Event-Driven Productivity Infrastructure

Hugh Taylor
- Lecturer, UC Berkeley School of Information
- Senior Marketing Manager, Microsoft

hughta@sims.berkeley.edu
www.hughtaylorite.com
(206) 384-1241

Abstract

The world of information technology is currently experiencing the parallel emergence of two separate paradigms, Event-Driven Architecture (EDA) and Productivity Infrastructure (PI). Each of these constructs has unique portent for the ways in which people interact with data and applications. However, there is also an exciting potential for the two constructs to work together in an integrated, synergistic fashion.

EDA is an approach to Service-Oriented Architecture (SOA) that creates an "enterprise nervous system," aware of changes in state that occur within applications, databases, as well as the outside world by publishing state information through XML to a message backbone, such as an Enterprise Service Bus (ESB). Event listeners, also connected to the ESB, distribute state change data to Service-oriented applications for processing and reaction, including human interactions. The EDA approach to enterprise architecture offers advantages in agility and segregation of concerns that benefit the utility of information systems.

Productivity infrastructure (PI) is an umbrella term to describe people's and organization's increasingly connected and synergistic use of phone, email, Internet, PDA, PDA, intranet, extranet, and desktop productivity applications. What was once a collection of essentially siloed productivity technologies and work flows – phone calls, emails, searching the Web, creating documents, using a PDA, and so on – are now merging into a combined infrastructure that drives personal and organization productivity. In brand name terms, productivity infrastructure is integrating the functionality of product sets such as Microsoft's Office System, Cisco's VOIP solutions, IBM's Lotus suite, and Google's Docs and gMail services, just to name a few.

This paper explores the potential integration of EDA with productivity infrastructure. Events that originate from applications in the EDA can be published to individual human users, or collaborative groups operating in the productivity infrastructure. The integration has the potential to connect real time operations of an enterprise with Web 2.0 technologies such as blogs and wikis, as well as a range of mobile computing technologies, in addition to standard portal interfaces. The paper focuses on the ways that productivity infrastructure empowers the human thinking and decision making that is often implicit in the process flow of an EDA. It looks at scenarios where EDA-PI integration can improve the speed, quality, and cost effectiveness of end users in a large enterprise environment. Further, the paper will examine the potential for real time human activities to become events themselves, which can flow in the other direction and drive application functioning in the EDA.

Table of Contents

Introduction

Jerry Seinfeld does a routine where he wonders how mankind discovered that glue can be made from horses. He describes a scene where someone is trying unsuccessfully to stick two pieces of paper together, when, suddenly a horse rides by, causing an instant revelation of potential… "Hey, wait a minute!" he imagines the brilliant inventor saying. "How come I never thought of that before?" So it is, too, in IT. Sometimes, trends of technological innovation exist in parallel for a period time before someone realizes that they can be put to work together, for a greater effect than either one on its own. There are some great examples of this from the history of technology, including the merging of recording technology and the telephone (creating the answering machine), the joining of the QWTRY keyboard with the cathode ray tube and the CPU (the modern computer), or the phone with the computer (networked computing).

Today, we are witnessing the parallel maturing of Event-Driven Architecture (EDA), typically a subset of Service-Oriented Architecture (SOA) and productivity infrastructure, two separate, but potentially synergetic information technologies. Each is powerful in its own right, but together, they can create transcendent event driven information processing environments. This paper will explore the potential integration of EDA with productivity infrastructure. In particular, we will focus on the ways that productivity infrastructure empowers the human thinking and decision making that is often implicit in the process flow of an EDA.

To illustrate the potential for the integration of productivity infrastructure and EDA, we will use a hypothetical example of a business that connects its purchasing process and related transactional systems with its productivity infrastructure, including email, VOIP, blogs, and instant messaging. I will use the case study of a hypothetical business to understand the EDA-PI connection because the current state of the technology does not reflect its future potential. At the conclusion of this paper, I will take a quick look at several of the emerging products that are on track to fulfill the potential of EDA-PI integration.

Background of Research

The subject of connecting back-end transactional systems with messaging infrastructure is not new. There are numerous examples of comparable structures, such as Interactive Voice Response (IVR) or SNMP alerts forwarded from system management consoles to pagers, and so forth. The concept of linking instant messaging, presence, and email with XML messages used in business transactions arises frequently in standards literature, commercial software development papers, and academic discussions of messaging and Service-Oriented Architecture. For example, there is the SOAP MailTo Command in SOA.[1] Due to the inherent commercial promise of EDA-PI integration, and the commercial basis for much of the installed base of infrastructure, much of the dialogue about EDA-PI integration is occurring in the commercial sphere. For example, IBM and Nortel announced in 2007 a plan to offer an integration between SOA and VOIP.[2] Or, the new "Oslo" SOA offerings from Microsoft contain a foundation for linking business process modeling, Web services, and email messaging. This paper tries to go one step further and explore a vision of EDA-PI integration that has not yet been achieved in the commercial world, though one which may become a reality in the near future.

[1] http://cpan.uwinnipeg.ca/htdocs/SOAP-Lite/MAILTO.pm.html
[2] Nortel and IBM Use SOA to Streamline Communications Among Customers, Employees, and Partners

EDA: A Working Systemic Definition

Event-Driven Architecture is an approach to enterprise software architecture that is characterized by the ability of the enterprise to detect events and react to them. Brenda Michelson, a technology analyst, writes, "In an event-driven architecture, a notable thing happens inside our outside your business, which disseminates immediately to all interested parties (human or automated). The interested parties evaluate the event, and optionally take action."[3] To keep the discussion simple, I am going to make the following assumption: Event-Driven Architecture in the modern sense is based on the set of XML messaging standards and inter-operability technologies collectively known as Service-Oriented Architecture (SOA). These technologies include SOAP Web Services, UDDI, WSDL, Enterprise Service Bus (ESB), and industry/commercial solutions such as Java Connector Architecture (JCA), Windows Communication Foundation (WCF), Java Message Service (JMS), and so forth. Though there are many different, equally valid approaches to the realization of EDA, I will focus on the SOA version because it now the de fact mode of implementation of the EDA paradigm.

Defining an "Event"

First, let's define what we mean by an "event." In life, an event is something that happens: a car drives by, a ball flies through your window, someone falls asleep – an action occurs. Alternatively, for our purposes, an event can also be an expected action that does not occur. If the temperature does not go down at night, that could be an "event." In systemic terms, an event generally refers to a change in "state". A change in state typically means that a data value has changed.

For example, if you make a credit card charge, there are several values that change in state as the transaction is processed. The value of your account balance will change as you make the charge, as will the state of your card from "quiet" to "transaction occurring," and so on. The shift from "balance = $500" to "balance = $550" is an event.

An event has three levels of detail. At one level, there is the basic fact that an event has occurred. An event has either occurred or not. This takes us to the second level of detail, which is the event definition. In order to recognize an event, an EDA must have a definition of what the event is. In the credit card example, the EDA must work with a definition of an event that says, in effect, a "change in balance" event has occurred whenever the balance value goes up. What if the balance goes down? Is that also an event? It could be, but the definition would need to take that into account. Thirdly, there is the detail of the specific event. How much has the card balance changed? In our example, the two-level event information structure would say, "So-and-so's Balance has changed. Amount = $50." All three factors – event notification, event definition, and event detail, are necessary when designing an EDA.

For an event to occur in an EDA, it must be in digital form, or a form that computer can understand. However, that does not mean that an EDA is exclusively the preserve of digital information. A change in state can also result from non digital information being translated into digital form. For example, a digital thermometer typically has some kind of analogue temperature sensor which inputs temperature information into the sensor and results in a digital value equated to temperature. The edges of an EDA may be full of analog information that is translated into digital data in order to trigger events.

[3] Michelson, Brenda – Elemental Links 2007

A key learning point here is to understand that virtually anything can be an event or trigger an event. Rainfall in Chad? Could be an event, if it is quantified and made available as a source of data. Stock market activity in Tokyo? Absolutely an event, assuming you know why you are interested. And on and on. An ideal EDA can be easily adapted to recognize events that occur anywhere. The trickiest word in the last sentence was "easily," a simple idea that can generate a lot of discussion and complexity.

EDA Overview

Armed with a sense of what an "event" can be, we can now add some flesh to the basic EDA definition articulated by Brenda Michelson. If an event is any "notable thing" that happens inside our outside our businesses, then an EDA is the complete array of architectural elements, including design, planning, technology, organization, and so on, which enables the ability to "disseminate" the event immediately to all interested parties, human or automated. The EDA also provides the basis for interested parties to "evaluate the event, and optionally take action."

The reason that EDA is a challenging concept is that it is so potentially broad. Just as almost any piece of data, analog or digital, can be an event, and any system in the universe can potentially be part of your EDA, where do you begin to draw the boundaries and definition of an EDA that makes sense to your organization? Though there is no bullet proof way to answer the question, I think that it makes sense to identify the main ingredients of an EDA, and build the definition from these constructs.

Event Producers or Publishers

In order to have an EDA, we must first have events… That may seem obvious, but a lot of otherwise sophisticated discussions of EDA either neglect or muddle up this central enabling concept. The EDA cannot work unless it has the ability to perceive that an event has occurred. For that to happen, the event must be created, and then published, for the components of the EDA called listeners to "hear" them.

The technologies that do this are known as "Event Publishers" or "Event Producers." With the broad definition of an "event," event publishers can take many different forms. Most are software programs, though an event publisher can also be a dedicated piece of hardware that translates analog data into digital form and feeds it into software that can detect an event. Here are some core ideas to keep in mind about event publishers:

Event publishers can be anywhere. Because events can occur outside of your enterprise, event publishers that relate to your EDA can be pretty much any place. Imagine the relationship between an airline EDA and the FAA radar tower. The radar tower, which serves many purposes, one of which is to be an event publisher, is completely separate from the airline's systems, yet it is part of the EDA.

Event publishers may or may not originate the data that is contained in the event itself. In their purest form, an event publisher generates a piece of data that is formatted to be "heard" as an event in accordance with the EDA's setup for this process. For example, a credit card processing system should automatically generate data that is EDA ready – it contains the card holder's identifiers, the time of the transaction, the amount, the merchant name, and so on. Of course, the data was created for the purpose of charging the card, not feeding the EDA, but it serves that purpose quite readily. In contrast, other event publishers need to translate data into a format that constitutes an event according to what the EDA requires. For example, there is no inherent event pattern in the Tokyo stock exchange index unless you specifically instruct an event publisher to transmit data about the index in a manner that makes sense to your EDA's purposes.

Event Listeners, or Consumers

Like event publishers, event listeners can be anywhere. In theory, the event listener (or consumer) has a communication link with the event publishers. That is not always the case, but we will work under that assumption for now. The event listener is a piece of technology – typically software based, but also hardware – that "knows" how to differentiate an event, as it is defined, from other data it receives.

In the simplest form of EDA, the event listener can only receive the specific event data that it is meant to hear. For example, an EDA for building security may be based on a burglar alarm whose "Event Listeners" can only hear one kind of event - the kind created by window break, detecting "event producers." There is just one kind of event that can occur: a break in. The real world, of course is more complex, and as we progress, we will get into more involved EDA setups.

Event listeners also need to know what they are "listening for"… An application that reads the data stream of the Tokyo stock market average is not an "event listener" until it has been instructed to "listen" for some specific type of event. For instance, the event listener must know that a gain of 5% or more in the average is an "event." A 4% gain is not an event. The event listener must be able to detect the event, and be capable of learning what the event is.

