UNIVERSIDAD AUSTRAL DE CHILE
FACULTAD DE CIENCIAS DE LA INGENIERIA
ESCUELA DE INGENIERIA CIVIL EN INFORMATICA

Adaptation of a software development methodology incorporating Social Network Analysis and Situational Leadership.

Tesis para optar
al título de Ingeniero Civil
en Informática

PROFESOR PATROCINANTE: JUAN PABLO SALAZAR
PROFESOR CO-PATROCINANTE: ALEX HILLS

NOMBRE DEL ALUMNO: WESLEY DAVID MATTHEWS SCHELE

VALDIVIA - CHILE
AÑO 2005

Valdivia, 07 de marzo de 2005

De: Juan Pablo Salazar Fernández
Profesor Auxiliar
Instituto de Informática

A: Miguelina Vega R.
Directora
Escuela de Ingeniería Civil en Informática

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación "Adaptation of a software development methodology, incorporating social network analysis and situational leadership", presentado por el alumno Sr. Wesley David Matthews Schele, mi evaluación del mismo es la siguiente:

Nota: **6,9** (Seis coma nueve).

Fundamento de la nota:

El presente trabajo de titulación explora de manera original e intuitivamente correcta la relación existente entre las ciencias sociales y la ingeniería del software. Los elementos incorporados a la metodología Scrum permiten suponer una mejora costo-efectiva a la misma en las diferentes etapas de desarrollo de un proyecto de software, beneficiando tanto al equipo de desarrollo como a clientes y usuarios. Es valorable además que los tests realizados, aunque insuficientes para probar de manera científica las bondades de las mejoras, hayan sido realizados en situaciones reales.

Es importante también destacar el estudio e interés en utilizar metodologías ágiles de desarrollo de software ya que, aun cuando en el ámbito universitario pueda ser más atractivo el estudio de complejas metodologías y métricas, éstas tendrán una aplicación reducida y circunscrita a grandes empresas y proyectos. En proyectos de menor envergadura los factores humanos pueden llegar a tener más peso que los aspectos técnicos, elemento que ha sido correctamente manejado en este trabajo.

Adicionalmente, considero que un trabajo de titulación no debe ser evaluado solamente por el resultado sino por el proceso mediante el cual se desarrolla. En este caso, considero altamente valorable la definición y seguimiento de un plan detallado de actividades, el cual se cumplió prácticamente sin desviaciones.

Aspecto	Evaluación
Cumplimiento de objetivos	7.0
Satisfacción de alguna necesidad	7.0
Aplicación del método científico	6.5
Interpretación de los datos y obtención de conclusiones	6.5
Originalidad	7.0
Aplicación de criterios de análisis y diseño	7.0
Perspectivas del trabajo	7.0
Coherencia y rigurosidad lógica	7.0
Precisión del lenguaje técnico	7.0

Sin otro particular, saluda atentamente a usted,

Juan Pablo Salazar Fernández
Profesor Auxiliar
Instituto de Informática

Universidad Austral de Chile

Instituto de Informática

Valdivia, marzo 04 de 2005
Comunicación Interna Nº 024/05

De: Sr. Alex Hills, Profesor Huésped
Instituto de Informática
A : Sra. Miguelina Vega Rosales, Directora
Escuela de Ingeniería Civil en Informática

Motivo: Informar Calificación Trabajo de Titulación

Nombre Trabajo de Titulación:
"ADAPTATION OF A SOFTWARE DEVELOPMENT METHODOLOGY INCORPORATING SOCIAL NETWORK ANALYSIS AND SITUATIONAL LEADERSHIP".

Nombre Alumno :
Wesley David Matthews Schele

Evaluación:

Cumplimiento	7.0
Satisfacción de alguna necesidad	7.0
Aplicación del método científico	6.6
Interpretación de los datos y obtención de conlusiones	7.0
Originalidad	7.0
Aplicación de criterios de análisis y diseño	6.4
Perspectivas del trabajo	7.0
Cohęrencia y rigurosidad lógica	6.8
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración	6.4
Nota Final	6.8

Sin otro particular, le saluda atentamente,

Alex Hilss
Profesor Huésped

Dirección : General Lagos 2086 – Campus Miraflores- Valdivia – Chile

Fono: 56 63 221427 · Fax: 56 63 293115 · email: instituto@inf.uach.cl

Valdivia, 07 de Marzo de 2005

De : Martín Gonzalo Solar Monsalves

A : Directora Escuela Ingeniería Civil en Informática

Ref. : Informe Calificación Trabajo de Titulación

Nombre Trabajo de Titulación:

"ADAPTATION OF A SOFTWARE DEVELOPMENT METHODOLOGY INCORPORATING SOCIAL NETWORK ANALYSIS AND SITUATIONAL LEADERSHIP"

Nombre Alumno:

Wesley David Matthews Schele.

Evaluación:

Cumplimiento del objetivo propuesto	7.0
Satisfacción de alguna necesidad	6.8
Aplicación del método científico	7.0
Interpretación de los datos y obtención de conclusiones	7.0
Originalidad	7.0
Aplicación de criterios de análisis y diseño	7.0
Perspectivas del trabajo	6.8
Coherencia y rigurosidad lógica	7.0
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración	7.0
Nota Final	**7.0**

Sin otro particular, atte.:

Martín Solar Monsalves

DEDICATORIA

Español

Para Vivi por todo.

Para mis padres por mi vida.

Para Owen y Erin por mi amor.

Para Keith por creer.

Para Juan Pablo, Alex y Jean Paul por guiar.

Para mi familia por estar ahí.

Para mis amigos por ser ustedes.

English

To Vivi for everything.

To My Parents for my life.

To Owen and Erin for my love.

To Keith for believing.

To Juan Pablo, Alex and Jean Paul for guiding.

To my family for being there.

To my friends for being you.

INDEX

1 ILLUSTRATIONS TABLE

1.1 Tables

1.2 Graphs

1.3 Illustrations

2 SÍNTESIS

Este trabajo es un estudio de metodologías de desarrollo de software y de como pueden ser optimizadas mediante la incorporación de métodos analíticos para el diagnostico de situaciones. Al incorporar SNA (Análisis de Redes Sociales) y Liderazgo Situacional, trata de resolver problemas comunes a todas las metodologías.

El trabajo primero establece la situación general de del desarrollo de software y luego analiza las metodologías mas populares disponibles. De la lista que se discute se selecciona SCRUM para su modificación. Luego de analizar SNA y el liderazgo situacional, se propone una modificación a la metodología.

La metodología se descompone en tres etapas, definición de requerimientos, desarrollo e implantación. Luego la metodología es modificada utilizando técnicas de SNA y se propone un modelo para un método cuantitativo para la evaluación de los niveles de preparación de liderazgo situacional. La nueva metodología modificada se denomina Social Scrum o S-Scrum. A continuación la metodología se prueba en un proyecto real. Finalmente se discuten las conclusiones referentes a la nueva metodología y a los modelos analíticos propuestos.

La metodología resultante tiene por objetivo ser de utilidad como una herramienta de evaluación y recomendación para jefes de proyecto y arquitectos de solución. Al proveer información adicional y métodos de control en varios puntos clave en la metodología de desarrollo seleccionada, se busca un mejor proceso, y una posible ruta a encontrarlo queda propuesta.

3 ABSTRACT

This work is a study of software development methodologies and how they can be improved by incorporating analytical methods to diagnose situations. By incorporating SNA (Social Network Analysis) and Situational Leadership, it tries to solve common problems to all methodologies.

The work first states the general situation of software development, and then analyzes the most popular methodologies available. Out of the list that is discussed SCRUM is selected for modification. After analyzing SNA and situational leadership, a methodology modification is proposed. The methodology is broken down into three stages; requirement definition, development and deployment. The methodology is then modified using SNA techniques and a model for a quantitative method for evaluating situational leadership readiness levels is proposed. The new modified methodology is called Social Scrum or S-Scrum. Next the methodology is tested within a real project. Finally conclusions about the new methodology and the analytical models that are proposed are discussed.

The resulting methodology is intended to be of use as an evaluation and recommendation tool for project managers and solution architects. By providing additional information and control methods at several key point in the selected software development methodology a better process is sought, and a possible route to finding it is suggested.

4 INTRODUCTION

4.1 General Information about the problem to be approached

One problem that haunts every software development project is cost control. Any professional that has been in charge of a software development project of any size is familiar with the problems of estimating the effort required to develop a solution. One of the main problems, especially in Latin America, is that clients rarely are willing to pay for a detailed design for their software before building commences. This is one of the core problems of software development (Others are: incorrect specifications, changing requirements, incorrect effort estimation, etc). Software, especially custom applications, is not considered an engineering project as much as an implementation of the customer's requirements. Most projects require cost estimation based on a small set of general requirements and include final requirement definition, project design and documentation as part of the construction project. [1]

Would anyone consider asking an architect what the cost of a house will be depending only on the number and type of rooms to be included? This problem appears especially strongly in ad hoc software solutions and normally is more noticeable in small client corporations. Because of the high costs of design and requirement definition, customers would rather run the risk of not having these items complete at the beginning of the project. When this happens, they may have to compromise some functionality because unanticipated requirements often cannot be correctly integrated in the project later on.

