In *Extreme Programming and Agile Methods: XP/Agile Universe 2002*, pp. 70–88, Springer Verlag, LNCS 2418, Editors, Don Wells and Laurie Williams.

Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell About It

Glen B. Alleman

Niwot Ridge Consulting
Niwot, Colorado 80503
galleman@niwotridge.com

Abstract: The selection, procurement, and deployment of an Enterprise Resource Planning (ERP) system is fraught with risk in exchange for significant business and financial rewards [26]. In many cases the packaged ERP product does not provide the entire solution for the business process. These gaps can be closed with third party products or by customizing existing products. Management of this customization, as well as the selection of the core ERP system has traditionally been addressed through high–ceremony, science–based, project management methods [13]. Well–publicized failures using this approach creates the need for new methods for managing ERP projects [11]. This compendium paper describes an alternative to the traditional high–ceremony IT projects management methods. Although many of the methods described are not new assembling them into a single location and focusing on a single issue provides the tools to make decisions in the presence of uncertainty, focus on the critical success factors, and address the managerial and human side of project management Agility allows the project management methods as well as the system to be adaptively tailored to the business needs.

1. Introduction

Using accepted standards for doing business significantly reduces the coordination efforts between business partners as well as internal information and workflow processes [46]. ERP provides the means to coordinate and manage this information, by integrating enterprise information and business processes.

Managing an ERP project is not the same as managing a large scale IT project. IT projects emphasize requirements elicitation, detailed planning, execution of identified tasks, followed by end–to–end delivery of business functionality. Even though this project methodology faces difficulty when scaled to larger projects, applying it to ERP projects creates further difficulties.

The ERP environment faces constant change and reassessment of organizational processes and technology [67]. The project management method used with ERP deployments must provide adaptability and agility to support these evolutionary processes and technologies [33]. The use of agile methods in the ERP domain provides:

- Increased participation by the stakeholders.
- Incremental and iterative delivery of business value.
- Maximum return on assets using a real options decision process.

1.1 What's the Problem Here?

The major problem with software development (and deployment) is managerial, not technical.

The notion that Commercial Of The Shelf (COTS) products are the solution to business problems *out of the box* has pervaded the literature [13]. The application of *scientific* management principles to these projects is understandable. The use of predictive strategies in this environment is inappropriate as well as ineffective since they do not address the emergent and sometimes chaotic behaviors of the market place, the stakeholders, and the vendor offerings.

This paper describes a method of augmenting structured project methods with agility to produce a new approach to managing ERP projects. This agile approach requires analytical tools for making the irrevocable decisions in the face of uncertainty found in the ERP domain. This approach provides methods for dealing with the interpersonal, stakeholder, and business process issues that arise in the rapidly changing ERP environment.

Agile methods provide the means to deliver not just pretend progress but real progress, measured as business value to all the participants – buyer, seller, and service provider.

1.2 What is an ERP Project?

The term *Enterprise Resource Planning*, coined in the early 1990's, is a software application suite that integrates information and business processes to allow data entered once to be shared throughout an organization. While ERP has its origins in manufacturing and production planning systems, it has expanded to *back–office* functions including the management of orders, financials, assets, product data, customer relations, and human resources.

Thinking about an ERP project as a large–scale IT deployment leads to several unacceptable propositions [13]:

- Spend $2 million, $20 million, or even $200 million up front for a new technology with a 50% to 70% probability of a partial or complete write off of the investment.
- If unwilling to write off the investment, double the original investment to complete the project successfully.

1.3 ERP Project Management and Normal Science

Modern project management is heavily influenced by the belief that a project management process can be improved by scientific methods [16, 26]. These include the beliefs create the myth that:

- Clear–cut investment opportunities with an explicit purpose, beginning, duration, and end can be identified early in the project.
- Low opportunity costs for each business or technical decision exist, in most instances with a reversible decision process.
- Feasible, suitable, and acceptable project attributes can be identified.
- Accurate predictions of project duration and resource demands are possible once the requirements have been defined.
- Worst–case consequences can be determined in advance.
- The failure of the project was due to lack of skills rather than inappropriate feasibility, suitability, or acceptability of the solution.

This is a normal–science view of project management. In the ERP domain it can be replaced with a post–modern view [1], in which there are:

- Highly uncertain facts about the project attributes.
- Constant disputes about the values and expectations.
- High decision stakes with irreversible consequences.
- Urgently needed decisions in the presence of insufficient information.
- Outcomes that affect broad communities of interest.

Agile methods do not mean that the normal–science model is irrelevant, just that such a model is applicable only when uncertainty and decision stakes are low [37].

A fundamental attribute of post–normal science is the reliance on heuristics [32, 51]. Using heuristics to guide the development using agile methods allows the management of ERP projects to be placed in a post–normal science context.

1.4 ERP Projects are New Ventures

The agile methods used to manage an ERP project can be taken from the *Venture Capitalist* approach rather than the *IT Managers* approach [3, 7, 8]. These methods include:

- *Staged Investments* – capital must be conserved.
- *Managed Risk* – all participants must share the risk.
- *It's the people stupid* – the composition of the participants is "the" critical success factor.

---

[1] Classical science and conventional problem solving were labeled "normal science" by Kuhn [53]. Post–Normal science acknowledges there is high system uncertainty, increasing decision stakes, and extends the peer review community to include the participants and stakeholders, who insure the quality and validity of the conclusions [37].

1.5 ERP is also Enterprise Transformation

Three major processes make ERP projects significantly different from traditional IT projects.