Event Processors

After an event has occurred, and been "published," and "consumed" or "listened to" – it must be processed. It does little good to have an event that is perceived, but not handled. An EDA without Event Processors brings to mind the great joke about economists. A man gets lost in a hot air balloon. After drifting for hours, he finally sees an economist standing on the ground. The man yells out, "Where am I?" To which the economist replies, "You're up in the air." Unlike the economist, who may know the facts but be unable to make any sense of them, an EDA should have the capacity to interpret the events it hears.

An event processor is invariably a piece of software. While it may or may not be part of some larger, more comprehensive suite of applications, an event processor is distinctive because it has the ability to assess events, determine their importance, and generate a reaction of some kind, even if the reaction is "do nothing."

Event Reactions

Following our chain of activities, we have an event, which is published by an event publisher, heard by an event listener, and processed by an event processor. Then, something (or nothing) needs to happen. Because "something happening" is inherently more interesting than "nothing happening," let's look at event reactions that require action.

Reactions to an event in an EDA vary widely, from automated application responses, to automated notifications sent either to applications or people, to purely human reactions based on business processes that occur outside of the EDA itself. In the purely automated application response category, we might see an EDA that reactions to an event by initiating an application level process. For example, in the credit card fraud example, the EDA might modify a variable value from "Normal" to "Warning" based on an event that suggested that fraud were occurring. If this reaction were coded into the event reaction, it will happen without any human involvement. Following from this, another related branch of processes might handle new charge requests on the account differently based on a “Warning” value than from a “Normal” state. Event processors and reactions can be linked and interdependent.

The event reaction might be machine-to-human. Continuing with our example, imagine that there is a customer service representative who sees all the new “Warning” value changes, and is prompted to call the cardholder to inquire about the status of their card. This approach to EDA is dependent on the human reaction to an event, a situation that may be good or bad, depending on the desired outcome. For example, many intrusion detection systems that monitor networks for unauthorized access attempts generate a great deal of false positives. The system itself, which is essentially a tightly defined EDA, will only be as good as the person who monitors it. Indeed, there are some intrusion detections that are not monitored at all. What these defectively implemented systems do is generate logs consisting of thousands of possible intrusion records, which are essentially useless. The takeaway here is that the human reaction is very much part of the EDA design, even though it does not involve technology per se.

Finally, there are EDAs where the event reaction is based on an entirely human set of processes, following the detection and processing of the original event. For example, an EDA might generate a number of alerts a number of stock market indicators. An investor then reviews the alerts and decides whether to buy or sell. Once again, the EDA design must take into account the human reaction as part of its eventual success or failure. And, with this type of complex human decision-making, it can be quite challenging to determine what the “right” reaction should be. Experts may differ on how to react to identical sets of event alerts. As they say on Wall Street, these differences of opinion make for a good horse race…

Messaging Backbone

The fifth and final core component of an EDA is the messaging backbone, the communication infrastructure – inclusive of hardware, software, network protocols, and message formats – that enables each piece of the EDA to communicate with one another. In order to serve an EDA effectively, a messaging backbone must have several characteristics. An EDA messaging backbone needs to be as near to universal as possible, meaning that it should enable messaging across multiple network transport protocols and data formats. In other words, it should be standards based, or have the ability to mediate across multiple messaging standards. It should be pervasive, i.e. far-reaching and universally accessible. In reality, this means that it is based on the Internet, though it need not be. It should be inexpensive to develop, maintain, and modify – a lot easier said than done, but this is a critical. Cost is the “invisible hand” that has killed many great EDA initiatives. Finally, it should enable a high level of de-coupling between event producers and event consumers. In reality, this usually translates into a “publish/subscribe” or “pub/sub” set up.

The messaging backbone is arguably the most essential piece of the EDA puzzle, for without it, there can be no EDA. Without the ability to communicate, the event listeners, producers, processors, or reaction processes, cannot work. Now, you might be thinking, yes of course they can – one can always create communication interfaces between systems. Of course, this has been true for many years. The reality, though, is that proprietary interfaces, which have been the traditional way to achieve connections, are costly to develop, maintain, and modify. So costly, in fact, that they have rendered the concept of a dynamic EDA virtually impossible to realize. As I have noted earlier, I will work with the assumption that the messaging backbone of the EDA will be an Enterprise Service Bus (ESB) or equivalent.

Assembling the Paradigmatic EDA

To get to the paradigmatic EDA – the one I will use as the reference point for the rest of this paper – involves connecting event producers, event listeners, event processors, and event reactions using a common messaging backbone. Figure 1 shows what this looks like in the plainest terms. Obviously, things can get wildly more complex in real life, but I thought it would be best to start with a very stripped down paradigm example to start.

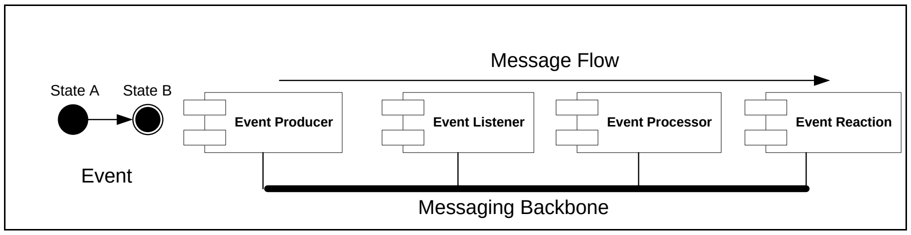


Figure 1
The paradigmatic EDA – consisting of event producer, event listener, event processor, and event reaction, all connected through a common messaging backbone

Like an "enterprise nervous system,", the EDA works as a whole, taking in signal inputs in the form of event data, processing it, and reacting to it based on some kind of intelligent model. Like the nerve endings that tell us when we are hot or cold, being touched, or falling through the air, event listeners in the EDA quantify information from the world and order it into a form that the EDA can understand. The messaging backbone of the EDA is like the nerve cells themselves, transmitting signals back and forth from various places in the body and toward the brain. The brain is like the EDA's event processing components. It takes in event data and decides how to respond to it. The event reaction components are like our limbs. Based on the input, we take action, or not.

The Flavors of EDA

There's an old joke about Las Vegas, where a guy wants to be seated at a restaurant and is asked, "Which section would you prefer, smoking or chain smoking?" EDA is the same. Almost every system has some degree of event orientation in it. After all, computing is essentially a matter of input, processing, and output, much like an EDA. Following from this, discussions of what an EDA actually looks like can be confusing because they tend to get overly broad and encompass, well, just about everything the IT universe. To keep our focus, let's look at several prime models of EDA that are in use today.

According to Manas Deb, there are two essential types of EDA: Explicit and Implicit.[4] In an explicit EDA, the event publishers send event data to known event listeners, perhaps even by direct hard coding of the event listener destination right into the event producer. Most current implementations of EDA are either partly or wholly explicit. As such they tend to rely on tight coupling of event producers and event listeners.

In contrast, an implicit EDA does not specify any dedicated connections between event producers and event listeners. The event listeners, or the event processors even, may determine the events to which they want to listen. The coupling between the event producers and listeners will be loose, or even completely de-coupled in an implicit EDA. As you might imagine, an implicit EDA is more flexible and dynamic than an explicit EDA. Historically, they have proved

[4] Deb, Manas EDA and Event Processing Essentials – Visual Studio Magazine 2006

too complex to implement. This is beginning to change with the advent and adoption of open standards.

Within EDAs themselves, there are three basic patterns: Simple Event Processing, Event Stream Processing, and Complex Event Processing, which is known as CEP. In simple event processing, the EDA is quite narrow and simple. The event producer's function is to generate event data and send it to the event listener, which in turn processes it in whatever manner is required. The thermostat example is emblematic of a simple event processing design. The furnace gets the signal and switches on or off, and that's about it.

Event stream processing involves event processors receiving a number of signals from event producers (via the event listeners) but only reacting when certain criteria are met. For example, the thermometer may send the temperature data every two seconds, but the event processor (thermostat) ignores all but the relevant "switch on" data point, which activates the furnace only when it is observed.

Complex event processing (CEP) takes the EDA to another level, but enabling it to react to multiple events under multiple logical conditions. So, for the sake of argument, let's say that we only want to buy US Treasury bonds when the Nikkei hits a certain number, and unemployment figures dip below 5%. The CEP listens to multiple event streams, and knows how to correlate them in a logical manner according to the objectives of the EDA. In other words, the CEP starts to "think" like a person, taking in data from unrelated sources on the fly, differentiating between useful and useless information, and acting accordingly. This is the ideal of the complex, implicit, dynamic EDA.

The Often Inadequate Human Link in the EDA

There are many instances where the corporate "nervous system" of EDA loops through a human decision making process. In the airline traffic case, it was the flight operations managers who were called upon to make critical decisions about flight prioritization based on input from the EDA. In the anti-money laundering case, bank fraud staff were fed information about suspicious transactions for their review and decision on actions. This EDA-human connection makes sense much of the time. Indeed, there is often no substitute for a person, and his or her awareness of multiple influencing factors, in a business decision making process. Artificial intelligence is suitable in some cases to make or support human decisions, but even in cases where decision making can be automated, there is frequently the need for a person to take responsibility for the decision. Alas, there is still no way to hold a computer accountable for the consequences of a decision that causes an airliner to crash or money to be stolen from a bank account. Given the inevitable presence of people in EDA-based decision process flows, one of the big challenges is relative inefficiency of human decision making.

Unlike computers, people are extraordinarily inefficient at decision making and are, in fact, quite high maintenance. While a computer can execute a decision algorithm at any time within a fraction of a second, people need to be present (awake and focused) to make a specific decision at a specific time. This is not efficient, and in some cases, may be harmful to the business process that the EDA is meant to serve. And, in many cases, people need to make decisions in groups, a situation where the inefficiency of communication compounds the delay and quality of the decision. For example, if the air traffic EDA enables rapid decisions about prioritizing flight departure times, but the key decision maker needs to consult with a superior, who is out to lunch, the whole process could be delayed to the point where it compromises the whole intent of the system.

In other cases, people may need to access external sources of information in order to make decisions that feed into an EDA decision making process. The bank anti-fraud staffer may have to review scanned document images to compare signatures on old checks before making a judgment call about whether to escalate a fraud investigation or notify law enforcement. This type of manual, or semi-manual process, can cause harmful inefficiency for the EDA. In this instance, the anti-fraud staffer might have to manually write down the name of the suspected account, exit the EDA interface, open a records management application and conduct a search for matching documents. After manually reviewing the documents, he or she may have to share the findings with other anti-fraud staffers or document review specialists before making a decision to escalate the case. Such a discussion could involve a combination of email, phone, fax, or instant messaging. None of this is particularly horrible, but the cumulative effect of faults in the communication and manual process flows – spread out across multiple concurrent fraud cases – can result in a significant drag on performance and sub-optimal anti-fraud efforts. The solution to these types of challenges is known broadly as "productivity infrastructure," and it is maturing today at a rapid rate.

Overview of Productivity Infrastructure

Productivity infrastructure (PI) is an umbrella term to describe people's and organization's increasingly connected and synergistic use of phone, email, Internet, PDA, intranet, extranet, and desktop productivity applications. What was once a collection of essentially siloed productivity technologies and work flows – phone calls, emails, searching the Web, creating documents, using a PDA, and so on – are now merging into a combined infrastructure that drives personal and organization productivity. In brand name terms, productivity infrastructure is integrating the functionality of product sets such as Microsoft's Office System, Cisco's VOIP solutions, IBM's Lotus suite, and Google's Docs and gMail services, just to name a few.