Since this problem cannot be solved without a culture change brought on by customer education, and, because making that possible would require extensive coordination among competing companies, software development professionals usually use a "workaround."

Most software development methodologies evolved to be able to assess the costs effectively based on a minimal definition of the project. The problem is that, to be able to assess cost effectively you need the project to be static, requirements to be defined at the beginning of the project and to stay as they were originally specified. What usually really happens is that the

more customers see of the project, the more they come to understand their own problems and the solutions that are offered, and the more they will modify the project. In other words, requirements are “volatile”.

Another problem with software development is assessing the cost of implementing a system within an organization. Usually the implementation cost of an Enterprise Resource Planning solution, or ERP[1], solution is approximately twice as much as the licensing fee. These problems are reasons that most software development projects have difficulty meeting deadlines and staying within budget. [1, 2, 3]

Most software development methodologies center on four critical points:

1. Quality (Clean and bug free code)
2. Cost control
3. Maintainability
4. Correct requirement implementation

In considering a design process it is logical to include as much information as possible that is relevant to the problem to be solved. An architect will talk to his clients to decide every aspect of the house and design it in accord to the clients’ way of living, expectations and intentions. This is also true for most engineering processes, but, in software design, probably owing to its relative immaturity and since software development has only evolved in recent decades, this is not the case. Compared to older design processes, the software design process has evolved as a loose collection of best practices (in the case of the lighter methodologies) to a structured and rigid form that is not flexible enough to satisfy the volatile requirements usually associated with the larger projects. This is especially true in systems that are meant for large numbers of users who will have different roles associated to their interfaces to the system, leading to a diversity of interfaces.

[1] ERP: Enterprise Resource Planning. An information system that integrates all manufacturing and related applications for an entire enterprise.

Now let's define the term "architecture" in relation to software. The best definition we have found so far is the one by Ralph Johnson [1]:

> In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers. [1]

Regarding this definition we ask: To achieve consensus among members of the design group, shouldn't there be complete information regarding the entire environment the design is going to be applied to? If an engineer is designing a structure, should he not consider the environment (climate, humidity, temperatures, etc.) that might affect it? What climate does software work in? Software works in the most temperamental, volatile, changing climate of all: organizations.

4.2 The human factor.

The only way to deal with our environment is to understand it. In an environment that may be volatile, changing and unstable, the cause of the instability is clear: people.

In order to ascertain procedures that will work in organizations, the key component of all organizations must be studied: people. Most software development methodologies consider only superficially what is happening within the client's organization. This may be a fatal flaw. The only element able to affect the outcome of a project is the development team: the team leader, clients and management. So why not focus on them during the development process?

This study centers on modifying a software development methodology (SDM) so that it includes the elements that have been described so that they can be measured and controlled. Once a better understanding of how development teams work and interact is established, it may be possible to aid development teams in becoming more efficient, cohesive units. The objective of this work is to modify an existing software development methodology to incorporate elements

that will consider human interactions as part of the development process. The modifications will require some additional information gathering and analysis during the project.

During this work Social Network Analysis (SNA) will be used to study communication patterns within organizations and development teams. This is a very powerful tool and is used frequently in consulting firms that study organizational behavior. [3, 15, 16]

5 OBJECTIVES AND RELEVANCE OF THE STUDY

5.1 General Objectives

1. Generate an improvement to a specific software development methodology (SDM) that allows a better definition of the environment of the design process, better control and earlier warning of critical situations during development and a smoother and more cost-effective implementation process for both developers and clients.

2. Through a better and more informed design process, obtain systems that are more in accord with an organization's work method, thus enabling better adoption of the new system and a longer system life span.

5.2 Specific Objectives

1. Select an adequate methodology for the incorporation of Social Network Analysis (SNA) into the different stages of development.
2. Structure the SNA process to optimize it for incorporation into the SDM.
3. Modify the SDM to incorporate SNA
4. Evaluate its efficiency through testing
5. Analyze the test results and document the entire new SDM.

5.3 Relevance of the study

Although much has been said about software development methodologies, many of the problems they try to resolve continue to exist. Our view of the problem is that although software development can be viewed as a process, process control methodologies (methodologies used in industry to control production processes) are not applicable. This is because a controllable process must have the following characteristics [5]:

- The process must be repeatable

- The process must be controllable from established key points that can be determined beforehand.
- The process works within an established environment.
- The expected results must be known beforehand.

Most of these characteristics do not apply to software development. Let's review them in order:

- The process must be repeatable

Almost every software development project is completely different from previous projects. If we need a copy of earlier projects, we can simply copy the results digitally, and there is no necessity to develop the product anew. Thus every project will incorporate tasks that others did not.

- The process must be controllable from established control points (points of the process that are significant or that generate information relevant to controlling the process), which can be determined beforehand.

Since every project is different, control points don't make sense since it is impossible to know the exact process to follow. The process must be controlled as a whole, with frequent inspections on every level if it is to be controlled effectively.

- The process works within an established environment.

The process can work within an environment, but since we are speaking of people that develop software, not machines, the environment is highly volatile.

- The expected results must always be known beforehand.

In most software development project the result is known at the beginning only vaguely, a mere statement of intentions. The process in itself will define the final product.

Since process-oriented control methodology is not applicable to most software development, we must look for controls that can be used to gather information about what is happening within a project and make informed decisions regarding its development.

Our study also considers the need to analyze issues related to human resources that the client and provider companies have allocated to the project. SNA until now has not been applied to software development as a tool for controlling the process. We know enough not to use SNA to try to find a way to make the process more standard or establish a more appropriate process. Rather, our intentions are to try to provide more information regarding human interactions within the project and insight on how to use that information to the benefit of the project. [2, 5, 15]

6 AVAILABLE SOLUTIONS REVIEW AND BASIC CONCEPTS

6.1 Existing information regarding the problem in question

This work concerns two main areas, software development methodologies (SDM) and analysis of human interactions during the process. For the latter, social network analysis (SNA) will be used in conjunction with the study of appropriate leadership for software development and how to achieve it.

Both of these areas have been extensively described in the literature . First we discuss SDM, and then we discuss SNA.

6.2 Software Development Methodologies (SDM)

Software development methodologies began to flourish in the early 80's when software projects grew in scale and became longer, requiring the projects to be maintained and perfected over time.

According to Martin Fowler;

> Methodologies impose a disciplined process upon software development with the aim of making software development more predictable and more efficient. They do this by developing a detailed process with a strong emphasis on planning inspired by other engineering disciplines - which is why we tend to refer to them as **engineering methodologies**." [4]

Engineering or "heavyweight" software development methodologies define a structured approach to software development, normally using several phases with defined inputs and expected outputs.

Although engineering methodologies have developed substantially over time, they are not very successful for software development, except in non-volatile environments, and they are

extremely unpopular due to the additional work overhead they require. When such a methodology is used, the additional work usually slows the development pace.

As a solution and also a standardization of best practices, lightweight methodologies have surfaced. Now they are known as “agile methodologies” given their speed and adaptability to different types of projects and environments.

Agile methodologies center on developing products as fast as possible, adapting as the environment may require, and welcoming change. They incorporate the client into the development process and use constant feedback to be responsive and adaptable.

The main differences between the two types of methodologies are [6]:

- **Agile methods are adaptive rather than predictive**. Engineering methods tend to try to plan out a large part of the software process in great detail for a long span of time. This works well until things change. Their nature is to resist change. The agile methods, however, welcome change. They try to be processes that adapt and thrive on change, even to the point of changing themselves.

- **Agile methods are people-oriented rather than process-oriented**. The goal of engineering methods is to define a process that will work well whoever happens to be using it. Agile methods assert that no process will ever make up the skill of the development team, so the role of a process is to support the development team in their work. [6]:

The following table describes the main differences between the two types of methodologies; this table describes the consensus[2]. Practitioners do not agree on all the points, especially project size.

