Improving IT Effectiveness

As we close the first decade of the 21st century, IT solution delivery continues to have a reputation for high cost and high risk, with underlying IT productivity substantially unchanged over the last 20 years. This paper discusses an underlying cause of stagnating IT effectiveness, and outlines a new approach to acquiring technology that can improve operating effectiveness while substantially reducing costs.

Key phrases include:

- Commoditization of IT is lowering software cost while improving quality and capability
- Inter-operation with, and integration of, commoditized software is easily achieved with modern integration approaches
- Decision making know-how is definitive core IP that provides competitive edge
  – it should be catalogued and automated
- 'Systems' are not synonymous with this core IP – they either contain it or they connect to it
- The majority of system code in use today is not core IP and can be commoditized
- Integrating bespoke decision-making components into commodity software systems gives rise to bespoke software outcomes at commodity software costs
- A sea-change in IT thinking and approach is required to obtain the significant benefits of commoditized IT

IT Effectiveness	Commoditization	Mass Customization
Decisioning	The IDIOM Value Proposition	Our Problem/Your solution

IT Effectiveness

For many executives it is difficult to 'put a finger on it', a troubling concern about IT effectiveness that is felt through high ongoing costs and a frustrating inertia that constrains business agility.

This feeling is aggravated by a complexity that is deeply entrenched in the semantics and approaches of today's IT. As the following excerpts (all published 2008) indicate, today's IT is a beast 'not yet tamed'.

- "The odds of a large project finishing on time are close to zero. The odds of a large project being canceled are an even money bet" – *Rapid Development: Taming Wild Software Schedules* (Paperback), By Steve McConnell[1]
- "What's more, the failures are universally unprejudiced" – From the IEEE article *Why Software Fails*[2]
- "If the assets of a failed project can all be considered waste, in 2006, software value was measured at 59 cents on the dollar" – Standish Group Report, SDTimes 9/1/2008
- "For example the cost of project failure across the European Union was €142 billion in 2004" – *The British Computer Society* (BCS), Dr John McManus and Dr Trevor Wood-Harper, June 2008
  http://www.bcs.org/server.php?show=ConWebDoc.19584

The inability to reliably deliver IT solutions within acceptable business parameters is a malaise that is 'universally unprejudiced', affecting systems both built and bought, new and old. One underlying cause is a continued presumption of a need to own the traditional line-of-business or enterprise scale system, where 'own' implies an exposure to both the development and the ongoing maintenance burden. In fact the scale of the traditional system is one of the major factors driving higher cost, risk, time-to-market, and system rigidity, as the following chart (derived from Capers Jones' book *Applied Software Measurement*[3]) indicates.

System Size in Units	50	500	5000	50000	500000	5000000
Over-runs [percentage]	0%	1%	11%	24%	46%	66%
Risk of Project Failure [percentage]	0%	2%	5%	9%	45%	70%
Annual Changes* [percentage]	(no data)	(no data)	10%	9%	7%	4.5%
Defects [rate per Unit]	0	5	20	40	60	90
Cost [rate per Unit]	10	13	19	24	37	51

* implying a measure of rigidity

1 & **2.** As quoted by http://www.codinghorror.com/blog/archives/000588.html, blogger Jeff Atwood
3. These approximate numbers were derived from "Applied Software Measurement", Capers Jones, published by McGraw-Hill.
     For a more authoritative discussion on this topic, see Capers Jones website http://www.spr.com/

Systems *per se* do not deliver business value; the value arises from specific features and functions that the system might provide. Given that ownership is expensive, then it becomes a question as to how much of the system code base needs to be 'owned' in order to derive the various benefits of the system? And do all benefits provide equal value to the organization, or do some benefits create value, while others are of lesser importance? Answers to these questions regarding cost and benefit are important if we are to determine more effective IT outcomes.

Until the current decade, cross system communications and integration technologies tended to be proprietary, and larger, fully integrated systems were a necessity, albeit at a high cost in terms of money, risk, time and functional rigidity. Today, the widespread use of standardized integration technologies, from web services to RSS to SQL, has enabled the rise of commodity components. Commoditization in this context means the competitive supply of like components, leading to reduced cost, competition driven improvements, specialization, and mass-market scale. Commoditized components now provide the raw materials for low cost composite systems that are both more sophisticated and more flexible than yesteryear's line-of-business or enterprise class systems.

By way of analogy, one could argue that because an organization needs power it should build or buy power-plants just as it does computer systems – in which case the equivalent of the legacy IT function would argue about the merits of gas turbines vs windmills, and whether these should be built or bought. Many of today's corporates still have the systems equivalent of a water-wheel or a steam engine in their computer room, and are debating whether to upgrade to a modern turbine or windmill. In fact, at least for power, these arguments ended generations ago when executives started buying power from the grid.

Owning the maintenance burden of most of the functionality provided by your computer systems is like owning a power station – a source of 'troubling concern'.

Commoditization

When contemplating acquisition of new software, executives are increasingly asking that important question: "How much of the code base does the business actually need to own to achieve its purpose?" – (i.e. does it need to own the power-plant?)