To understand the importance and impact of productivity infrastructure, let's use the creation of a sales proposal as a baseline example of the kind of unstructured type of workflow that typically challenges information workers to be productive. In contrast to structured tasks, such as those performed by customer service agents at a call center, a great deal of business work today involves unstructured tasks, which are unpredictable in terms of work flow step order, location of needed information and stakeholder identities, roles, and responsibilities. In the case of creating a sales proposal, there may be a number of different approval patterns, issues to be resolved, and decision makers involved at any given time. The processes, people, and underlying data and documents required to create the sales proposal may change from case to case. Though the process will contain the basic flow shown in figure 2, in reality each situation will be slightly different. Managing this subjectivity within a tight time frame is the essence of productivity infrastructure.

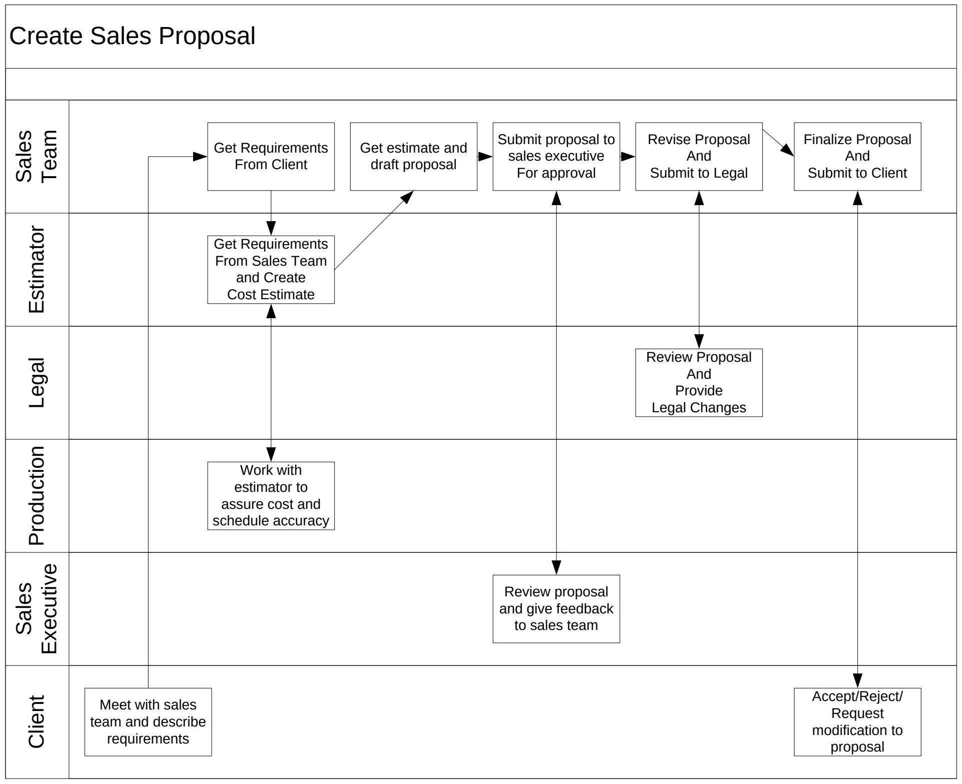

Create Sales Proposal
Sales Team
Estimator
Legal
Production
Sales Executive
Client
Get Requirements From Client
Get estimate and draft proposal
Submit proposal to sales executive For approval
Revise Proposal And Submit to Legal
Finalize Proposal And Submit to Client
Get Requirements From Sales Team and Create Cost Estimate
Review Proposal And Provide Legal Changes
Work with estimator to assure cost and schedule accuracy
Review proposal and give feedback to sales team
Meet with sales team and describe requirements
Accept/Reject/ Request modification to proposal

Figure 2
In the process flow of creating a sales proposal, multiple people and groups must collaborate and share documents and information, often in real time.

Each step in the creation of the sales proposal, as shown in Figure 3, involves multiple people, often from different work teams, in the sharing of information, documents, and knowledge. The more efficiently the people involved in completing this multi-step, multi-player process can work get their work done, the better off the organization will be in productivity terms. The impetus behind the development of productivity infrastructure is the drive to enable workers in unstructured information work to get more done in less time, with less expenditure of resources.

As Figure 3 shows, the steps in the sales proposal development process map to capabilities in the productivity infrastructure. In this example, VOIP technology speeds up the process of connecting the customer with the sales rep by automatically connecting a landline call to a mobile device. All participants in the process use email to communicate, with relevant stakeholders able to share links to document repositories where proposal templates and other data are stored for common use. A portal UI links stakeholders to the workflow management process as well as collaboration sites that contain blogs and wikis that publish up to date information needed for crafting the proposal. Social computing technologies allows individuals to understand connections between stakeholders that may not be apparent through job titles or task assignments. Team sites enable stakeholders to make sure that their proposal is in alignment with team objectives and business goals, or clarify approvals required for the proposal. The capability to see presence enables real time instant messaging or web conferencing that facilitates rapid resolution of open issues for the proposal.

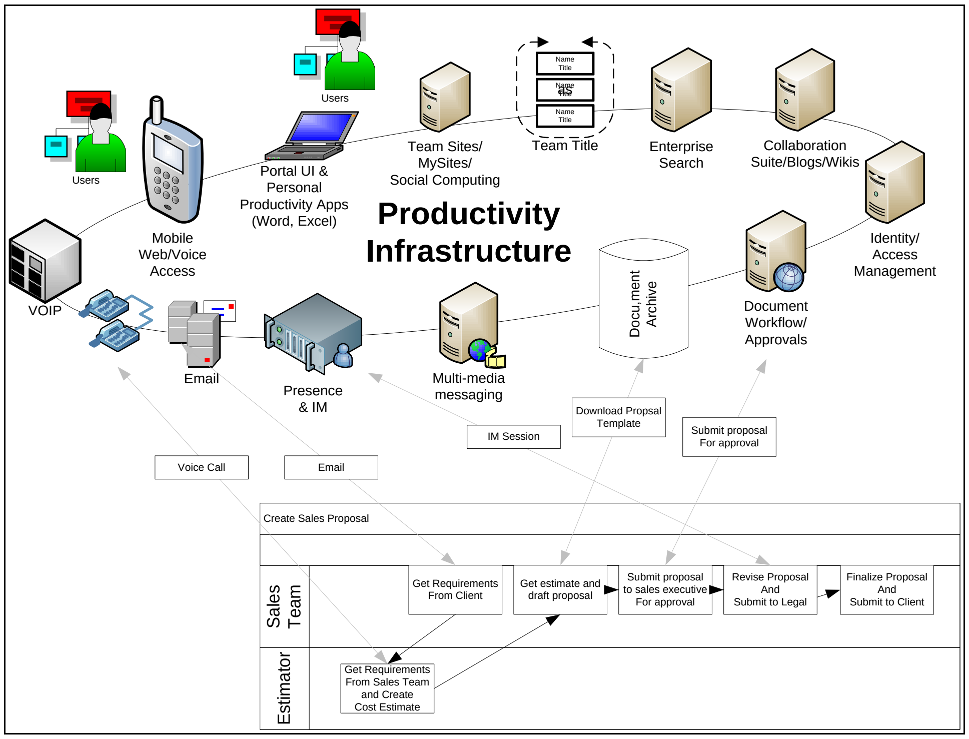

Users
Users
Mobile Web/Voice Access
Portal UI & Personal Productivity Apps (Word, Excel)
Team Sites/ MySites/ Social Computing
Team Title
Enterprise Search
Collaboration Suite/Blogs/Wikis
Identity/ Access Management
Productivity Infrastructure
VOIP
Email
Presence & IM
Multi-media messaging
Docu,ment Archive
Document Workflow/ Approvals
Download Propsal Template
Submit proposal For approval
IM Session
Voice Call
Email
Create Sales Proposal
Sales Team
Estimator
Get Requirements From Client
Get estimate and draft proposal
Submit proposal to sales executive For approval
Revise Proposal And Submit to Legal
Finalize Proposal And Submit to Client
Get Requirements From Sales Team and Create Cost Estimate

Figure 3
Productivity infrastructure, which links workflow, collaboration, email, phone, mobile, document repositories, and Web 2.0 technologies such as blogs, wikis, mysites, and social computing, can drive efficiencies in the unstructured, multi-player, iterative process flow required to create a sales proposal.

If everyone involved in creating the sales proposal were sitting in the same room at all times, then there would be no reason for the investment in productivity infrastructure. In fact, that's how business worked until about 1890, when the telephone started to move people away from physical proximity to their business partners. Today, of course, the groups of people involved in getting business processes accomplished are almost never all together, and certainly not in any reliable pattern or schedule that will allow them to get time sensitive tasks done. And today, as we often find, it's not just being able to communicate with others in a collaborative process that makes things flow smoothly. To get the tasks accomplished, each participant needs to know who the other players are, who they report to, what their priorities are. On top of all that, participants also often need access to information that is not easily located without the assistance of others in the group. To make access to needed information available without requiring time consuming conversations or emails, participants need to be able to search for and find what they are looking for on their own.

Productivity infrastructure has the potential to drive more efficient work flows for unstructured tasks, assuming it is implemented properly. The time cycle for completing the entire process flow for creating a sales proposal, for example, becomes shorter with good productivity infrastructure, and the time investment of each participant goes down as well. Ideally, the accuracy and quality of the final product improves as an added bonus. However, productivity infrastructure can be complex to deploy, as it can raise some odd security and compliance concerns. I mention this here just to assure you that I am not all starry-eyed about the ease of deploying such a comprehensive and interconnected infrastructure, and neither should you. Nevertheless, I do believe that productivity infrastructure, once in place, has portent for EDA.

The Potential Benefits of EDA-PI Integration

To see the potential for benefits of integrating EDA with PI, we should think about the advantages of linking the corporate “nervous system” of EDA with the comparable organizational nervous system of PI. With EDA, corporate systems can detect changes in state that affect business. Wherever the EDA needs human input, PI can speed up the EDA’s reaction time to the state change. PI can also improve the quality of the human input, because it can link people with data sources, and each other, with high efficiency. Ultimately, there is the potential for the creation of loops of interaction between EDA and PI, where state changes noticed by the EDA elicit reactions from people though the PI, who in turn input their own changes of state to the EDA.

