, with proper feedbacks.

For 7. Project management view: Will plan and follow up the next product release, according to MRs accepted by a CCB meeting.

Three **meta-process phase** categories:

8. MP1-MP3, Process model definition (Meta.Def): to formally define and analyze/simulate process models.

Main characteristics: BA.ACT, PR.ORG, US.VIEW, VA.EVOL.

9. MP4-MP5, Process model enactment (Meta.Enac): to execute a process model.

This implies workspace management, guiding the user and invoking tools, assembling metrics during process enactment, cooperating with other processes/users etc.

Main characteristics: BA.ACT, BA.HUM, US.TOOL, US.INTE, US.VIEW.

10. MP6, Process assessment and evolution (Meta.Evol): This evolves the process model **after** start-up, based on collected metrics and observations – either to adjust (e.g. replacing a sick developer) or to improve (e.g. adding a review).

Main characteristics: BA.ACT, PR.ORG, US.VIEW, VA.EVOL.

6.1 Cross-match of 9 process elements and 10 process categories

Using a scale 0..5¹, where 5 is best.

| Process category | BA.ACT | BA.PROD | BA.HUM | PR.ORG |
|------------------|--------|---------|--------|--------|
| 1. Cler.Work | 4 | 2 | 1 | 1 |
| 2. Tech.Work | 2 | 5 | 3 | 0 |
| 3. Groupware | 2 | 1 | 5 | 2 |
| 4. Business | 5 | 2-4 | 2 | 2 |
| 5. Coop.Work | 3 | 5 | 3 | 2 |
| 6. SW.Prod | 4 | 3 | 2 | 2 |
| 7. Proj.Mgmt | 4 | 1 | 2 | 5 |
| 8. Meta.Def | 2 | 3 | 3 | 5 |
| 9. Meta.Enac | 5 | 4 | 4 | 4 |
| 10. Meta.Evol | 2 | 3 | 3 | 4 |

Figure 3: Process categories according to BA and PR process elements.

| Process category | US.WS | US.TOOL | US.INTE | US.VIEW | VA.EVOL |
|------------------|-------|---------|---------|---------|---------|
| 1. Cler.Work | 1 | 3 | 4 | 2 | 3 |
| 2. Tech.Work | 5 | 4 | 5 | 3 | 2 |
| 3. Groupware | 2 | 1 | 2 | 1 | 5 |
| 4. Business | 1 | 1 | 2 | 3 | 2 |
| 5. Coop.Work | 4 | 1 | 5 | 4 | 4 |
| 6. SW.Prod | 3 | 2 | 4 | 3 | 4 |
| 7. Proj.Mgmt | 2 | 1 | 1 | 4 | 4 |
| 8. Meta.Def | 2 | 2 | 3 | 4 | 1 |
| 9. Meta.Enac | 5 | 5 | 4 | 4 | 3 |
| 10. Meta Evol | 1 | 2 | 2 | 4 | 5 |

Figure 4: Process categories according to US and VA process elements.

¹Perhaps a trinary scale will suffice? – i.e. NO, PARTLY, YES.

6.2 Evaluation of some PMLs / PSEEs

1. **ADELE** at LGI in Grenoble since 1982.
ADELE-DB: versioning, own DDL/DML with dynamically bound schema and triggers, workspace support.
ADELE-TEMPO offers object roles (dynamic types) and process interconnections.
NB: later APEL high-level PML not considered.
2. **EPOS** at NTNU (ex. NTH) in Trondheim since 1989.
Uniformly versioned EPOSDB to store reflexive process models in nested transactions. Process model in SPELL (Prolog) is a typed “dataflow” network, with BMS-connected tools.
3. **SPADE** at Politecnico di Milano since 1990.
Process model is a generalized and reflexive Petrinet with tokens denoting OO products, all stored in O2. Tool/user communication separately via a BMS (DEC-FUSE).
4. **Process Weaver** by Cap in Grenoble since 1988 (now FlowWeb).
Process model is an extended Petrinet. Activity modeling by project clusters, with file workspaces and BMS-connected tools.
5. **Process Wise Integrator** by ICL since 1986.
Role–activity–interaction paradigm, strong on evolution. Roles represent humans or tools. Own PML is translated to PS-Algol.

