OOPSLA '92
Vancouver, British Columbia, Canada
5 – 10 October 1992

Addendum to the Proceedings

Workshop Report—Team Approaches to OO Design

Report by:

Steven D. Fraser
Lynn Marshall
Bell-Northern Research Ltd.

Tony Bailetti
Carleton University

Large complex real-time systems are developed by design teams using object-oriented analysis and design methodologies. This workshop focused on the processes and the representations used to create and manipulate OO designs. The context of the participants (AT&T, BNR, CAI, Digitalk, Espirito Santo Data Informatica, IBM, NRC, OTI, University of Newcastle-upon-Tyne, and Xerox) included both pure OO systems and systems with a large legacy of non-OO designs and code. Each participant contributed a paper to detail their experience with OO design techniques and to provide some background on their current system size; design tools; and design process. Presentations by the participants were limited to a brief five minute statement of position. This permitted the rest of the day (over seven hours!) to be spent brainstorming and organizing ideas related to team approaches to OO design. Workshop activities included:

- *brainstorming*: participants generated three to five ideas related to their thoughts and experiences on team approaches to OO design
- *presentation*: all ideas were presented in sequence to the participants
- *categorisation and category naming*: the brainstormed ideas were grouped together into categories of "similarity"
- *category prioritization*: the workshop evaluated the categories and proceeded to rank them in the order required to achieve the biggest advance in a six month research window (Figure 1)
- *success measures*: metrics to evaluate success were brainstormed.

The output of the workshop consisted of ten categories of ideas related to team approaches to object-oriented design and a priority ranking (Figure 1). Each category is listed alphabetically (Category *x*: ...) with its constituent set of ideas (in italics). Comments, both serious and *off-the-wall*, annotate each idea. The category with the greatest assumed impact is at the top of the figure. Prioritization was determined by inferences drawn from group responses to the question: In the context of OO Team Design (in the short term): is *{category 1}* more likely to have impact than *{category 2}*?

A brief description of the process used to facilitate the workshop, based on the Interpretive Structural Modeling methodology developed by Warfield [Warfield76], follows in the Appendix.

Overall this summary represents *work-in-progress* rather than a final report. Workshop participants agreed to remain in touch by E-mail to follow-up on the initial interest generated in Vancouver.

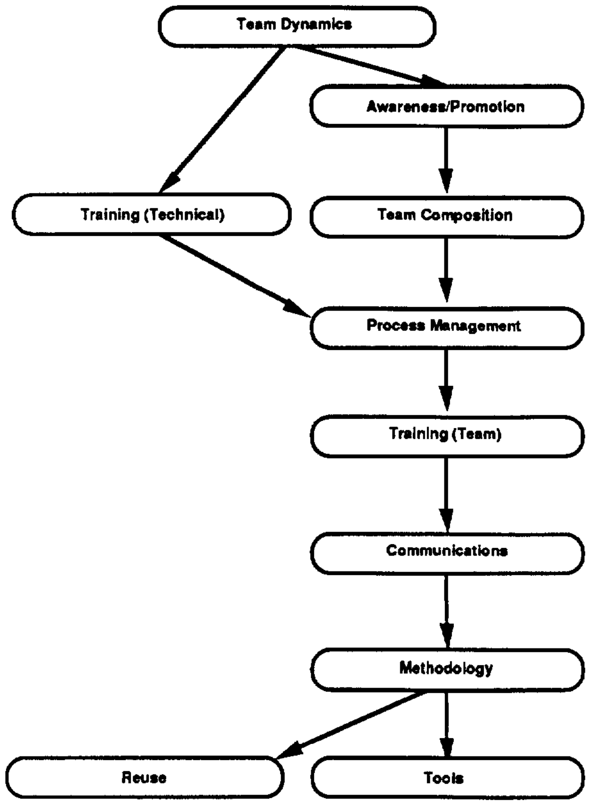

Figure 1. First-cut prioritization of concept categories.

Concept Categories

Category 1: Awareness/Promotion:

Clearly identify the "real" objective (it's NOT just "to be object-oriented").

Can you use existing evaluation criteria to judge?

Use business criteria (business driven and not technology driven).

Is solution a real objective?

This is very true, we get hung up on the object-oriented words.

Get rid of the sliver bullet mentality

Yes very much so, but we will need many bullets of some type.

Buy-in of technical members of the team for trying OO techniques

Learn from their experience?

Expand management awareness and participation

How do we get them to attend?

Awareness of what?

They need to get back in touch with what goes on "in the trenches."

Rearrange the process—make management team players.