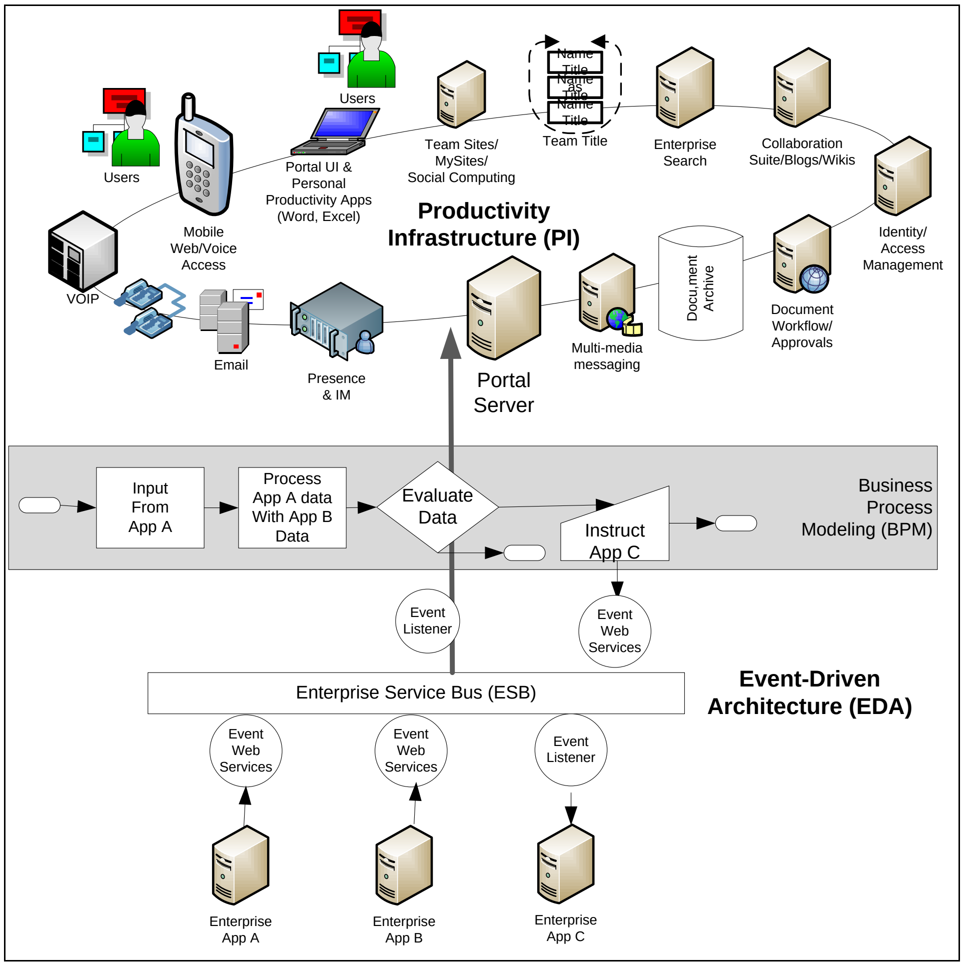

Users
Users
Team Title
Team Sites/ MySites/ Social Computing
Enterprise Search
Collaboration Suite/Blogs/Wikis
Portal UI & Personal Productivity Apps (Word, Excel)
Mobile Web/Voice Access
Productivity Infrastructure (PI)
Identity/ Access Management
VOIP
Docu,ment Archive
Document Workflow/ Approvals
Email
Presence & IM
Multi-media messaging
Portal Server
Input From App A
Process App A data With App B Data
Evaluate Data
Instruct App C
Business Process Modeling (BPM)
Event Listener
Event Web Services
Enterprise Service Bus (ESB)
Event-Driven Architecture (EDA)
Event Web Services
Event Web Services
Event Listener
Enterprise App A
Enterprise App B
Enterprise App C

Figure 4
The integration of productivity infrastructure and EDA, connecting people and enterprise apps through an ESB and event Web services.

As shown in Figure 4, the integration of productivity infrastructure and EDA can be understood by considering a simple business process model that involves inputs from two enterprise systems that must be evaluated by people. Event Web services on applications A and B publish data about their state to the ESB, and on to an application built using a Business Process Modeling (BPM) tool. The process model calls for people to assess the data presented by the states of applications and B, and for them to make a decision about what the states mean, and then take action either by instructing application C or terminating the process without taking action. The PI is designed to notify the decision makers of the change in state. Once the decision is made, the reaction to the change in state flows back to the EDA through an event Web service located in the PI.

If there were just one person who could make the decision called for in the process, off the top of his or her head, there wouldn't be much need for the kind of elaborate setup called for in Figure 4. However, let's suppose that the decision being made in the process flow is complex, high risk, multi-stakeholder, and time sensitive. Imagine, for example, that it involves the decision to manufacture goods with costly inventory. The decision could have impact on financial statements, factory capacity, even labor unions. In that kind of situation, a tight integration between the decision makers and the EDA could have a real impact on the business.

In the case of a manufacturer setting inventory levels, the time required to make a decision – the right decision, that is – is highly relevant to business success. If the manufacturer guesses wrong, and either overstocks an item whose product life is on the decline, or under stocks a hot seller, the financial results will be less than optimal. In these kind of situations, even the difference of an hour or two, or the lack of a few critical nuggets of business data, can have an impact on the bottom line. Imagine, for instance, if you decided to order a truckload of component parts for the manufacture of a product that was later determined to be unneeded. It might not be the end of the world, but it would create an accounting and logistical hassle to return the order. Multiply this type of problem across a large, global company, and the effect on earnings could reach into millions of dollars of direct and indirect costs. Consider, for example, the necessity of engaging accounting staff unnecessarily due to faulty decision making. This ???

In addition to offering a shorter decision cycle time, the integrated EDA-PI approach has the potential to enable a higher quality of decision than the current state of integration between PI and enterprise systems. Keeping in mind with this example that we are dealing here with decisions that cannot be automated through rules engines, consider the factors that affect the quality of decision making amongst multiple stakeholders. In our view, the quality of a decision depends on the financial consequences of the decision. The decision that saves or makes the most money is the best one. Of course, there is a range of quality decisions between best and worst, but the goal should be to strive for the best decision in the largest number of cases. This concept is known as the "decision yield".[5]

As anyone who has worked in a large, distributed organization could tell you, the quality of a decision depends on multiple interdependent factors, including knowledge of who the stakeholders are for a particular decision, equal simultaneous access to information, and equal understanding of information. Quality of decision making also depends on a productive engagement of stakeholders inside an organizational hierarchy. The higher level stakeholder may have the ability to overrule the correct decision through innate power, and the smarter subordinate may not have, or want, the opportunity to oppose the incorrect decision. Of course, productivity infrastructure cannot help an organization overcome this hierarchical flaw in process on its own. However, by providing open access to shared opinions and corporate knowledge, and real time access to multiple points of view, the hierarchy effect can be mitigated in favor of discussion and group learning.

Integrating EDA and PI can improve the efficiency of the information workers who are tasked with making business decisions. Ultimately, this can result in reductions in overhead or increased utilization of staff for strategic business purposes. I felt this point was worth making because I have heard many dialogues about the value of SOA and EDA that make the assumption that there is a high efficiency analysis apparatus available to parse the output of these systems.

[5] Taylor, James and Raden, Neil – *Smart Enough Systems* – Prentice Hall 2007

This is not necessarily so, and indeed, a lot of approaches to SOA dead end into an empty seat called "stakeholders" and fail to generate good ROI as a result.

The Productivity Infrastructure itself can function as an event producer as well. A simple example might be the presence of a stakeholder being announced as an event. Another example, which touches on an exciting new area of PI, is the concept of "active search" within the enterprise and its potential to function as an event producer from within the PI.

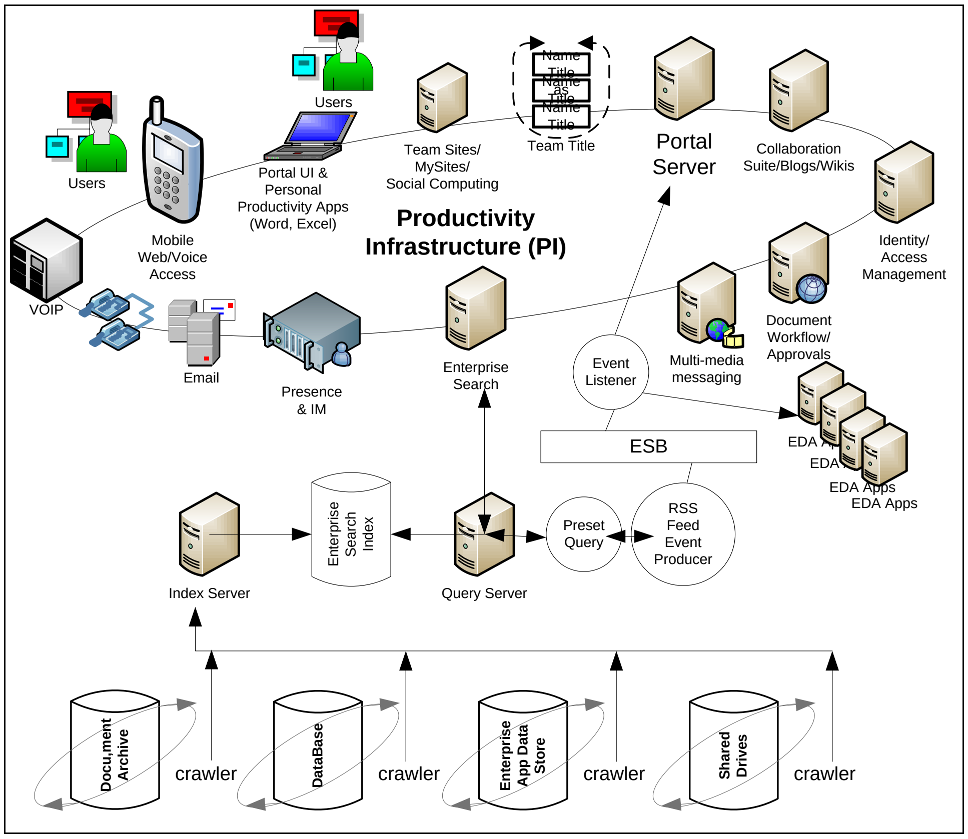


Figure 5
Active search involves the use of an enterprise search solution, which "crawls" repositories of unstructured data, indexes them, and then sends RSS feeds in response to preset search queries. The RSS feeds, which function to indicate the presence of specific information in unstructured data, can serve as event producers.

Figure 5 depicts an enterprise search solution allowing a productivity infrastructure to function as an event producer. To see how this works, we must first understand the process of an enterprise search solution, which is an increasingly common fixture in today's enterprises. Like a Web search engine, the enterprise search solution contains three core elements: a query server, an index server, and "crawlers," which read through documents and other data sources and feed their findings into the index server. The enterprise search index, like their corollaries in the Web search world, is a massive and exhaustive directory of information located within the enterprise. The enterprise search solution operates by taking queries from end users through a front end UI (eg a search box in a portal interface), sending the queries through the query server to the index and returning matching results back to the end user through the UI.

Some enterprise search solutions offer the ability to conduct active search, a process wherein certain queries are stored and continually re-run, with the search results being published to end users through RSS (Real Simple Syndication, a form of XML). So, for example, an end user in a real estate development firm could use an enterprise search solution to query the company's internal document libraries for a data that matches the keyword of a particular neighborhood. If the user types "Chelsea" into the search box on the portal UI, that query will return any documents or other indexed data files that contain the word "Chelsea," and the user could then learn about projects or people involving Chelsea. In an active search mode, the user could store the "Chelsea" search and instruct the search solution to issue him an RSS feed every day that contained the latest search results for "Chelsea" without the user having to run the query every day himself.

From an EDA perspective, it is possible to imagine how this active search function could turn into an event producer. The stored query, and resulting RSS feed, could be designed to publish changes in state that exist within unstructured data environments. In our real estate example, the query might feed into an algorithm that determines whether specific people or company divisions are working in Chelsea, a change in state that could drive action through the EDA. A CRM system attached to the EDA could flag the activity in Chelsea, based on the event data published through the enterprise search solution and PI, for follow up by account executives in the neighborhood.

There is even the potential for EDA and PI to inter operate as looping, connected halves of a bigger EDA. Events published out through the EDA inform stakeholders and drive action through the PI, which in turn publishes back event data about stakeholder activities, presence, and data creation through the PI. Admittedly, this level of sophistication is fairly futuristic, even for this forward looking paper. However, I believe the potential for productivity improvements and information worker empowerment through the integration of EDA and PI are powerful and promising.

ProdCo, An EDA-PI Integration Scenario

To see how EDA and PI could be integrated, I will use the example of a custom manufacturing business. In order to optimize the learning experience, w are going to keep the example fairly simple and focus on the aspects of this hypothetical business that are most general to all businesses. This company, which I'll call ProdCo, could stand in for a mass of businesses that perform custom services on a job-by-job basis. Within ProdCo, I will focus on the sales proposal and order fulfillment process to highlight the potential EDA-PI integration.