	Agile Methods	Heavy Methods
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project Size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to Change	Change Adaptability	Change Sustainability
Culture	Leadership-Collaboration	Command-Control
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Team Size	Small/Creative	Large
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in the project	End of the project

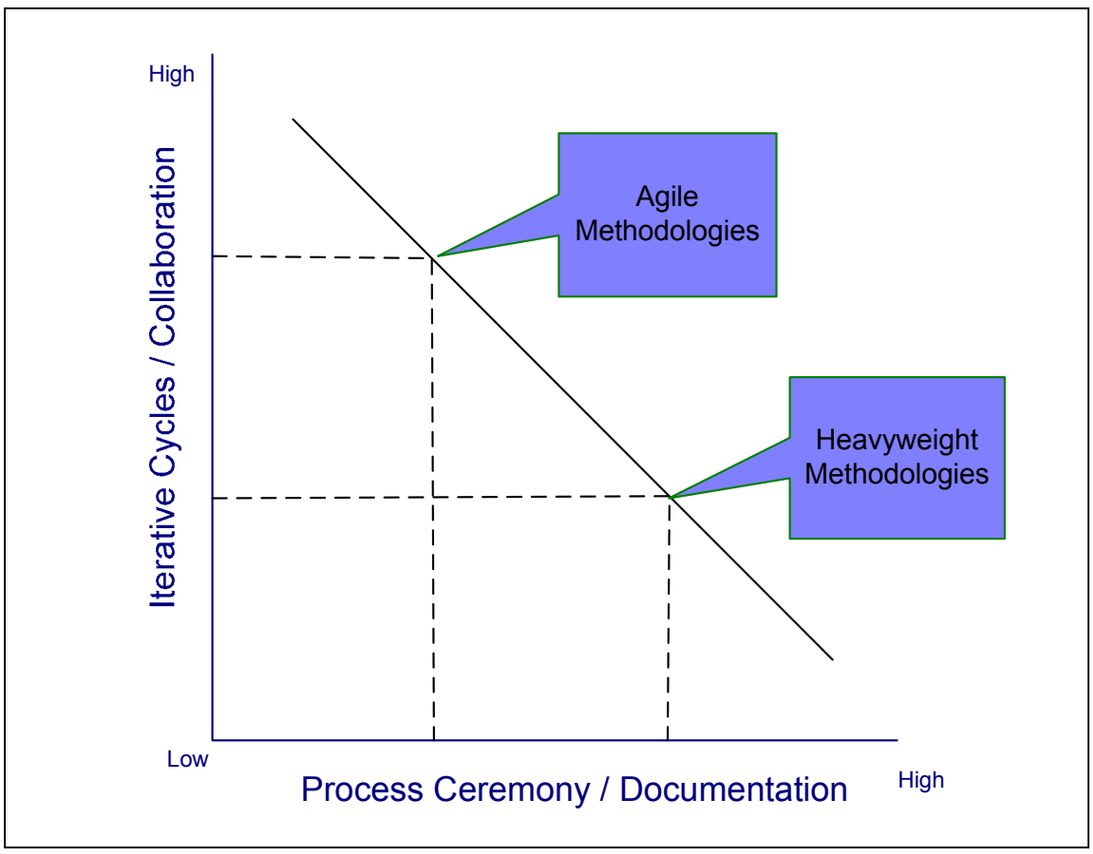
Table 1 Agile vs. Heavy Methodologies [6]

The following table shows the applicability for each kind of methodology --the types of project for which each is best suited.

	Indicator for Agile Methods	Indicators for Heavyweight Methods
Objective	Rapid Value	High Assurance
Scope (requirements)	Subject to change Largely emergent Unknown, Uncertain	Well Known Largely Stable
Resources (money, infrastructure)	Uncertain budget Money tight	Sufficient Budget
Time	Unclear & Not well Defined Milestones	Clear & Defined Milestones
Risks	Unknown risks Major Impact New Technology	Well understood risks Minor Impact
Architecture	Design for current needs	Design for current and future needs
Developers	Agile, co-located, collaborative	Process-oriented, Adequately Skillful
Customers	Collaborative, dedicated, co-located, knowledgeable	Knowledgeable, representative, collaborative
Refactoring/Cost of Change	Inexpensive	Expensive

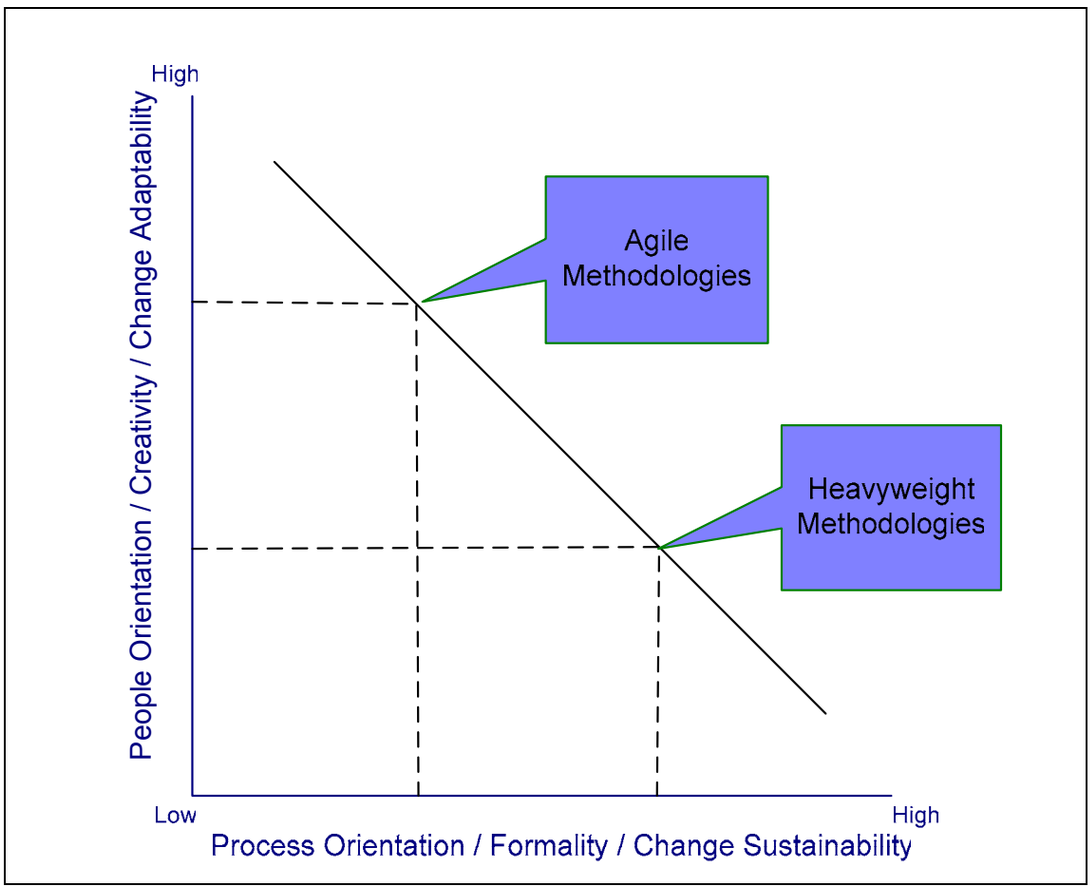
Table 2 Methodology Applicability [6]

The following graph illustrates the characteristics of the two types of methodologies regarding collaboration and iteration costs vs. documenting more thoroughly and maintaining a stricter formality during the development process. The graph illustrates one of the main differences between the methodologies. Specifically, agile methodologies rely on many iterations to achieve a goal, and heavy methodologies depend upon documentation and written definitions to be able to establish the scope of a project correctly. Engineering methodologies must be able to do this to try to stay within budget. Agile methodologies try to stay in budget by reacting with measured response to changes in the definitions throughout the project.



Graph 1 Ceremony/Documentation vs. Cycles/Collaboration [6]

The last graph in this section illustrates that an agile methodology can handle volatile requirements better -- but at the cost of maintainability. This means that the more adaptive to change a methodology is, the less documentation and structure it allows. Thus, a system that was developed by one team may be harder for another team to maintain. [6, 8, 12]



Graph 2 Methodology Orientation Comparison [6]

In this project both types of methodologies were analyzed to determine which is more suitable for the intended modifications. Even though at first glance it would appear that the more flexible ones would allow a better integration of new processes, all options are considered. Some of the methodologies in each category are:

Heavyweight Methodologies

- UML

- SDLC

Agile Methodologies

- TSP (Team Software Process)
- XP (Extreme Programming)
- Cockburn's Crystal Family
- Open Source
- Highsmith's Adaptive Software Development
- Scrum
- Feature Driven Development
- DSDM (Dynamic System Development Method)

Here we analyze a small subset of these, corresponding to the ones that are most used. An objective of this Thesis is that the proposed methodology changes might be adopted to some extent by software development companies, and this can only be achieved by using one of the mainstream methodologies.

6.3 Additional Information about Software Development Methodologies

6.3.1 Heavyweight Methodologies

Most project management methodologies have evolved based on principles derived from traditional process control methodologies. Process control methodologies can be mostly categorized into two models: “Defined” and “Empirical” [5].

The defined process control model requires that every piece of work be defined and understood. Given a series of concrete inputs, the output is generated using the same process every time. The process can be controlled at several established control points based on the assumption that the process is repeatable and predictable.

The empirical model provides and exercises control based on frequent inspections and adaptation for processes that are imperfectly defined and generate unpredictable and unrepeatable outputs.

Heavyweight methodologies are based on the “defined” model. They structure the stages of the software development process and suppose that in general the process has a beginning, after which it moves through defined stages to a known conclusion and results. They are based on the typical waterfall life cycle and, once begun, advance forward through all different stages producing tangible results late in the process. They rely heavily on design and documentation for the definition of the final product. [5, 7]

6.3.2 Agile Methodologies

Agile methodologies are based on what Babatunde Ogunnaike [5] calls “empirical” process control models. These models try to expect the unexpected. Based on these considerations, agile software development methodologies were born. They were defined from best practices found in successful teams and companies, and they have evolved to augment team cohesion, wellbeing and customer satisfaction.