- *Process reengineering* – is about replacing business processes that have evolved historically within the organization with new and innovative processes embodied in the ERP system. If the business needs aren't met in some way by the ERP system, there is a temptation to *customize* it. If this is done, an instant legacy system is created with the similar maintenance and support problems as the previous system.
- *Package the delivery of IT capability* – is about staging the delivery of system components and their business value to maximize these resource investments by the continuous delivery of business value.
- *Shift toward business processes modularity* – is about modularizing the *architecture* of the organization as well as the software. There is technical architecture, data architecture, application architecture, and enterprise architecture. The deployment of ERP impacts all four of these architectures.

1.6 What is Architecture and Why Do We Care?

One approach to agile deployment of ERP systems is to begin with system architecture. Several benefits result:

- *Business Processes are streamlined* – through the discovery and elimination of redundancy in the business processes and work artifacts.
- *System information complexity is reduced* – by identifying and eliminating redundancy in data, software and work artifacts.
- *Enterprise–wide integration is enabled through data sharing and consolidation* – by identifying the points to deploy standards for shared data, process, and work artifacts.
- *Rapid evolution to new technologies is enabled* – by isolating the data from the processes that create and access this data.

Architecture is a set of rules that defines a unified and coherent structure consisting of constituent parts and connections that establish how these parts fit and work together [69]. Many of the attributes of building architecture are applicable here. Form, function, best use of resources and materials, human interaction with these resources, reuse of design, longevity of the design decisions, and robustness of the resulting entities are all attributes of well designed buildings and well designed software systems [1, 2].

While architecture does not specify the details of any implementation, it does establish guidelines to be observed in making implementation choices. These conditions are particularly important since ERP architectures embody extensible features that allow additional capabilities to be added to previously specified parts [56].

In the COTS domain, architecture provides the guidance to the development team to direct their creativity.

2. How to Implement an ERP System

IT projects traditionally use formal management processes for the acquisition or development, deployment, and operation of the system that emphasizes planning in depth. This approach organizes work into phases seperated by decision points. Supporters of this approach emphasize that changes made early in the project can be less expensive than changes made late in the project.

In the past this approach has been called waterfall. [2] The waterfall approach contains several erroneous assumptions that negatively impact ERP projects:

- *Planning* – It is not humanly possible to produce a plan so that its implementation is merely a matter of executing a defined set of tasks.
  - Plans for complex projects rarely turn out to be good enough for this to occur.
  - Unanticipated problems are the norm rather than the exception.
- *Change* – It is not possible to protect against late changes.
  - All businesses face late changing competitive environments.
  - The window of business opportunity opens and closes at the whim of the market, not the direction of the project manager.
- *Stability* – Management usually wants a plan to which it can *commit*. By making this commitment, they give up the ability to take advantage of fortuitous developments in the business and technology environment [72].
  - In a financial setting this is the *option value* of the decision.
  - Deferring decisions to take advantage of new information and new opportunities is rarely taken into account on IT projects [74].

2.1 The Road to Hell is Paved with Good Pretensions

The erroneous assumptions in §1.3 create a dysfunctional relationship within the project that undermines its effectiveness. This dysfunctional relationship is created when:

- The client pretends it is possible to define milestones and deliverables far in advance. The client then creates a project plan that formalizes these milestones.

---

[2] The term *waterfall* has been used many times as a *strawman* by the agile community. In fact very few pure waterfall projects exist today. This is not to say there are not abuses of the concept of waterfall – sequential development based on the simple algorithm `REPEAT [Design, Code, Test] UNTIL Money = 0`. In practice, development and deployment processes based on incremental and iterative methodologies are the norm. The literature contains numerous references and guidelines to this iterative project management approach dating back to the 1980's [65].

- The vendor pretends that it can meet these milestones in order to get the business.

Both parties maintain the illusion of good project management by pretending they know how to meet these milestones, when in fact they are headed for failure.

2.2 Planning in the Presence of Uncertainty

Plans are unimportant; planning is essential – D. D. Eisenhower

The rules of thumb for applying agile processes are built around the increasing levels of uncertainty experienced by the project [31].

- *A clear future* – a single consistent view of the outcome.
- *Alternative futures* – a small set of outcomes, one of which will occur.
- *A range of futures* – many possible outcomes.
- *True ambiguity* – no specified range of outcomes.

The higher the degree of uncertainty the more effectively agile methods can replace high–ceremony methods [10, 70]. In the presence of, the difficulty of planning does not remove the need for planning – it simply changes its purpose:

- Plan in order to gain understanding.
- Plan for unanticipated events – this is called risk mitigation.
- Don't take planning too seriously – the original plan is simply a guide to the future – it is not *the* future.

2.3 Avoiding Dysfunctional Relationships

Using the three key aspects of a *Venture Capital* methodology reviewed in §1.4, ERP projects can as if they were of as business ventures [13]. Using a post–normal methodology, ERP management includes:

- *Staging* – deploying all the ERP features at once to gain the benefits of the integration and infrastructure is not a good V*enture Capital* decision.
  - Different projects have different cash flow requirements therefore different deployment requirements.
  - Capital investment moves to locations with *acceptable* or *low* cash flow requirements.
  - The risk / reward proposition must be reasonable for the capital investment requirements.
- *Incentive alignment and risk sharing* – among the parties, cooperative problem solving is a critical success factor.
  - Vendor and system integrator payments should be linked to the accomplishment of real tasks, not milestone dates.
  - Senior managers' compensation should be based on successfully delivering components of the project in an incremental, iterative manner with measurable business value.