Technical brief idea: One page with picture, key concept written for high level managers.

Produce a relevant working example

Example of What?

Do agree we need examples.

Show significant improvement (versus standard process) on OO projects.

How?

Find a metric, standard process, may win sometimes.

Demonstrate a real project versus standard paper.

Takes time to get results.

Motherhood.

Get a selection of successes/failures using OO techniques outside your own organization.

Provide specialized "Awareness" training in current OO design methodologies for senior executives (decision makers).

Maybe they could spend some time just observing how OO development works on a day-to-day basis.

Need specialized training to "rock" ideas. Bored with day-to-day activities.

This is definitely needed and can be effective.

Change the organization—make senior executives team players.

Senior executives aren't interested in details of technology.

Bridge the existing process with an OO Flavor process.

Only if there is a demonstrable benefit.

If people are prepared to bridge—show them the benefits.

Can't be perceived as dollars = sideways step by management.

Category 2: Communications:

Use "standard" design notation to facilitate understanding and enable design reuse.

What is standard?

What do you do with out-dated standards?

A standard isn't a standard until it gets used at least twice!

It is too early to standardize.

NO, need concept understanding over standardization of notation!

Need common representation of "concept" but this is easier than a common methodology.

Select a language approach that can do as a communication means.

Is English included, or do you mean a formal language?

I don't understand is this a LAN, WAN or some other technology?

What about visual languages?

How do we represent team dynamics, team emotions?

Language is insufficient!

Need a syntax, plus a text!

Category 3: Methodology:

Simplified methods which support planning phases!

Select principles for good OO design.

Exactly what do you mean?

Knowledge of design methodologies.

Multiple methodologies?

Same methodologies?

Evaluation criteria?

How detailed is the knowledge required?

I'm not sure a team really needs to know that much about others, but should know/understand fully about their chosen method

Team knowledge, or experts on the team?

Pick an OO development methodology and train people.

Right, let the team customize it themselves, creates better buy-in!

Why? We need to understand our problems before picking a methodology or tool!

No one existing methodology satisfies every requirement!

Wing it!

Adopt a methodology, either an existing one or just formalize one for use locally.

Good idea!

Who says you have to follow a formal one as a religious experience!

Achieves teamwork!

Why?

Something is better than nothing?

Broader experience in various design methodologies.

Might be confusing.

Okay if selected subset of available methods used?

What will this information be used for?

Need accurate development focus!

Is the language that important?

Category 4: Process Management:

Start collecting and analyzing metrics instead of talking about them

Random metrics get forgotten when the heat is on.

What are metrics anyway?

How do they fit into the process?

Pick metrics that are meaningful to the organization.

Establish incentive programs for design and reuse ("Credit" of contributions to team and "Debit" of acquisition from the team)

What kind ... cash?

Is competition within a team desirable?

An unjust reward system can cause poor morale?

Don't know how to measure reuse yet!

Reuse incentive program.

Should be orthogonal to "project management."

Good: reuse is also a means to "Software Engineering Training."

"Reuse" makes sense in the business world.

Category 5: Reuse:

Adapt for the future from experience gained.

How specifically do we adapt?

Example: Creating a database of experiences, somehow categorized for easy access.

Great! How should we approach this?

What is your measure of experience? How to start?

Creating, storing, retrieving, adapting, modifying, applying team; generated artifacts; Use of reasoning techniques.

How do we share the "good stuff" without sharing unwanted artifacts?

Sometimes you don't have time to sift through all that bulk.

Category 6: Team Composition:

Overall system architecture group, one object model, champion design reuse, OO expertise.

Not sure what it means to "Champion Design Reuse."

Need to spread expertise from kernel group.

As long as this group isn't a bunch of bureaucrats (it must be technical).

May lead to "religious wars" and end up with multiple competing groups of this kind.

Seed development teams with system architects, facilitators and OO Zealots.

This is key, especially on early projects.

Don't forget to seed middle management too.

Overall staffing and organizing is key.
Why not "homogenize" the staff—no different skills.
Problem: development team may "keep" the architects, better to create an informal team.

Join a cross-functional team, participants from many areas of product development.
Question of integration-how do you get them speak the same "language."
Well, here we are!
This sounds long-term.
Not a three to six month project!

Establish cross-functional teams in all development actions.
Does this mean right through from functional specification to product testing/verification?
What is the goal of this?
What are cross-functional teams?

Category 7: Team Dynamics

Treat geographic/political separation as a central problem, not a side issue (e-mail; shared books; video links; video conferences; anthropologist ...).
Is it a control problem or something that requires processes we are not use to.
Very important for reusability.
Amen; Neat!