Several of the leading professionals in this field have joined forces to improve these methodologies and compare their observations. The resulting organization is called the Agile Alliance, and they have defined the principle behind agile software development in their manifesto

Agile methodologies can be described as methodologies that focus on delivering as much functionality as possible, within the least time, and maintaining communications among all participants to ensure that the project will remain on track. They rely heavily on prototyping and frequent releases. All agile methodologies consider that the main drivers in successful software development are the people involved in the process and not the process itself. [5, 8, 11, 12]

6.3.2.1 The Agile Manifesto

A group of individuals all working on agile software development methodologies, has come together to define what an agile methodology should be and what goals it should pursue. They have established the key principles in their manifesto:

> We are uncovering better ways of developing software by doing it and helping others do it.
>
> Through this work we have come to value:
>
> Individuals and interactions over processes and tools
>
> Working software over comprehensive documentation
>
> Customer collaboration over contract negotiation
>
> Responding to change over following a plan
>
> That is, while there is value in the items on the right,
>
> we value the items on the left [3]

These are the basic principles behind every agile methodology; the basic notion is that the focus of any project should be about better results end better relations instead of the process itself.

[3] Source: http://agilemanifesto.org/

6.4 Social Network Analysis (SNA)

Social network analysis (SNA) is both a theoretical perspective and a set of methods. In terms of theory, SNA extends and complements traditional social science by focusing on the causes and consequences of relations between and among sets of people rather than on the characteristics of individuals. Methodologically, SNA focuses on the measurement of relationships between people. By quantifying the relationships between people, social network analysts can apply models and techniques that are commonly used in the social and natural sciences.

Two distinct approaches to SNA arise from two distinct historical traditions. The sociocentric (whole) network approach comes from sociology and was heavily influenced by the work of Georg Simmel. Sociocentric network analysis involves the quantification of relationships between people within a defined group – a classroom of children, a board of directors, the residents of a village or town, the trading partners in a bloc of nations. By representing relationships as numbers, powerful mathematical and statistical techniques can be applied.

Sociocentric network analysis begins with the assumption that members of a group interact more than would a randomly selected group of similar size. In other words, people that know each other interact more than strangers put together randomly. The focus is on measuring the structural patterns of those interactions and how those patterns explain outcomes, like the concentration of power or other resources, within the group. Sociocentric network analysts are interested in identifying structural patterns that can be generalized, and in this sense they are like physicists or economists who are interested in modeling behavior. The egocentric (personal) network approach arose from anthropology and traces its roots to A. R. Radcliffe-Brown, among others. This form of SNA is almost always about people rather than about groups. An egocentric network comprises the people (what social network experts call “alters”) that a person (referred to as “ego”) knows. An egocentric network thus may have, as its members, spouses, children, cousins, co-workers, church members, book club members, or friends. So the personal network of an elementary school teacher may contain her husband, her son and daughter, all of their

friends and relatives, her own friends and relatives, her coworkers, students, parents of students and members of her church. But she may have more family relations than, for example, the CEO of a large company who has less time to maintain those relationships.

Egocentric SNA is concerned with making generalizations about the features of personal networks that explain things like longevity, consumer and voting behavior, coping with difficult life situations, economic success or failure, and so on. With its focus on individuals, the egocentric network approach has been more germane to studies of community than the sociocentric network approach. It is also possible to treat organizations, classrooms, communities or even nations as the “ego” in an egocentric network study.

For this work, and for some other types of problems, the sociocentric is appropriate. The metrics that will be used here have been created in sociocentric studies.[7, 14, 15]

Even though SNA has evolved substantially and there are several possible ways to approach SNA, we will take one in particular that seems most suitable. A particular concise definition of this approach is the one by Valdis Krebs:

> Social network analysis [SNA] is the mapping and measuring of relationships and flows between people, groups, organizations, computers or other information/knowledge processing entities. The nodes in the network are the people and groups, while the links show relationships or flows between the nodes. SNA provides both a visual and a mathematical analysis of human relationships. Management consultants use this methodology with their business clients and call it Organizational Network Analysis [ONA].[7]

A method to understand networks and their participants is to evaluate the location of actors in the network. One example is finding the centrality of a node. This measure can help determine the importance, or prominence, of a node in the network. Network location can be different than location in the hierarchy, or organizational chart.

When considering the analysis of an organization, one must be aware of which are the relevant aspects of a social network. Next we will describe some basic elements.

a. Network Measures[7]

Consider the following example to understand the different metrics used in SNA: This graph was created using UCINET, it shows several nodes (people in the network) that are communicating with each other. Some do not communicate, but those who do are connected by an edge.

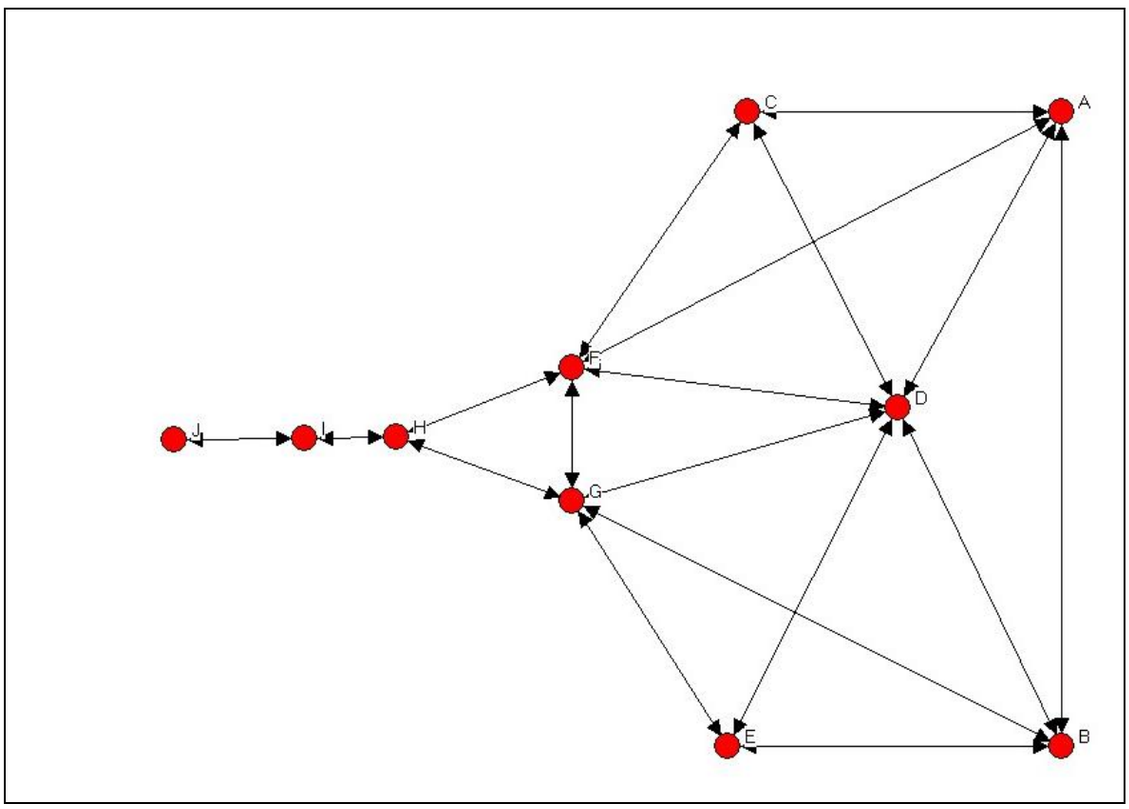


Illustration 1 Sample SNA Network[7]

Degrees

Social network researchers measure network activity for a node by using the concept of degrees -- the number of direct connections a node has. In the network above, “D” has the most direct connections in the network, making hers the most active node in the network. She is a 'connector' or 'hub' in this network. Common wisdom in personal networks is "the more connections, the better." This is not always so. What really matters is where those connections lead to -- and how they

connect the otherwise unconnected! Here “D” has connections only to others in her immediate cluster -- her clique. She connects only those who are already connected to each other.

Betweenness

While “D” has many direct ties, “H” has few direct connections -- fewer than the average in the network. Yet, in many ways, she has one of the best locations in the network -- she is *between* two important constituencies. She plays a 'broker' role in the network. She plays a powerful role in the network but that she is a single point of failure. Without her, “I” and “J” would be cut off from information and knowledge in “D's” cluster. A node with high betweenness has great influence over what flows in the network.

Closeness

“F” and “G” have fewer connections than “D;” yet the pattern of their direct and indirect ties allows them to access all the nodes in the network more quickly than anyone else. They have the shortest paths to all others, i.e., they are close to others. They are in an excellent position to monitor the information flow in the network -- they have the best visibility into what is happening in the network.

Boundary Spanners

Nodes that connect some groups to others usually end up with high network metrics, e.g., betweenness and closeness. Boundary spanners such as “F”, “G”, and “H” are more central than their immediate neighbors, whose connections are only local, within their immediate cluster. Boundary spanners are well-positioned to be innovators, since they have access to ideas and information flowing in other clusters. They are in a position to combine different ideas and knowledge, found in various places, into new products and services.