Sales Proposal and Order Fulfillment at ProdCo

The sales and fulfillment process at ProdCo is touched by a group of teams and individual stakeholders. To get a proposal out to a client and then fulfill on the order, the Sales, Marketing, Production, Legal, Finance, and C-Suite executives need to be involved. Sales, of course, is the main point of contact with the client. Marketing engages in the process to assist with pricing and discount programs that may be tied to particular campaigns. Finance is involved to assure that the pricing and costing of the job are appropriate, and that the HR aspects of the job are properly considered. Finance also weighs in with sourcing decisions and execution. Production is responsible for actually doing the work. The C-suite has oversight, especially if the job is large or strategically significant.

Figure 6 shows the essential flow for ProdCo's proposal-fulfillment-assessment process. The sales team drafts the proposal and circulates it through reviews by legal, production, finance, and the C-suite. If there are revisions, the review cycle may repeat in whole or part. Once the client approves the proposal, the job goes into production, and is fulfilled. Finance becomes involved again for sourcing of materials and overseeing labor expenses, as well as invoicing and collection. At the end of the process flow, the finance department publishes the results of the job – if it made or lost money compared to the estimate – and all other departments receive this information and update their own knowledge bases. Or not…

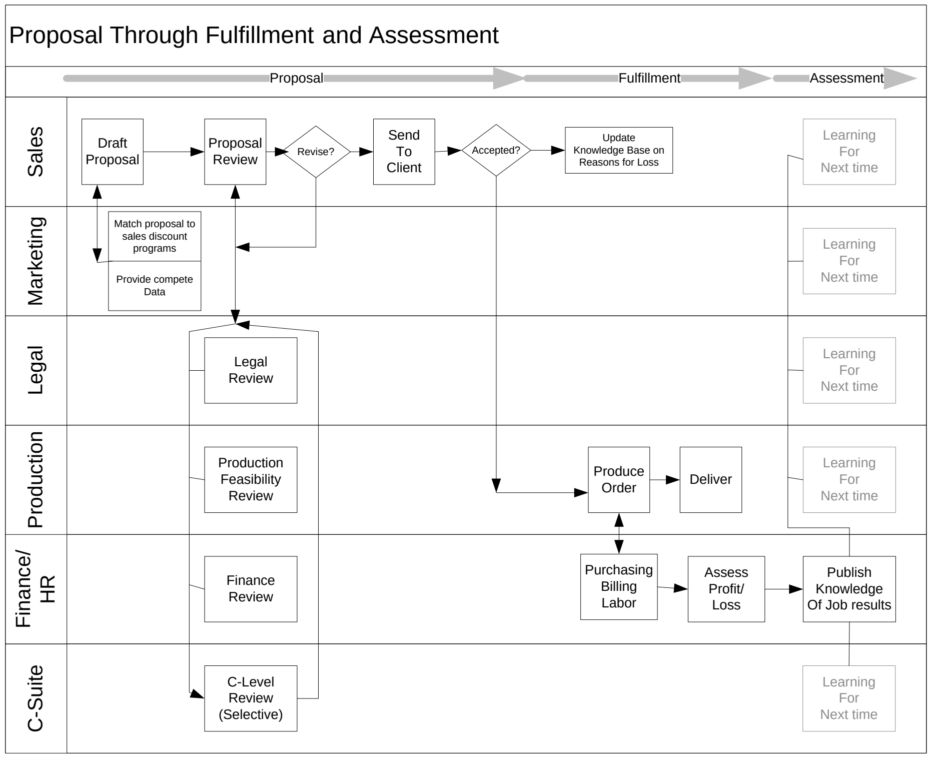


Figure 6
ProdCo's process flow for creating a sales proposal, fulfilling the job, and assessing its profitability.

There are a couple of flaws in this process design, though to be fair to ProdCo, it's about as good as it can be given the current state of technology. For one thing, the review loops for the proposal may be a lot more complicated and subjective than any process model can approximate. And, the assumption that the process flow makes is that everyone has access to relevant information on a timely basis. The marketing department may not know, for example, that certain types of projects lose money, so they ought to be dropped from the discount plan, and so forth. Most problematic, though, is that the process is very inefficient. The people and groups involved in this process waste time managing and finding the information they need to get the process finished as well as communicating with one another. A closer look at the way ProdCo has set up its productivity software and enterprise architecture can reveal some of the causes of this inefficiency.

Prod Co's Current Productivity Tools

Figure 7 shows how ProdCo has set up its productivity tools and enterprise applications. The sales and marketing teams have access to the CRM system, while finance and production use the ERP solution. Collaboration inside each team, and between teams, is a fairly ad-hoc affair, with stakeholders emailing files back and forth and saving them on departmental shared drives that each team can access through a portal interface. It is possible for a non team-member to access a portal, but that person must first be granted access rights by a departmental administrator. Each team portal has calendaring capabilities, and everyone is able to schedule meeting the email suite. The c-suite executives use a business intelligence dashboard that is fed by the finance staff because it does not tie directly into the ERP system.

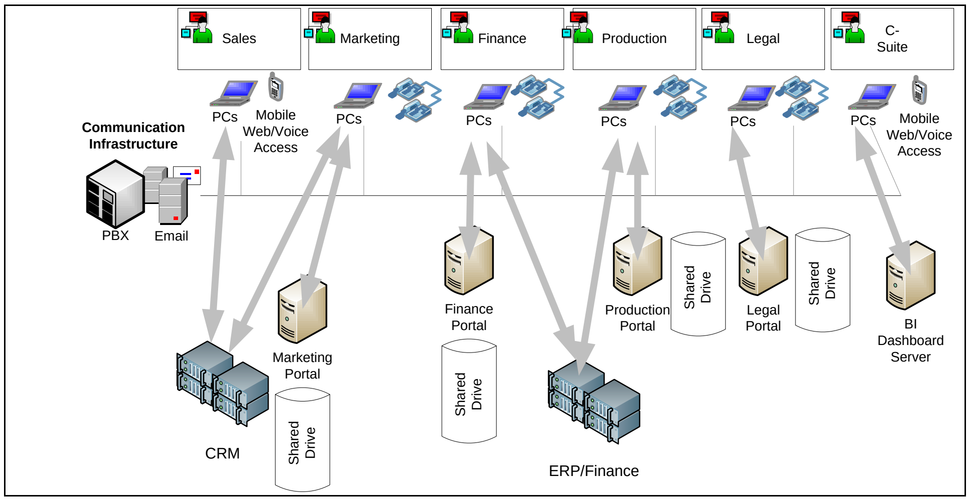


Figure 7
The ProdCo teams involved in creating sales proposals and fulfilling orders use two enterprise systems – CRM and ERP – and a slew of ad-hoc productivity solutions, including share drives, email, and voice communication.

ProdCo's productivity solutions and enterprise applications are too siloed to be highly efficient. There is no efficient way to share information or documents across the entire company – at least in a way that does not open access to the document to every single employee, which nobody wants. Connections between systems are haphazard or non-existent. So, for example, there is no way to seamlessly import the terms of a sales proposal into the ERP system. It must be re-entered when the order goes into production. Approvals on each phase of a proposal, and its subsequent production phase, are conducted by email and phone. The ERP system does have automated approval functions, where executives can sign off on purchase orders, overtime schedules, and so forth. However, the problem is that these approvers must often communicate with others, such as the legal department, before proceeding. The efficiency of the automated approval function is mitigated by the slow, unstructured process of communication within the firm. And, there is no way to keep track of recurring patterns of unstructured workflow that could save time in the future.

For example, imagine that certain types of orders require materials to be sourced from Mexico. The procurement staff has learned from experience that in the summer months, the heat inside the trucks coming from Mexico is so intense that it can ruin the parts in transit. They know now either to order these parts in advance, or actually pay for a refrigerated truck. Of course, this is more expensive, though paying the expense is preferable to delays and missing parts. However, the procurement staff has no way to keep this relevant fact in front of all stakeholders at all times.

The ERP system has a "notes" section, where the procurement staff can write down a reminder to order a refrigerated truck with that SKU. Yet, when the sales team wants to make a deal, or the marketing department wants to create a discount campaign, they do so without realizing that their margin is lower than normal on the item that includes the Mexican components. The c-suite, too, may lack visibility into the issue, and wonder why margins are low on this type of product.

ProdCo's Proposed EDA

Being wise and forward-looking, ProdCo has decided to invest in shifting its architecture towards SOA and EDA. Figure 8 shows how this would work. ProdCo would install an Enterprise Service Bus as an integration layer that exposes the functionality of both the CRM and ERP systems as sets of Web Services.   A portal server would provide access to the various department portals and also make certain CRM and ERP functions available to users who didn't have access to either the CRM or ERP client. The departmental portals themselves would remain essentially untouched, though their provisioning could now be centrally controlled and extension of existing portals could be governed more thoroughly than before. Communications would remain a silo.

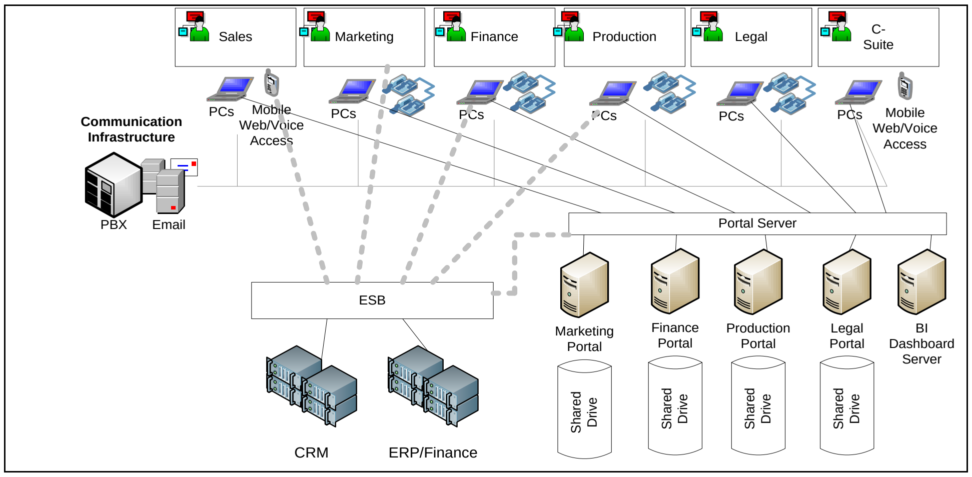


Figure 8
ProdCo is considering a move towards an SOA and EDA, where the CRM and ERP/Finance applications would be exposed as Web Services through the Enterprise Service Bus. The departmental portals would be accessed through a central portal server, though their content and administration would remain essentially siloed. The communication infrastructure would remain unchanged.

To appreciate how the EDA approach would work for ProdCo, let's look a small part of the production process, the ordering of supplies based on a bill of materials. Figure 9 shows the flow for the bill of materials (BOM) process as well as the matchup between the process steps and the Web services enabled in the EDA. Certain steps, though, such as selecting the winning bids from the RFQ, are still human processes and cannot be fully automated. Other steps, such as requesting procurement, may be semi-automated, wherein the ERP system does all the work once a person has approved the request for RFQ.

