FROM: INTERNET:owner-coad-letter@oi.com, INTERNET:owner-coad-letter@oi.com
TO: (unknown), INTERNET:COAD-LETTER@OI.COM
DATE: 9/19/95 3:59 PM

Re: 19a--Managing--"Embracing New Technology"

Sender: owner-coad-letter@oi.com
Received: from oak.zilker.net by dub-img-4.compuserve.com (8.6.10/5.950515)
id PAA28190; Tue, 19 Sep 1995 15:55:39 -0400
From: <owner-coad-letter@oi.com>
Received: from oi.com by oak.zilker.net (8.6.10/zilker.1.96)
id OAA20296; Tue, 19 Sep 1995 14:28:28 -0500
Received: by oi.com (4.1/single.1.1)
id AA10389; Tue, 19 Sep 95 13:30:24 CDT
Date: Tue, 19 Sep 95 13:30:24 CDT
Message-Id: <9509191830.AA10389@oi.com>
Subject: 19a--Managing--"Embracing New Technology"
To: coad-letter@oi.com
Sender: owner-coad-letter@oi.com
Reply-To: owner-coad-letter@oi.com

The Coad Letter
Issue 19a
Category: Managing
Title: "Embracing New Technology"
Date: Tuesday, September 19, 1995

Dear Friend,

Thank you so much for the suggestions so many, many of you sent in.

While reading some of your suggestions, I realized that "secrets of multiproject OT managers" was too narrow a topic. Seemed like the challenges we face are more about...

Embracing New Technology

along with some object-specific challenges along the way.

Yesterday, the traveling team visited ImageBuilder Software, in the Portland, Oregon area. I am writing this issue from Portland, just before leaving for our second stop on our week-long US tour.

Thanks for joining in on the tour via e-mail. I hope you'll enjoy this and other snapshots along the way. :-)

Sincerely,

Peter Coad

---

EMBRACING NEW TECHNOLOGY
Secrets of Multiproject New Technology Managers

Peter Coad

Work-In-Progress / Tuesday, September 19, 1995

---

CONTENTS



## PREFACE

This is a working draft of the material. The outline and its contents will most certainly change over time. Indeed, one or two more drafts within the next week or so is very likely.

Please write up and send in your comments, suggestions, and anecdotes via e-mail to coad@oi.com

I hope you find this evolving work helpful.

Special thanks go to Dwayne Towell, Terry Hamm, David North, Jonathan Kern, my friends and colleagues at NTT Data, and the many readers of The Coad Letter (especially <short list>). This special report would not be possible without your kind and thoughtful participation. Thank you very, very much.

## INTRODUCTION

There is nothing new under the sun.

Over the past four decades, every new software development advance has proclaimed itself as something "new, different, and radical" and requiring new management techniques. <insert quotable quotes. candidates: assembly language, FORTRAN, C, structured design, objects, distributed objects>

After seeing this happen time and time again, patterns emerge. A pattern is a template; it's a template derived from experiencing something again and again -- and then abstracting from those experiences.

This purpose of this special report is to set forth such a template... for managers, team leaders, and team members who will inevitably live through many new technology advances during their careers.

Specific comments come from managers of multiple object technology projects, working at these companies:

(ibs) = ImageBuilder Software
Dwayne Towell and Terry Hamm
Multimedia apps

(aig) = Artificial Intelligence Group
David North
Retail and Wholesale apps

(qti) = Quantum Technologies International
Jon Kern
Manufacturing apps

PROJECT MANAGEMENT OBJECTIVES
- Critical success factors
  - achievable objectives
  - juggle cost, schedule, resources
  - manage expectations
  - remove obstacles
  - deliver a product of acceptable quality
- For new technology
  - managing acceptance and use across...
    - a pilot project
    - a real project
    - across multiple projects