Peripheral Players

Most people would view the nodes on the periphery of a network as *not* being very important. In fact, “I” and “J” receive very low centrality scores for *this* network, e.g., betweenness and closeness. Yet, peripheral nodes may be connected to networks that are not currently mapped. “I” and “J” may be contractors or vendors that have their own network outside of the company -- making them very important resources for fresh information not available inside the company!

Network Centralization

Individual network centralities provide insight into the individual's location in the network. The relationship between the centralities of all nodes can reveal much about the overall network structure. A very centralized network is dominated by one or a few very central nodes. If these nodes are removed or damaged, the network quickly fragments into unconnected sub-networks. A highly central node can become a single point of failure. A network centralized around a well connected hub can fail abruptly if that hub is disabled or removed. A less centralized network has no single points of failure. It is resilient in the face of many intentional attacks or random failures -- many nodes or links can fail while allowing the remaining nodes to still reach each other over other network paths. Networks of low centralization fail gracefully.

Other Network Metrics

1. Structural Equivalence - determine which nodes play similar roles in the network
2. Cluster Analysis - find cliques and other densely connected clusters
3. Structural Holes - find areas of no connection between nodes that could be used for advantage or opportunity
4. E/I Ratio - find which groups in the network are open or closed to others
5. Small Worlds - find node clustering, and short path lengths, that are common in networks exhibiting highly efficient small-world behavior

<u>Acquiring information to build the network</u>

While traditional SNA relies heavily on expensive surveys and interviews for data, a mass of inexpensive data tied to relationships has become available in recent years. Credit card transactions, cell phone calls, global positioning systems (GPS) data, Web site access, e-mail, Internet messaging, ATM transactions, border control, electronic toll payment and supermarket “loyalty” cards are just some of the data points that are being captured. Analysis of this information can help reveal the patterns of relationships.

6.4.1 Metric calculation

Next we will explain how to calculate the metrics that will be used in our study.

To explain metric calculation, we use the following sample network:

Example Network

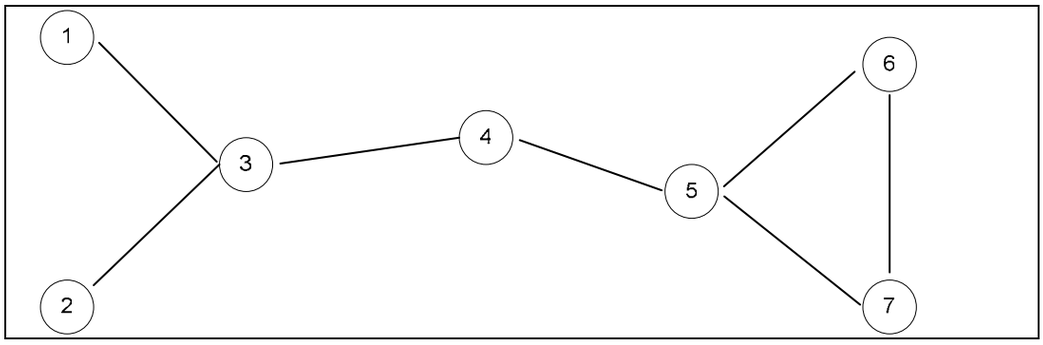


Illustration 2 Sample Network for metric calculation explanation

Closeness[18]

Definition: total distance of given node from all others

The centrality of a point can be measured by summing the geodesic distances from that point to all other points in the graph. Actually, this is a measure of point decentrality or inverse centrality since it grows as points are far apart, and centrality in this context means closeness. If we let

$d(p_i, p_k)$ = the number of edges in the geodesic linking p_i and p_k then a measure of the decentrality of a point p_k is

$$C_c(p_k)^{-1} = \sum_{i=1}^{n} d(p_i, p_k)$$

This measure is dependent upon the number of points in the network from which it is calculated. We cannot, therefore, compare values of $(C_c(p_k))^{-1}$ for points drawn from graphs of different sizes. So again it would be useful to have a measure from which the impact of graph size was removed.

Beauchamp (1965) has solved this problem. He suggested that the relative point centrality of a point p_k be defined as

$$C_c(p_k)^{-1} = \left[\frac{\sum_{i=1}^{n} d(p_i, p_k)}{n-1} \right]^{-1}$$

$$= \frac{n-1}{\sum_{i=1}^{n} d(p_i, p_k)}$$

In our example:

Closeness		
Node	Score	Standardized Score
1	1/16	6/16 = 3/8
2	1/16	6/16 = 3/8
3	1/11	6/11
4	1/10	6/10 = 3/5
5	1/11	6/11
6	1/15	6/15 = 2/5
7	1/15	6/15 = 2/5

Betweenness[18]

Partial betweenness can be defined in terms of probabilities, if we assume two points p_i and p_j are indifferent with respect to which of several alternative geodesics carries their communications, the probability of using any one is

$$\frac{1}{g_{ij}}$$

Where g_{ij} = the number of geodesic paths from *i* to *j*

The potential of point p_k for control of information passing between p_i and p_i then may be defined as the probability that p_k falls on a randomly selected geodesic connecting p_i and p_j

$g_{ij}(p_k)$ = the number of geodesics linking pi and p_j that contain p_k

Then

$$b_{ij}(p_k) = \frac{1}{g_{ij}} \times g_{ij}(p_k)$$

$$= \frac{g_{ij}(p_k)}{g_{ij}}$$

is the probability we seek; it is the probability that point p_k falls on a randomly selected geodesic linking pi with p_j.

To determine the overall centrality of a point p_k, we sum its partial betweenness values for all unordered pairs of points where i<> j<>k:

$$C_B(p_k) = \sum_{i=1(i\neq k)}^{n} \sum_{j=1(j\neq k)}^{n} b_{ij}(p_k)$$

Where n is the number of points in the graph.

Freeman (1977) proved that the maximum value taken by $C_B(p_k)$ is achieved only by the central point in a star. It is

$$\frac{(n-1)(n-2)}{2}$$

Therefore, the relative centrality of any point in a graph may be expressed as a ratio,

$$C_B'(p_k) = \frac{2C_B(p_k)}{(n-1)(n-2)}$$

For this network, (7-1)(7-2)/2 = 15. Note that node 5 has a little smaller centrality score that node 3 and 4 because the connection between node 6 and 7 reduces the controllability of node 5.

In our example:

Betweenness		
Node	Score	Standardized Score
1	0	0
2	0	0
3	16/3	16/45
4	13/3	13/45
5	13/3	13/45
6	0	0
7	0	0

Cutpoints

Another important consideration is whether a user is an information bridge between other users that otherwise would be disconnected from the network.

One example is two groups that are connected by only one user..

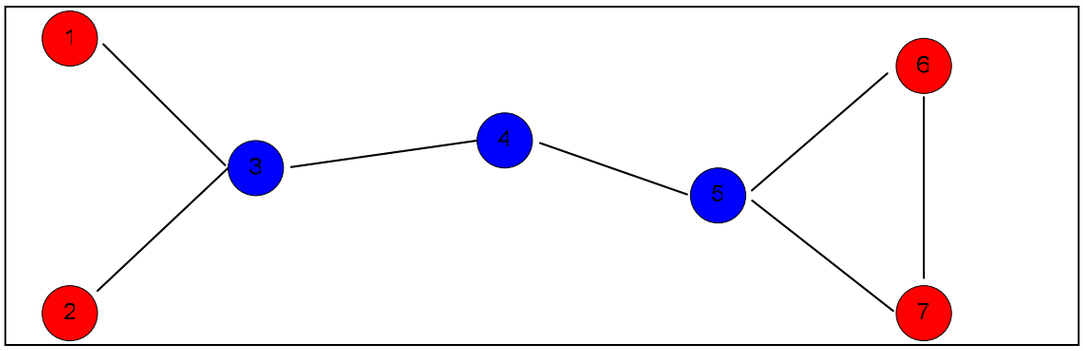


Illustration 3 Network Cutpoint Example

As we can see in Illustration 3, nodes 3, 4 and 5 are cutpoints for the network. If any of these were missing we would have a disconnected network.

6.4.2 Description of the SNA Analysis Process

All SNA analysis processes are based on two stages:

1. Information collection
2. Information analysis

The first of the two stages can be achieved in several ways; some of the most common are:

1. Direct surveys
2. Phone record recollection
3. Mail server log analysis
4. Workflow systems analysis

Undoubtedly there are many other ways to determine the amount of communication within an organization, and any way that can be used systematically is a valid means of compiling information. Normally, researchers must use acceptable methods. For example if a company is not willing to have someone go through a mail server log, then other methods must be used. Another example: some companies might not have available phone call detail recording (CDR) records. Information collection is done by a combination of available methods.

The second stage, information analysis, is normally done using software. There are several products on the market. Some are commercial, and others are free. The main differences between the products are the clarity of the user interface, the quality of the graphs produced, and the number of filters and analysis tools available. We will compare some of the most common products available.

6.5 Basic Situational Leadership concepts

One concern is that most leaders of software development teams started as software developers themselves but have very little experience or training in effective leadership. A simple solution is to provide the basic tools so that the team leaders of software development companies can identify the type of team they are working with and use the appropriate type of leadership as described in the Situational Leadership Model.

