SKILLS. TALENT. $OURCE. OUT$OURCE.

written by Manoj Khanna | February 3, 2003

The traditional definition of outsourcing describes using external agents to perform one or more organizational activities – purchasing of goods or services. Mostly in our IT segment we see outsourcing as contract programmers to third party facilities. And now there is something else too associated with it. And what's that?

The usual way of outsourcing are getting very common practices with every company. The unusual ways are trending in. The global village is here. In my business I'm only six-tenths of a second (speed of light measurement) away from the other business person around the globe. If I need to partner with somebody in IT – I can always easily look into Bangalore/Hyderbad, India or Belarus, Russia, and Vancouver, Canada.

According to Economist, "Any activity that relies on a screen or a telephone can be carried out anywhere in world." A-N-Y-W-H-E-R-E-I-N-T-H-E-W-O-R-L-D! Thus making services such as insurance processing, customer care solutions, banking-services, home-security, health-care delivery, accounts-management as exportable as automobiles or TVs or VCRs. Major airlines have their back-office works transferred to countries like India, and Phillipines. Even systems like EMS in Australia are controlled and monitored remotely in offices like, Singapore, Malaysia, Sri Lanka, Indonesia, and Taiwan.

So what's the age of outsourcing in which we are living in? But in the first place the question arises "W-H-Y-O-U-T-S-O-U-R-C-E?" Its a competitive world. And in a competiting market in order to $ave those extra bill$ is as equally important as to sustain yourself in the business. Why pay for a service $100 when you can get it done for $30 or less. So that's why O-U-T-$-O-U-R-C-E!!

Wages in the United States, Germany, Switzerland, United Kingdom, Japan are…ludicrously higher than the rest of the world. We like high wages and high ranking on any competitive table. And if we would like to continue this position or stay near the top, then we must work on the next act.

Its a revolution in cognitive processing. We all are in technology business, from travel-services-organizations, to twelve-table-restaurants, to financial-services, the list goes-on. Downsizing is not the solution. Finding new ways to enhance revenues – n-e-w-p-r-o-d-u-c-t-a-n-d-i-n-n-o-v-a-t-i-o-n is the key for the $ucce$$ gate!

And I would like to conclude with James Morse saying, “The only sustainable competitive advantage…comes out-innovating the competition.”

Think revolution, not evolution.

SKILLS. TALENT. $OURCE. OUT$OURCE.

written by Manoj Khanna | February 3, 2003
The traditional definition of outsourcing describes using external agents to perform one or more organizational activities – purchasing of goods or services. Mostly in our IT segment we see outsourcing as contract programmers to third party facilities. And now there is something else too associated with it. And what’s that?

The usual way of outsourcing are getting very common practices with every company. The unusual ways are trending in. The global village is here. In my business I’m only six-tenths of a second (speed of light measurement) away from the other business person around the globe. If I need to partner with somebody in IT – I can always easily look into Bangalore/Hyderbad, India or Belarus, Russia, and Vancouver, Canada.

According to Economist, “Any activity that relies on a screen or a telephone can be carried out anywhere in world.” A-N-Y-W-H-E-R-E-I-N-T-H-E-W-O-R-L-D! Thus making services such as insurance processing, customer care solutions, banking-services, home-security, health-care delivery, accounts-management as exportable as automobiles or TVs or VCRs. Major airlines have their back-office works transferred to countries like India, and Phillipines. Even systems like EMS in Australia are controlled and monitored remotely in offices like, Singapore, Malaysia, Sri Lanka, Indonesia, and Taiwan.

So what’s the age of outsourcing in which we are living in? But in the first place the question arises “W-H-Y-O-U-T-S-O-U-R-C-E?” Its a competitive world. And in a competiting market in order to $ave those extra bill$ is as equally important as to sustain yourself in the business. Why pay for a service $100 when you can get it done for $30 or less. So that’s why O-U-T-$-O-U-R-C-E!!

Wages in the United States, Germany, Switzerland, United Kingdom, Japan are…ludicrously higher than the rest of the world. We like high wages and high ranking on any competitive table. And if we would like to continue this position or stay near the top, then we must work on the next act.

Its a revolution in cognitive processing. We all are in technology business, from travel-services-organizations, to twelve-table-restaurants, to financial-services, the list goes-on. Downsizing is not the solution. Finding new ways to enhance revenues – n-e-w-p-r-o-d-u-c-t-a-n-d-i-n-n-o-v-a-t-i-o-n is the key for the $ucce$$ gate!

And I would like to conclude with James Morse saying, “The only sustainable competitive advantage…comes out-innovating the competition.”

Think revolution, not evolution.