- There must be no conditional support. Every one should have some skin in the game. It's going to get ugly no matter what happens, so conditional support is the kiss of death for an ERP project.

- *People are the key to success* – any successful venture is based on having the right people. The right team with a mediocre idea is better than the wrong team with a good idea.

3. Agile Methods and ERP Systems

Agility is the ability to create and respond to change… agile organizations view change as an opportunity, not a threat [43].

3.1 Agile Method Background

In the 1980's the development of many large software applications was factory–centric. Large volumes of code were generated by equally large volumes of programmers [15]. The consequences of this horde approach have been well documented [24, 25]. As early as 1956 the concept of software process entered the lexicon [18]. The discussion of software process improvement has a long history, with varied results even to this day [14, 22, 65, 64].

In recent years, the landscape has changed dramatically for both the suppliers and consumers of software. Time to market pressures, rapidly changing requirements, the Internet, and powerful programming languages have placed new forces on traditional software development organizations [10]. These forces have been felt in the COTS integration domain as well [57, 78].

One source of modern process improvement was initiated by Royce [65]. From this, iterative methods improved on the original waterfall process. The mid–1980's produced several new processes including the spiral model of Boehm, which evolved from a risk management point of view [3]. Process programming emerged from formal modeling techniques in the late 80's [58, 59]. Software process improvements continue to occupy an important place in research as well as the commercial market place [3, 19].

The concept of agility has been discussed in detail in the hardware domain [41]. Similar research and discussion is just starting to take place in a manner for the software domain. This leaves a gap in the academic approach to the subject. This gap has been filled by anecdotal accounts of agile processes being applied in a variety of development domains, but an extensive survey of the taxonomy and processes have not been conducted [9, 10, 27, 28, 43].

3.2 Pre–Paradigm Issues with Agility

The gap in the agile process theory represents the normal evolution of any intellectual venture. The current agile processes could be considered to be in a pre–paradigm state [3]. This is a state in which the inconsistencies in the current paradigm (high–ceremony methods) are resisted until a new paradigm emerges [53]. Some questions are appropriate for these emerging agile methods:

- Can these methods be evaluated using the scientific principles found in the high–ceremony methods?
- Can the management of ERP systems acquisition and deployment be reduced to a set of scientific principles?
- How does the paradigm of agility compare with the more traditional methods described in §3.1?
- How are gaps in the current high–ceremony methods filled by agile methods?

3.4 Agile Project Management Principles

It is common to speak of agile methods in the context of the *lightweight* activities used to manage the development or acquisition of software. These activities include requirements, design, coding, documentation, and testing processes using a minimal set of activities and artifacts needed to reach the end goal – a working software system.

Applying the concept of *agility* to the management of a software project is a natural evolutionary step from high–ceremony processes. However, several questions need to be answered by the agile process before proceeding:

- How can these minimalist approaches be applied in a COTS integration environment while still maintaining the necessary integrity of the delivered product – cost control, functional capabilities, resource management, and timely delivery?
- Which project management process simplifications are *appropriate* for the ERP domain and which are not?
- Are all lightweight and agile project management process steps applicable to the ERP problem domain? If not, which steps are applicable [48]?

3.5 An Agile ERP Delivery Process

Agile methods emphasize rapid and flexible adaptation to changes in the process, product, business, and deployment environment [9, 10]. This is a generic definition of agile and not very useful without some specific context. Before establishing this context though, any agile process must include three major attributes. It must be:

---

[3] A paradigm is "… essentially a collection of beliefs shared by scientists, a set of agreements about how problems are to be understood." A pre–paradigm is characterized by an abundance of initiatives, the development of standards, and the increasing use of methods and structure [42, 60].

- *Incremental, Iterative, and Evolutionary* – allowing adaptation to both internal and external events.
- *Modular and Lean* – allowing components of the process to come and go depending on specific needs of the participants and stakeholders.
- *Time Based* – built on work cycles, which contain feedback loops, checkpoints, and guidance on using this information in the next cycle.

3.6 An Options Approach To Decision Making

In the 1980's Barry Boehm established a framework for an economics–oriented approach to software development focused on cost estimation [20, 21]. These concepts have been extended in many directions, including the economic tradeoffs made during COTS product deployment [35]. The selection, deployment, and operation of a complex software system is subject to a high degree of uncertainty. Reasons for this uncertainty are numerous: general macroeconomic influences, changing stakeholder requirements, and changing demands from customers and consumers for specific capabilities [38].

Classical financial analysis techniques, such as discounted cash flows (DCF), calculation of net present value (NPV), or internal rate of return (IRR), are not capable of dealing with this uncertainty.

DCF treats assets as passively held, not actively managed, as they would be in an ERP project. ERP projects have the flexibility to make changes to investments when new information is obtained. Treating this flexibility as an *option* allows decisions to be made in the presence of uncertainty. The fundamental advantage of this *real options* framework (versus financial options) over the traditional DCF legacy IT project framework is that the resulting valuations incorporate the value by making *smart* choices over time in the presence of changing information and risk assessments. [4]

Many of the choices in the selection and deployment of an ERP system are made without the theoretical or conceptual foundations described in the previous paragraphs [72].

An important distinction between software development decision–making and COTS decision–making is that COTS decisions are often irrevocable.

Individual software modules cannot be refactored or redacted since the source code is not available.

Performing the economic tasks above without some quantitative tools to guide the decision maker leads to poor choices at best and chaos at worst. Chasing the next optimization, gadget, or latest vendor recommendation has become all too common.