Regimen for technical integration.
What is technical integration?

Team recognition program for incentives to participation.
What kind of software reuse?
Oh Great!—reward people for not being able to shut up.
Well, we need something to get people out of their burrows.

Stress small/informal meetings over large formal meetings.
Good on efficiency as long as shared vision.
Both are effective if the meeting has focus.
Good idea to exchange real experience informally.
Yes, cross-functional teams have only representatives, not entire departments.
SWAT Teams.
How do you pick the right reps?

Build team skills: select team members; train team members; and reward teams.
Essential; Train in team building.
Sometimes you don't have control over team selection, you need to figure out ways to deal with what you have.
Bonus programs for designs, reuse etc.
How is quality determined?
An unjust reward system can cause more problems than it solves

Team (people-human) Interaction approaches (project orientation, management, "social" structures).
Tell me more!
How do you get everyone to cooperate?
This is the most interesting aspect to me—how?
Yes!—we need to involve behavioural studies

Category 8: Tools

Need tools to refine hierarchies based on usage (feedback from team).
Is there a process for this or a methodology?
Why refine if they are "correct?".

Better tools for combining the efforts of individuals.
Make them or find existing ones?
What kinds of "efforts?".
Design, prototype, implementation, etc.?
Composite or hierarchical?
Select tools to adequately support method and processes?
That's the problem, isn't it?

Teach users to distinguish between classes and chunks of reusable code, which may be smaller or larger than a class.
Good idea, methods that go beyond classes in terms of encapsulation and reuse units.
Why not formalize this, e.g. fragment packages?

Technology to browse repositories of reusable code.
Specially "distributed" repositories.
No!—need generic architecture (has anyone actually had success with repository browsers???).
This is good (and doable today)!

Apply team facilitation tools to OO design, e.g. JAD, CSCW, etc.
Before a general purpose tool can exist, you need the experience of having done it manually.
This may not be a short-term idea.

Tools to span design and implementation.
Auto-generation of prototype.
No, tools to span (not auto-generate).
No, auto generation is actually slower, however, reverse-generate (code $\Rightarrow$ design) is useful.
Need to at least keep the two in synch.

Use available tools effectively.
What do you mean by effectively?
Great when possible.

What do you do when the tools don't go far enough ... suffer?
Adapt new tools?

Get a tool that supports teamwork.
Does this mean facilitate teamwork?
A team management tool?
Yes, also the metaphors sometimes don't really tell us how a team really uses the tool.
Yes, at the point we understand what "support" means.
A single tool is probably not good enough.
Maybe there is too much emphasis on tools (they are an aid, but not the answer).
Is this teamwork in design or implementation?

Find/develop tool(s) for rapid evaluation of designs.
Necessary for gaining confidence that the idea works.
Prototyping?
Is there a way we can do this without giving the impression that there is a product to sell?
Whenever we do this in our company, our marketing group goes out and sells prototypes! (Right On)!
Evaluation of designs, how by quality?
Are prototypes "throw-away" by definition?
Or can they evolve to products?
What are the criteria (acceptable for evaluation) for evaluating designs?
We need to develop rule-of-thumb measurements.
Agree, need tools and conventions.
The tools are easy, getting the rules is the hard part!

Tools for evolution—taking the changes for one "environment" forward to related, but different environments.
How do we abstract out the parts that can be brought forward?
These things are often orthogonal to project domains.
Great, now how do we do it?
Answer is through a toolkit (some companies are doing this now)!

Category 9: Training (Team)

Train and evaluate teams and individuals in teams.
As team members?
With respect to OO experience?
For what purpose?
Team building skills?
Who does this?

Training for effective team behaviors for all members of the team.
This assumes someone somewhere knows what is "effective behavior."
Each member has its own limitations.

Category 10: Training (Technical)

Training of cross-functional team in simplified OO methodology
Which one?
Big job!
Sounds good, elaborate.

Educate/train in ideas of object-orientation.
Train whom?
Design?
Implementation?
Teamwork?
OO is a different way of thinking (paradigm shift).
Key concepts are more important than any method.
Use reuse "toolkit" to train (very effective from experience).

Bring in noted expert to keep momentum going.
The expert must be immersed for a long time for this to work.
May fit your prelim to his/her solution.
Needs team acceptance.
Be clear on WHY the outside expert is being brought in—if it is to keep the momentum going—their solutions won't be coming out.
This is not always possible.

Proposed Metrics: Measures for Success

The associated category name from the previous section is given in italics.