.NET. WINDOWS. J2EE. LINUX. UNIX/MAC. THE WAR.

written by Manoj Khanna | February 3, 2003
.NET stands up against J2EE

Windows cheaper than Linux

Arguments supporting both platforms

- Regardless of which platform you pick, new developers will need to be

  trained (Java training for J2EE, OO training for .NET)
- You can build web services today using both platforms
- Both platforms offer a low system cost, such as jBoss/Linux/Cobalt for

  J2EE, or Windows/Win32 hardware for .NET.
- Both platforms offer a single-vendor solution.
- The scalability of both solutions are theoretically unlimited.

Arguments for .NET and against J2EE

- .NET has Microsoft’s A-team marketing it
- .NET released their web services story before J2EE did, and thus has

  some mind-share
- .NET has a better story for shared context today than J2EE
- .NET has an awesome tool story with Visual Studio.NET
- .NET has a simpler programming model, enabling rank-and-file

  developers to be productive without shooting themselves in the foot
- .NET gives you language neutrality when developing new eBusiness

  applications, whereas J2EE makes you treat other languages as separate

  applications
- .NET benefits from being strongly interweaved with the underlying

  operating system

Arguments for J2EE and against .NET

- J2EE is being marketed by an entire industry
- J2EE is a proven platform, with a few new web services APIs. .NET is a

rewrite and introduces risk as with any first-generation technology

- Only J2EE lets you deploy web services today
- Existing J2EE code will translate into a J2EE web services system

  without major rewrites. Not true for Windows DNA code ported to .NET.
- .NET web services are not interoperable with current industry

  standards. Their BizTalk framework has proprietary SOAP extensions and does

  not support ebXML.
- J2EE is a more advanced programming model, appropriate for

  well-trained developers who want to build more advanced object models and

  take advantage of performance features
- J2EE lets you take advantage of existing hardware you may have
- J2EE gives you platform neutrality, including Windows. You also get

  good (but not free) portability. This isolates you from heterogeneous

  deployment environments.

- J2EE has a better legacy integration story through the Java Connector

  Architecture (JCA)

- J2EE lets you use any operating system you prefer, such as Windows,

  UNIX, or mainframe. Developers can use the environment they are most

  productive in.

- J2EE lets you use Java, which is better than C# due to market-share

  and maturity. According to Gartner, there are 2.5 million Java developers.

  IDC predicts this will grow to 4 million by 2003. 78% universities teach

  Java, and 50% of universities require Java.

- We would not want to use any language other than C# or Java for

  development of new mission-critical solutions, such as a hacked

  object-oriented version of C, VB, or COBOL.

- We are finding most ISVs and consulting companies going with J2EE

  because they cannot control their customers’ target platforms. We believe this application availability will result in J2EE beginning to dominate more

and more as time goes on.

In conclusion, while both platforms will have their own market-share, we

feel most customers will reap greater wins with J2EE. We feel the advantages

outweigh those offered by Microsoft.NET. That is our preferred architecture,

and we stand behind it.

As we know the field is the real test and these head to head comparisons

will be more believable by the end of 2003 when the real cost of people and

maintenance and interoperability and up time and down time.

Ready. FIRE! Aim.

written by Manoj Khanna | February 3, 2003
The biggest problem faced by business today is not learning but forgetting. And as Tom Peter quotes *“Forgetting is the key*

activity…the primary activity…these days." Forget it!

"You can't live without an eraser". (Gregory Bateson, Cybernetics)

Ready. Fire. Aim. Or: Just do it. We forget…and then we try to reinvent the wheel. But where we'll be going then. Some of the most innovative successes came from forgetting…the past…and then venturing into N-E-W! But how to i-n-n-o-v-a-t-e? Forget the details. Rapid prototyping is the core competency among the innovation's winners. *"Effective prototyping may be the most valueable 'core competence' an innovative organization can hope to have."* (Michael Scharge, author and tech maven)

What's then Rapid prototyping? It's N-O-T off-the-shelf technique. Its cultural. Thus needless to say the-way-we-do-business-around-here is basically driven by quick-and-dirty tests and experiments. The normal practices are free flowing exchange around the rough models.

So how can I (MK), you or anybody can implement such a culture? The process is subtle, but with all due surprises its the way of life that makes an impact on the innovation potential. Think about the text book – think about chemistry as a subject you learned while in school. The-prototypers-breed.