[4] There are other methods to aid in decision–making as well as options pricing – utility theory and dynamic discounted cash flow are examples. Each of the approaches makes assumptions as to the applicability, advantages, and disadvantages [75].

3.7 A Quick Options Tutorial

An options based decision process can be used in agile ERP deployment to great advantage [10, 35, 36, 45]. An option is a contract that confers its holder the right, *without obligation*, to acquire or dispose of a risky asset at a set price within a given period of time. The holder may exercise the option by buying or selling the underlying asset before its expiration date if the net payoff from the transaction is positive.

If the holder does not exercise the option by the expiration date, the option expires as worthless. The value of the option is the amount one would pay to buy the contract if it were traded in an open market. An option that gives the right to acquire an asset is a *call* option; an option that gives the right to dispose of an asset is a *put* option.

The value of an option is linked to its asymmetric nature – the holder has the right, but not the obligation, to exercise the option. The exercise takes place only if and when it is beneficial to do so [29, 30, 55].

3.8 Important Assumptions About Real Options

The use of an Options–Based decision process in software development has been popularized in the *eXtreme Programming* methodology [17]. For the options based decision process to be properly applied several conditions must exist:

- An option has value only if there is uncertainty in the outcome, resulting value, or impact on future decisions. In software defining the dimensions of this uncertainty is difficult [61].
- The decision process and the consequences of the decision must be irreversible.
- Irreversibility implies that the optionable asset is *scarce* and difficult to replicate in a timely manner. In the case of software decision processes, the scarce item is *knowledge* about the underlying technical and business processes in the form of *core competencies* [51]. Project success is related to the maturity of an organization, it capabilities in dealing with projects, uncertainty, and abilities to learn from the past [62, 68].

In the absence of these conditions, an options–based decision making process may have little to offer.

There are several theoretical difficulties as well with the options concepts presented in [17]:

- In software development the underlying asset is not actually traded.
- In other cases the asset exists only as a result of exercising the option and is not tradable independently from the decision process.
- In richly traded markets there is information about uncertainty and values. In the low volume world of IT projects, obtaining valid data about future values, treating this data consistently, and dealing with the unqualified effects of staff, business processes, and changing markets results in a very different valuation process. [5]

---

[5] This argument is presented in Strassmann's *The Squandered Computer* [73]. In this work he dismisses the use of real options in constructing values in incomplete markets. Such markets are where prices are not in the space of the market. Strassmann is correct in that the use of

Using the natural uncertainty of the ERP domain core competencies can be used to produce *complete market information*, to apply the options–based decision process to advantage [6, 34, 71].

3.9 Measuring Value in COTS Project Management

Much of the agile literature discusses *value creation*. Several questions arise in the context of options theory:

- In what dimensions and units is the value measured?
- How are the contingent future payoffs valued?
- What is the role of risk–aversion in valuing contingent payoffs?
- How can tradeoffs in multi–dimensional value spaces be evaluated?
- How can the value of an option be determined in the presence of uncertainty and incomplete knowledge?
- How can core competency be used as the source of the options value as discussed in §3.7?

These questions are just beginning to be addressed in the agile literature. Some answers can be found in utility theory and multi–objective decision–making [42]. Even in the absence of the answers to these questions, agile methods can be valuable in the management of ERP systems deployment, since asking the questions focuses the attention of all participants on the uncertainty and irreversibility of the decision process [70].

4. Applying Agile Methods to ERP

Agility implies a systematic vision of the outcome – an *intelligent* action or *ingenium* that makes it possible to connect separate entities and their outcomes in a rapid and suitable manner.

> *Ingenium: … the way which we build while going [77].*

Agile methods possess values and principles that can be considered heuristics that guide the process application using the mechanism of *Ingenium*.

---

arbitrage–based pricing techniques are not theoretically appropriate for IT projects. Gathering valid data is the source of the problem here. However when markets are incomplete, the data required for pricing can be obtained from experts in the problem domain. An estimate of the *likelihood of change* is produced in the normal course of software engineering management [23].

4.1 Agile Project Values

The set of underlying *values* for an agile project include: [6]

- *Communication* – of information within and outside an agile project is constant. These communication processes are essential *social* activities for the project participants.
- *Simplicity* – defines the approach of addressing the *critical success factors* of the project in terms of the simplest possible solution. See Fig. 3 for the ERP CSF's.
- *Feedback* – "optimism is an occupational hazard of software development, feedback is the cure" [17].
- *Courage* – important decisions and changes in the direction of the project must be made with courage. [7] This means having the courage *not* to engage in non–value added activities or artifacts.
- *Humility* – the best project managers acknowledge they don't know everything and must engage the stakeholders to close the gaps.

4.2 Applying Agile Principles

Using these agile values, the following principles create the foundation for managing ERP projects in an agile manner. [8]

- *Assume Simplicity* – as the project evolves it is assumed that the simplest solution is best. [9] Overbuilding the system or any artifact of the project must be avoided. The project manager should have the courage to not perform a task or produce an artifact that is not *needed for the immediate benefit of the stakeholders*.
- *Embrace Change* – since requirements evolve over time, the stakeholder's understanding of these requirements evolve as well. Project stakeholders themselves may change as the project makes progress. Project stakeholders may change their point of view, which in turn may change the goals and success criteria of the project. These changes are a natural part of an ERP project.
- *Enabling The Next Effort* – the project can still be considered a failure even when the team delivers a working system to the users. Part of fulfilling the needs of the stakeholders is to ensure the system is robust enough to be extended over time. Using Alistair Cockburn's concept, "when you are playing the software development game your secondary goal is to setup to play the next game" [27]. The next phase

---

[6] These values are taken from the agile Modeling and eXtreme Programming resources. They are not unique since they can be traced to the earliest project and program management sources. See [66] for a good review of adaptive and agile process in the aerospace business. But they do represent a cohesive set of values and principles articulated by the agile community.