Ken Blanchard,, and Paul Hersey [3] created a model for “situational leadership” in the late 1960's. It allows one to analyze the needs of the situation one is dealing with and then adopt the most appropriate leadership style. It has been popular with managers over the years because it is simple to understand, and it works in most environments for most people.

Situational Leadership is based on two basic concepts.

1. Not all teams are alike, and all go through various stages as they evolve into more independent and efficient work forces. This can be observed in the team readiness level chart.

Readiness Level	Ability	Willingness
R1	Unable	Unwilling or unconfident
R2	Unable	Willing and confident
R3	Able	Unwilling or unconfident
R4	Able	Willing and confident

2. Each individual team requires different styles of leadership depending on their determined Readiness Level. We can observe the different styles of Leadership in the chart on the following page

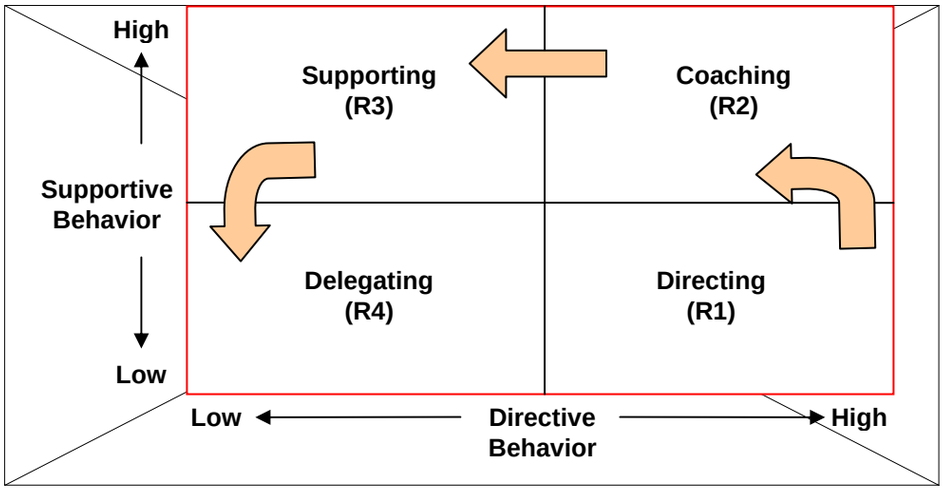


Illustration 4 Situational Leadership Readiness vs. Direction Model [3]

Once the appropriate readiness level has been determined, the appropriate leadership style can be applied. Descriptions for the four leadership styles are:

Directing Leaders define the roles and tasks of the 'follower', and supervise them closely. Decisions are made by the leader and announced, so communication is largely one-way.

Coaching Leaders still define roles and tasks, but seeks ideas and suggestions from the follower. Decisions remain the leader's prerogative, but communication is much more two-way.

Supporting Leaders pass day-to-day decisions, such as task allocation and processes, to the follower. The leader facilitates and takes part in decisions, but control is with the follower.

Delegating Leaders are still involved in decisions and problem-solving, but control is with the follower. The follower decides when and how the leader will be involved. [3]

7 PROJECT DEVELOPMENT

7.1 Part 1. Evaluation, Selection and Justification of Software Development Methodology to be used

During this stage of the project several software development methodologies will be evaluated to determine which is more suitable to the project's requirements. The evaluation will be based on how useful the methodology is considered to be and if it is possible to modify it to incorporate the intended improvements. Once one is selected, we will discuss the modifications that were introduced into the methodology.

In considering the various software development methodologies one of our main questions was: Since any of these methodologies can benefit from incorporation of SNA, is it possible to generate a separate additional process that can run parallel to a software development methodology and allow it to benefit from the use of SNA? We decided that, although that idea seems reasonable, it is beyond our reach and experience at the time. Even though the difficulty of approaching the project in this manner was evident, we continued to see ways that it would be beneficial. This consideration was one of the determining factors in the final SDM selection.

The selection process needed to consider these aspects for our project:

- The structure that the methodology uses should not be rigid. It should allow modification without major impact on established users.
- The methodology should be widespread.
- The methodology should be viable for any size project, not limited to small or large projects.

These three considerations alone ruled out any of the heavyweight methodologies because:

- Heavyweight methodologies are high in ceremony and protocol. They require heavy documentation and set control points, therefore they are not easily modifiable and if

modified would imply a considerable delay while development teams learned the process anew.

- They are limited to larger projects because of the high cost overhead they impose on any project that uses them.
- Since heavyweight methodologies are complicated, and the learning process is tedious, any modifications to the methodology wouldn't be easily accepted by current users.

Even though both types of methodologies have strengths, we believe that agile methodologies, with the improved information that will be available through the modifications, can be used for any project size. On the other hand, a modified heavyweight methodology probably wouldn't be cost effective for small projects

These considerations left us only with agile methodologies as an option.

7.1.1 Evaluation of Methodologies

As previously mentioned, the project needs the use of an agile methodology. Next several of the most popular methodologies will be described [9,11,12] and evaluated to determine which one will finally be used for the project.

Extreme Programming (XP)

Characteristics

- Rapid Advancement
- 1-4 week iterations
- User stories
- On-site customer is required
- Heavy emphasis on testing
- Do the simplest thing possible
- You Aren't Gonna Need It (YAGNI)

XP is best suited for

- loosely-defined or volatile requirements
- Team has or can develop strong engineering skills and practices
- Customers can be involved on a daily (hourly) basis

Scrum

Characteristics

- 30-day iterations called "sprints"
- Self-organizing teams
- No specific engineering practices prescribed (but many Scrum teams are adopting much of XP)
- Uses generative rules to create an agile environment for delivering projects

Scrum is best suited for

- Requirements are changing or emergent

- Willing to let the team self-organize
- You need a management framework more than a set of engineering practices
- You need a proven, scalable agile process

DSDM

Characteristics

- Highly iterative
- Strong emphasis on prototyping
- Uses time boxes to control scope
- Strong focus on business value

DSDM is best suited for

- The project has tight time constraints
- The application is interactive or UI intensive
- You have clearly identifiable users
- The project is small or can be made small by decomposing it
- Requirements can be prioritized
- Requirements are not clear or change frequently

Crystal Clear

Characteristics

- Incremental delivery on a regular basis
- Progress tracking by milestones based on software deliveries and major decisions rather than written documents
- Direct User Involvement
- Automated Regression testing of functionality
- Two user viewings per release
- Workshops for product and methodology tuning at the beginning and middle of each increment

Crystal Clear is best suited for

- You want to do a lot of the process definition yourself
- You have a small, collocated team
- Requirements are mostly known or knowable in advance
- Project involves loss of Comfort or Discretionary Money, not Essential Money or Life

Feature-Driven Development (FDD)

Characteristics

- Serve as primary unit of work
    - Similar to XP stories or Scrum backlog items
    - Small enough to do in two weeks
- Feature Set
    - Collection of features
    - Assigned to a Chief Programmer and team
- Major Feature Set
    - A domain area, one or more Feature Sets
- Domain Object Modeling: Exploration and explanation of the domain of the problem. Results in a framework where features are added.
- Developing by Feature: Developing and tracking the progress through a small list of functions.
- Individual Class (Code) Ownership: Each class has a single person nominated to be the one responsible for the consistency, performance and conceptual integrity of the class.
- Feature teams: refers to small dynamically formed teams.
- Inspection: refers to the use of the best known defect detection mechanisms.
- Regular Builds: Refers to ensuring that there is always a running demonstrable system available. Regular builds form the baseline to which new features are added.
- Configuration Management: Enables the identification and historical tracking of the latest versions of each completed source code file.

- Progress reporting: Progress is reported based on complete work to all necessary organizational levels.

FDD is best suited for

- Willing to trade some agility for a well-defined way of scaling
- Organization has solid UML skills
- Most requirements are knowable in advance or somewhat stable
- You do not view self-organizing teams as a critical success factor

The following table illustrates the coverage for the different aspects of software development or each different Methodology.

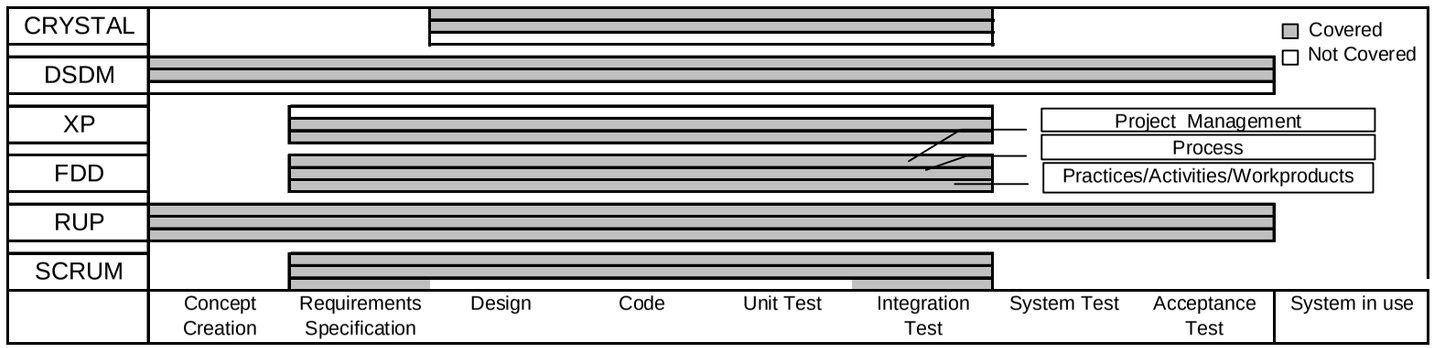


Table 3 Methodology Development Process Coverage

7.1.2 Selection Process

After evaluating possible methodologies to select the best option the following were considered based on user reviews and on worldwide adoption of the methodology. These were our top options:

1. Extreme Programming (XP)
2. Scrum
3. Feature Driven Development (FDD)
4. Crystal Clear
5. DSDM
6. RUP

Note: RUP is not always considered an agile methodology, but it is not a heavyweight. Based on its efficiency, high levels of adoption and since it is relatively familiar to programmers using UML, we included it in our study as an agile methodology.