By asking this question, an organization is positioned to take a more discriminating view regarding the provision of the various features and functions that it requires. Many of the features and functions in today's systems are being commoditized, and are available as low cost services, or as low cost or open source products.

When the organization shifts its focus to take a commodity view of required features and functions, and away from a presumption of the need to build or buy a full business system, then it subtly reclaims a business centric viewpoint. The nature of the business case is transformed from the risky justification of a large IT capital expenditure (that is usually based on point-in-time requirements that cannot easily be verified – at least, not actually tested for

correctness, consistency or completeness), to identifying and acquiring specific lower cost benefits when and as needed. Executive focus is now shifting towards understanding what needs to be internalized in order to protect and leverage the core productive IP of the organization for competitive edge; and by way of contrast, what can be sourced from commodity suppliers. So, the key issue is not whether to build or buy systems, but how to develop and leverage the core IP that drives competitive edge at the lowest cost and with the greatest ongoing flexibility.

"Consumerization of technology should be a broad manifesto for change in corporate IT and enterprise vendors," according to Mirchandani, Computerworld blogger[4]. "Let's face it – we are slower, uglier, exorbitantly expensive, obsessed with security and compliance." As he sees it, it's time for an extreme makeover.

For IT infrastructure and operations, commoditization is well established and continues, for example, with cloud computing. Renting raw capacity in the cloud for the same cost as the depreciation on equivalent purchased systems achieves more than a stable, low cost infrastructure; it also loses much of the burden of ownership – the internal support, housing, and management overhead that contributes to the 'troubling concern'.

A similar shift is now appearing with regard to applications. Most processes are common across like businesses – the very existence of the major system vendors attests to this. So we are now seeing commoditization of many support processes, as specialist vendors provide more capable processes either online as services, or as downloadable, installable software. By definition, these commodity software processes are generic in nature – processes from generic databases to CRM and sales management, rather than the core production processes that are associated with competitive edge. Purchase and use of these commodity processes is often a business decision as much as a technical one – more akin to the process of selecting a mobile phone on the basis of features and performance, rather than on the basis of an 'under the hood' technology evaluation.

Mass Customization

If all processes are commoditized, and all organizations have more or less equal access to the lowest cost supplier, how does an organization differentiate itself and, as an MIT Sloan report[5] claims, join the mere 7% of corporates who derive the benefits of increased growth and reduced spend from effective IT.

The next wave in the commoditization of IT, and the focus of this briefing, addresses this problem directly through "mass customization" of software. Mass customization is "the use of flexible computer-aided manufacturing systems to produce custom output" and "is the new frontier in business competition for both manufacturing and service industries. At its core is a tremendous increase in variety and customization without a corresponding increase in costs. At its limit, it is the mass production of individually customized goods and services. At its best, it provides strategic advantage and economic value."[6]

4. http://blogs.computerworld.com/blogwatch
5. "Avoiding the Alignment Trap in Information Technology", MIT Sloan Management Review as reported by TechRepublic's Jason Hiner http://blogs.techrepublic.com.com/hiner/?p=615,
6. http://en.wikipedia.org/wiki/Mass_customization
7. http://en.wikipedia.org/wiki/Haecceity
8. For a conceptual discussion on decisioning see http://www.idiomsoftware.com/pdfs/IDIOM%20Decisioning.pdf

Before we can apply the concept of mass customization to software components, we need to understand precisely what it is within a system that should be customized, and how the customization should be applied. To do this, we need to define and capture the 'haecceity'[7] of the organization (from the Latin haecceitas, which translates as "thisness"). Haecceity "denotes the discrete qualities, properties or characteristics of a thing which make it a particular thing" – it is an organization's "thisness" that defines the essential customization requirement, just as the personality of an individual drives the customization of a pair of jeans. The key manifestation of an organization's "thisness" is how it applies its proprietary knowledge to everyday business events through decision making. The unique decision making knowledge of an organization can be captured and embedded within commodity processes to create "Dynamic Business Applications", which are prime examples of mass-customization at work.

By applying the concepts of mass customization to the manufacture of the software that defines competitive edge, and by acquiring commodity software for the remaining non-core features and functions, we can derive bespoke system outcomes at commodity system prices.

Decisioning

The term 'decisioning'[8] is in growing use in the business systems lexicon. The concept of 'decisioning' recognizes the fundamental importance of decision-making to any organization. An organization's decision making is the proprietary know-how that drives value and differentiates one organization from another in the marketplace. Because of this fundamental importance, decision-making is usually visible as policy, from where it can be distilled into precise 'decision models'. Decision models are independent, auditable, and electronically testable specifications of corporate policy that can align your systems to your strategy, while at the same time improving operational efficiency and agility.

Decisioning is the management approach that supports the development and use of decision models in a 'decision centric' organization.

Decision models can be rendered simultaneously into human readable documentation (PDF) and computer readable source code, to provide a live, persistent, and direct link between corporate policy statements and operational computer systems. This explicit linkage provides useful context for systems acquisition projects, and can be used with an appropriate methodology to guide project development or component acquisition, reducing project risk, cost and time by significant margins. And because they are decision making proxies for the organization's subject matter experts, decision models put the business in day-to-day control of automated system responses – analogous to a remote control for the organization's computer systems.

