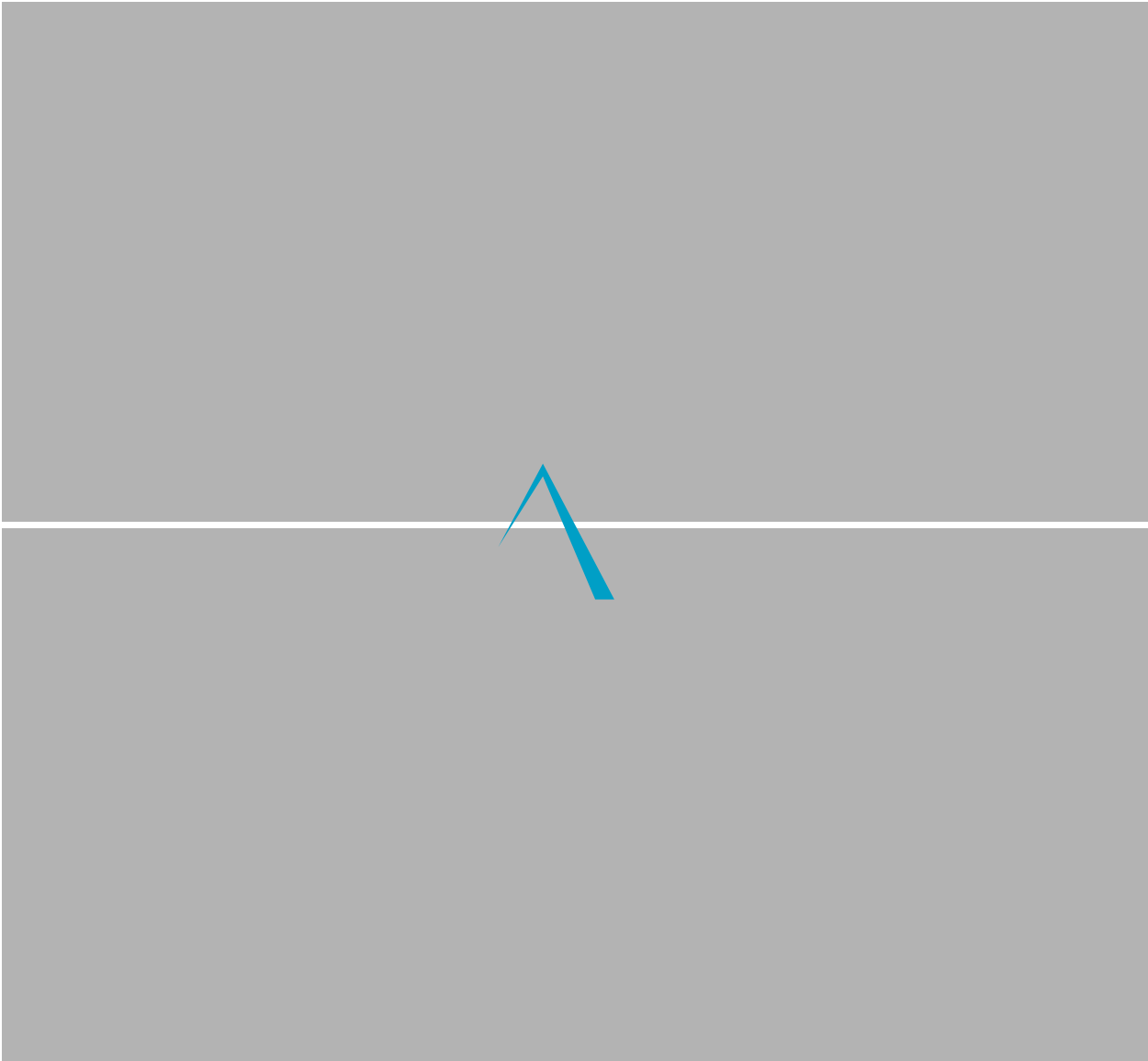




Integrity in Product and Software Development

mindAmp Viewpoint





Integrity in Product and Software Development

*Conceptual integrity and
execution integrity.*

In the realm of product and software development, we define integrity as simply the ability to stand by an idea. Of course, there's really nothing simple about that ability. It's a lot easier to make assumptions about various moral and ethical attributes that you possess than to simply and resolutely stand by an idea or set of ideas.

There are two forms of integrity that form the core of how we develop and deliver software: conceptual integrity and execution integrity.

Conceptual Integrity

Conceptual integrity is the act of designing, developing, and delivering software with a central unifying theme. It is consistently the most overlooked aspect of most software development. We deliver software that has a central unifying theme both in terms of its operation as well as in terms of its internal structure.