[7] This term is used in [17]. I do not consider it the same courage found in soldiers, firefighters, and police officers.

[8] These principles are attributed to Scott Ambler and are adapted with permission to the ERP acquisition and deployment domain. [4, 5]

[9] This may not always be the case for ERP, but it is a good starting point.

may be the development of a major release of the system or it may simply be the operation and support of the current system.

- *Incremental Change* – the pressure to *get it right the first time* can overwhelm the project. Instead of futilely trying to develop an all–encompassing project develop a small portion of the system, or a high–level model of a larger portion of the system. Evolve this portion over time, and discard portions that are no longer needed in an incremental manner.
- *Maximize Stakeholder Value* – the project stakeholders are investing resources — time, money, facilities, and etc. — to create a system to meet their needs. Stakeholders expect their investment will be applied in the best way.
- *Manage With A Purpose* – by creating artifacts that have stakeholder value. Identify who needs the artifact. Identify a purpose for creating the artifact.
- *Multiple Project Views* – considering the complexity of any modern information technology system construction or acquisition process, there is need for a wide range of presentation formats in order to effectively communicate with the stakeholders, participants, and service providers.
- *Rapid Feedback* – the time between an action and the feedback on that action must be minimized. Work closely with the stakeholders, to understand the requirements, to analyze those requirements, and develop an *actionable* plan, which provides numerous opportunities for feedback.
- *Working Software Is The Primary Goal* – not the production of extraneous documentation, software, or management artifacts. Any activity that does not directly contribute to the goal of producing working software should be examined to determine its value.
- *Travel Light* – since every artifact must be maintained over its life cycle. The effort needed to maintain these artifacts must be balanced with their value.

These principles need a context in which to be applied. More importantly they need specific *actionable* outcomes within that context.

5. An Agile Application Example

Much of the agile literature provides recommendations and guidelines independent of a business domain. What is needed is a domain specific set of principles and practices that can serve as a *checklist* for getting started [44].

5.1 ERP Functional Domains

There are numerous ERP business domains and functions within those domains. Narrowing the domain from this long list will help focus the case study context. The business domains in which ERP plays a critical role includes:

Domain	Functions
Product Line Management	Program Management, Product Data Management, Quality Management, Asset Management
Supply Chain Management	Networking, Planning, Coordination, Execution
Customer Relationship Management	Customer Engagement, Business Transactions, Order Fulfillment, Customer Service
Financials	Financial Operations, Accounting, Corporate Services
Human Resources	Administration, Payroll, Organizational Management and Development, Time Management, Legal Reporting, Strategies
Procurement	Indirect Materials Procurement, Direct Materials Procurement, Electronic Tendering, Integrated Analytics

Fig. 1.

These functions are too broad for a useful example of agile deployment. One functional area that impacts many business processes is Product Data Management (PDM). PDM systems manage product entities from design engineering through release to manufacturing. In the ERP taxonomy in Fig. 1., PDM is a good starting point for an example.

5.2 PDM Domain Relationships with ERP

Engineering processes drive product development, so engineering tools are at the heart of the PDM systems interaction with the engineering user community. Engineering processes are good candidates for agile deployment, since the process improvement aspects of engineering processes can usually only be discovered by putting them into practice, experimenting with various tools and user interactions, and evolving these business processes to deal with unknown and possibly unknowably demands from the market place.

5.3 Agile PDM Domain Practices

Using the agile principles stated above describes specific actions that implement these principles for a PDM deployment within an ERP project.

Principle	Applied in the PDM Domain
Assume Simplicity	▪ COTS products define the requirements, more than the users do. Don't make changes in the system if it can be avoided. Start with the Out Of The Box system and discover gaps. Fill the gaps with other COTS products when ever possible ▪ Separation of concerns is a critical success factor for both products and processes. Base these separations on the business architecture of the system, and then apply the technical architecture. ▪ Decoupled work processes create architectural simplicity. Search for opportunities to decouple work processes along technical architecture boundaries. ▪ Stateless management of connections between application domains isolates components. ▪ Minimize product structure attribute creation early in the deployment cycle. ▪ Provide a means to add product attributes and relations to the object model later in the deployment cycle. ▪ Start with the simplest business process; verify the system can be deployed against these processes. Progress to more difficult processes but always search for the simplest solutions.
Embrace Change	▪ Object architectures enable change, but ruthlessly maintain proper object attributes. Modularity, information hiding, and other object attributes can pay large dividends over time. ▪ Model based thought processes focus requirements. ▪ Continuous delivery using selected products. Focus on vertical versus horizontal delivery (making the disk move on the first instance and every instance after that). ▪ Isolate components to provide a replaceable architecture.
Enable the next effort	▪ Architecture driven planning in depth is the primary role of the project manager and the architecture staff. ▪ Use a *Battle Planning* paradigm for the daily project activities – it's chaos at the low level and big picture strategy at the high level. ▪ Focus on values for today, while keeping the generation of value in the future in mind. ▪ Continuously evaluate the future opportunity costs.
Incremental Change	▪ Plan globally, implement locally, guided by architecture. ▪ Rapid planning in depth is not an oxymoron. COTS integration is an experience–based discipline. Skills are important, but are secondary since most decisions are irrevocable.
Maximize Value	▪ Put tools in the hands of the users. Discover what we have to do for the people who have to do the work. ▪ Provide these tools in a rapid, efficient, and beneficial manner, with the minimum of resources and disruptions to the ongoing operation. ▪ The stakeholders define the dimensions of value, ask them what they want, when they want it, and how much they're willing to pay.
Manage with a Purpose	▪ Architecture centered management places the proper boundaries on creativity. ▪ Always define the outcome of an action: who benefits? How can this benefit be recognized? What does this benefit cost? ▪ Never confuse effort with results.
Multiple Views	▪ Objects (static and dynamic) are nice but they don't show the business process. ▪ Data flow is nice but it doesn't show the underlying business object architecture. ▪ Control flow is useful for business process improvement, but be careful about redundant data and persisted entities. ▪ Event and data source and sink can be used for isolating business process boundaries. ▪ Business processes can be used to define the highest level boundaries. ▪ Interface exchange artifacts are critical for maintaining separation of concerns. ▪ Inversion of control – the identification and management of the interface control points is a critical success factor.