Decisioning is the unique 'thisness' that can be implemented into any process to make the behavior of the process proprietary – the embedded decisioning makes an otherwise generic process bespoke in terms of its behavior, without the cost, risk, and time-to-market that is the hallmark of a hand-crafted process. Decision modules can be strategically deployed to bring bespoke behavior to an otherwise commodity based system. Furthermore, by using advanced model driven code generation, the development cost, risk, and time parameters of producing bespoke decision modules is on par with the best value commodity offerings.

IDIOM's measured experience is that using a decision modeling approach for the analysis and production of decision modules is 20 times more effective than producing the same code through a traditional SDLC approach. And this advantage is retained through all future changes.

The transition to a 'decision-centric' approach is relatively simple from a technical perspective. From an organizational perspective, it implies a mild and generally cost beneficial adjustment in the division of responsibilities between the business process owners and those who are responsible for the technical aspects of process implementation. This adjustment in the relative ability of the business and technology process owners to manage the organization's automated processes is an important step in improving IT effectiveness that can benefit from being executive led. These changes should be matched by an approach to systems acquisition that has the following characteristics:

- Strong executive ownership of the organization's value strategy and supporting policies.
- Delegation of the ownership of specific policy implementation to business functional units – this includes ownership of decision making.
- IT refocuses on the engineering of the underlying processes, and on providing the infrastructure capability and capacity that the processes require. Decision models from the functional units must be acquired and loaded into the processes as required.
- A tool is required to manage the decisioning 'contract' between IT and the functional units.
  IDIOM Decision Manager can be deployed and managed by IT, and used by the functional units to define and deploy decisioning as per their delegated authority.
- IT systems development strategies must adapt to recognize this new, direct functional unit involvement in process control.

The IDIOM Value Proposition

Implementing the changes above leads to wholesale improvement in business performance as follows:

Strategy and Policy Benefits

- When the approach is fully implemented, operational decision making can be progressively automated and updated independently of systems development priorities.
- Decision making behavior is defined, catalogued, and managed by the functional unit that is charged with implementing the relevant policy.
- Decision making can be audited and verified, and shown to be aligned with and arising from agreed policy.
- Decision making can be tested for correctness, completeness and consistency, thereby validating the assumptions underlying the relevant policies at the same time as proving the efficacy of the deployed decision-making.
- By using decisioning strategies to control processes, other corporate resources and activity (both systems and manual) can be coordinated and aligned with corporate policy.
- Decision automation means faster, more accurate and more consistent transactions at lower cost across the board.
- Decision enabling new and existing processes can drive process re-engineering and process improvement on a large scale.

Functional Unit Benefits

- Responsible business owners are empowered and in more direct control of the available system resources.
- The relationship between business owners and IT becomes more structured with more predictable outcomes.
- The organization becomes more agile when system behavior is driven by business defined decisions without translation through the traditional software 'SDLC'.

Information Technology Benefits

- With decision making as 'content', systems development is de-risked, with lower cost, more reliable systems delivery.
- Choice of build, buy, or rent of system processes becomes easier when the corporate IP is already external to the process and cannot be compromised.
- IT ceases to have implied responsibility for business outcomes, and can focus on higher quality process engineering and capacity management.
- There need never be another legacy system – legacy processes can be transformed by appropriate transfusion of the embedded decisioning.

> **The IDIOM measure of IT effectiveness:** Percent automation of policy defined decisions * Number of system participants * Average transaction value / IT cost

Our Problem/Your solution

The power of decision based technologies to drive commodity agile, low cost solutions is so substantial, and the strategic implications so significant, that it may seem 'unbelievable'.
So we have made it very easy for you to engage with us and verify our claims.

- The easiest way is to call us. We will provide you with an introduction to our product and approach via GoToMeeting, after which we can discuss how these could be applied to your business.
- If you like what you hear, then we will suggest that we do a few days paid engagement to produce a small pilot that resolves a real problem in your business – worst case, a few days work for each of us to deliver the solution to the problem. No license fee is required.
- If you like what you see, then we can engage further to build more complex solutions, and/or to work with your in-house team to do the same. Only when you are entirely comfortable that our product works as claimed in your environment will we approach you for a license fee.
  (See http://www.idiomsoftware.com/purchases/assistedpilot58.asp for details of this approach.)
- Alternatively, you can experiment with our **IDIOM** Lite product if you prefer a do-it-yourself approach. We will still be happy to provide a GoToMeeting review session to get you started. **IDIOM** Lite is now available for free download, and low cost, pay-as-you-go code generation. Payment is only required after **IDIOM** produces bug-free, ready-to-go source code. This code is generated to decisioning specifications developed and tested by you or your analysts using the **IDIOM** Decision Manager.
  (See http://www.idiomsoftware.com/purchases/selfservice58.asp for details of this approach).

For more information please contact:

Mark Norton, mob +64 21 434669, mark.norton@idiomsoftware.com
Mark Hamilton, mob +64 21 748751, mark.hamilton@idiomsoftware.com