*Reject democracy and
embrace meritocracy.*

How do we do it? We use it by completely rejecting democracy and consensus organizing principals for designing software. Each project is assigned a leader who is completely responsible for the design, development, and delivery of the software that will meet and exceed the expectations of our customer. Each person working for the customer on that project has an assigned role and responsibility, and the architecture of the software is designed such that each person can take responsibility for one or more components within the framework of the previously established theme.

Over twenty-five years ago, Fred Brooks wrote in *The Mythical Man Month* words that we believe deeply:

I will contend that conceptual integrity is the most important consideration in system design.

We all know conceptual integrity when we see it: the Apple Macintosh has it while the Windows operating system doesn't. Conceptual integrity resonates in our mind as an elegant unity of design; once we understand a small part of the design, the remainder of the design seems elegant and



Conceptual integrity is the primary tool that is available to maximize ease of use.

natural. There are no discordant surprises; the overall feature/functionality is harmonious.

This philosophical view of conceptual integrity is the one we tend to think of most often; however, there is also a quantitative rendering of the concept of conceptual integrity that we think is quite apt. Conceptual integrity in this sense is given by the following equation

In this equation, conceptual integrity is defined as the total number of functions exposed by a system (e.g., the total number of functions of an API) subtracted by the minimum number of functions that must be understood in order to

$$ConceptualIntegrity = \frac{\sum Functions - Functions_{Minimum}}{\sum Functions}$$

understand the next API. The best possible conceptual integrity would tend toward 1, and the worst possible conceptual integrity would tend toward 0.

Ease of use is closely coupled to the conceptual integrity of a system. Ease of use is linked to conceptual integrity through the following equation

where conceptual complexity is the inverse of conceptual integrity.

The goal must be to decrease conceptual complexity as

$$EaseOfUse = \frac{\sum Function}{ConceptualComplexity}$$

Conceptual integrity and ease of use are not vague, amorphous concepts.

much as possible while offering as much functionality as possible. Conceptual integrity and ease of use are not vague, amorphous concepts but rather are rigorous, quantitative entities that may be directly manipulated to increase the usefulness of a system.

Execution Integrity

If conceptual integrity is the most important consideration in system design, then execution integrity is the most important consideration in system implementation.



*Doing
what
we say,
when
we say we'll do it.*

Execution integrity is the act of doing what we say, when we say we'll do it. This is a rarity in software development where most projects are delivered behind schedule, with poor quality, reduced feature/functionality, and over budget.

How bad is execution integrity within the software business?

- ⌘ The average software project finishes 220% behind schedule and 190% over budget. [Construx Software, 1999]
- ⌘ The average software project is 50% late, 45% over budget, and a colossal failure in customer satisfaction. [KPMG, 1997]
- ⌘ In 1995, 85% of software projects finished over time or budget. 1/2 of projects double cost estimates. [SEI, 1999]
- ⌘ Statistics reveal that the average software project exceeds its budget by 90% and its schedule by 120%. [Standish Group, 1998]

We make commitments to our customers in four fundamental areas: feature/functionality, quality, schedule, and price. And we stake our own profitability on our ability to execute with integrity with our fixed price, fixed schedule proposal process.

How do we do it? There's no magic to it. We plan, we execute to that plan, and we adjust our planning and execution continuously the dynamic requirements of our customers. Our plans are incredibly detailed and yet flexible enough to enable us to make adjustments due to newly discovered external and internal issues before there is any impact on a customer deliverable. This is particularly important since our average time between customer deliverables tends to be seven to fourteen days; without a tremendous amount of attention to our detailed internal plans, it wouldn't be possible to maintain this deliverable rate successfully.

*How many
organizations
link project planning to the
customer deliverable
continuously,
instead of linking project
planning only to the
initial quote?*

Of course, everyone uses project planning tools, so what's the difference between mindAmp and others? In addition to the depth of the planning we do, we have a series of custom tools that allow us to link the planning process to the customer deliverable. These tools allow us to operate at a finer granularity than the typical "run Microsoft Project and then code" company and even allow us to operate more efficiently



