Term Paper

On

Software Project Managemen

{Topic: => Managing Distributed Projects}

**More Free Term Papers On Site: Www.Maturski.Org**

**Introducti On:**

In Today’s Organizations the Common Unit of Work Is the Project. Turner Defines a Project As “an Endeavor in Which Human, Material, And Financial Resources Are Organized in a Novel Way, To Undertake a Unique Scope of Work, For a Given Specification, Within Constraints of Cost and Time, So As to Achieve Beneficial Changes Defined by Quantitative and Qualitative Objectives”. Projects Have Moved From Being Simple Phenomena to Manage to More Complex Entities Spanning Geographical Locations, Multiple Occurrences, And Different Organizational Affiliations, With It Being the Key Enabler for the Transformation. For Instance, A Co-Located Program Involves Multiple Projects Running at One Location, Whereas a Distributed Project Is a Single Endeavor Conducted From Multiple Locations. Finally, The Most Complicated Scenario Is Multiple Projects Conducted at Multiple Locations.

Complexities Can Be Attributed to Managing Multiple Interdependencies Across Time, Space, And Projects Knowledge in Projects Calls for a Close Look at Insights Generated Within Each Individual Project, Such As Schedules, Milestones, Meeting Minutes, And Training Manuals. Individual Project Members Need to Know When, What, How, Where, And Why Something Is Being Done and by Whom, With the Goal Being to Promote Efficient and Effective Coordination of Activities. From the Macro Perspective, An Organization Must Have an Inventory of All Projects Under Way at Any Given Time, Or Knowledge About Projects. This Aids in the Planning and Controlling of Resources to Maximize Utilities. Knowledge Includes Employee Assignments to Projects, Return On Investment, Cost and Benefit Analysis, Deadlines, And Customer Commitments and Expectations. It Is Common for Such Knowledge to Be Generated at Regular Intervals, Such As in Weekly, Monthly, Or Bi-Monthly Reports. Knowledge From Projects Is a Post Hoc Analysis and Audit of Key Insights Generated From Carrying Out Projects. This Knowledge Is a Key Determinant of Future Project Success, As It Aids Organizational Learning. These Three Categories Call for Distinct Roles by It to Enable Effective and Efficient Knowledge Management.

***Communication Is Key: =>***

**Despite Their Geographic Separation; Distributed Developers Need to Stay in Constant Contact. Conference Calls and Email Are Basic and Important Ways That Should Be Used Regularly. Even When Time Zone Differences Require Some Team Members to Participate During Unusual Hours, There Is Still No Substitute for Talking to Each Other.**

However, Today the Networked Team has Some New Choices to Consider As Well:

**Threaded Discussions: Many Software Tools Facilitate Newsgroup-Like Discussions That Allow Multiple Developers to Communicate Asynchronously On Any Topic. Better Than Email, Threaded Discussion Tools Focus Debates On Specific Issues and Save the Dialog for Later Use. Some Tools Can Link Discussions to Specific Projects, Tasks, Or Other Development Assets.**

**Instant Messaging: I'm Is Starting to Find It's Way Into the Business World As a Valuable Tool. Lower Friction Than the Telephone, Yet More Interactive Than Email, I'm Is Valuable for Asking a Quick Question or Holding an Informal Chat. Like Threaded Discussions, Many I'm Tools Can Save Dialogs for Later Review. To Avoid Privacy Concerns of I'm Messages Traversing Public Networks, Teams Should Consider an I'm Tool That Can Be Deployed On the Corporate Intranet.**

**Blogging: Web Logging or "Blogging" Is the Process of Instant Publishing to a Web Page. "Blogs" Typically Contain Short Messages That Follow a Chronological Order. Blogging Was Created by Individuals Wishing to Chronicle Daily Work, Personal Experiences, Or Just Random Streams of Consciousness. Like Im, Blogging Is Beginning to Find It's Way Into the Corporate Setting. Team Members Can Use Blogging to Publish Their Progress, And the Web Interface Makes It Easy for Everyone to See Each Other’s Notes.**

**Web Conferencing: When Real-Time Communication Is Needed, Conference Calls Can Be Significantly Enhanced with Web Conferencing. Web Conferencing Can Be Used for Group Presentations, Interactive Planning and Review, And Even Unstructured Brainstorming Sessions. Both Pay-For-Use Web Conference Services and Commercial Software Products That Can Be Centrally Installed Are Available. Depending On the Software, Web Conferencing Products May Include Collaboration Features Such As White Boarding, File Sharing, Instant Messaging, And Cooperative Editing.**

Depending On Bandwidth Availability, Teams May Also Want to Consider Emerging Technologies Such As Voice-Over-Ip and Even Video-Over-Ip. Anything That Promotes Effective Team Communication Should Be Considered.