JUSTIFICATION AND READINESS
- Reasons for adoption
- Advantages, compared to other approaches
  - (ibs)
    - manage complexity
    - need to think about fewer parts at a time
- Drawbacks
  - (ibs)
    - small project, so not that big of a risk
    - try it on a small project; no problem falling back to C
    - biggest risk: tools were pretty simple in 1989
- Expected results
  - (ibs)
    - reuse, reliability
    - success probably depends more upon the developer
    - it's technology that might help
    - working with new technology is another way to keep developers
- Assessing readiness / preparing for
- Selling senior management
  - find someone willing to listen (an early adopter)
    - (ibs)
      - our company embraces change; it's okay to try new technology
      - find a manager who is a new technology advocate
      - then build a small pocket of consensus.
      - projects are short (one year or less); risk is low

(we could always fall back to older technology and still get the job done)
- making a defensible business case
  - measurable business benefits for a new technology?
    - not likely
  - increase sr. management visibility into what's really going on, in a project
  - increase sr. management frequency at seeing working results, so they can know what progress is really being made (or not being made)
- change is change is change
  - change ripples through an organization
  - few changes are merely a "quick technical add-on"
- pilot project(s), then serious ones
  - reduce risk, allow time to learn, accept some false starts
- justification of investment
  - education, tools, consulting

- Criteria for selecting...
  - any project
    - (ibs)
      - interesting work
      - good match to our capabilities
      - challenging, yet within reach
  - your first new technology pilot project
    - (ibs)
      - who will work on it
        - experienced, interested, and available at the right time
      - totally new (no old code to deal with)
      - 3 people, 3-4 months
      - afterwards, assign those 3 people to 3 different teams, to evangelize, to communicate lessons learned

PROJECT PLAN

- Outline
  - (ibs)
    - informal...
      - description
      - risk, contingencies
      - living document!
      - somewhat client-dependent (client may own the plan)
- Scope
- Balancing schedule, budget, and resources (people, tools)
  - (ibs)
    - sr. project managers are responsible
    - 1-2 day spec; then lead works out a schedule
    - breakdown tasks
    - project team and framework team work together on estimates
  - estimating heuristics
    - how long?
    - how much?
    - how many people?
    - personal "fudge factors?"
    - (ibs)
      - project teams --

- we track what we've been off in the past
  - and add a fudge factor each time (say, 20%)
- framework team --
  - we double our estimate, to add time for communicating with project teams, an essential ingredient for our success.

- milestones
  - by activity? by measurable results?
  - at what intervals?
  - (ibs)
    - external (client-visible) milestones
      - alpha, beta, final (sometimes several of each)
      - frequency: every 1-2 months; client gets working software
    - internal milestones
      - what
        - spec
        - task list and schedule
        - get something up (a trivial screen)
        - then by features or groupings of features
          - product features
          - framework features
        - plus internal milestones that are placed 1-2 weeks before external milestones
      - frequency
        - every week or two; code goes to QA
    - acceptance
      - internal -- word of developer, accepted by team leader
      - alpha -- all features
      - beta -- all level 1 bugs corrected
  - when to commit to a schedule?
  - what do you do during planning?
  - how do you adjust during execution?
    - (ibs)
      - if meeting the schedule is the highest priority,
        - then we add resources to meet it

Process

- waterfall, incremental
  - (ibs)
    - incremental
      - multimedia scene by scene
        - in effect, by feature or feature category
      - sometimes we break a scene into two increments:
        - rough out, then fill out
      - size of increments: whatever fits into about 1-2 weeks of development time
  - defining milestones; determining milestone completion

Multiple releases

- how to plan, measure, control?
  - (technical issues, administrative issues)

Progress measurements (and actions taken)

Keeping on course

- feature creep? feature drift?
- (ibs)

- change controls are in the contract
  - some changes are free
  - some changes cost a bit
- feature too hard? then we endeavor to negotiate a substitute.

