UsefulMetaphors

Some vivid metaphors to remind you about key principles

These should be grouped into categories of some sort. Each will likely also become its own Wiki with a good name.

- **Tower of Babel**
  - **Army, navy, air-force interpretation of "secure the building": the air force takes a 3 year lease with an option to buy**
  - **The goal is converged clear and ubiquitous vocabulary**

- **The many faces of an object (interfaces, ports, types)**
  - **Hotel guest and airline traveller**
  - **But shared attributes (wallet), dependent behaviors (stranded overnight at airport)**
  - **Document as seen by Spell checker vs. Layout manager**

- **The label on the box (type specifications need information models)**
  - **To use a machine you need the knobs, levers, and label (& gauges)**

- **Multiple viewpoints that must be consistent (like home architecture)**
  - **don't want a/c vent coming out on top of fan**
  - **candidate for more vivid metaphor**

- **Abstract value types (for params and attributes)**
  - **The power of Date, Date Range, Moving Window, Customer Identification, Account Query**

- **Tornado-like Process with zooming in and out**
  - **Broad and smooth at high level, narrow and dirty near ground**
  - **Repeatedly lifts off and touches down**

- **Don't get paralyzed**
  - **Zoom in and out**
  - **Switch viewpoints info, state, process, examples, ui...**

- **Crisp models do not emerge on day 1**
  - **You have one when previously complex things become simple to express**
  - **Clue off the abstract descriptions the business user uses**
  - **Make many small refactorings, few big ones, timing won't always be ideal**

- **Make implicit concepts explicit**
  - **Many refactorings of models (and code) take a concept that is implicitly referred to, and making it explicit as an attribute, assoc, or type.**

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files
1) RiskyBusiness.pdf

Last changes
1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages
1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

- Alan Birchenough's jigsaw metaphor for how many projects are run (a.k.a. DivideAndBeConquered)
  - To fail to make a jigsaw ...
  - ... cut up board
  - ... each team paints any piece of any picture in any color they like (WeHaveGoodPeopleHere, they know what to paint)
  - ... collect the pieces and put together the jigsaw puzzle
  - ... oops, the picture is nonsense. Where did we go wrong?

- Importance of abstraction
  - what vivid metaphor?

- The importance of clearly stated goals
  - maintain optimal risk to return relationship by controlling portfolio while environmental conditions change
  - what vivid metaphor to use?
    - maybe the M.Hammer BPR one of eliminating the invoice and invoice reconciliation process altogether (you know when you have to pay, so you pay)

- The problem is outside your software; importance of understanding the domain
- Alternately, use this metaphor for outside, boundary, inside; or why/how analysis.
  - The LCD projector example
    - Boundary: spec of power, signal in, image out
    - Outside: what problem am I solving? Get information from laptop to people.
      - Must understand nature of problem domain: how many people? where are they? how much information?
      - How do I solve the problem? Needs projector and wires, but also needs people in same place, same time, presenter present, etc.
    - Inside: internal design of projector

- Generic technical infrastructure
  - How HTTP put/get/post supports most any domain

- Architecture (style) as being about standards
  - Find patterns mapping logical interaction to HTTP put/get
  - SOAP as one example of this mapping
  - How SOAP does not go far enough in standardization e.g. it leaves the identification of the operation, and the resource (s) requested, undefined
    - How this makes some standard infrastructure services very hard e.g. security

- Needless creativity

- Abstract <> Fuzzy

- 2 Minute Rule

- **How to think invariants**
- **Refinement and traceability**
- **Object Info type**
  - **Given object type T e.g. Patient**
  - **and need to pass in/out patient records**
  - **add a derived attribute /info: Patient Info**
  - **if needed, write derivation invariant**
- **UI as**

Last modification date: Tuesday 15 of April, 2003 [12:15:53 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... **comment:**

attach

(0 comments) [Show comments] Hide comments]

CamAsaThinkingTool

CAM is more a way of thinking about problems than just filling in templates

The acid test of a real method or technique is whether it helps the user solve harder problems than they otherwise would be able to tackle. CAM is a good example of a method that does this.