6.3 Assessment of 5 PMLs / PSEEs against 9 process elements

| PML/PSEE | BA.ACT | BA.PROD | BA.HUM | PR.ORG |
|----------|--------|---------|--------|--------|
| 1. ADELE | 3 | 5 | 2 | 2 |
| 2. EPOS | 3 | 5 | 1 | 2 |
| 3. SPADE | 4 | 3 | 1 | 1 |
| 4. PrWe | 4 | 1 | 2 | 2 |
| 5. PWI | 4 | 3 | 3 | 2 |

Figure 5: PMLs/PSEEs according to BA and PR process elements.

| PML/PSEE | US.WS | US.TOOL | US.INTE | US.VIEW | VA.EVOL |
|----------|-------|---------|---------|---------|---------|
| 1. ADELE | 5 | 4 | 4 | 2 | 3 |
| 2. EPOS | 4 | 4 | 2 | 2 | 4 |
| 3. SPADE | 3 | 3 | 3 | 2 | 3 |
| 4. PrWe | 2 | 3 | 3 | 4 | 2 |
| 5. PWI | 2 | 4 | 4 | 2 | 4 |

Figure 6: PMLs/PSEEs according to US and VA process elements.

6.4 Assessment of 5 PMLs/PSEEs against 10 process categories

| PML/PSEE | Cler.Work | Tech.Work | Groupware | Business | Coop.Work | SW.Prod | Proj.Mgr |
|----------|-----------|-----------|-----------|----------|-----------|---------|----------|
| 1. ADELE | 2 | 4 | 1 | 2 | 4 | 4 | 1 |
| 2. EPOS | 1 | 3 | 1 | 3 | 3 | 3 | 2 |
| 3. SPADE | 1 | 2 | 1 | 3 | 2 | 3 | 2 |
| 4. PrWe | 3 | 1 | 2 | 4 | 3 | 1 | 4 |
| 5. PWI | 3 | 2 | 2 | 3 | 2 | 2 | 2 |

Figure 7: PMLs/PSEEs according to process categories for emphasis and domain support.

| PML/PSEE | Meta.Def | Meta.Enac | Meta.Evol |
|----------|----------|-----------|-----------|
| 1. ADELE | 3 | 4 | 3 |
| 2. EPOS | 3 | 4 | 4 |
| 3. SPADE | 2 | 4 | 3 |
| 4. PrWe | 2 | 5 | 1 |
| 5. PWI | 3 | 4 | 3 |

Figure 8: PMLs/PSEEs according to process categories for meta-process phases.

6.5 Some more detailed comments

ADELE: strong on product modeling, cooperation, workspace control, automated tasks, and dynamic process model.

EPOS: same, but stronger on model evolution.

SPADE: strong on activity and partly product modeling, model evolution, general tool architecture.

Process Weaver: strong on simple workflow, only product placeholders, some clerical support, pragmatical UI, general tool architecture.

PWI: strong on activity modeling, user modeling, model evolution, partly tool architecture.

All PMLs/PSEEs: weak on organization, groupware, user modeling, user views, early meta-process phases, and late meta-process phases wrt. evolving the real production process.

And evolution is paramount in software process.

6.6 Future Work on PML/PSEE Assessments

Few such assessments have been published so far.

Remains to:

- **Validate** the given process elements and categories.
More on software management, testing, and meta-process?
- **Detail** process categories into useful reference models, cf. the ISPW example.
Make **process model assistant, with reusable experience base?**
- **Assess** PMLs/PSEEs more in-depth wrt. to the above.
- **Compare** with work in business management, office automation, hardware production etc.

Continue the work in the ESPRIT-projects PROMOTER, PROTEUS, and PERFECT.

Thanks go to Grenoble process modeling group (Cap, LGI, CRISS).

7 Proposals for future PSEEs and PMLs

PML design ideas:

- **Main view of process model:** flow diagrams on www. Annotated with pre/post-conditions, scripts etc. for formal analysis and computerized enactment.
- **A central experience base,** with reusable process / product / quality models on www or in RDBMSes, cf. TAME [BR91] and EPOS [Min96].
- **Local project databases,** similarly.
- **A set of process tools** for model editing, presentation, query (4GL), analysis and synthesis, utilizing the www and with a **standard user interface.**
Architectural layers, such as:
 4. Project management, of high-level process.
 3. Process Engine for medium-level process.
 2. Transaction and Version Manager.
 1. DBMS, for data modeling and basic facilities.
- **Decentralization and distribution** of models and associated metrics, by the www.
- **Using Java as a bottom-level PML,** using Java interpreters as Process Engines to execute “active” applets (process scripts) on process model documents.
- **Local product workspaces (WS),** with development tools.
- **Process model evolution,** incremental and interpretative execution. Dynamic binding of human agents or tools.