***Shared Repositories: =>***

**In Typical Projects, Developers Must Share Many of the Assets Used to Specify, Design, Implement, And Test Software. In the Past, Network Bandwidth and Even the Tools Themselves Prevented Remote Developers From Having Direct Access to the Same Repositories Used by Developers Operating Behind the Corporate Firewall. Today, A Broad Array of Software Management Tools Is Internet-Enabled. Combined with Better Availability of Reliable Bandwidth, These Tools Increasingly Make It Feasible for All Team Members to Share Common Repositories.**

Shared Repositories Provide Developers with Up-To-Date Information, Thereby Reducing Conflicts. Internet-Enabled, Repository-Centric Tools Are Available for Key Development Processes Such As

* Requirements Management: Formal Requirement Specifications, Dependencies, Responsibilities, And Change History
* Source Code Management: Versioned File Assets Such As Design Documents, Source Code, And Binary Files
* Change Management: Bug Reports, New Feature Requests, Defect Tracking, And Traceability
* Project Management: Project Schedules, Task Breakdowns, Work Assignments, And Progress Reports
* Test Management: Test Plans, Test Cases, And Test Results

Moreover, Some Internet-Enabled Tools That Help to Manage These Processes Are Starting to Provide Integrated Collaborative Features, Such As Threaded Discussions and Peer-To-Peer Messaging. These Tools Further Foster Team Synchronization.

Large Projects and Large Project Teams May Require Access to Many Corporate Resources. In These Scenarios, Teams Can Benefit From Resource Portals That Consolidate and Concentrate Disparate Information Sources. For Example, Web-Based Development Resource Portals (Drps) Can Provide Access to Research Materials and Project Information From Multiple Repositories, Focused On the Needs of Developers. Integrated Search and Discovery Capabilities Provide a Central Place to Traverse a Broad Array of Information. For Effective Distributed Development, The Key Is to Surface Information Stored in Corporate Repositories to All Members of the Team.

***Keep Remote Developers Involved: =>***

**Perhaps the Most Important Way to Keep Remote Developers On Track Is to Keep Them Included. Every Team Member Needs Periodic Reassurance That They Are Important to the Project. The More a Remote Developer Feels in the Loop, The More Likely They Are to Contribute Effectively. There Are Many Ways to Ensure That Distant Team Members Are On Track While Fostering Their Sense of Involvement. Here Are a Few Ideas:**

**Show-And-Tell: Remote Team Members Should Periodically Demonstrate Their Work Via Online Presentations. They Should Also Be Included in Training Sessions, Turnover Meetings, And Even Customer Presentations. Participation in Important Milestones Such As These Allows Distant Developers to Showcase Their Work and Demonstrate Their Expertise.**

**Reviews: Periodic Design and Code Reviews Are Good Ways to Keep Tabs On Remote Developers’ Progress. In Order to Keep Them From Feeling Singled Out, They Should Participate in Reviews of Other Team Members’ Work As Well. One Approach Is to Perform Code Walkthroughs, In Which the Author Drives the Presentation. This Approach Rotates Each Team Member’s Responsibilities and Maintains a Sense of Equality.**

**On-Site Visits: Teams Shouldn’t Forget the Importance of Periodically Inviting Remote Developers to Headquarters or Other Development Centers. Conversely, Most Remote Developers Appreciate Having Occasional Team Meetings at Their Locale. Digital Collaboration and Information Sharing Is Helpful, But Nothing Can Replace the Interaction and Camaraderie of Occasional Face-To-Face Meetings.**

A Close Examination of the Codification and Personalization Approaches Led Us to Draw Parallels to Two Popular Models of Computing: Client-Server and Peer to Peer (P2p). The Client-Server Paradigm, Wherein a Centrally Located Resource Is Used by Multiple Clients to Request Services for Task Accomplishment, Is Common in Most Distributed Computing Environments. P2p Is a Rather Recent Computing Paradigm in Which All Nodes Can Take the Role of Either Client or Server. A Node Can Request Information From Any Other Node, Or Peer, On the Network and Also Serve Content. Due to the Centralization of the Main Resource Provider, Client Server Computing Is Similar to the Codification Strategy, Whereas the Distributed Nature of P2p, In Which Each Node Owns and Makes It's Resources Available to the Network, Can Be Viewed As Parallel to the Personalization Strategy. Hence, We Term Codification and Personalization As the Centralized and the P2p Approaches, Respectively, To Knowledge Management.

## *Managing Distributed Testing =>*

## There's a Great Deal Talked About Different Types of Testing: Web Testing, Regression Is Testing, User Testing and of Course Black Box Testing, White Box Testing, Even Gray Box Testing. But There Is Another Type of Testing That Receives Less Coverage, Distributed Testing. Here I Shall Attempt to Explain What We Mean by Distributed Testing, And How It Compares with Non-Distributed Testing.