In general, such methods or techniques have the following characteristics:

- **They require the novice to go through a small but worthwhile amount of pain, as they traverse the learning cycle:**
  - **unconscious incompetence followed by**
  - **conscious incompetence followed by**
  - **conscious competence followed by**
  - **unconscious competence.**
- **The techniques act as thinking tools; in other words they effectively direct the user's attention to aspects that may have been missed, or to relationships between those aspects that may have received insufficient attention. (EdwardDeBono defines a thinking tool as an "attention directing device", which I think is a very useful perspective.)**
- **Once mastered to any degree (even slight), these methods can produce results that suprise the user. In other words, the user can actually discover things about the problem or its solution, which they were previously unaware of.**

In the author's experience, this "thinking tool" effect is light years ahead of doing pretty documentation in terms of yielding method benefits.

All this constrasts strongly with the usual ways in which methods or techniques are used, and the surrounding attitudes:

- **We know what we want to do - WeHaveGoodPeopleHere - let's just figure out how to spell it in this new fancy method.**
- **Better still, let's just fill in these templates they've given us! (See LetsJustFillInTemplates.)**
- **All methods are the same anyway. It's all just top down decomposition. (Author note: Unfortunately, this attitude can easily lead to the DivideAndBeConquered antipattern.)**

Last modification date: Tuesday 15 of April, 2003 [13:32:21 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... comment:

attach

(0 comments) [Show comments| Hide comments]

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

CAM : a Community Site

Search

in:

Entire Site go

Login

logged as: admin
Logout

User Defined Menus

[-] Click to Edit [+]

Menu

home
user preferences
categories
[-] Wiki [+]
[-] CMS [+]
[-] Forums [+]
[-] File Galleries [+]
[-] FAQs [+]
[-] Trackers [+]
[-] Admin (click!) [+]
Users
Groups
Cache
Modules
Links
Hotwords
RSS modules
Menus
Polls
Backups
Mail notifications
Search stats
Chat
Categories
Banners
Edit templates
Admin drawings
Dynamic content
Cookies
Content templates
HTML pages
Shoutbox
Referer stats
Wiki Import dump
phpinfo

Quick edit a Wiki page

edit

Last forum topics

Last Created FAQs

NotJustFillingInTemplates

Templates are fine things, but...

Every non-practitioner's (i.e., manager's) ideas about the core of any method worthy of the name seem to revolve around document templates nowadays.

There are many reasons why this is silly; here are a few for starters.

- **A method's real benefit is as a thinking tool, i.e., something which focuses attention on things that matter and preferably suppresses detail about things that don't. (See CamAsAThinkingTool.) Templates do nothing to enhance the thinking process; if anything, they cause the practitioner's brain to seize up. On the contrary, whiteboards are much more useful when thinking than Word documents.**
- **Even if templates did not have this inhibiting effect, they still promote the idea that software engineering is somehow an exercise in "filling in the blanks" in some well-understood structure, whereas, as we all know, ours is not a profession in which mass production can be successful.**
- **Every project needs to slice and dice the simple forward engineering waterfall life cycle in a different way. As soon as you start defining increments, iterations, parallel working, phases, reverse engineering, gap analysis, etc., document templates become useless because the high-level structuring of the documents no longer matches the chosen life cycle (or "route" in Catalysis terms).**
- **Even if project variability was less of an issue, any fixed order of exposition is contrary to the principles of LiterateModeling?, and hence more a display of compliance with arbitrary rules than of a commitment to communicate with ones colleagues.**

At some low level of granularity, though, we do get down to a level where elements are relatively atomic and stable. (Think of a Catalysis action spec, for example.) These MiniTemplates? are really useful and reusable elements from which to build up larger deliverables. (I only wish we had a ConcreteSyntaxEditor? to support the handling of these smaller pieces.)

Last modification date: Tuesday 15 of April, 2003 [09:28:29 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... comment:

attach

(0 comments) [Show comments| Hide comments]

print edit remove lock perms history backlinks similar undo export

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

CAM : a Community Site

Search

in:

Entire Site go

Login

logged as: admin
Logout

User Defined Menus

[-] Click to Edit [+]

Menu

home
user preferences
categories
[-] Wiki [+]
[-] CMS [+]
[-] Forums [+]
[-] File Galleries [+]
[-] FAQs [+]
[-] Trackers [+]
[-] Admin (click!) [+]
Users
Groups
Cache
Modules
Links
Hotwords
RSS modules
Menus
Polls
Backups
Mail notifications
Search stats
Chat
Categories
Banners
Edit templates
Admin drawings
Dynamic content
Cookies
Content templates
HTML pages
Shoutbox
Referer stats
Wiki Import dump
phpinfo

Quick edit a Wiki page

edit

Last forum topics

Last Created FAQs

LetsJustFillInTemplates

Templates are fine things, but...