Awareness:
Measured successful if not one manager discusses the silver bullets (ever).
% of new project using technique or % of $ going to OO teams.
Number of "adoptions."
Increased management support for new design process.
Shift in emphasis/vocabulary.
# of people that like doing OO.

Change
- Maximal change represented by minimal information.
- Ease or speed of change measured reduction in development time.

Communication
- Basic for teamwork measurable success if ... single view evokes consistent interpretation.

Methodology
- Project a success.
- Completeness of the final project as it meets original requirements.
- If it facilitates understanding of the design without an interpreter (from management to programmers)
- % of new work done in adopted methodology or in OO technology in general

Reuse
- % of a system's code taken verbatim from the reuse repository

Team Composition
- Productivity

Team Dynamics
- Amount of re-work required
- People satisfaction
- Degree of team participation in meetings
- Conflict resolution via collaboration
- Team identity (visible)
- Personality conflicts
- Frustration at meetings
- Amount of common ground established
- Easy transition of members across teams
- Time spent in meetings

Technical Training
- # of programmers completing curriculum

Tools
- Ease of maintenance
- Improvement in productivity (hard to quantify)
- Number of users as a % of total

Appendix: Workshop Facilitation Process

McGoff et al. [McGoff90] has documented IBM's experiences with Group Decision Support Systems (GDSS) as a tool to facilitate general meeting activities. It was observed that the parallel, anonymous and collaborative group communication facilitated by GDSSs effectively and efficiently addresses complex problems. The application of group decision support tools to software development has been previously reported by IBM and Boeing [McGoff90, Boeing91, Post92] as being very successful. For example, Boeing has reported savings of 91% in days of flowtime (development interval) and a reduction of 71% in labor hours. Groupware is defined [Ellis91] as:

> *computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment*

The workshop was facilitated by the Interpretive Structural Modeling (ISM) technique developed by Warfield [Warfield76]. In dealing with complex tasks, ISM uses mathematical models to provide the type of high-level support that is currently lacking in GDSS tools. An important aspect of task complexity concerns the amount and degree of interaction between different factors. The key limiting factor when dealing with complexity is the limitation of human short-term memory. Generally, the span of immediate recall is just seven to nine chunks of information [Miller56]. Thus, when dealing with a problem that has many interaction factors, the ability to keep everything in play and develop the most effective solution is quickly exceeded.

Structural modeling breaks complexity down into manageable pieces of information and helps the human mind deal with them. This is accomplished by focusing the team on the relationship between only two concepts at a time. Not only is this pair of items comprehensible to the mind, it also has the powerful effect of revealing the paradigms that each individual holds about a particular situation. Kelly [Kelly55] discovered that a subject, limited to the consideration of only two or three notions at a time, would discover what Kelly called *personal constructs* about the world. Current terminology labels these personal constructs as paradigms or *mental models*. The technique has the unique power to reveal and coalesce assumptions which team members hold implicitly about the field of endeavor they are exploring. Senge [Senge90] comments that mental models are often not recognized for their importance:

The problems with mental models lie not in whether they are right or wrong by definition, all models are simplifications. The problems with mental models arise when the models are tacit when they exist below the level of awareness.

Personality and private agendas can interfere with the process of sharing. Therefore, it is important to decouple personality from the process [Warfield73] since people and organizations have goals and objectives that they are unwilling to express. Or if they were expressed, there would be a substantial amount of bitterness engendered, leading to confrontations, which would in turn lead to unreconcilable conflicts. Because individuals and organizations have "invisible intent," any visible portrayal of intent is said to be at best incomplete and probably hypocritical.

ISM (Interpretive Structural Methodology) has five basic phases: preparation, brainstorming, voting, model construction, and model interpretation. The first two phases capture the collective knowledge of a group and the last three organize that knowledge in a meaningful way. In a typical ISM intervention, the preparation phase involves the formulation of a trigger question and a relational term. The trigger question focuses the generation of issues and the relational term enables the comparison of interactions. For managing the brainstorming phase, the Nominal Group Technique (NGT) is used to generate as many ideas as possible. Once a set of issues has been generated in the preparation and brainstorming phases, interactions between issues are determined in the voting phase. Computer support, based on mathematical algorithms, is used to record the relationships interpreted between elements, extract a structure from the complex set and graphically model the specific relations and overall structure. Computer support reduces the cognitive effort of the participants. Table 1 (based on the work of Cole and Nast-Cole [Cole92]) outlines the phases of group progression from conception to completion.