With Tet Where the Terminology Is Very Important. Running Tests On Several Machines at Once Is Called Remote Testing, And Remote Testing Is Non-Distributed.

### *Non-Distributed Testing =>*

So, Let's Start with Non-Distributed Testing. It Is More Frequently Used. Non-Distributed Tests Are Those That Run On a Single Computer System and Do Not, Normally, Involve Any Form of Interaction with Other Computer Systems.

I Say Not, Normally, Because There Are Some Tests That Are Run From a Host Machine to Test a Target Device That Is Running a Real-Time or Embedded Operating System. These Tests Can Be Configured and Run As Non-Distributed Tests. But the Testing of Real-Time and Embedded Systems Is a Subject in It, And Will Be the Subject of Another Article.

To Make It Even More Complicated, We Can Divide Non-Distributed Tests Into Two Further Categories: Local and Remote Testing.

Local Testing Involves Running Test Cases On a Local Computer System. So I Might Run a Local Test On My Laptop, Wherever I Am. I Don't Need to Be On a Network to Run a Local Test. By Comparison Remote Testing Does Require That I Have a Network Connection. It Allows Me to Use This Connection to Run a Test On Another Computer System. This Is Very Useful As It Allows Me to Run Tests On Other Processors On the Network Without Leaving My Desk, And to Collect the Results Back at My Desk. I Might Run Remote Tests On Several Machines, Concurrently, (Let's Call This Simultaneous Testing) Within One Test Suite, And Collect All of the Results Back at My Desk.

With Simultaneous Testing, Even When I'm Running Tests On Many Machines at Once, Those Tests Do Not Involve Any Interaction Between the Different Processors or the Tests That They Are Running.

###

### Distributed Testing =>

Distributed Testing Is Different. Because a Distributed Test Case Consists of Two or More Parts That Interact with Each Other. Each Part Being Processed On a Different System. It Is the Interaction Between the Different Test Case Components That Sets Distributed Testing Apart. Typically It Is This Interaction Between Different Computer Systems That Is Under Test. For Example, Testing a Client-Server Application or the Mounting of a File System. All of the Test Cases Processed On All of the Different Processors Contribute Towards a Single, Common, Result.

This Is Not the Same As Simultaneous Testing. Because Even Though Simultaneous Testing Involves Different Test Case Components Being Carried Out On Different Processors, And Contributing Towards a Single Result, There Is No Interaction Between the Test Cases or the Processors. As Noted Above It Is This Interaction That Sets Distributed Testing Apart.

A Further Challenge That We Have to Face with Distributed Testing Is That of Platform. For Example Testing a Client Server Application May Involve Using a Windows Client to Access One or More Unix Servers, And Controlling the Whole Process From a Linux Desktop. So Our Environment has to Be Written at a Level Capable of Working Across All of These Platforms.

### *Test Scenarios =>*

Once We Have Set Up Our Hardware Environment and Established Our Network Connections Between the Different Systems We Need to Describe the Way in Which We Want to Carry Out Our Test Cases. We Can Do This in a Test Scenario. This Lists All of the Test Cases and Describes How They Are to Be Processed. The Description Is Provided in the Form of a Directive. For the Serial Processing of Test Cases On a Local Machine We Don't Need Any Directives, Just a List of Test Cases in the Order That They Are to Be Processed. But the Test Scenario Adds a Powerful Capability to Describe Tests That, For Example, Should Be Repeated a Number of Times or for a Period of Time.

For Remote or Distributed Testing We Use Remote or Distributed Directives Which Define Which Systems Will Run Which Parts of the Test Case. Other Directives Allow Us, For Example, To Run a Number of Tests in Parallel. The Power of the Directives Is Further Enhanced As We Can Nest Them One Inside the Other. So, For Example, We Can Use Parallel and Remote Directives to Do Simultaneous Testing, With Different Tests Being Carried Out On Different Systems at the Same Time.

The Distributed Directive Is Used to Define Distributed Tests, And the Scenario File That Contains the Directive Is Reads by a Controller (See Figure), Which Allocates the Different Parts of the Tests to Separate Control Services. One for the Local System (The Control Console), And One for Each Remote System. Note That These Are Logical Systems and That More Than One Logical System Can Be Resident On the Same Physical Device. A Test Suite May Comprise Many (Up to 999) Remote Systems All of Which Interact and Contribute Towards a Single Test Result. We Feed in One Scenario File and Get Out One Results File.