Rapid Feedback	▪ Continuous engagement with the stakeholders. ▪ War room mentality in which the participants are fighting the system not each other. ▪ Continuous delivery of functionality.
Working Software	▪ COTS products change this concept, but system integration efforts are just as difficult and important. ▪ Continuous delivery using standard products with the minimum of customization. ▪ Use the vendor's tools to get something working fast. ▪ Avoid customizing a COTS product if at all possible.
Travel Light	▪ Analyze and Model once, publish many. Use technology to reduce the *white space* in the process and organization. ▪ Move fast and light. Use experience based behaviors and high–level specifications to guide architecture. Low–level specifications add NO sustaining value in an ERP system. Working code is the value to the stakeholders. ▪ Working software is the final specification. Use specifications to capture knowledge that will be needed independent from the working software. This can be interface specification for 3rd parties, justifications for the decisions, and other *tribal knowledge* conveyance materials.

Fig. 2.

5.4 Agile ERP Heuristics

Agility is about being adaptive. Heuristics are a way of learning from the past and adapting to the future. There is a large body of heuristic–oriented guidelines for programming languages and other low level development activities.

The following are broadly applicable heuristics in the ERP integration problem domain [66]:

- Choose components so they can be implemented independently of the internal behavior of others. Ask the vendor: *Can I replace your product with another?*
- The number of defects remaining undiscovered after a test suite is proportional to the number of defects found during the test. The constant of proportionality depends on the thoroughness of the test, but is rarely less than 0.5 in the traditional *test– last* environment.
- Very low rates of delivered defects can be achieved only by very low rates of defect insertion throughout the development process. This is the primary contribution of development processes like Extreme Programming [17].
- The system must be grown not built. The use of evolutionary development and deployment are critical indicators of an agile organization. Without this the process is not agile.
- The cost of removing a defect from the system grows exponentially with the number of cycles, since the defect was inserted. Constant and complete test processes are an indicator of agile organizations [22].
- Personnel skills dominate all other factors in productivity and quality [49].
- The cost of fixing a defect does *not* rise with time. It may be cheaper to discover a requirements defect in final use testing than in any other way. Continuous releases are critical in agile organizations. Put the system in the hands of the stakeholder as often as possible.

- Architecture–based processes provide a touchstone when things go wrong – and they will go wrong. The project manager should always ask the question *how does this proposed change fit into the architecture, change the architecture, or affect the architecture in some way*?

5.5 The Critical Success Factors For Agile ERP

The question of what specific agile processes are to be applied in the ERP domain can be addressed by focusing on the Critical Success Factors related to ERP [63]. One such list includes [70]:

Top Management Support	Project Champions
Management of Expectations	Vendor/Customer Relationships
Use of Vendor's Tools	Careful package selection
Project Management	Steering Committee
Use of Consultants	Minimal Customization
Business process reengineering	Defining the Architecture
Dedicated resources	Project Team Competence
Change Management	Clear Goals and Objectives
Education on Processes	Interdepartmental Communication
Interdepartmental Cooperation	Ongoing Vendor Support

Fig. 3.

Applying agile principles and practices in support of these CSF's will guide the project manager to the agile methods of addressing the complexities of ERP deployment.

6. Call to Action

It is both interesting and significant that the first six out of sixteen technology factors associated with software disasters are specific failures in the domains of project management, and three of the other technology deficiencies can be indirectly assigned to poor project management practices [50].

The management of an ERP deployment involves requirements gathering, vendor selection, product acquisition, system integration and software development, and finally system deployment and operation. It involves risk management, stakeholder politics, financial support, and other intangible roles and activities that impact project success. By applying the values and principles of agile methods along with risk management, clearly defined and articulated critical success factors, architecture driven design, people management, the agile team can deliver real value to the stakeholders in the presence of uncertainty while maximizing the return on assets and minimizing risk.

References

The following resources have been used as assemble the compendium of ideas presented in this paper. Since many of the ideas presented here are not mine, I give full acknowledgement to the original source and authors of the materials presented here.