Every non-practitioner's (i.e., manager's) ideas about the core of any method worthy of the name seem to revolve around document templates nowadays.

There are many reasons why this is silly; here are a few for starters.

- **A method's real benefit is as a thinking tool, i.e., something which focuses attention on things that matter and preferably suppresses detail about things that don't. (See CamAsAThinkingTool.) Templates do nothing to enhance the thinking process; if anything, they cause the practitioner's brain to seize up. On the contrary, whiteboards are much more useful when thinking than Word documents.**
- **Even if templates did not have this inhibiting effect, they still promote the idea that software engineering is somehow an exercise in "filling in the blanks" in some well-understood structure, whereas, as we all know, ours is not a profession in which mass production can be successful.**
- **Every project needs to slice and dice the simple forward engineering waterfall life cycle in a different way. As soon as you start defining increments, iterations, parallel working, phases, reverse engineering, gap analysis, etc., document templates become useless because the high-level structuring of the documents no longer matches the chosen life cycle (or "route" in Catalysis terms).**
- **Even if project variability was less of an issue, any fixed order of exposition is contrary to the principles of LiterateModeling?, and hence more a display of compliance with arbitrary rules than of a commitment to communicate with ones colleagues.**

At some low level of granularity, though, we do get down to a level where elements are relatively atomic and stable. (Think of a Catalysis action spec, for example.) These MiniTemplates? are really useful and reusable elements from which to build up larger deliverables. (I only wish we had a ConcreteSyntaxEditor? to support the handling of these smaller pieces.)

Last modification date: Tuesday 15 of April, 2003 [09:36:47 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... comment:

attach

(0 comments) [Show comments| Hide comments]

print edit remove lock perms history backlinks similar undo export

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

CAM : a Community Site

Search

in:

Entire Site go

Login

logged as: admin
Logout

User Defined Menus

[-] Click to Edit [+]

Menu

home
user preferences
categories
[-] Wiki [+]
[-] CMS [+]
[-] Forums [+]
[-] File Galleries [+]
[-] FAQs [+]
[-] Trackers [+]
[-] Admin (click!) [+]
Users
Groups
Cache
Modules
Links
Hotwords
RSS modules
Menus
Polls
Backups
Mail notifications
Search stats
Chat
Categories
Banners
Edit templates
Admin drawings
Dynamic content
Cookies
Content templates
HTML pages
Shoutbox
Referer stats
Wiki Import dump
phpinfo

Quick edit a Wiki page

edit

Last forum topics

Last Created FAQs

DivideAndBeConquered

Divide in haste, repent at leisure

A sardonic friend and former colleague of mine once commented on the way our little company's software team tried to develop its products. He said "Oh I see. You take a problem and decompose it into a number of much larger problems."

There are a few things that often seem to go wrong with this beguilingly simple idea of "divide and conquer".

- **Component partitioning is tricky. Despite the fact that WeHaveGoodPeopleHere, you need a firm concept of "component", an architecture discipline that separates viewpoints (as CAM does), and a number of well-tried objectively-reviewable component partitioning heuristics. IntuitiveTopDownDecomposition is not usually adequate.**
- **Similarly, on the program/project management side of the house, any division of labor introduces problems of its own. Despite the fact that WeHaveGoodPeopleHere, they will not be able to compensate for a fundamentally uncoordinated approach by means of water cooler conversions alone. As with architecture, the antidote to this problem is to use a number of well-tried process patterns for division of labor, factoring and joining of models, coordination and synchronization, backed up by a rigorous discipline of project estimating.**
- **Both these sets of problems can be made worse by over-confident management based on early poorly thought-out decisions. A favorite kicker is to back everyone into an impossible end date as well.**

BTW I wouldn't even bother to write some of this obvious stuff if I weren't continuing to see it go wrong even in 2003.

Last modification date: Tuesday 15 of April, 2003 [13:16:41 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... comment:

attach

(0 comments) [Show comments| Hide comments]

print edit remove lock perms history backlinks similar undo export

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

CAM : a Community Site

IntuitiveTopDownDecomposition

The top down approach has its faults

My old boss Michael Jackson has written eloquently and amusingly about the problems of top down development. I cannot rival his skill in this area, so here I just offer a quick paraphrase, and a reference to one of his books, which I highly recommend for all sorts of reasons.