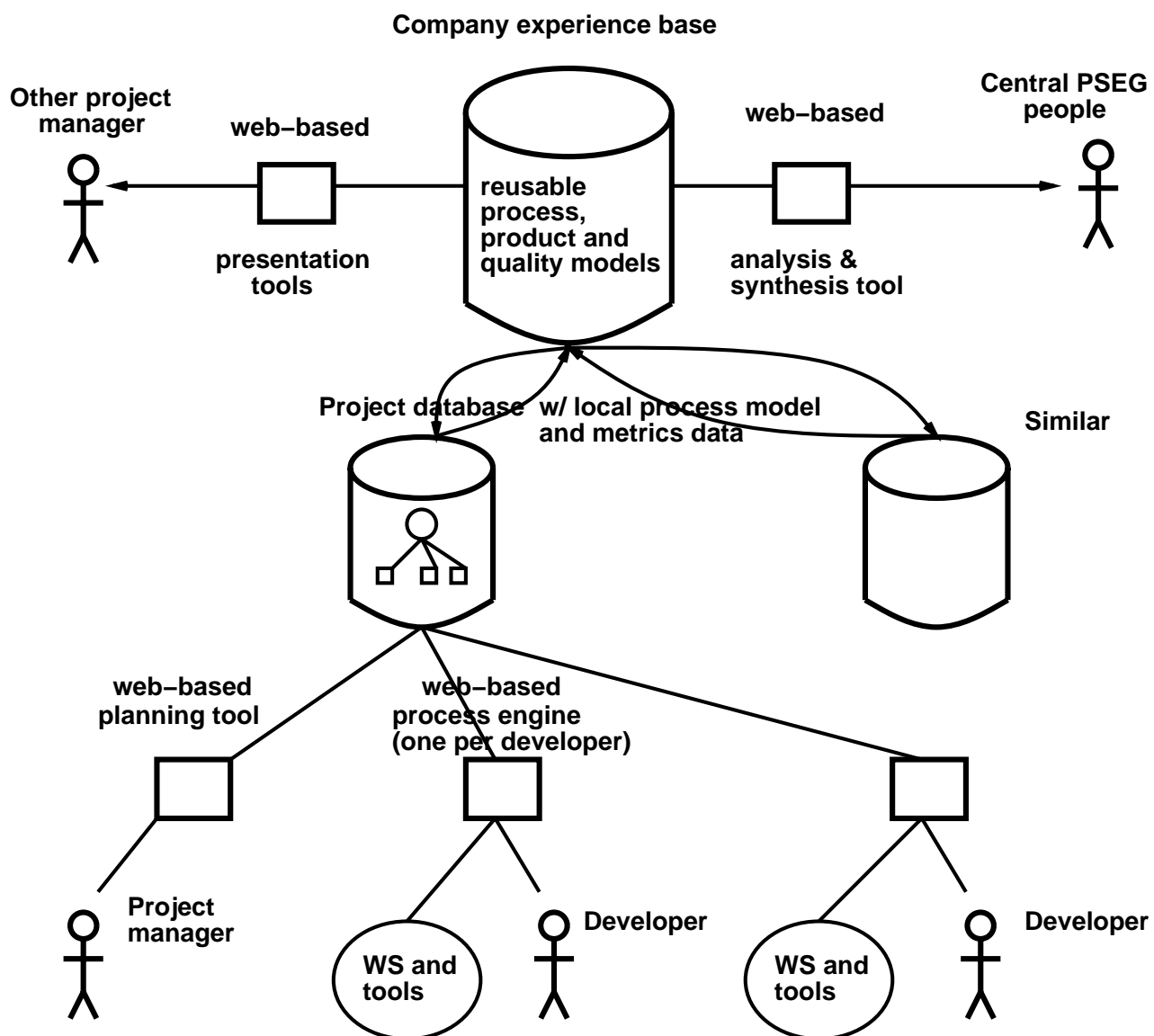


Figure 9: Proposed process support architecture on the www.

Pragmatic PSEE principles:

- Low start cost for commitment, training and tool support.
- Reuse existing process handbooks on www, make them “active”.
- Phase in use of metrics as part of the process model.
- Enable overall distribution and reuse of models.

8 Conclusion on PMLs

Much research on “correct” linguistic paradigm for core PML.
Rather federated set of PMLs – and partial (sub)models?

Reason: PSEE developers do not control the PML design space!
Look at STEP/CDIF, WPMC/PIF, UML ...

PML issues:

- **Complex** area, conceptually and implementationally.
Specific for SPI – dynamics, human-orientation, improvement?
- Agree on common **concepts / formalisms / taxonomy?**
- **Interoperability and Standardization:**
Standard support technologies (www, java),
i.e. modularisation and open systems.
- **Integration with Experience Database:**
Metrics, reusable process and quality models,
product libraries, dissemination.
- **Engineering of total process models:**
planning and analysis, consistency, reuse, configuration, ...
- **Flexible user-level evolution of the process model:**
Model accessible and changable by most process agents,
from adding new users or access-rights to structural changes.
- **Spectrum of support** (views, user in control):
reasoning, explanation, awareness, guidance,
non-intrusive automation and monitoring ...
- **Be practical, listen to users:**
distribution (www-based workspaces), experience database,

add-on to existing environment (e.g. project tools).