***Figure 1: Simple Architecture Diagram for Distributed Testing***

###

### Synchronization =>

Ensuring That All of the Tests Happen On All of the Systems in the Correct Order Is the Greatest Challenge in Distributed Testing. To Do This We Synchronize the Test Cases Either Automatically, At System Determined Points (E.G. The Start and End of Each Test Case), Or at User Determined Points. Synchronization Is the Key to Distributed Testing and Is Important Enough to Merit Separate Consideration.  (More . . . )

The Challenge of Distributed Testing Lies Both in Synchronization and the Administration of the Test Process: Configuring the Remote Systems; Generating the Scenario Files; And Processing the Results to Produce Meaningful Reports. Being Able to Repeat the Tests Consistently, And to Select Tests for Repetition by Result I.E. Regression Testing. And Doing This Repeatedly and Across Many Different Platforms, Unix, Windows and Linux.

Here, We Look at the Implications of These Approaches On the Aggregation, Transfer, And Sense Making of Knowledge in Non-Collocated Work Environments. Drawing On the Strengths and Limitations of Each Technique, We Propose a Hybrid Model. We Begin by Comparing the Approaches, Focusing On Three Dimensions: Sharing, Control Of, And Structuring of Knowledge.

***Sharing =>***

*Many Studies Report That Members of Organizations Fear That Sharing Their Knowledge with the Community at Large Makes Them Less Valuable to the Organization. As Such, The Idea of Contributing to a Central Repository Does Not Jibe Well. In the Centralized Approach, There Are Inherent Delays Between the Moments the Knowledge Is Created in the Minds of Individuals and When It Is Posted to the Repository. Individuals May Delay Posting Not Only for Gate Keeping Purposes but Also to Allow for Confirmation of Events, Sometimes to the Point of Irrelevance. This Defeats the Concept of Real-Time Availability of Knowledge, As Insights Not Captured Immediately Are Lost. As Individuals Are More Likely to Store the Draft Notes and Working Documents of Insights On Their Local Repositories Than On the Main Server, This Concern Is Minimized in the P2p Approach. Moreover, The P2p Approach Fosters Dialogue Among the Various Agents of the Team and Develops a Spirit of Community, As Each Agent Interacts with Peers to Gain Knowledge. Hence Socialization and Externalization Is Mandated, Which Is Pivotal for Tacit Knowledge Exchange .*

***Control =>***

*The Centralized Approach Detaches the Contributor From His or Her Knowledge. Once Posted Centrally, The Author Loses Control Over Knowledge Access and Usage. In the P2p Approach, Each Member of the Organization Retains His or Her Knowledge, As Well As Explicit Control Over It's Visibility.*

Members Are Connected to Their Peers in the Organization and Can Choose What Knowledge to Share. Since Individuals Have Control Over Their Own Knowledge Repositories, They Are Less Likely to View Sharing of Knowledge As a Threat to Their Value.

***Structuring*** *=>*

*Knowledge Contained in the Central Repository Is Structured On Such Dimensions As Teams, Products, And Divisions, Enabling Faster Access Times to Required Elements. This Facilitates the Use of Filtering and Categorizing Mechanisms for Sifting. However, The Nature of Centrality Calls for Global Filtering and Categorizing Schemas, Which Are Not Optimal in All Cases. Setting Global Thresholds for Relevance, Accuracy, And Other Attributes for Knowledge May Lead to Loss of Knowledge, As Insights Considered Important for One Project May Be Lost Due to Filters. The Significant Costs in Categorizing Information by Appending Appropriate Key Words and Metadata to Knowledge Prior to Posting It Are Borne by Everyone, Whereas the Benefits of Better Retrieval Times Are Selectively Reaped*

These Characteristics. Also Make the Centralized Approach Useful for Storing Structured Knowledge About and From Projects. Requirements for Knowledge About Projects Do Not Change Frequently. As Such, Having Structured Approaches for Retrieval Is Facilitated Via a Centralized Approach. But Only a Small Percentage of the Organization Uses Knowledge About Projects for Budget Preparation, Staff Allocation, And Other Control Purposes. Hence, Storing Such Knowledge in a Central Repository Is of Minimal Value to the Remaining Employees, Or the Majority of the Organization. However, The P2p Approach Is Not Advisable Here Due to Difficulties in Filtering, Categorization, And Coordination of Disparate Knowledge Sources.

Using the Centralized Approach to Store Knowledge From Projects Helps Make Lessons Learned From Past Endeavors Available to Organizational Members at Large, But Centralization Is No Panacea. Once Again It May Be Difficult to Contribute Knowledge If Appropriate Categories Do Not Exist or the Repository Is Not Amenable to Customization. As the Repository Is Built On the Premise That All Members of the Organization Need to Access the Knowledge Base, Care Is Taken to Provide a Shared Context. This Entails Knowledge Contributors Making Extra Effort to Ensure Their Thoughts and Insights Can Be Understood by Their Peers Once Entered Into the Knowledge Repository. Developing a Ranking Mechanism to Indicate the Relevance of Results Becomes Easier Owing to the Structured Categorization of Knowledge and the Shared Context. Thus, Transfer of Knowledge From the Provider to the Consumer Is Improved by the Centralized Approach.