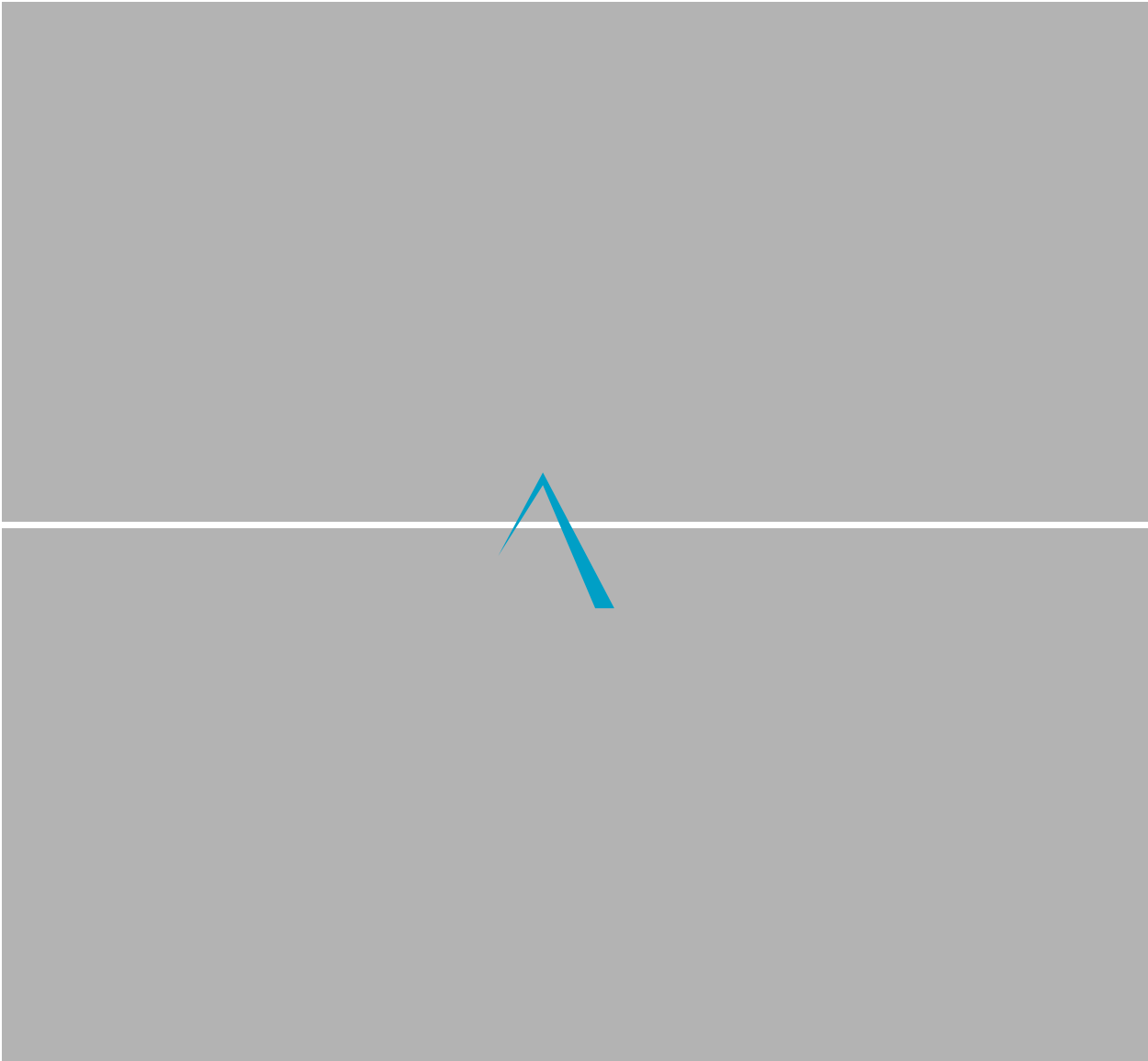
*Conceptual integrity
is a key measure
of the ability
of a
system
to support a
viral adoption
vector model.*

and precisely than organizations that use best-case methodologies and processes. This is the backbone of execution integrity.

What's the Payoff?

The payoff of conceptual integrity are systems that are inherently more powerful and yet easier to use. This leads to a decreased barrier to entry and an increased barrier to switch - key ingredients supporting increased revenue and the possibility of supporting a viral adoption vector of use.

The payoff of execution integrity are systems that are delivered incrementally and predictably. This means that marketing, sales, and services can interoperate with development in order to maximize the success of the company as a whole both in gaining continuous feedback concerning the system and in building marketing, sales, and services programs that may be delivered with precision and effectiveness with relation to the delivered system.





mindAmp

www.mindAmp.com

© 2001 mindAmp Corporation. All Rights Reserved.

Contact information:

Voice: [US] 803.892.3479

Fax: [US] 803.892.6055

E-Mail: Campbell@mindAmp.com

Headquarters

mindAmp Corporation

2130 Shull Avenue

Gilbert, SC 29054