Acknowledgements

Go to the EPOS team, and the SPIQ and PROMOTER colleagues.

References

- [BG94] Victor R. Basili and Scott Green. Software Process Evolution at the SEL. *IEEE Software*, pages 58–66, July 1994.
- [BR91] Victor R. Basili and Hans D. Rombach. Support for Comprehensive Reuse. *Software Engineering Journal (special issue on Software process and its support)*, 6(5):303–316, September 1991.
- [CE94] Reidar Conradi and Jacky Estublier. The Major Software Process Categories: Taxonomy and Technology Assessment. In *Magne Haveraaen (ed.): Proc. Norsk Informatikk Konferanse – NIK’94, 14–16 Nov. 1995, Molde. Tapir Forlag, Trondheim.*, pages 53–66, November 1994.
- [Dem84] W. Edwards Deming. *Out of Crisis*. MIT Center for Advanced Engineering Study, Cambridge, MA, USA, 1984.
- [Der92] Jean-Claude Derniame, editor. *Proc. Second European Workshop on Software Process Technology (EWSPT’92), Trondheim, Norway. 253 p.* Springer Verlag LNCS 635, September 1992.
- [EDA97] Jacky Estublier, Samir Dami, and Mahfoud Amieur. High level Process Modeling for SCM systems. In *Reidar Conradi (Ed.): “Proc. 7th International Workshop on Software Configuration Management (SCM’7)”*, pages 81–97, Boston, USA, 18–19 May 1997. Springer Verlag LNCS 1235.
- [Fer93] Christer Fernström. Process WEAVER: Adding Process Support to UNIX. In *[Ost93]*, pages 12–26, 1993.
- [Håk94] Morten Håker. *Evaluering av Prosjektstyringsverktøy*. PhD thesis, NTH, Trondheim, December 1994. 79 p. (diploma thesis). EPOS TR 239.
- [Har87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3), June 1987.
- [Høy97] Geir Magne Høydalsvik. *Experiences in Software Process Modelling and Enactment*. PhD thesis, IDI, NTNU, March 1997. 237 p. (IDI-rapport 6/97, dr.ing. thesis 1997:32 at NTNU).
- [Kel89] Marc I. Kellner. Software Process Modeling Support for Management Planning and Control. In *[Per89]*, pages 8–28, 1989.
- [Mad92] Nazim H. Madhavji. Environment Evolution: The Prism Model of Changes. *IEEE Trans. on Software Engineering*, SE-18(5):380–392, May 1992.
- [Min96] Minh N. Nguyen and Reidar Conradi. Towards a Rigorous Approach for Managing Proces Evolution. In *Carlo Montangero (Ed.): “Proc. 4th European Workshop on Software Process Technology (EWSPT’96)”*, pages 18–35, Nancy, France, 9–11 Oct. 1996. Springer Verlag LNCS 1149.
- [Mye89] Brad A. Myers. User-Interface Tools: Introduction and Survey. *IEEE Software*, 6(1):15–23, January 1989.

- [NWC97] Minh Ngoc Nguyen, Alf Inge Wang, and Reidar Conradi. Total Software Process Model Evolution in EPOS. In *ACM/IEEE International Conference on SW Engineering (ICSE'97)*, May 1997. (Accepted for ICSE'97, May 1997, Boston, USA, 9 p.).
- [Ost93] Leon Osterweil, editor. *Proc. 2nd Int'l Conference on Software Process (ICSP'2), Berlin*. 170 p. IEEE-CS Press, March 1993.
- [Per89] Dewayne E. Perry, editor. *Experience with Software Process Models, Proc. 5th International Software Process Workshop, 1989*, Kennebunkport, Maine, USA, October 1989. IEEE Computer Society Press.
- [PWCC95] Marc C. Paulk, Charles V. Weber, Bill Curtis, and Mary B. Chrissis. *The Capability Maturity Model for Software: Guidelines for Improving the Software Process*. SEI Series in Software Engineering. 640 p. Addison-Wesley, 1995.
- [Rei90] Steven P. Reiss. Connecting tools using message passing in the field environment. *IEEE Software*, July 1990.
- [Tot97] Terje Totland. *Enterprise Modeling as a Means to Support Human Sense-making and Communication in Organizations*. PhD thesis, IDI, NTNU, March 1997. ca. 250 p. (forthcoming PhD thesis June 1997).
- [Vot93] Lawrence G. Votta. Comparing One Formal to One Informal Process Description. In *Wilhelm Schäfer (ed.): Proc. 8th International Software Process Workshop (ISPW'8), Dagstuhl, FRG, IEEE-CS Press*, pages 145–147, March 1993.
- [Win88] Terry Winograd. A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction*, 3(1):3–30, 1988.