John Deere, A Global Producer of Tractors and Other Equipment, Uses the Personalization Approach to Knowledge Management. Deere has Established Communities of Practice (Cops) For Facilitating Knowledge Exchange. It has Recognized Hundreds of Cops, Approximately 65 Of Which Use Jd Mind Share As a Technology Solution. Knowledge Is Exchanged Within These Cops Via Videoconferencing, Email, And Discussion Groups. Using the Personalization Approach, Deere has Experienced the Challenges Discussed in This Article, Including the Difficulty of Sharing Knowledge Among Cops Due to Lack of Shared Context and Varying Schemas of Knowledge Representation.

***A Hybrid Approach* =>**

To Gain the Benefits of the Centralized and P2p Approaches and Overcome Their Limitations, We Propose a Hybrid Model with Two Components. At the Core of the System Is the First Component: A Central Repository Holding Popular Knowledge (Knowledge About and From Projects). This Repository Serves As an Index to the Second Component: Knowledge Available by Peers (Knowledge in Projects). By Storing Knowledge About and From Projects in a Central Repository, We Ensure the Following:

• Maintenance of a Shared Context, Thus Improving Means of Exploration of Knowledge;

• Ease of Access, As Knowledge About Projects Is Well Structured and Stored in a Central     repository;

• Ease of Transfer of Knowledge From Projects Throughout the Organization;

• Enhanced Validity of Knowledge From Projects, Since Only Validated Knowledge Makes It's Way     to Central Storage. Members Whose Knowledge Is Stored Centrally Can Also Be Rewarded for It's      high Value; And

• Easy Identification of the Source of Knowledge About and From Projects.

A Centrally Located Index for Knowledge in Projects, The Second Component of the Central Repository, Helps Foster an Efficient Coordination Mechanism, As This Index Contains the Sources of Knowledge in Projects. It Can Serve As the Knowledge Dictionary to Integrate Individual Knowledge. Common Terms and Categories Can Be Assigned, Along with Facilities to Serve As an Organizational Thesaurus. We Assert That Knowledge in Projects Must Be Exchanged Via P2p Approaches. Knowledge Capture Becomes Efficient and Effective, As Each Project Team Can Set Up It's Own Protocols, Build Categories, And Develop Filtering Mechanisms. As Each Project Is Unique and Each Team Is Different, Allowing for the Establishment of Flexible Knowledge Creation and Exchange Protocols Is Pivotal. Such an Approach Prevents the Loss of Knowledge Considered Relevant to One Project, Site, Or Stakeholder, Since Local Parameters Will Not Affect the Global Knowledge of the Organization.

Moreover, It Promotes Efficient Sharing of Knowledge Among Team Members, As the Fear of Making One’s Insights Available Throughout the Organization Is Absent. If an Employee Is Interested in a Knowledge Object Dealing with an Ongoing Project, He or She Can Refer to the Index and Obtain the Source. The Employee Can Then Request That Knowledge Directly From the Source.

Furthermore, Maintenance of Local Repositories Becomes Simple, As Each Team Is Given the Right to Purge Their Local Repositories Periodically, Therefore Deciding Which Knowledge Is Relevant and Useful and Which Is Not. This Not Only Prevents Irrelevant and Outdated Search Results, But Also Helps Improve Access Times. It Also Helps Circumvent the Version Control Problem by Constantly Updating the Repository While Purging Old and Irrelevant Knowledge.

Motorola, A Major Provider of Wireless Communications, Semiconductors, And Advanced Electronic Components, Systems, And Services, Takes an Approach to Knowledge Management That Conforms to the Hybrid Model. Motorola’s Major Business Segments/sectors Are the Personal Communication Segment (Pcs), Network Systems Segment (Nss), Commercial Government and Industrial Systems Segment (Cgiss), Semiconductor Products Segment (Sps), And Other Product Segment (Ops). Motorola has an Internal Portal Where Knowledge Objects (From and About Projects) Are Stored in a Central Location for Employee Access and Use Across Sectors. Examples of These Documents Include White Papers, Feature Requirements Documents, Project Test Reports, And Data Reports. These Documents Are Posted for the Software/hardware Features for a Given Product in a Sector.