Ready. FIRE! Aim.

written by Manoj Khanna | February 3, 2003
The biggest problem faced by business today is not learning but forgetting. And as Tom Peter quotes *"Forgetting is the key activity…the primary activity…these days." Forget it!*

“You can’t live without an eraser”. (Gregory Bateson, Cybernetics)

Ready. Fire. Aim. Or: Just do it. We forget…and then we try to reinvent the wheel. But where we’ll be going then. Some of the most innovative successes came from forgetting…the past…and then venturing into N-E-W! But how to i-n-n-o-v-a-t-e? Forget the details. Rapid prototyping is the core competency among the innovation’s winners. *“Effective prototyping may be the most valueable ‘core competence’ an innovative organization can hope to have.”* (Michael Scharge, author and tech maven)

What’s then Rapid prototyping? It’s N-O-T off-the-shelf technique. Its cultural. Thus needless to say the-way-we-do-business-around-here is basically driven by quick-and-dirty tests and experiments. The normal practices are free flowing exchange around the rough models.

So how can I (MK), you or anybody can implement such a culture? The process is subtle, but with all due surprises its the way of life that makes an impact on the innovation potential. Think about the text book – think about chemistry as a subject you learned while in school. The-prototypers-breed.

RAPID. FORWARD. FAST. VISION. MANAGEMENT.

written by Manoj Khanna | February 3, 2003

There is a great deal of progress in the past two decades in Software Development Management, in terms of improving software quality, development productivity and overall development process. The progress over the last two decades specified what to do with the programs like Sourcecode Configuration Management (ClearCase, CVS, etc.) and Defect Tracking System. On the other hand, the techniques learned are impossible to put into a program.

“VISIBILITY” is the most important element of project, and so is its “ABILITY TO CHANGE DIRECTIONS”. A highly visible project tells it’s leader where it is at all times.

In a “VISIBLE” scenario, the project instruments everything with tests. Not regressing the bench version, and making every feature available in bench version a deliverable. Thus making the status of the project always visible.

To be visible, a project must write Programmer Tests (sometimes called Unit Tests). Done right, they reduce the odds of defects to so low that the few bugs you actually get will simply go into your requirements tracking system.

For direction changes the project must have a configuration tool (tools such as CVS, ClearCase, etc.) and the prescence of copius tests. This enables the leader to request features in the most important order, and any feature which is not requested cause no waste code and eventually no waste of work.

Thus this signifies all the tests that ensure visibility also permit steering. Tests will tell you what’s done and what isn’t.

In other cases, when there is no code produced then a record of document reviews is a good indicator of project status. Also its important to track what’s going on with change requests and defect reports coming into the project. A

good risk register is also important.

A good quality system can go a long way to improve the development process, though it is up to you to perform the implementation. There are several documents which define a quality system. Two of those which are well-known are:

– IEEE Standard 1298

– ISO Standards 9001:1994 and 9000-3:1997 (or the later versions).

There isn’t any “canned” software development system that works across all types of software development projects. In fact, some software gurus adhere to a “contingent” approach to software development. Why? Because there are a broad variety of types of software development projects, there is, thus, a need to use a broad variety of software development approaches, depending on the type of project involved.

RAPID. FORWARD. FAST. VISION. MANAGEMENT.

written by Manoj Khanna | February 3, 2003
There is a great deal of progress in the past two decades in Software Development Management, in terms of improving software quality, development productivity and overall development process. The progress over the last two decades specified what to do with the programs like Sourcecode Configuration Management (ClearCase, CVS, etc.) and Defect Tracking System. On the other hand, the techniques learned are impossible to put into a program.

“VISIBILITY” is the most important element of project, and so is its “ABILITY TO CHANGE DIRECTIONS”. A highly visible project tells it’s leader where it is at all times.

In a “VISIBLE” scenario, the project instruments everything with tests. Not regressing the bench version, and making every feature available in bench version a deliverable. Thus making the status of the project always visible.

To be visible, a project must write Programmer Tests (sometimes called Unit Tests). Done right, they reduce the odds of defects to so low that the few bugs you actually get will simply go into your requirements tracking system.

For direction changes the project must have a configuration tool (tools such as CVS, ClearCase, etc.) and the prescence of copius tests. This enables the leader to request features in the most important order, and any feature which is not requested cause no waste code and eventually no waste of work.

Thus this signifies all the tests that ensure visibility also permit steering. Tests will tell you what’s done and what isn’t.

In other cases, when there is no code produced then a record of document reviews is a good indicator of project status. Also its important to track what’s going on with change requests and defect reports coming into the project. A

good risk register is also important.

A good quality system can go a long way to improve the development process, though it is up to you to perform the implementation. There are several documents which define a quality system. Two of those which are well-known are:

– IEEE Standard 1298

– ISO Standards 9001:1994 and 9000-3:1997 (or the later versions).

There isn’t any “canned” software development system that works across all types of software development projects. In fact, some software gurus adhere to a “contingent” approach to software development. Why? Because there are a broad variety of types of software development projects, there is, thus, a need to use a broad variety of software development approaches, depending on the type of project involved.