1. Alexander, Christopher, *A Timeless Way of Building*, Oxford University Press, 1979.
2. Alexander, Christopher, *Notes on the Synthesis of Form*, Harvard University Press, 1964.
3. Agresti, *New Paradigms for Software Development*, IEEE Computer Society Press, 1986.
4. Ambler, Scott, www.agilemodeling.com.
5. Ambler, Scott, *Process Patterns* and *More Process Patterns*, Cambridge University Press, 1998 and 1999.
6. Amram, Martha, and John Henderson, "Managing Business Risk by IT Investment: The Real Options View," *CIO Magazine*, March 1999.
7. Amram, Martha, Nalin Kulatilaka, and John Henderson, "Taking an Option on IT," *CIO Magazine*, June 15, 1999.
8. Amram, Martha and Nalin Kulatilaka, *Real Options: Managing Strategic Investments in a World of Uncertainty*, Harvard Business School Press, 1999.
9. Aoyama, Mikio, "New Age of Software Development: How Component Based Software Engineering Changes the Way of Software Development," *1998 International Workshop on Competent–Based Software Engineering*, 1998.
10. Aoyama, Mikio, "Agile Software Process and Its Experience," *International Conference on Software Engineering*, 1998.
11. "APICS Survey," *CPIM Journal*, **46**, 17 July 2000.
12. Arthur, W. Brian, "Increasing Returns and the New World of Business," *Harvard Business Review*, 74(4), pp. 100–110, July–August 1996.
13. Austin, Robert D. and Richard L. Nolan, "How to Manage ERP Initiatives," Working Paper 99–024, 1998.
14. Baker, F. T. "Chief Programmer Team Management of Production Programming," *IBM Systems Journal*, 11(1), 1972.
15. Basili, Victor, "Iterative Enhancement: A Practical Technique for Software Improvement," *IEEE Transactions on Software Engineering*, 1(4), December 1975.
16. Bateman, T. S. and C. P. Zeithaml, *Management: Funciton and Strategy*, Irwin, 1990.
17. Beck, Kent, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 1999.
18. Bennington, "Production of Large Computer Programs," *Symposium on Advanced Computer Programs for Digital Computers*, sponsored by Office of Naval Research, June 1956. Reprinted in *Annals of the History of Computing*, October 1983, pp. 350–361. Reprinted at *ICSE '87*, Monterey, California, March 30–April 7, 1987.
19. Boehm, Barry, "Get Ready for Agile Methods with Care," *IEEE Computer*, pp. 64–69, January 2002.

20. Boehm, Barry, and Kevin Sullivan "Software Economics: A Roadmap," in *The Future of Software Engineering*, special volume, A. Finkelstein, Ed., *22nd International Conference on Software Engineering*, June, 2000.
21. Boehm, Barry, "Software Engineering Economics," *IEEE Transactions on Software Engineering*, 10, January 1984.
22. Boehm, Barry, "Anchoring the Software Process," *IEEE Software*, pp. 73–82, July 1996.
23. Boehm, Barry, et. al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.
24. Brooks, Fred, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Software*, 20(4), pp. 10–19, April 1987.
25. Brooks, Fred, *The Mythical Man–Month*, Addison Wesley, 1995.
26. Charette, Robert N., "Large–Scale Project Management is Risk Management," *IEEE Software*, pp. 110–117, July 1996.
27. Cockburn, Alistair, http://crystalmethodologies.org/
28. Cockburn, Alistair, http://members.aol.com/humansandt/crystal/clear/
29. Copeland, Thomas E. and Philip T. Keenan, "How Much is Flexibility Worth?" *The McKinsey Quarterly*, 2, 1998.
30. Copeland, Thomas E. and Philip T. Keenan, "Making Real Options Real," *The McKinsey Quarterly*, 3, 1998.
31. Courtney, Hugh, "Making the Most of Uncertainty," *The McKinsey Quarterly*, 4, pp. 38–47, 2001.
32. Davis, Alan M., "Fifteen Principles of Software Engineering," *IEEE Software*, 11(6), pp. 94–96, November/December, 1994.
33. Earl, Michael, Jeffery Sampler, and James Short, "Strategies for Reengineering: Different ways of Initiating and Implementing Business Process Change," Centre for Research in Information Management, London Business School, 1995.
34. Earl, Michael, "Information Systems Strategy: Why Planning Techniques are Never the Answer," Centre for Research in Information Management, London Business School, 1995.
35. Erdogmus, H., "Valuation Of Complex Options In Software Development," *First Workshop on Economics–Driven Software Engineering Research*, *EDSER–1*, May 17, 1999.
36. Flatto, Jerry, "The Role of Real Options in Valuing Information Technology Projects," *Association of Information Systems Conference*, 1996.
37. Funtowicz, S. and J. Ravetz, "Post–Normal Science: A New Science for New Times," *Scientific European*, pp. 95–97, March 1992.
38. Gattiker, Thomas and Dale Goodhue, "Understanding the Plant Level Costs and Benefits of ERP: Will the Ugly Duckling Always Turn Into a Swan?" *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.
39. Georgescu–Roegen, Nicholas, *The Entropy Laws and Economic Progress*, Harvard University, 1971.
40. Glass, Robert, "Failure is Looking More Like Success These Days," *IEEE Software*, January / February 2002.
41. Goldman, Steven. L., Roger N. Nagel and Kenneth Preiss, *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*, Jossey–Bass, 1994.