Each Employee Can Customize a Portion of the Portal, Thus Fostering the P2p Approach. Functionality Exists for Storing Their Documents On the Central Repository with Password-Protected Locations for Confidentiality Purposes. Passwords Are Issued by the Owner (Knowledge Provider and Document Author) Of the Site, So If an Individual First Accesses the Site He or She Will Be Instructed to “contact the Site Owner/administrator for Password.” This Guarantees That the Material Is Not Openly Available and People Trying to Access It Have a Conversation with the Original Source or Owner of the Material Knowledge, Just Like a P2p Exchange. Search Functionalities Allow Employees to Locate Knowledge Sources; However, A Full-Fledged Indexing System Is Not Present. Thus, Motorola Follows a Hybrid Approach to Knowledge Management Much Like the One Proposed Here.

***Managing Distributed Teams: =>***

Web Methods Development Across Distributed Teams Offers Many Benefits but Also Introduces Significant Challenges. Some Research Indicates That More Than 90% Of Communication Effectiveness Is Determined by Non-Verbal Cues. Communications Between Teams in Multiple Locations and Multiple Time Zones Can Be Challenging Because in Person Discussions Are Often Replaced by Telephone Calls and Written Communications. Language and Cultural Differences Provide Additional Opportunities for Misunderstandings. Team Members Also Do Not Have the Opportunity to “see” Their Coworkers at Work and Thus Do Not Always Have Good Visibility Into What They’re Delivering. The Resulting Misunderstandings Coupled with a Lack of Visibility Can Lead to Mistrust Between Coworkers Limiting the Productivity of the Entire Team. Team Server Helps Bridge That Gap by Increasing Both Visibility and Accountability.

There Are Two Basic Ways to Distribute Web Methods Projects Across Multiple Teams. One Is to Distribute the Effort Across Teams Performing the Same Function (Like Development or Testing) And Another Is to Assign Functional Teams to a Specific Location.

**Distributed Functional Teams**

for Example, Let’s Say Your Project’s Development Team Is Distributed Across 2 Or More Regions. How Do You Hand Off Development Artifacts Between Developers? How Do Developers Tell Each Other Exactly How Far They Got? And How Do They Tell Each Other What Changes Were Made to the Program Logic?

Team Server Gives Team Members the Ability to Manage Multiple Lines of Code, Track All Changes to the Program Logic, And See Exactly What Changes Are Applied by Each Developer. This Added Level of Visibility Minimizes Miscommunication and Allows All Developer to See Exactly What Changes Other Developers Are Making to the Project.

***Distributed Teams Organized by Function =>***

A Tightly Managed and Automated Application Development Process Allows Distributed Teams to Work More Confidently. Team Server Provides a Framework Upon Which Automated Processes Can Be Built.

• Development Teams Can Very Easily Hand Off New Builds to the Qa Team.

• Qa Teams Can Use Promotion Rules to Automatically Reconfigure the Build for Deployment to the     qa Environment. They Can Also Use Team Server to Verify That, With the Exception of the      new Code, The Current Qa Environment Is a Mirror Image of the Prod Environment.

• The Administrators Within the Operations Team Will Have the Same Abilities As the Qa Team. In    addition, All Actions Will Be Audited so Administrators Will Be Able to Know Who, When, And    what Changes Have Been Made to Any of the Environments.

1.) Software Development Lends Itself Well to Distribution of Tasks.

The Question Then Becomes Why Distribute Your Software Development? There Are a Number of Reasons, Two of Which Are Primary:

* High Quality Talent Is Not Always Local
* Isolation Works - Cannot Get "Do Not Disturb" Around the Office, And This Is a Real Important Issue. There Are References Given That for Example One Phone Call to a Developer Will Make Them Take 10-15 Mins to Get Back On Track.

2) How Do I Mitigate / Manage Risk in a Distributed Project?

There Are a Number of Answers to This Since There Are Actually a Number of Types of Risk to Mitigate:

* How Do I Communicate Effectively Communicate Over Large Distances?
* How Do I Cope with Giving Up Control (Not Micromanaging by Having to Have Everyone Right There)?
* How Can I Tell If Someone Is Thrashing From a Distance?

Basically, You Can and Will Lose Control of Developers Even If They Are Collocated. The Trick Is to Be Able to Identify It, With Local or Remote Staff. Thrashing Is When a Developer Gets Stuck and Will Not Ask for Help. This Can Happen Even in a Cube, And for Weeks at a Time, So This Is Not New Even with Remote Staff. The Issue Is Again How to Identify It.

The Solution to These Comes Down to Using Scrum As Your Project Management Methodology, And This Will Assist in Identifying These Issues. Through This You Gain Control Over You’re Through Empowerment and Even If They Are Not Right There.

 Which Also Leads to the Question of Isn’t It Important to Have Face to Face Interaction? Well, It Is to an Extent. As Mentioned Later It Is Important to Make Sure That Most People Know Each Other by Face Having Had Some Type of in Person Interaction so That You Can “see” The Person That Is Remote, Meaning That You Can Get a Read On What That Person Is Saying Between the Lines. Several Techniques for How to Do This Are Given Later. Also, Stress That Communications by E-Mail for Important Items Is Not the Most Effective Means of Getting Things Done. Stress Using Phone, Skype, Web Cams, Or Getting in Person If Possible.