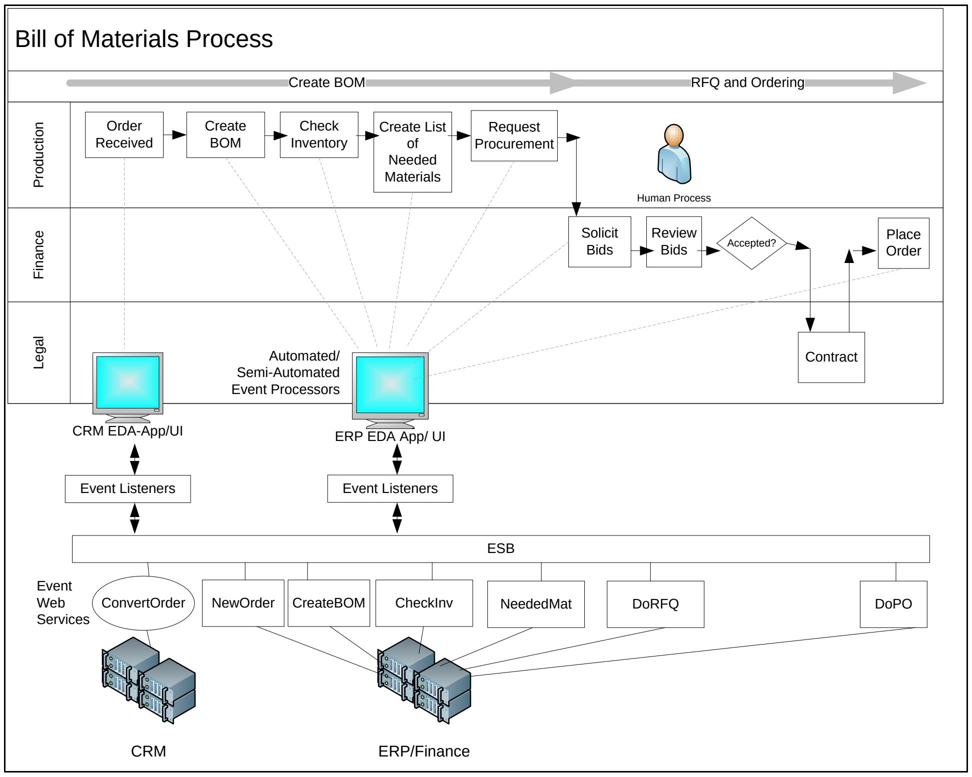


Figure 9
ProdCo's EDA exemplified through the process flow of creating a bill of materials and requesting quotes (RFQs) from suppliers.

As shown in Figure 9, the bill of materials process consists of several automated/semi-automated steps that work off the event Web Services, as well a few that are completely human. Sales proposals exist in the CRM system, but when a customer makes the decision to buy, under the new EDA approach, the CRM system's ConvertOrder Web service publishes the state change of the proposal from "Pending" to "Accepted." The event listener in the ERP EDA application receives this state change information and, in reaction, takes the order data from the transfers it into the ERP system. This step could be fully automated, meaning it could occur simply because of programmatic instruction, or it could be semi-automated, where a person gets a signal through the UI that the order is ready to convert and the human action of clicking a button on the interface actually completes the transfer from CRM to ERP systems.

The process flow then follows a series of pub/sub steps. The ConvertOrder web service in the CRM system results in the creation of a new order for production in the ERP system. The NewOrder Web services publishes that its state is "New," which is listened to by the CreateBOM (Create Bill of Materials) Web Service. The Web Services CheckInv (check inventory) and NeededMat (Needed Materials) create a list of materials that are needed for the job but which are not in stock. The DoRFQ Web Service is activated by the NeededMat Web Service's publishing of its change of state (from Nothing to List). Each of these Web Services can be invoked manually or through a automated process.

Sending the RFQs out to vendors, though, is a manual step even if the actual work is done by the system. A procurement person should be active the process to make sure that the RFQ goes to suppliers who are appropriate for the materials needed. However, even this could be a fully automated process that reacts to the event of "DoRFQ" by generating a Request For Quote electronically and sending it to pre-approved vendors through email or online notification calling them to a vendor portal.

Assuming the EDA is implemented correctly, it should be able to confer some advantages to ProdCo's operations. It renders the back end more flexible than a conventionally architected system. The EDA makes it simpler, faster, and cheaper to implement changes in the process flow for RFQs, and enables streamlined reporting and aggregation of data for consolidation of purchase orders and vendor management. The standards-based interfaces to the end user are also potentially more cost effective to maintain and modify as changes take place in the process flow over time.

However, the human decision elements of the process flow are not much affected by the EDA. The act of soliciting bids from vendors is still either wholly or partly human, as is the selection of winning bids. It would be possible to automate both of those processes, though, and even include some fairly sophisticated rules sets to assure best practices in procurement. Taking the human beings out of the picture for selecting vendors, for example, might occur if ProdCo could implement a set of business rules that awarded contracts to vendors with the lowest price, the best ranking for quality from production operations, and a consistent track record of on time delivery. If this automation were implemented, the bill of materials process could proceed seamlessly without the messiness and delay of human actions. This might be a desirable goal, and it might even be possible. Yet, if we are realistic and look at our own experiences in corporate life, we will realize that this level of automation is not practical or wise.

Returning to our example of goods from Mexico that melt in hot trucks in the Arizona sun, (and I'm not making this up, either. It happened to a colleague of mine) we can see that there are still instances where there is a need for subjective human knowledge to get to the best possible business result from a process flow, even with an advanced EDA in place. And, this is where the efficiency of ProdCo's EDA starts to falter.

As the automated, or semi-automated process of converting an order from proposal to sale, and the derivative bill of materials/RFQ steps cascade out from the entry of the sale into the ERP system, we still dead end at a person – or group – that needs to decide which vendors get to bid on the order. In the best case scenario, the vendor selection is done by an experienced person who understands all the subjective issues involved and acts promptly and decisively. In a less rewarding scenario, the decision is made by a distributed group of people who may individually lack the knowledge of the subjective challenges to getting an optimal procurement accomplished. There are many scenarios in between, such as an inexperienced procurement person who cannot process the vendor selection quickly due to lack of information, or one who makes the wrong decisions based on lack of knowledge, or even a lack of awareness that certain types of knowledge are required to make the decision. In this situation, we see the promotion of the person who knows that one must request a refrigerated truck from Mexico, and his or her replacement being a person who doesn't have any idea that such a problem exists. The new person will proceed, with the best of intentions, to repeat a mistake that has long since been solved.

There are several aspects to this poor quality decision cycle. If the communication process that connects the people in the decision loop is detached from the EDA and procurement interfaces, then the communication itself risks being inefficient and inaccurate. For example, imagine that the vendor selection process is dependent on people reading long email threads from the bottom up, assessing the situation, and making recommendations. (Surely, we've all been there…) This communication pattern is less than ideal for rendering a consistent, rapid set of correct decisions. However, due to the subjective nature of most unstructured processes, such as selecting a vendor using group knowledge as the basis for the decision, email threading is probably unavoidable. At the very least, it is clumsy and unreliable. Even if everyone involved is paying attention and very well informed, the process could bog down if one or more participants is unreachable (or un-findable) – to the extent that production orders could back up due to communication breakdowns in the procurement process.

ProdCo's EDA also lacks the capacity to store organizational knowledge. The specialized knowledge about the subtleties of procurement is not stored in a fashion where stakeholders can easily find or use the information. One enterprising person might create a procurement best practices document that he or she can use, and perhaps even share with others. But, if that document is lost on a shared network drive, its contents may never reach other stakeholders who need it. If the creator of the document moves on, the document will likely disappear. A dedicated and well-managed procurement team could also create an intranet site on the portal server for collection of practices and settling of decision issues. This approach has some benefits over a totally uncoordinated procurement method, but it might still result in time lags and communication mistakes if the stakeholders still need to toggle back and forth between their intranet team site, email, and the actual ERP system to make and implement decisions. At the very least, it is a largely unrepeatable pattern.

The integration of ProdCo's Productivity Infrastructure with its EDA is a major contributor to solving this dilemma, where lost knowledge, lack of knowledge, and poor communication mitigates the positive impact of an EDA and improvement business process management. PI integration cannot solve this problem all by itself, of course. There are myriad challenges related to training, knowledge preservation, best practices documentation, and so forth, that are required to assure good human decision cycles in a process flow. However, the chances of ProdCo attaining the best human decisions in the EDA environment are greatly enhanced by the integration of the EDA and the PI. Without this integration, the likelihood that ProdCo will optimize the organizational impact of the EDA are slight.

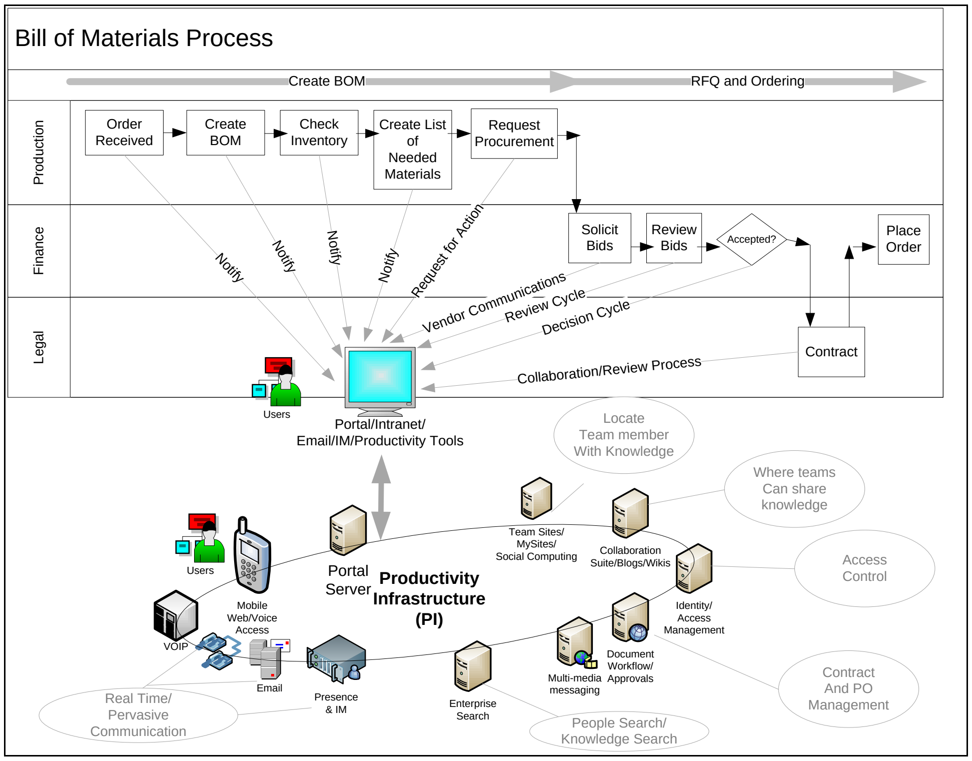
A Better Proposal: EDA/PI Integration

I may be displaying some hubris by characterizing EDA/PI integration as a better proposal. The truth is, it's a very new area and still quite theoretical, even more so than EDA, which is itself somewhat more of a vision than a reality. However, as we go through the potentialities of EDA/PI integration in the ProdCo case, I think you will see some exciting possibilities for improving the way work gets done in practical terms.

Unstructured tasks, such as procurement staffers rounding up best practices data from diverse stakeholders, tend to be messy and unpredictable. Given that reality, it is challenging to design any kind of technological solution that will consistently solve the problem and make the unstructured tasks faster, simpler, and cheaper to execute. In fact, there really isn't much in the way of standard language or practice to even describe the kind of problems that IT solutions need to solve for unstructured tasks. For this reason, I am going to attempt to work backwards from identification of unstructured task problems, to causes that can be remedied by EDA/PI integration, and build a solution approach from there.