42. Hammond, J. S., R. L. Keeney, and H. Raiffa, *Smart Choices: A Practical Guide to Making Better Decisions*, Harvard Business School Press, 1999.
43. Highsmith, Jim, *Adaptive Software Development*, Dorset House, 1999.
44. Hoffman, Hubert and Franz Lehner, "Requirements Engineering as a Success Factor in Software Projects, *IEEE Software*, pp. 58–66, July / August 2001.
45. Holland, Christopher and Ben Light, "A Critical Success Factors Model for ERP Implementation," *IEEE Software*, pp. 30–36, May / June 1999.
46. Huber, Thomas, Rainer Alt, and Huber Österle, "Templates – Instruments for Standardizing ERP Systems," *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.
47. Jeffries, Ron, *Extreme Programming Installed*, Addison Wesley, 2000.
48. Jones, Capers, Software Assessments, Benchmarks, and Best Practices, Addison Wesley, 2000.
49. Jones, Capers, "What it Means to be Best in Class," Version 5, February 10, 1998.
50. Jones, Capers, *Patterns of Software Systems Failures and Success*, International Thompson Computer Press, 1996.
51. Kogut, Bruce and Nalin Kulatilaka, "Strategy, Heuristics, and Real Options," *The Oxford Handbook of Strategy (2001)*, Chapter 30, 2001.
52. Kogut, Bruce and Nalin Kulatilaka, "What is Critical Capability?" Reginald H. Jones Center Working Paper, Wharton School, 1992.
53. Kuhn, T. S., *Structure of Scientific Revolutions*, Chicago University Press, 1962.
54. Kulik, P. and R. Samuelsen, "e–Project Management for the New Reality," *PM Network Online*, PMI Management Institute, 11 April 2001.
55. Leslie, Keith J. and Max P. Michaels, "The Real Power of Real Options," *McKinsey Quarterly*, 3, pp. 97–108, 1997.
56. Morris, C. and C. Ferguson, "How Architecture Wins Technology Wars," *Harvard Business Review*, pp. 86–96, March–April 1993.
57. Oberndorf, Patricia and David Carney, *A Summary of DoD COTS–Related Policies*, SEI Monographs on the Use of Commercial Software in Government Systems, Software Engineering Institute, Carnegie Mellon University, 1998
58. Osterweil, Leon J., "Software Processes are Software Too," *Proceedings of the 9th International Conference on Software Engineering* (ICSE 1987), pp. 2–13, March 1987, Monterey, CA.
59. Osterweil, Leon J., "Software Processes Are Software Too, Revisited," *Proceedings of the 19th International Conference on Software Engineering* (ICSE 1997), pp. 540–548, May 1997, Boston, MA.
60. Pajares, Frank, "The Structure of Scientific Revolutions," Outline and Study Guide, Emory University, http://www.emory.edu/EDUCATION/mfp/Kuhn.html.
61. Potters, Marc, et. al. "Financial Markets as Adaptive Ecosystems," May 31, 2001. arXiv:cond–mat/9609172.
62. Remy, Ron, "Adding Focus to Improvement Efforts with PM3," *PM Network*, July 1997.
63. Rockart, J. F. and C. V. Bullen, "A Primer on Critical Success Factors," Center for Information Systems Research, Working Paper No. 69, Sloan School of Management, MIT, 1981.
64. Royce, Walker, *Software Project Management*, Addison Wesley, 1998.

65. Royce, Winston W., "Managing the Development of Large Scale Software Systems," *Proceedings of IEEE WESCON*, pp. 1–9, August 1970.
66. Rechtin, *System Architecture: 2nd Edition*, CRC Press, 2000.
67. Ross, Jeanne, "Surprising Facts About Implementing ERP," *IT Pro*, pp. 65–68, July / August 1999.
68. Saures, Isabelle, "A Real World Look at Achieving Project Management Maturity," *Project Management Institute 29th Annual Seminars/Symposium*, October 9–15, 1998.
69. Shaw, Mary and D. Garlan, *Software Architecture*, Prentice Hall, 1996.
70. Sommers, Toni and Klara Nelson, "The Impact of Critical Success Factors across the Stages of Enterprise Resource Planning Implementations," *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2000.
71. Sullivan, Kevin. P., William G. Griswold, Yaunfang Cai, and Ben Hallen, " The Structure and Value of Modularity in Software Design," *Proceedings of the Joint International Conference on Software Engineering*, September 2001
72. Sullivan, Kevin, P. Chalasani, S. Jha, and V. Sazawal, "Software Design as an Investment Activity: A Real Options Perspective," in *Real Options and Business Strategy: Applications to Decision–Making*, edited by Lenos Trigeorgis, Rick Books, 1999.
73. Strassman, P. A., *The Squandered Computer*, The Information Economics Press, 1997.
74. Szulanski, Gabriel, "Unpacking Stickiness: An Empirical Investigation of the Barriers to Transfer Best Practices Inside the Firm," INSEAD Study, *Academy of Management Best Paper Proceedings*, pp. 437–441, November 1995.
75. Teisberg, E. O., "Methods for Evaluating Capital Investment Decisions under Uncertainty," in *Real Options in Capital Investment: Models, Strategies, and Applications*, edited by L. Trigeorgies, Praeger, 1995.
76. Thorburn, W. M. "The Myth of Occam's Razor," *Mind* 27:345–353, 1918.
77. Vico, Giambattisa, (1668–1744) "Method of the Studies of Our Times," Naples, Italy, 1708.
78. Wallnau, Kurt, Scott Hissam, and Robert Seacord, *Building Systems from Commercial Components*, SEI Series in Software Engineering, Addison Wesley, 2002.

Glen Alleman is the Chief Technology Officer of Niwot Ridge Consulting specializing in enterprise application integration, system architecture, business process improvement, and project management applied to manufacturing, electric utility, petrochemical, aerospace, process control, publishing, and pharmaceutical industries.