3) Distribute Software Development Leads to Greater Agility.

This Is True When Combined with Scrum and Short Cycles / Iterations. Doing This Can Lead to Very Quick Releases When Properly Defined. This Is Actually a Large Topic to Discuss That I Will Elaborate On Independently in Another Post.

4) Use Available Tools to Assist in Managing the Distributed Agile Process.

There Are a Number of Tools Available to Minimize Distance Between Team Members:

* Scrum: Methodology for Collaboration and Identifying / Assigning Work.
* Tfs / Scrum: Tools for Running Scrum in a Distribute Manner.
* Skype: For Communicating Amongst the Team with Low Cost.

And Make Sure That the Scrum Master Ensures That Everyone Uses Them and Collaborates Daily Through the Daily Scrum.

5) Identifying and Addressing “thrashing” With Remote Members.

* How Can Thrashing Be Identified Over Skype? This Basically Comes Down to Being Able to Ask a Simple Question: “why Are You Working for so Long On This?” If You Then You Get a Long Answer, Basically a Laundry List of Seemingly Unrelated Statements to the Problem at Hand, Then They Have Probably Tried Lots of Things Except Asking for Help.
* Also, If You Run One Week Sprints, You Don't Have to Care Too Much About Thrashing As Long As the Work Is Done at the End of the Sprint. If It Is Not Done at That Time, Then You Are Never More Than One Week Behind and You Can Then Identify Those People Who Need Help Through the Lack of a Deliverable.
* It Is Also Important to Start with Several Small Architectural Sprints (And Put These in the Wiki - My Input). These Can Be Used to Define a Baseline for How the System Should Be Implemented and Reviewed Against.

6) How to Perform Scrums with Remote People

* Keep Iterations Short, Perhaps Just One to Two Weeks. This Will Ensure That Thrashing Does Not Exceed More Than One Iteration.
* Use Tfs with Scrum Feature to Manage the Work Items in the Iteration and Allow All Remote Team Members to Collaborate.
* Extensively Utilize Skype As Well As Plug-Ins for White Boarding and Sharing. Skype Cast Daily and Make Sure You Go Around the Horn and Use Pti (Pardon the Interruption - Time Is Up).
* Prevent “hiders” In the Scrums by Making Sure That Scrum Masters Make Sure Everyone Participates.
* Be Diligent to Make Sure That Scrums Do Not Devolve Into Status Meetings; Remember That They Are for Solving Issues and Giving Marching Orders.
* What's the Effective Size of a Scrum - 10 People, Then Do Scrum of Scrums

7) Team Collaboration

What Techniques Are There to Build Collaboration Amongst the Remote Workers?

* Kick Off the Project in Person with Everyone Present No Matter the Cost As It Will Be Saved Later and It Will Allow Distributed People to Get to Know Each Other
* Explain the Business in the Kick Off, As Developers Will Need to Understand This to Work Independently, And May Also Likely Take Control of Business Features Through the Scrum Process
* Get Developers to Think As a User, Having Them Always Think in the Frame of Reference of a User Operating the System.
* Empower the Developer to Make Decisions As It Will Mitigate Thrashing, And Make Them Feel Free to Ask for Input On the Next Sprint While Working On the Current One.

8) How to Keep Remote Developers Feeling As Though They Are Part of the Team?

The Problem(S) Here Are:

* How to Keep Remote Staff Happy and Motivated
* How to Keep Remote Staff Feeling That They Are a Part of the Team

Solutions Are:

* Operate with a Staff Augmentation Model
* Engage with Local Recruiters to Find What Local Talent You Can
* Do a Staff Rotation Model, Were Every Person in a Particular Office Should Within a 2 Year Timeframe Work in Other Offices for Extended Periods (2-4 Weeks), And Always Have One Person From Every Office in Another Office to Build True Relationships.
* These Techniques Will Also Let You Get to Know the People so You Can Understand Them in the E-Mail

9) How About Dealing with the Cultural Differences?

* It’s Stephen’s Opinion That the Younger Generation Is Much More Liberal and Technology has Transcended the Cultural Borders Therefore Mitigating Those Issues.

10) In General:

* It Seems As Scrum Master Merges with Product Manager a Lot. This Is Something That I Was Aware of From Experience, But It Was Nice to Hear Stephen Express It Too.
* Stephen Had a Great Example of How to Use Current Tools to Manage a Crisis. A Remote Worker Said He Had a Power Outage. Stephen Used Google Earth and Local Search to Find Him an Internet Cafe to Go Work At.
* Carl Asked If It Would Be Possible to Use Second Life for Remote Interaction? I Kind of Like This Idea and May Look Into It.
* Richard Feels That He Can Handle Only a Maximum of 0.25 Interruptions Per Hour (2 Per 8hr Work Day).