First, what problems does ProdCo face with procurement, and which can be traced back to inefficient unstructured tasks and poor EDA integration? Wasting of time ranks high on the list because slower procurement typically translates into slower production. Then there is the loss of knowledge over time, which can result in slow procurement or other costly errors in production. Then, there are just plain mistakes made through poor communication or inadequate decision making processes.

Backing out of these problems, we can get to a set of business objectives for the integrated EDA/PI. ProdCo needs to have rapid procurement that is accurate. They need high quality and consistent knowledge transfer as team members move in and out of roles. And, they need transparent, well-documented decision processes that result in accurate, timely procurement without imposing an undue administrative burden. This last point is relevant because it is almost always possible to cure a process by larding it up with many onerous administrative tasks and parameters. The net effect, though, is usually counter to the goal of efficiency. Getting the right process in place without choking the team members with bureaucracy is a fine balancing act.



Bill of Materials Process
Create BOM
RFQ and Ordering
Production
Finance
Legal
Order Received
Create BOM
Check Inventory
Create List of Needed Materials
Request Procurement
Solicit Bids
Review Bids
Accepted?
Place Order
Contract
Notify
Notify
Notify
Notify
Request for Action
Vendor Communications
Review Cycle
Decision Cycle
Collaboration/Review Process
Users
Portal/Intranet/ Email/IM/Productivity Tools
Locate Team member With Knowledge
Where teams Can share knowledge
Access Control
Contract And PO Management
People Search/ Knowledge Search
Real Time/ Pervasive Communication
Users
Mobile Web/Voice Access
VOIP
Email
Presence & IM
Portal Server
Productivity Infrastructure (PI)
Team Sites/ MySites/ Social Computing
Collaboration Suite/Blogs/Wikis
Identity/ Access Management
Document Workflow/ Approvals
Multi-media messaging
Enterprise Search

Figure 10
Matching the RFQ process flow to a productivity infrastructure, showing the connections between process steps and collaboration, communication, and knowledge sharing areas of the PI

In Figure 10, we see how productivity infrastructure can help ProdCo's procurement staff collaborate with one another, as well as other groups, share knowledge, and communicate efficiently in the fulfillment of the RFQ process. In the early steps of the process, from "Order Received" through "Create List of Needed Materials," the procurement staff are notified of changes in order status and impending RFQ workflow. The specific mechanism of notification could vary, though it would probably be an email alert or a change in an order status screen on the intranet. As the procurement staff need to solicit and review bids, they can use the team sites and search features of the PI to find expertise that may rest with individuals throughout the company. In our example, if the procurement staffer searched for the name of the product that gets shipped from Mexico, he or she might be directed to a wiki or blog that communicates the salient details of shipment in hot weather that would enable even an inexperienced procurement staffer to avoid the problem that has plagued others. Once that kind of knowledge is extant and searchable, it is harder for the organization to lose. The Web 2.0 type of features that allow individual users to create their own material easily – but also securely – is an underpinning of successful PI.

Stakeholders in the process can work on documents, such as contracts and RFQs, in a virtual collaboration environment and document management system. Throughout the process is communication, pervasive and real time, through email, phone, mobile devices, IM, and Web meetings. The net effect of this sophisticated PI is a faster procurement cycle with greater sharing of knowledge in real time and preservation of knowledge. PI has the potential to help ProdCo realize its business objective of rapid, accurate procurement.

You might imagine, based on robust productivity functionality shown in Figure 11, that even without integration of the underlying enterprise EDA applications, you would be ahead of the game. Having solid connectivity between stakeholders in real time, and streamlined access to documents and knowledge is a big boost to productivity, even without tight integration with ERP and CRM. However, as you start to tie the two sets of systems together, the benefits become striking – assuming of course that the organization and users are up to the challenge of making it all work.

EDA/PI Integration Requirements

What will it take to integrate ProdCo's PI and EDA? To keep it simple and focused, so we can learn, we will look at the functional requirements for EDA/PI integration as they relate to the procurement example only. The following are the requirements that would drive improved productivity in the procurement process through connection between the underlying EDA-enabled ERP and CRM applications and the productivity infrastructure. We will also assign this hypothetical project a name. We are considering integrating ProdCo's PI with its EDA, so the name PIEDA fits well. It sounds a bit like a geek fraternity, which in a sense, it is.

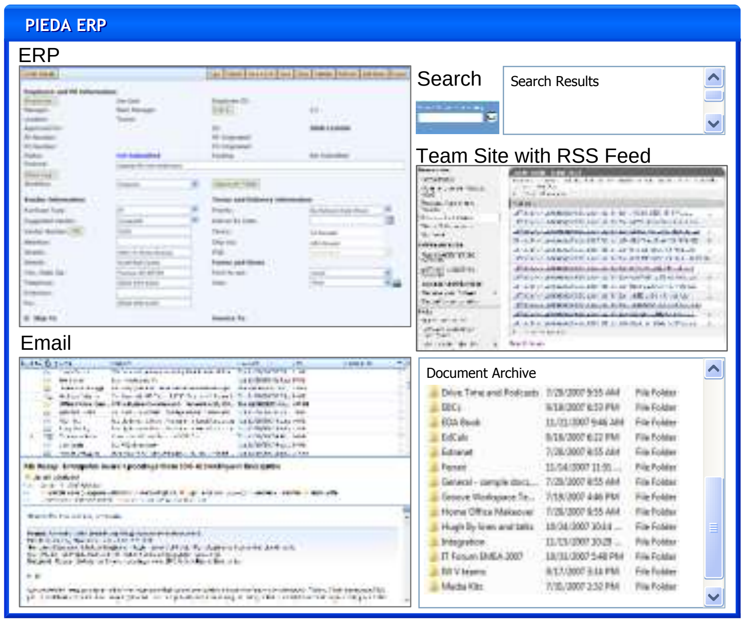


Figure 11
Wireframe mock-up of a composite UI for PIEDA, showing a single interface that contains ERP features, search, email, team site with RSS feed, and a document archive

One of the first requirements that the PIEDA team will have to figure out is the interface. There are two basic choices: integrate the productivity tools into the EDA-enabled ERP and CRM applications, or surface the ERP and CRM applications through the productivity infrastructure tools. Both options require the use of APIs and custom tooling for implementation, though neither requires creating interfaces wholly from scratch. The major productivity suites are available with APIs and development kits that enable integration of interfaces and routing and transformation of messaging to and from enterprise systems.

The first option, which is shown in Figure 11, puts the EDA-enabled ERP app into the same UI as email, document archive, search, and a team site with RSS feeds. In this kind of unified dashboard, the procurement staffer can work on specific RFQs and have a view of his or her email, relevant documents and team updates without leaving the ERP app. In the second option, the ERP app might appear as a side bar in the email client, for example. The best practice for this entire issue might be to do some research first, consulting the end users and showing them the mockups to get input on how they prefer to work before committing to one approach or another.

As shown in Figure 11, one of the most basic features of PIEDA is the ability to automate notification of procurement process status changes to stakeholders. When an order is received from the CRM system, the end user is notified, perhaps through an email alert. Same thing when the "Create BOM" process is executed in the ERP system - the procurement person is notified, and so forth, through Check Inventory, and Create List of Needed Materials. This real time (or rapid) notification of end users of changes in state in the EDA serves to prompt action on the part of the end user. For example, if the procurement staffer is alerted that an order is going into the "Check Inventory" state, then he or she can consult the team site for background information on this type of order and be ready to handle the vendor selection right away. In this way, PIEDA connects the "enterprise nervous system: of EDA right to the human thought processes necessary for completion of the process flow.

Following the process flow shown in Figure 11, the next requirement for PIEDA we need to address is the request for action that occurs when the process reaches the "Request Procurement" stage. When the bill of materials has been generated, and the inventory checked, and the list of needed materials drawn, the procurement is now ready to begin. PIEDA will alert the procurement staffer that he or she needs to create an RFQ and solicit bids. At first, this would probably be an email that requests that action be taken. To make the process efficient, though, the email should contain an embedded link that will take the end user to exactly the right screen in the ERP app where the RFQ can commence. If the end user simply receives an email notifying him that he needs to go to the ERP system and start an RFQ, and making him look it up (perhaps by copying and pasting a job number into a search field) we have not accomplished very much in terms of efficiency.

This innate linking from email, blog entries, IM texts, and documents through to the actual job page on the ERP system is an essential requirement to make PIEDA worth the effort and expense. Users should be able to toggle effortlessly between PI and EDA without having to look up the job detail in the middle of the process.

As the RFQ and bids go through the review loops, PIEDA needs to keep decision makers close to each other and to the related documents. The ERP app needs to show the presence of stakeholders so that all the people involved in the process can connect with one another – either through IM, email, phone, or by looking up information that was authored by a particular person or team. In other words, if a procurement staffer receives an alert that he must review a draft of an RFQ, the draft of the document should appear in the same interface as the alert. Then, the draft should show the presence of its authors, with their availability for IM, email, or phone instantly visible. Alternatively, a user should be able to link on the "mysite," teamsite, or blog of any document author or group of authors in order to learn more about what they know about the procurement process.

It should be possible in PIEDA to conduct a search through the documents, blogs, wikis, teamsites, mysites, and ERP business data to find relevant information about the procurement, or the project itself. Through an enterprise search interface, and back-end enterprise search engine, the end user needs to be able to look up any missing information about the procurement. In our example, if the procurement staffer searches for the component that needs to be shipped in a refrigerated truck, he or she should see the pertinent notes about shipment in the search results –

or at the very least, get a search result that points to documents that contain the needed information.

A more sophisticated version of the enterprise search scenario described above involves including people in the search results. If the end user conducts a search for the bill-of-materials components, the search results could contain both a list of documents related to the components as well as the authors, including their presence and knowledge contributions to the organization. That way, if a procurement staffer cannot find exactly what he is looking for, he can click on the results of a people search and connect to expertise either live or through a published knowledge contribution such as a blog or wiki.

Business intelligence should be a requirement for PIEDA. Though not perhaps a drop-dead necessity, BI gives stakeholders the ability to analyze data and create reports that lead to organizational knowledge. For example, it may not be known that certain component gets destroyed by heat in transit. If a procurement staffer ran a report on job orders that exceeded planned costs, he or she might notice that certain types orders that contained a specific component (which melted in transit) all resulted in poor financial results for their respective orders. Of course, such BI functionality already exists in many ERP systems. The challenge for PIEDA is to make it available to users easily, through the portal front end of the productivity infrastructure. This availability helps expose the knowledge potential of the business intelligence. And, going further, with RSS feeds and subscriptions, it becomes possible to alert people of new knowledge without the receivers of the knowledge needing to know in advance to look for it. Finally, with an EDA-PI connection, the RSS feed itself becomes an event with the potential to trigger action.

The collaborative workflow for document creation needs to be manifest through PIEDA. As end users crate documents and revise them, they must be able to see who has contributed to the drafts and be aware of their presence if they need to contact them. The document management functions of PIEDA need to be able to route documents along approval paths, allowing for final approvers to be aware that drafts have been created for their review. The ability to set up an instant online Web meeting or conference call to review a document in real time should be built into PIEDA.

Bottom line, PIEDA needs to provide users with real time (or near real time) access and awareness about events that occur in the ERP and CRM systems. PIEDA needs to give users access to the people who create documents and data inputs to the system. The goal is streamlined, rapid decision making that results in the best possible decisions. PIEDA needs to stimulate the creation and distribution of knowledge in an effortless manner. Users need to find or receive knowledge passively.

PIEDA's Target Architecture

Having outlined PIEDA functional requirements, the challenge now is to relate them to a target architecture for the development of a working EDA. After all, the goal of PIEDA is to reduce the amount of random and unstructured communication between people as they manage the information in the CRM and ERP systems. Doing this means connecting events in those systems to automated, and semi-automation actions in the productivity infrastructure. The first step in this process is to understand how the functional requirements map to events in the overall architecture.