The ISM methodology was implemented in BNR Prolog at Bell Northern Research's Software Engineering Centre [Pollard93]. This BNR GDSS is a flexible, fluid methodology that supports group members in selecting the most appropriate strategy to deal with complexity. After a finite set of issues has been generated in the preparation and brainstorming phases of the meeting, the BNR GDSS poses a series of queries like: *Does Concept A* (and here a relational term is used) *aggravate* (for example) *Issue B*. Other examples of intrinsically transitive contextual relations similar to *aggravate* [Warfield74] are: *is included in, is less than, supports, implies,* and *causes.*

Group Activity	Brief Description
Forming	Determine group's membership, focus and orientation
Storming	Group grapples with issues of power, control, and authority the primary focus is leadership
Norming	Once leadership has been determined, the group develops team coordination and integration (leadership and facilitation shifts from an individual to the group)
Performing	The team becomes productive, energetic and effective
Adjourning	Groups adjourn when their work is completed, they are reorganized, or their mission changes a postmortem is useful to evaluate the successes/failures of the team

Table 1: Group Lifecycle from conception to completion

All team members have the opportunity to vote yes or no. The answer to each query is used to determine the next query. A yes vote will usually result in a connection between the two issues. As the query process continues, a structural model is automatically constructed by the software. Structural modeling makes no assumptions about the structure of a problem except to assume that one exists. It also makes no assumptions about the importance of ideas. Structural modeling lets the problem reveal itself to the group and indicates the subsequent use of a more focused problem solving method. ISM facilitates the thorough testing of new ideas and the structured evaluation of seemingly plausible, yet perhaps off-the-mark assumptions that would otherwise lead the team astray.

As with any other group structuring technique, ISM does have its limitations. One practical difficulty that has emerged from the preliminary analysis of post-meeting assessments is that an inverse relationship exists between the precision of group thinking and their sense of satisfaction with the process. Group members view modeling as hard work. This is contrasted with the "fun" of brainstorming. As sessions become more intense, that is, more modeling oriented, the post-session evaluations noted a decrease in participant satisfaction.

Bibliography

[Allen87] Gill Allen, "Team techniques in system development" in *Datamation*, Vol. 33,No. 22, 1987, pp. 88-92.

[Boeing91] Boeing Computer Services-Professional Services Organization *TeamFocus* Group, *TeamFocusTechnology Evaluation Report*, The Boeing Company, October 1991.

[Cole92] P. Cole and J. Nast-Cole, "Group behavior and evolution: A primer on group dynamics for groupware developers" in *Groupware: Software for Computer-Supported Cooperative Work*, IEEE Computer Society Press, Washington, 1992, pp. 44-57.

[Ellis91] C.A. Ellis, S.J. Gibbs, and G.L. Rein. "Groupware: Some issues and experiences" in *Communications of the ACM*, Vol. 34, No. 1, January 1991, pp. 38-58.

[Kelly55] G.A. Kelly, *The Psychology of Personal Constructs*, W.W. Norton, New York, 1955 (2 vols.).

[McGoff90] Chris McGoff, Ann Hunt, Doug Vogel, and Jay Nunamaker, "IBM's experiences with group systems" in *Interfaces*, Vol. 20, No. 6, November-December 1990, pp. 39-52.

[Miller56] G.A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information" in *The Psychological Review*, Vol. 63, No. 2, 1956, pp. 81-97.

[Pollard93] Carol Pollard and Clifford Saunders, "The case for interpretive structural modeling as a technique for enhancing electronic meeting system support" to appear in *The Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*, IEEE, January 1993.

[Post92] Brad Quinn Post, "Building the business case for group support technology" in the *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, IEEE Computer Society, Vol. IV, 1992, pp. 34-45.

[Senge90] Peter M. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday, New York, 1990.

[Warfield73] John N. Warfield, "Intent Structures" in *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-3, No. 2, March 1973, pp. 133-140.

[Warfield74] John N. Warfield, "Developing Subsystem Matrices in Structural Modeling," in *IEEE Trans. on Systems Man and Cybernetics*, Vol. SMC-4, No 1, January 1974, pp. 74-80.

[Warfield76] J.N. Warfield, Societal Systems: *Planning, Policy, and Complexity, Interactive Management*, New York, 1976.

Contact information:

Steven D. Fraser
Lynn Marshall
Bell-Northern Research Ltd.
Box 3511, Station C
Ottawa, Ontario, Canada.K1Y 4H7
Voice: (613) 763-3073
Fax: (613) 763-4222
Email: *sdfraser@bnr.ca*

Tony Bailetti
Carleton University
Ottawa, Ontario, Canada.