***Problem in Successful Distribution: =>***

Distributed Projects Create Unusual Challenges Since the Project Team Members Are Not Collocated (Not Physically Together). As a Result, The Following Issues Can Become Impediments to the Success of a Project.

• Lack of Trust Between the Geographically Dispersed Teams.

• Unwieldy Amounts of Time Spent On Communication.

• Inability to Foster a “one-Team” Feeling Due to Cultural Differences.

• Lack of Participation From Team Members During Common Meetings.

• Lack of Identity with the Project Team, As Team Members in Different Geographies May Speak Different Languages And/or Have Different Project Practices.

These Stumbling Blocks Have Become Nightmares for Many Software Project Managers Facilitating Distributed Projects. Here Are Few To-Dos to Add to Your Communication Arsenal If You Are Assigned to Manage a Distributed Project.

• Find and Document the Overlap Time Between Different Geographically Distributed Teams (Don't Forget Daylight Savings Time).

• Publish the Instant Messaging (Im) Addresses of All the Team Members (And the Best Time to Reach Them).

• Make Sure That Each Key Stakeholder has Access to All Conference Call Access Details (Web and Telephone).

• Gather and Share the Vacation Details for Different Teams On a Shared Calendar.

• Publish a Schedule of Daily Stand-Up Meeting Between Geographically Dispersed Teams. Stand-Up Meetings Are Better Than Sit-Down Ones. Attendees Focus, Because No One Wants to Stand for a Long Time.

• Publish the Name and a Headshot (Photo) Of Each Team Member. Identify a Back-Up Contact Person for Each Key Role.

• Set Up a Common Location for Sharing Project Artifacts Between the Teams (Documents/reports/templates).

Besides Enhancing Your Communication Strategy, There Are Logistics Issues That Need to Be Addressed to Promote Superior Communication Among Distributed Teams.

• Invest in High-Quality Speakerphones for All Locations. When Participating in Conference Calls, Assurances of Stable Phone Connectivity Between Different Teams Will Go a Long Way to Building Camaraderie Between Those Teams Participating.

• Place the Phone in a Big Enough Room for the Team Members to Gather Comfortably for a Call. Equip the Room with a Large Table, As You May Want to Provide Food for Those Participants Meeting at Unusual Hours. Add White Boards, So Notes From Phone Conversations Can Be Jotted Down Quickly in a Way Everyone in the Room Can View.

• Budget Funds for a Few Team Members to Travel to Other Team Sites, Perhaps During the Initiation Phase or While Planning Quality Assurance Processes.

• Create a Project Dashboard (Use Any Collaboration Tool) For Teams to Communicate Their Issues. Share These Dashboard Images Between the Teams Whether They Use Online Tools, Or Only Have the Technology for Digital Photo Sharing.

• Publish the Overall Goals and Targets of the Project at a Common Location for Everyone to See, Even Telecommuting Team Members Working From Home.

• Arrange Presentations by Business Sponsors and Insist Key Teams Members From Every Location Participate in Hearing These Presentations.

As Virtual and Distributed Teams Become More Common, You Can Increase Your Chances of Success with Innovative Communication Techniques.

***References: =>***

* 1. Http://www.Google.Co.In/url?sa=t&Source=web&Cd=1&Ved=0cb8qfjaa&Url=http%3a%2f%2fwww.Controldraw.Co.Uk%2fpresentations%2flargeprojects.Ppt&Rct=j&Q=managing%20distributed%20projects&Ei=tzfctii8boe4vqok65jcca&Usg=afqjcnhqgm5k0izi7 Tkpjf4t2y4nk8isg&Cad=rja.
	2. Http://www.Soberit.Hut.Fi/t76.5633/2006/management%20of%20distributed%20projects/frontpage.Html.
	3. Http://www.Me2team.Com/moreinformation/papers/whitepapers/managingprojects/tabid/164/default.Aspx.
	4. Http://www.Projectperfect.Com.Au/info Managing Offshore Projects 1.Php.
	5. **Http://www.Google.Co.In/url?sa=t&Source=web&Cd=25&Ved=0cceqfjaeobq&Url=http%3a%2f%2fwww.Springerlink.Com%2findex%2f5ummu9yy9x0fcrp6.Pdf&Rct=j&Q=managing%20distributed%20projects&Ei=1jjctpzudoaevqpyn Tdca&Usg=afqjcngmlvbxk-Ozp6zxhkdkn6f2283pfw&Cad=rja**

**More Free Term Papers On Site: Www.Maturski.Org**