Measurement and control
- what do you measure?
  - (ibs)
    - features completed
    - bug rate; levels 1-4 (1 is worst kind); meeting certain milestones means no known bugs of a certain severity level.
- what is helpful; what is not

TEAM MEMBERS

Skills
- joy of learning, love to learn
  - (ibs)
    - self-motivated ones
    - track record for originating new ideas
    - for new technology, a lack of experience with earlier technology may be a plus (less change resistance)
- technology-specific experience
  - project experience, workshop experience, books, articles
- looking across multiple dimensions
  - conation (the will to do)
    - quick start, fact finder, implementer, follow-though
  - intelligences
    - linguistic, mathematical-logical, spatial, bodily-kinesthetic, intrapersonal, interpersonal
  - learning styles
    - visual, auditory, kinesthetic

Selecting team members
- selecting from a talent pool
- reserving the right of refusal
- partial vs. full-time dedication
  - (ibs)
    - always prefer full-time assignments
    - might have a part-time maintenance responsibility, too
    - partial dedication is much less efficient
    - as it is written, "no man can serve two masters"
- why, when, and how to say farewell to a team member

Keeping key players
- most important benefits, qualities
- competition other projects, other companies
- holding till the next project
- (ibs)
  - short projects (3-6 months) means that inadvertent people mismatches are short-lived
  - project teams don't stick together; for the most part, people are reassigned at the end of a project.

Moving to multidisciplinary teams
- putting together team members that competed in a past approach
- (ibs)

- it's important to give someone clear authority to make decisions; would prefer to not put old rivals on the same team, if at all possible.
- Recruiting experienced developers

TEAMS

- A team and its members; a project and its teams
  - size
    - (ibs) 3-5 people
  - skill mix
    - (ibs)
      - lead, 2-3 programmers, an artist, a media coordinator, a QA person
      - lead reports to a senior project manager (someone watching over about 4 projects at a time)
      - mix of experienced developers and recent grads
  - assignments for each team member (dividing up the work to be done)
    - (ibs)
      - by aptitude, experience, interests
      - by features or feature categories
      - by similar features (reuse code, or at least reuse their thinking and learning about building such a feature)
  - who sees the big picture for an app?
    - (ibs)
      - small apps, with hope that everyone does develop the big picture
  - who owns what?
  - who decides which way to go?
    - "my way is better" problem
      - (ibs)
        - educate; impose the change
        - resolve technical squabbles at a technical (not executive level)
    - "no clear leader" trap
      - (ibs) fully agree -- that's a trap
        - committee decisions foster change resistance (and delays)
  - with multiple teams, who insures that the pieces fit together?
    - (ibs)
      - the (technical) manager who is responsible for the teams
      - the teams themselves; each is developing software and must get the software working together every week or two.
- Decisions
  - sole tech leader?
  - senior tech team?
    - size?
    - skill mix?
    - how often?
- Transition to team leadership
  - (ibs)
    - take a close look at one's skill set
    - don't drag someone along; career counseling
    - provide a technical advancement path
- Team location
  - same room? adjoining offices? same floor? same building?

- same state? same country? same planet?
- (ibs)
- same room (low modular walls within)
- feel very strongly about positioning teams together
- not by discipline (programmers, artists, ...)
- why: effective communication
- we do a lot of "communication by running around"
- projects change, and yes, people change offices, too.

Effective communications
- within a team
- between teams
- with customers, business experts
- working together
- how soon? how much? how often?
- ways of working together
- assessment of needs/tasks/problems?
- models? role playing of major interactions?
- with sr. management
- managing expectations (initially; along the way)

PEOPLEWARE

Growing a vital corporate culture
- (ibs)
- fun place to work
- accept a 40-hour work week for what it is: 40 hours
- exceptions 2-3 times per year, for 2-3 weeks at a time; no more

Little things that show you care
- (ibs)
- goodies
- project T-shirts
- small tokens for accomplishments
- box of programmer food
- "reward kit" when a product ships
- lunch together
- conference
- good experience with platform-specific programming conferences

Keeping your top talent
- (ibs) this is something we are very good at.
- interesting work
- new challenges, new opportunities
- some autonomy, some creative freedom
- assignments with good interpersonal fit
- variety (not stuck doing the same old thing)
- compensation
- salary and benefit surveys, to compete in local market
- incentive stock options
- location
- being in a community that has a culture supportive of longer-term employment at a company

Training and consulting (mentoring)
- outside? inside? mix?
- when helpful? when not? when time to change?