The evaluation parameters were as follows:

Poor	2
Good	4
Very Good	6

Table 4 SDM grading system

The variables for the evaluation process were defined based on what was considered most relevant for the project success. The weights of the variables were assigned according to how important each variable was perceived in this particular project. These tables are relevant only to the work at hand and should not be considered as an evaluation of the quality or capacities of the methodologies.

The results were the following:

SDM Comparison Matrix						
	XP	SCRUM	FDD	Crystal	DSDM	RUP
Ease of Use (20%)	4	6	4	4	4	2
Impact upon Implementation (20%)	4	6	4	4	4	4
Large project capacity (20%)	6	6	4	2	4	6
Methodology Flexibility (20%)	4	6	4	4	4	2
User adoption (20%)	6	4	2	4	2	6
Totals	**4,8**	**5,6**	**3,6**	**3,6**	**3,6**	**4**

Table 5 SDM evaluation results

The table above shows the grades for each software development methodology, all of which are agile methodologies. The reason for Scrum's higher grades is that Scrum is basically a control method that can be applied to any type of project, not necessarily software development. It may be used in conjunction with other methodologies and is extremely light. Of the other methodologies, XP and RUP are better known and have been more heavily tested and adopted. Our main concern with both of these is that they incorporate more structure and thus become more difficult to modify. Also XP has not been used extensively for large projects, whereas Scrum has been, and the process and approach have been documented and are very simple and intuitive.

7.1.3 Definition and Justification

After reviewing the options, we chose Scrum based on its flexibility and ease of use. Scrum is a very light methodology in the sense that it can be implemented with minimal impact, and its benefits appear almost immediately, allowing the team leader to understand what is happening with the project at any given time and make decisions accordingly. It emphasizes customer satisfaction and the building of relationships.

Scrum has one other major benefit: It is not only is a methodology, but it can be used as a control wrapper over other methodologies. It is already commonly used as a wrapper for XP. The wrapper idea was extremely appealing because it allowed us to elegantly resolve our original preoccupation; Scrum allowed our new set of improvements to be used with any other methodology as long as Scrum was used as a wrapper. This means that the methodology can be used as an outer shell that envelopes an underlying methodology. For example a company that uses XP might use Scrum additionally over XP to have better control over the project, or to facilitate team coordination and management reports.

7.1.4 Introduction to Scrum

In this section the selected methodology will be described in detail. [17]

Scrum is an agile software-management process characterized, among other things, by quick daily meetings to report on project status. It has not received nearly as much attention as Extreme Programming (XP) or Rational Unified Process (RUP), but it is gaining popularity and is simple to understand. Like XP and RUP, Scrum tries to address the shortcomings of traditional software processes, where the assumptions that software development can be repeatable and well defined were often flawed.

Scrum approaches software development from a patterns perspective. Software development might not always be strictly repeatable and well defined, but certain patterns emerge. You can apply these patterns to address the issues of software development and achieve a highly effective and productive process.

Like many agile methods, Scrum puts a strong emphasis on communication, teamwork, and flexibility. By adopting practices that enhance these aspects of the process, you can plan the complexities of software development to a degree, but you can also tackle unplanned or unexpected events effectively to maintain forward progress.

Software process elements (patterns) in Scrum include Backlog, Sprints, Scrum Meetings, and Demos.

Next we will describe these elements briefly.

Backlog consists of the list of prioritized requirements or features the customer wants the team to build. Backlog is a dynamically changing list that management constantly reassesses. The customer, marketing department, and developers can all add new items to the Backlog, but only the product manager is allowed to change the items' priorities.

Sprints are the basic units of scheduling, typically 30 days or less of development activity. A Sprint consists of a preallocated work unit from the Backlog that composes the work and features to be completed during the Sprint. During a Sprint, the Backlog that the Sprint addresses is static—no new items may be added and the prioritization remains fixed for that portion of the Backlog.

Scrum Meetings are short daily meetings (targeted at 15 minutes) held without fail by the Scrum team. Each team member is expected to answer three standard questions: What did you do since the last team meeting? What obstacles are you encountering? What do you plan to accomplish by the next team meeting? A Scrum Master leads the meetings and tracks the responses from all team members. This information is the primary source of process measurement, monitoring, and documentation. The key idea behind Scrum Meetings is: If you have a function or process that is difficult to predict, sample more frequently to determine where things are going. This provides better feedback for both managers and developers, and it keeps people synchronized with the true progress of the whole team.

Finally, a **Demo** is the culmination of a Sprint and results in the functionality delivered to the customer. This might be an informal or formal delivery, or it might simply be a demonstration of the work the team has achieved to date. The key here is that the work in a Sprint must be organized so the end result is a piece of functioning software that shows the new capabilities added.

Scrum does not focus on a large list of specific practices, phases, or milestones that would define an entire product lifecycle. It focuses more on defining some enabling practices and patterns that allow you to move quickly and deftly while minimizing the risk the project. Scrum is not a fully defined, by-the-numbers process you can use to perform all aspects of software development, but you can combine it with other patterns and processes to complete and complement your software process of choice.

Scrum is best suited for companies with highly motivated, independent teams of developers. But management and customers must be willing to ease the reins of

control and let the developers work without much supervision and monitoring. Management must be firm in selecting the set of features that comprise the Backlog addressed by a Sprint and then stick to that list until the next Sprint. The remaining Backlog can evolve constantly during the Sprint, but when the next Sprint is started, it too takes a static set of items off the top of the list that becomes the feature set for the next Demo.[17]

Next we will show sample Scrum project documentation:

Example Project Backlog

This is the general backlog for the project where management and the team define what the goals of the project should be and the general tasks than should be performed. The tasks to be developed in each Sprint are coded according to the pertaining sprint color. The color coding can be viewed in the table below the mentioned one.

Tasks	Days	Hours	Assigned to
Informe 1 : Consolidated	12,5	100	Team Lead
Informe 2 : Telephony	10	80	Team Lead
Informe 3 : Internet	5	40	Team Lead
Informe 4 : Security	5	40	Team Lead
Informe 5 : Zones	11	88	Team Lead
Informe 6 : Enterprise (Market Segment)	11	88	Team Lead
Informe 7 : Mass (Market Segment)	11	88	Team Lead
Totals	53	524	

Table 6 Example Project Backlog

Color Coding
SPRINT 1
SPRINT 2
SPRINT 3

Table 7 Sprint color coding

Example Sprint Backlog

Table 8 shows a Backlog after Sprint completion. Each task has been fully defined and described. Each task has been assigned an effort (amount of man-hours). Each of these tasks was worked on during the Sprint.

Project: Telco Reports				
Nr.	Module	Description	Effort (Man Hours)	Assigned to
Tasks For Sprint				
	GenRep	**Report 1 : Consolidated**		
		Indicators Chart	8	Developer 1
		Graph 1 : Ingresos Acum. + Costos Acum.	12	Developer 2
		Graph 2: Accumulated Income Composition. By business area	8	Developer 1
		Graph 3: Services 12MM	12	Developer 2
		Graph 4: Arpu Service + churn acum. + sales	8	Developer 1
		Graph 5 : Mou 12MM	0	Developer 1
		MOU SLM TB	4	Developer 1
		MOU SLM RP	4	Developer 1
		MOU TL	4	Developer 1
		MOU CACC	4	Developer 1
		Graph 6 : Traffic Minutes 12MM	0	Developer 2
		SLM	4	Developer 2
		TRAFICO LOCAL (TL)	4	Developer 2
		CACC Carriers	4	Developer 2
		CACC Mobil	4	Developer 2
		CACC Local	4	Developer 2
		CACC Internet	4	Developer 2
		Graph 7 : Reclamos Técnicos 12MM por área de negocio	12	Developer 1
	GenRep	**Report 2 : Telephony**	0	
		Graph 1: Acum. Income + Acum Costs.	8	Developer 2
		Graph 2: Accumulated income Composition. by zone	8	Developer 1
		Graph 3 : Services Telephony 12MM	8	Developer 2
		Graph 4 : Arpu Services + Churn + Sales	8	Developer 1
		Graph 5 : MOU 12MM	0	Developer 1
		MOU SLM TB	4	Developer 1
		MOU SLM RP	4	Developer 1
		MOU TL	4	Developer 1
		MOU CACC	4	Developer 1
		Graph 6 : Traffic Minutes 12MM	0	Developer 2
		SLM	4	Developer 2
		TRAFICO LOCAL (TL)	4	Developer 2
		CACC Carriers	4	Developer 2
		CACC Mobil	4	Developer 2
		CACC Local	4	Developer 2
		CACC Internet	4	Developer 2
Total			172	

Table 8 Example Sprint Backlog

Example Sprint Burndown Chart

The following chart is an example of a typical project burndown, and a graph showing the burndown percentage and tendency line. The Burndown is the amount of effort (man hours) that the project required at its beginning vs. how many are still pending at each scrum meeting.