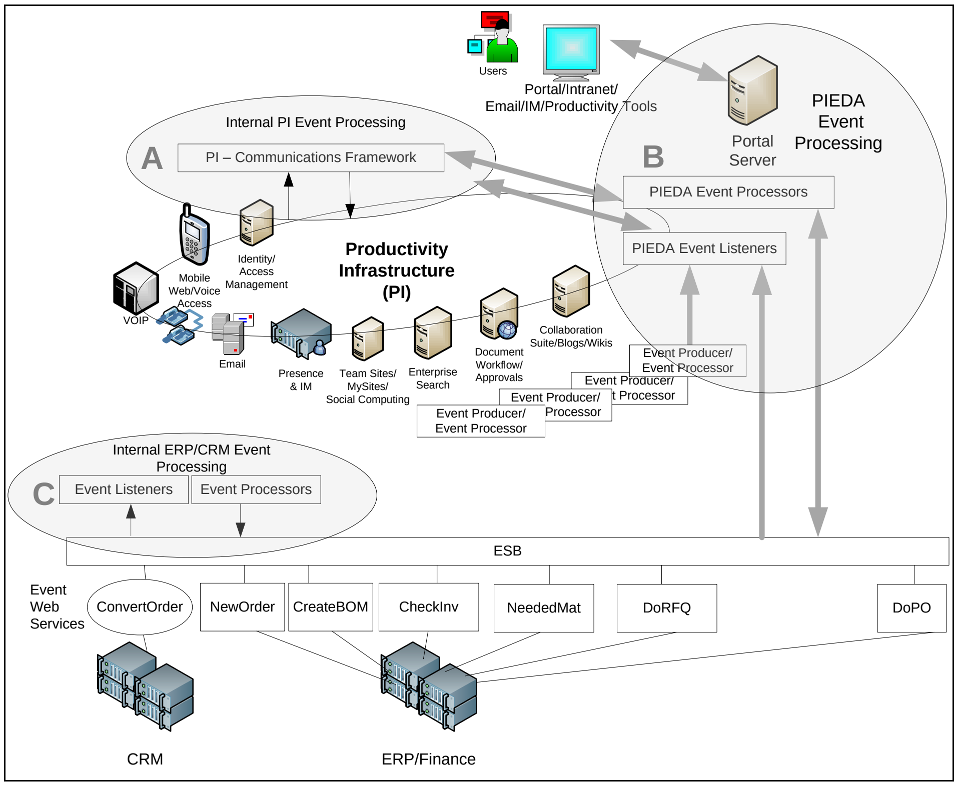


Figure 12
Target architecture for PIEDA, showing three separate areas of event processing: The CRM and ERP systems cycle event data between themselves (area C), as does the productivity infrastructure (area A). PIEDA (Area B) listens for event data from both, and produces its own events. All three event processing areas listen for each other's events.

Figure 12 shows a target architecture for PIEDA. Notable is the contrast to earlier EDA examples we have explored, which have one area of event processing. PIEDA has three. There are events produced, listened for, and processed at the level of the CRM and ERP systems, shown as Area C in the figure. In addition, the productivity infrastructure itself has a whole event processing setup, shown as Area A in the figure. This is an interesting discovery for anyone getting under the hood of a productivity suite: It has many event-driven features right out of the box. Whether you're dealing with Microsoft SharePoint and Office, IBM Lotus, or others, you will find a handsome compliment of event processing going on inside the application suite. For example, SharePoint has numerous automated notifications of document draft changes and blog posts, and so forth.

The third area of event processing – Area B - is PIEDA's actual integration between areas A and C. As depicted in Figure 12, PIEDA's event processing area is contained within the portal server. However, it need not be. We are approach PIEDA as if it were going to be developing using custom development APIs built onto the portal server. When undertaking a project such as PIEDA, many different alternatives may in fact be more attractive. For example, it might make the most sense to build PIEDA on top of the ERP-CRM service bus or develop it on its own standalone application server. The bottom line, though, is it will have to be built. At this time, there is no out of the box solution for the kind of functionality envisioned for PIEDA.

Figure 12 is a rather complicated picture, unfortunately. However, though PIEDA's A-B-C setup is complex, the beauty of it is that areas A and C can operate quite well on their own, even if you take out B, the PIEDA event processing center. PIEDA is an incremental upgrade to the architecture, and the event processing capabilities of the PI and the back-end are not reliant on it. I mention this because some might look at Figure 12 and decide that the effort is not worth the complexity. On the contrary, the productivity gains should justify the work of making it happen. Plus, if it's done right, the complexity of PIEDA need not result in an excess of administrative load or inhibitions of process agility.

To illustrate how the target architecture for PIEDA fulfills on the functional requirements, I'll walk through one specific example the event flow. Using the requirement of "Request Procurement," let's examine the specific EDA functions that PIEDA needs to realize in order to make the requirement work. The requirement specifies that the ERP system inform that procurement staffer that procurement is needed for a list of items. The following flow of process steps describes how this requirement could be fulfilled, adding in the events that are produced and listened to at each step.

Step	**Event Producer(s)**	**Event Listener(s)**	**Event Processor(s)**	**Reaction**	**Comment**
Notify procurement staffer that procurement is needed	DoRFQ (Area A)	PIEDA (Area B)	Email Server (Area A)	Send email that carries event state: Order Number SKUs needed	Email should contain dynamic links to underlying documents and presence information for stakeholders
IF collaboration is needed, create IM session to discuss RFQ	PIEDA Portal Server (Area B) CreateIM	PIEDA (Area B)	IM Server (Area A)	Create IM session that is populated with event state information	Requires XML carried on SIP
IF documents are needed to support procurement, pr-actively link staffer with documents	PIEDA Portal Server, (Area B) working through Word Processor – PostDocument	PIEDA Portal Server (Area B)	PIEDA Portal Server (Area B)	Publish document update in document library on PIEDA	Document needs to contain author presence data, and links to author mysite and teamsite
Generate vendor solicitation emails	PIEDA Portal Server (Area B) CreateVendSol	PIEDA Portal Server (Area B)	Email Server (Area B)	Create email, populated with RFQ data	RFQ data should be in state carried through event publication

Table 1
Connecting event producers, listeners, processors, and reactions to process steps in "Request Procurement"

Once again, our old EDA friend – carrying state – surfaces in PIEDA. In order for an email or IM session to be generated as a reaction to an event, the event processor must have access to the event state data in the event message. For example, if the user initiates the CreateIM command, which starts an instant message session with another stakeholder, the requirement is that the IM session will automatically contain data about the specific procurement under discussion, as well as links to the ERP system files as well as documents, that need to be discussed. To do this, the CreateIM command needs to contain a function that lets it search for already published event data about the procurement in question. In effect, CreateIM is a two-state command series that looks like this:

1) WHEN user initiates CreateIM command for procurement XYZ, SEARCH message queue of events published for DoRFQ and FIND procurement XYZ data
2) INITIATE IM Session (through SIP), carrying XYZ data as XML bound to SIP

I use this IM example for a reason, namely that SIP to XML integration is not easy, and the standards involved continue to evolve. Binding SOAP to SIP also challenging, to the point where some sources say it is not possible, or reliable. However, this is exactly the kind of integration that is needed to make EDA-PI integration a success. And, the integration needs to be flexible, allowing low friction for rapid changes in configuring IM session generation with changing business process models. It will take more than average effort to get it to work, but PIEDA should generate greater than average ROI.

The broader point is that PIEDA – area B – needs to be a flexible switchboard connecting the event activities with Areas A and C in order to create ROI and justification of the whole project. If PIEDA is too rigid, and changes are time consuming or expensive, then the EDA PI integration will not serve its business purpose. While this statement could be made about almost any application or architecture project, it is particularly true for PI. PI is inherently more unstructured, more unpredictable than conventional IT situations.

Implementation

Given the complexity of PIEDA, achieving the high degree of flexibility envisioned above is a double challenge. Not only does your development team have to create an architecture that is wholly new, and do so with custom development, they must create an architecture that is highly flexible too. Some wise people might declare the challenge not worth it at the outset.

While these wise people wouldn't be entirely wrong for wanting to avoid the hassle of entering uncharted waters, the good news is that PIEDA can actually be deployed in very small increments. Looking at Figure x, you can understand that Area A already exists in some form. Virtually every sizable business in the world has an email system, an intranet of some kind, and suites of productivity applications. VOIP and corporate IM are on the ascent as well. Building Area A from scratch is not an issue. Area C exists in its component parts, and exposing Web services and creating an EDA is a decision that is separate from building PIEDA. There would be a good rationale for creating an EDA at the CRM and ERP system level for its own sake.

The key takeaway for thinking about implementing PIEDA is to understand that Area B can be developed in stages. In fact, it can be deployed one feature at a time after a core set of EDA infrastructure pieces have been put in place. Unlike Area C, which needs a certain number of event producers, listeners, and processors to hit critical mass and function, PIEDA can start quite small. PIEDA can take one instance of event processing, such as "Request Procurement" and put it into full effect. This approach might even be the optimal way of getting it off the ground.

Given the fickle nature of human-machine interaction, which is truly at the heart of PIEDA, it might be smart to design, test, and deploy in tiny increments. While the dev team might think it's really cool to have an automated IM session generated, users might scoff at such a feature. To save time and resources, and assure the highest level of success, PIEDA's implementation plan should include a thorough usability testing and feedback cycle. End users need to be included intensely throughout PIEDA's lifecycle. This is especially relevant because PIEDA does not rely on pre-packaged software, which usually undergoes its own round of usability tests and market research. In the future, though, PIEDA type platforms might become common. Today, though, it's an unexplored frontier. (Free tip for readers: If anyone is brave enough to stake a claim in this space, venture capitalists might find it interesting..)

Again, in the spirit of not repeating what has already been discussed in great depth previously, let’s touch briefly on governance. Governance is where the complexity of PIEDA can be a challenge. With PIEDA, you are talking about integrating EDA components on multiple platforms, with all the federation and mediation that involves. And, you will confront the fact that productivity infrastructure typically comes with its own built-in identity management apparatus that you can’t really switch off. It may be possible to adapt the PI’s identity management system for use in the broader EDA. Or, you may be able to federate the identity management and governance solutions. Lifecycle and provisioning will also be built in to the PI, and it is almost certain that you will need to keep those functions in place, despite the fact that you will not be able to use those functions for life cycle and provisioning of the EDA components. All of these challenges can be solved, however, with time and effort. In planning for PIEDA, assume that governance will be a major investment.

Conclusion

We are at the very start of the movement towards sophisticated integration of EDA and PI. Early examples of this technological paradigm include the Microsoft-SAP Duet product, which integrates Outlook email with SAP applications, or the Lotus Sametime integration with WebSphere-based business applications. An interesting trend in the evolution of EDA-PI is the use of the “cloud” in linking events and productivity tools. Early examples of this include Facebook.com’s ability to create discussion threads and mobile alerts based on postings to a user’s page by other network members or Salesforce.com’s connectivity with the Outlook email client, taking advantage of Web Services.

It is likely that more sophisticated EDA-PI integrations will continue to develop as the workforce becomes more accustomed to pervasive, mobile computing as a mode of work. The need for disparate groups of people to communicate smoothly on fast-moving business issues regardless of location or computing platform should drive ever increasing capabilities in EDA-PI solutions. This is all the more relevant as a younger generation, who have known nothing other than mobile computing, arrives in the workplace and expects to conduct business in real time across a myriad of temporal and spatial factors.