- for the first project
- for subsequent projects
- Encouraging (overcoming resistance to) change
  - those who find difficulty
  - those who find it hard to let go of a past approach
  - those who outright refuse to change
  - (ibs)
    - keep going, hope they'll come along
    - just a few holdouts; they'll change of move on
    - education helps in this area, but not always
- Integrating on-going professional development
- Quality standards (ISO 9000-3)

REUSE
- What?
  - reuse what?
    - code
    - design
    - programming experience
    - building blocks for app developers
    - an overall approach, for building a family of apps
    - domain understanding
  - use again -- by more than one person, team, app, problem domain
- Who?
  - who builds it? who reuses it? who maintains it?
    - (ibs)
      - some false starts
        - we tried building for reuse on a project.
          - that contradicts the primary success measures for a project.
        - we tried building for reuse by committee.
          - someone had to be responsible.
        - we tried building for reuse in an ad hoc way.
          - vision was needed.
        - we tried building for reuse on a part-time basis.
          - yet full-time development assignments always work out better.
      - success with
        - small, dedicated team
          - fewer opinions, fewer arguments
          - very experienced team members
        - features
          - app teams and project proposal teams suggest new framework features
          - framework team, working with senior project managers, prioritizes those features
            - criteria: likelihood of reuse, level of difficulty for app developers, needs of future apps
        - ownership
          - framework team responsible for all framework code development and maintenance
          - note that bug fix urgency is often much higher for a framework developer

People and reuse
  getting people to build things for reuse
    rewards? dictatorial rule?
  getting people to reuse
  getting people to design and code for maintainability
Getting started
  (ibs)
    at first, it was easier to scrap previous results and start
      over each time (great pain to reuse existing code; we did
      it at times; it still was a great pain to do, though)
Discovering something with potential for reuse
  before building something for reuse, build it as part of an app,
    without worrying about building for reuse
    (ibs)
      we learned some things this way that we would have never
        learned, if someone had expected us to build it right
        the very first time.
    (ibs)
      don't expect a project to develop reusable software.
      why: a project has different goals, different urgencies
  look for what could be reused
    by apps within the problem domain you are working in
Organizing for reuse
  develop a skeleton, a framework, for a family of apps in the same
    basic business area
  a framework establishes an overall structure, a perspective that
    other developers can build upon
Building for reuse
  build for reuse across a family of apps
    (ibs)
      fit it into a growing framework
      always use a separate reuse team, when building for reuse
   use the "reusable" code the first time -- and refine it
    (ibs)
      begin to see what assumptions you've made that might
        impede subsequent reuse
      the framework, a hierarchy of application-specific classes,
        made reuse practical for us.
      so we've been developing C++ apps for 6 years; over the past
        year, we've gained business advantage from reuse -- and
        will continue to do so as long as we build software in
        the same basic problem domain.
Getting others to reuse
  reuse implies benefits, at some cost
    (ibs)
      framework users get a lot, yet must work within the
        organizational structure imposed by a framework
      make it easier for project teams to justify using the
        framework (easier for them to build their apps)
      weed out developers with a "not invented here" attitude
  freedom on amount of reuse
    (ibs)
      project teams free to reuse that which will help them reach
        their goals

- project teams free to use whatever version of the framework they want (freeze; most current; or every couple of weeks)
- keep in the design loop, encouraging reuse along the way
  - (ibs)
    - framework developers are available for design brainstorming, discussions, inspections, and reviews -- design issues, including (but not limited to) the effective use of the framework
    - in this way, framework developers gain insights into on-going successes and failures that developers are having when using the framework
    - framework developers spend one-half of their time here
- documentation
  - (ibs)
    - "how to think about it" document
    - all other documentation for framework users is in the header files
- training
  - lots of practical, how-to examples
    - (ibs)
      - demo program
      - startup program, ready to start building an app
      - how to do movies, sound channels, etc.