Here is the book reference:
Jackson, M.A. (1995), "Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices", Acm Press Books.

And here follows the paraphrase.

Problems with the intuitive light-of-nature decomposition of architectures, programs, projects, organizations and so on, are:

- **The approach makes you face the most difficult decisions (what is the top level partitioning of the major pieces?) before you have understood the problem. You are just partitioning vague abstractions at this stage; the consequences of the design don't emerge until you start running into trouble at the more concrete levels.**
- **Once problems do start to emerge, you face a huge refactoring problem to fix the higher levels. Often it is uneconomic to do this, with the result that everyone has to live with a broken structure until the next bad reorganization comes along.**
- **Top-down is a completely general problem-solving strategy and as such:**
  - **It can be made to fit pretty much any problem BUT**
  - **It has nothing specific to say about any particular problem.**

When adopted amongst software professionals, this procedure often forms a fatal synergy in combination with the WeHaveGoodPeopleHere bogus line of argument.

A hierarchical structure is, however, a nice way to structure descriptions of things we already understand pretty well.

Last modification date: Tuesday 15 of April, 2003 [13:04:21 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... **comment:**
attach

(0 comments) [Show comments| Hide comments]

print edit remove lock perms history backlinks similar undo export

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns

CAM : a Community Site

Search

in:

Entire Site go

Login

logged as: admin
Logout

User Defined Menus

[-] Click to Edit [+]

Menu

home
user preferences
categories
[-] Wiki [+]
[-] CMS [+]
[-] Forums [+]
[-] File Galleries [+]
[-] FAQs [+]
[-] Trackers [+]
[-] Admin (click!) [+]
Users
Groups
Cache
Modules
Links
Hotwords
RSS modules
Menus
Polls
Backups
Mail notifications
Search stats
Chat
Categories
Banners
Edit templates
Admin drawings
Dynamic content
Cookies
Content templates
HTML pages
Shoutbox
Referer stats
Wiki Import dump
phpinfo

Quick edit a Wiki page

edit

Last forum topics

Last Created FAQs

WeHaveGoodPeopleHere

Good people are no substitute for good methods

I couldn't resist putting this topic in to complement some of the other rants I have added to this Wiki recently.

I believe it is important for us to speak out when needed against the rather silly but often-encountered argument that if someone is a "good person", they:

- **Are omniscient: They know so much that they don't need to model any of it;**
- **Are infallible: They can architect and design whole systems without ever going through a detectable or reviewable reasoning process;**
- **Are telepathic: They can compensate for problems introduced at a management level by use of the DivideAndBeConquered anti-pattern;**
- **Must be cosseted: They would suffer terrible limitations to their creativity if forced to follow any sort of discipline;**
- **Would be insulted by the idea that their work could profit from any kind of review.**

These arguments are usually followed up by the statement that everyone around here is good in the sense described above, so basically, we don't need no stinkin' method. (But we think CAM is terrific. No, really...)

I need hardly point out that, contrary to these arguments, modeling, reviews, discipline and careful management have an immensely tonic effect on the creativity and morale of already-good people. (Watts Humphrey even has figures that show how people cheer up when they start using the Personal Software Process and Team Software Process, despite the heavy emphasis on defect tracking and relentless root-cause analysis.)

Last modification date: Tuesday 15 of April, 2003 [13:28:58 CDT] by AlanBirchenough

name	uploaded	size	dls	desc
No attachments for this page				

Upload file: Browse... comment:

attach

(0 comments) [Show comments] Hide comments]

print edit remove lock perms history backlinks similar undo export

1) CAM Architecture Styles for Capital Markets
2) CAM Tool Support
3) Origins of CAM

Last Files

1) RiskyBusiness.pdf

Last changes

1) WikiWants
2) ProjectBookRequirements
3) Use Case
4) SandBox
5) ProjectBook
6) CAM Welcome
7) CamAsaThinkingTool
8) WeHaveGoodPeopleHere
9) DivideAndBeConquered
10) IntuitiveTopDownDecomp

My Pages

1) CAM Welcome
2) ModelsAndCode
3) KevinIME
4) CAMwithRose
5) BlackAndWhiteIsRelative
6) PortSpecVsComponentSpe
7) BlackBox
8) WhiteBox
9) EffectiveSiteUsage
10) LevelsVsConcerns