- Measuring reuse
- Paying for reusable software
  - overhead, at first
  - projects that use the framework, after that
  - (ibs)
    - projects are the framework team's clients
    - projects that use the framework are charged for it
    - project cost estimates include framework usage costs.
- Business advantage from reuse
  - competitive advantage
    - (ibs)
      - same development time, with more and more features delivered in that time

QUALITY ASSURANCE

- Project standards
  - documentation
    - format and layout
    - how much?
      - overall
        - (ibs)
          - a balancing act between being responsive and being absolutely complete
      - spec
        - (ibs) 2-3 page product description, features list
      - design
        - (ibs)
          - whiteboards and large post-it notes
          - overview design description
          - comments in header files (re: interface)

- comments in code files (re: internals)
  - how much documentation burden, beyond content taken directly from engineering work itself?
  - how to decide when you have enough content?
  - who writes it? who reads it? who maintains it?
- coding standards
  - (ibs)
    - what's acceptable, what's not -- by example
- inspections and inspection lists
  - (ibs)
    - design and code inspections
    - opportunity to evangelize new approaches, encourage along the way
    - ego is an issue for some (limiting factor, if cannot resolve)

- Testing
  - who
    - who makes a good tester
  - who tests what
    - part done by developer
    - part done by test team
  - approaches used
    - white box
    - black box
      - (ibs)
        - mostly black box testing
    - scenarios
    - regression
  - plan
    - how developed
      - (ibs)
        - test plan is developed from the (informal) product spec (description and features list)
    - how used
      - (ibs)
        - test plans and scripts -- help testers get ready to test
        - actual tests require more of a "feel" for the product, what to exercise to find defects
  - tools
    - (ibs) when using a framework, coverage is hard to check (since each app is likely to use only part of the framework).
- Process standards
  - SEI process maturity model
    - vs. getting the job done
- Quality standards
  - ISO 9000
    - vs. getting the job done

RISK

- Positively courting risk
- Identification and assessment
  - (ibs)
    - senior team leaders meet regularly
      - project-specific problems

- risks of doing things different ways
- where is the market going? what's the impact on how we are doing business?

Mitigation
- milestone by working results, rather than activity completion

When problems seem insurmountable
- revisiting budget, schedule, resources
- establishing conditions for stopping a project

TRANSITIONING FROM ONE TECHNOLOGY TO ANOTHER

Education
- books, workshops, mentoring
- (ibs)
- video series
  - covered the spectrum of C++ syntax
  - would have been far better if the focus had been on how to use C++ effectively
- evangelism, from early project success
  - initial team of 3 succeeded, then gradually introduced new concepts on three subsequent projects (what works; what doesn't)
- quarterly in-house presentations
  - object-oriented, C++ tips, platform-specific techniques
- reading time
  - encourage the reading of technical books while waiting for a compilation
- platform-specific programming conferences
- encourage self-motivated ones to stir up others
- over time, develop a domain-specific framework
  - this educates, pulls everyone along

Technology transition
- all new
  - new software, using a new technology
  - (ibs)
    - first project was new software
- mix, adding in
  - adding to existing software, using a new technology for the new parts
  - (ibs)
    - used C libraries as-is (from past project work), e.g., functions for drawing things to the screen
- mix, replacing old
  - transitioning existing software, from one technology to another, redeveloping that part of the system

TOOLS

Project tools and reports
- for planning, estimating, measuring, controlling
- (ibs)
- QA tool called Track (just started using)
- Microsoft Project

Technology tools
- requirements, design, construction, test
- (ibs)

C++, cross-platform source code control
whiteboards and large post-it notes
maintenance (on-going development)
performance estimation, measurement
documentation tools
Build vs. buy
are you in the tool-building business?
Getting tools to work together

BEST ADVICE -- some "top 10 or so" lists
"Top 10 or so" list of things to DO
(ibs)
1. Get right people (skills, attitudes, preferences).
2. Complete a product spec.
3. Find ways to reuse code and effort.
4. Gradually develop a domain-based app framework.
5. Recruit team leaders with vision.
6. Emphasize communications between teams.
("No matter what they tell you, it's always a people problem." Weinberg)
7. Give teams some autonomy.
e.g., when a project downloads the latest framework
e.g., how much of the framework a project makes use of
8. Build a prototype and get a "buy off" from the customer.
Having a prototype is worth a million words (a helpful communication tool).
9. Seek out fun projects (interesting, fun to work on).
10. Gain experience by taking many small steps.
(Everything gets better with experience).
11. Keep teams small (3-6); keep framework team even smaller (2).

"Top 10 or so" list of things to NOT DO
(ibs)
1. Don't over-staff (keep teams small; allow time to get the job done).
2. Don't operate in panic mode. Or in overtime mode.
(occasional extra hours are okay, just not the norm; managers work those extra hours, too; meals, too)
3. Don't assume people are interchangeable (not so).
4. Don't throw away experience (capture experience in your framework).
5. Do not design by committee (one or two people decide).
6. Don't oversell the prototype (keep their imaginations from running away).

"Top 10 or so" things that we would have done differently
(ibs)
1. Exclusive, separate framework team sooner.
2. Have the design that has taken us numerous iterations to get to.
3. Projects that we took on that were not good for people or for the company (wisdom for what business we will do, though).
4. Would have stressed education much more. If it helped people

learn and change earlier, that would have made it worth it.

"Top 10 or so" risks and corresponding actions
(ibs)
1. Building a framework is a risk. Will payoff ever happen?
Began as part of a working project.
Then pulled out the team, a non-revenue producing team.
Add costs whenever they bid a project.
Payoff appeared on second or third project.
2. Project-specific risks
Technical risks (feature with acceptable performance)
Attempt to alleviate through prototyping (scope the problem; see if you can get in the ballpark).
People risks (key people leave)
Recruit through referrals within your company (at least half).
Challenge, fun, and rewards.
Team risks (teams not gelling together)
Skill levels
Rankings, subjective evaluations (annual)
Project leaders with great people skills -- counseling, encouraging communication

FAVORITE BOOKS
"Peopleware"
Tom DeMarco and Tim Lister
Dorset House
"Secrets of Consulting"
Jerry Weinberg
Dorset House
(ibs)
"Debugging the Development Process"
Steve MacGuire
Microsoft Press

OBJECT TECHNOLOGY NOTES -- selection, use, and lessons learned
OOA/OOD method
(ibs)
informal
CRC cards, features lists
whiteboard sketches of class hierarchies
whiteboard sketches of object interactions
related to language choice?
patterns use?
OOA & non-OOD mismatch?
kept in-synch with on-going development?
performance issues? (poor performance often means no product)
language
(ibs)
C++
reason: easier, lower-risk transition of C programmers and C apps
related to tool choice (environment; version control)?
related to staff skills?

- impact on productivity?
- impact of mixed languages? allow multiple languages?
- OOD & non-OOP mismatch

tools
- (ibs)
  - Borland C++ (support for templates)
  - Metroworks, Source Safe

GUI

data management
- object apps and relational databases mismatch

object infrastructure
- class libraries
  - catalogs? useful?
- frameworks
  - a hierarchy of classes, especially suitable across a family of apps within a problem domain (or analogous domain)
  - (ibs)
    - example
      - Visual (Thing), specializing into Box, Text, AnimatedBox
      - Button, specializing into many, many kinds of buttons
    - use
      - app developers take advantage of the interfaces and code that's already in place
- patterns

reuse mechanisms
- inheritance
- components
- libraries
- legacy software wrappers
- cut-and-paste from a code repository

configuration management tools
- identification, control, and accountability

testing tools

distributed processing (ORB? client-server?)

=====

---

Peter Coad
Object Intl, Inc. Education-Tools-Consulting. Founded in 1986
"Helping teams deliver frequent, tangible, working results"
8140 N MoPac 4-200, Austin TX 78759 USA
1-512-795-0202 fax 1-512-795-0332
direct line 1-919-851-5422 fax 1-919-851-5579
FREE newsletter. Write majordomo@oi.com, message: subscribe coad-letter
coad@oi.com info@oi.com http://www.oi.com/oi_home.html
pgp: 3DBA 3BDD 57B6 04EB B730 9D06 A1E1 0550 (public key via finger)