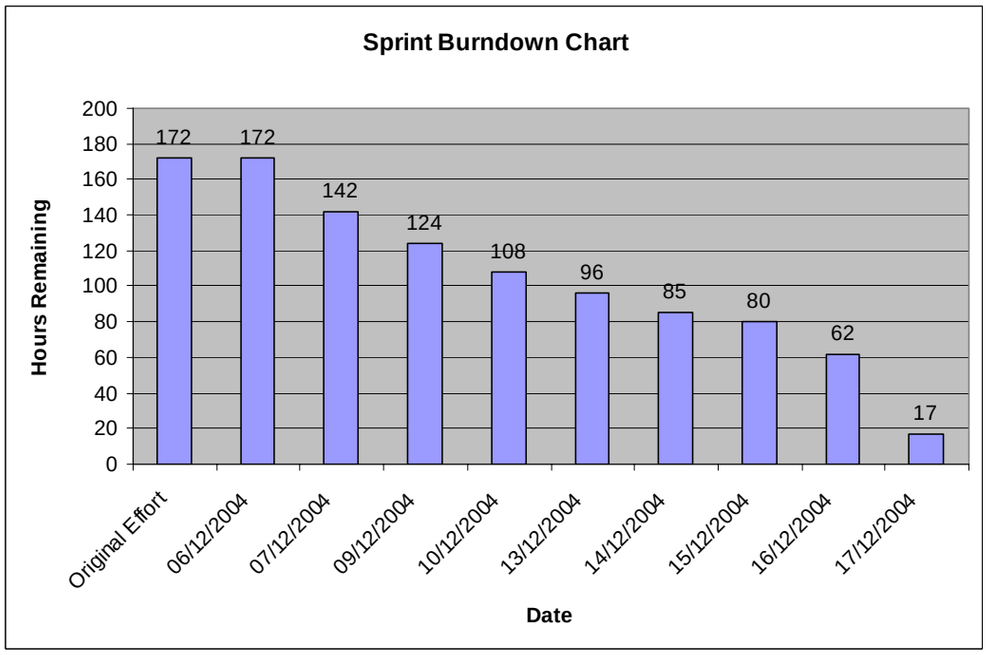
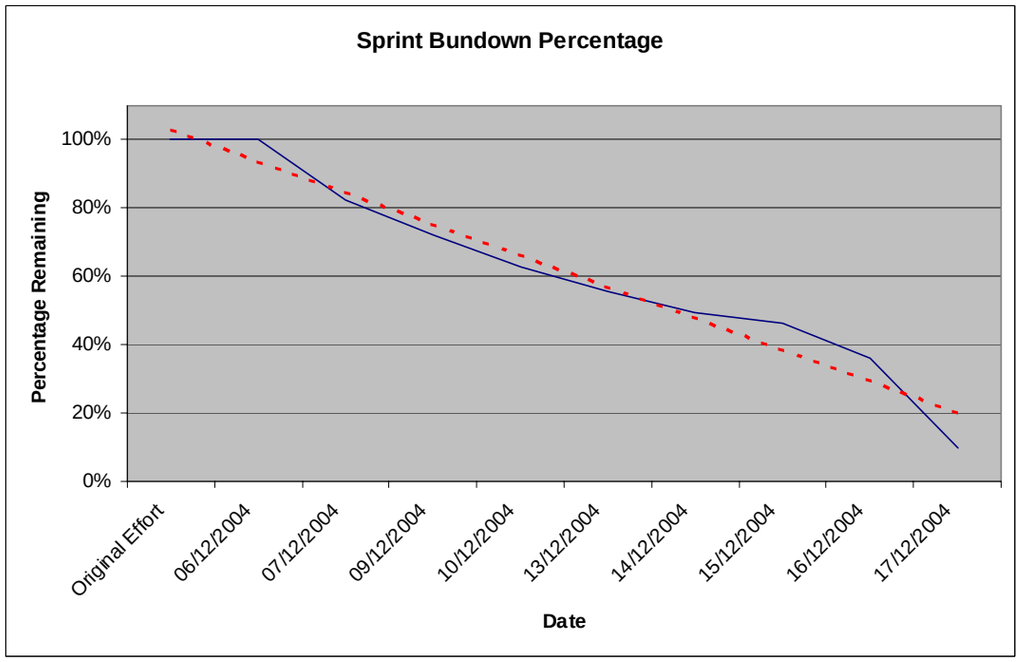


Table 9 Example Sprint Burndown Chart

In the graph above the effort is specified along the Y axis, and each scrum meeting date is shown on the X axis. We can see how the amount of effort remaining for the sprint decreases at each meeting. This process is called the Burndown. Each sprint has a Burndown Tendency, which is basically the tendency line for the Burndown graph. We can see the tendency line in the graph on the following page. The graph shown is the calculated tendency for the Burndown Chart shown above. We can see that in this case the tendency line will reach 0 effort remaining after the sprint has concluded, this can happen if the requirements changed drastically during the sprint, the effort was underestimated, the teams productivity level were inconsistent with past references, among others.



Graph 3 Sprint Burndown percentage

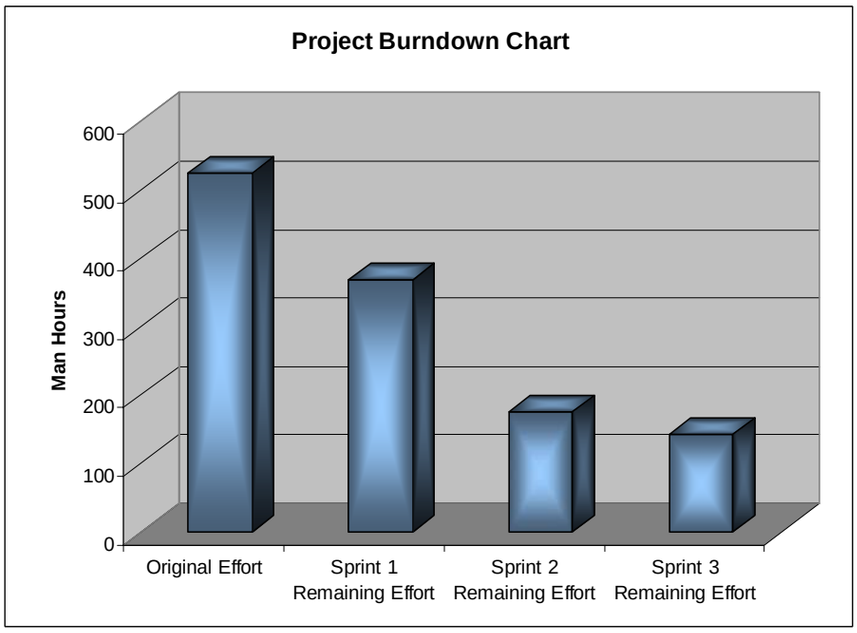
Example Project Burndown

The following table shows the amount of effort assigned to each Sprint and how much was completed during each sprint.

	Original	Completed	Remaining
Original Effort	524		524
Sprint 1 Remaining Effort	172	155	369
Sprint 2 Remaining Effort	217	194	175
Sprint 3 Remaining Effort	176	32	143

Table 10 Example project effort distribution

The graph below is a representation of the burndown process of each sprint in relation to the whole Project. Each sprint brings the team closer to project completion (0 man hours remaining)



Graph 4 Example Project Burndown

The last graph shows the same information as the last graph but in the line graph (Graph 3) we have added a tendency line to it, if the tendency line doesn't point toward zero during the Project development, the team may be facing some manner of problem.

7.2 Part 2. Evaluation, Selection and Justification of SNA Tools and Methodologies to be used

In this section the available SNA methodologies and tools will be reviewed for selection according to applicability to this work.

7.2.1 Evaluation of Methodologies

Information collection methods should usually be selected based on information availability at the organization to be analyzed. If we intend to formulate a standard process that can be repeated at any organization, the only way to do it is by selecting the most simple and straightforward approach. This means direct surveys that can be defined beforehand.

One of the main problems with conducting surveys is the formulation of the survey, preparation of the documents, and data input and preparation.

Since we planned on conducting several surveys, and after determining that no SNA-oriented survey software was available, we considered it worthwhile to develop our own survey module. The result is SNASurvey, a simple Delphi-based client running on an Access database. The Survey can be printed and filled out on paper, or also via a Web interface that can be easily posted on any web server.

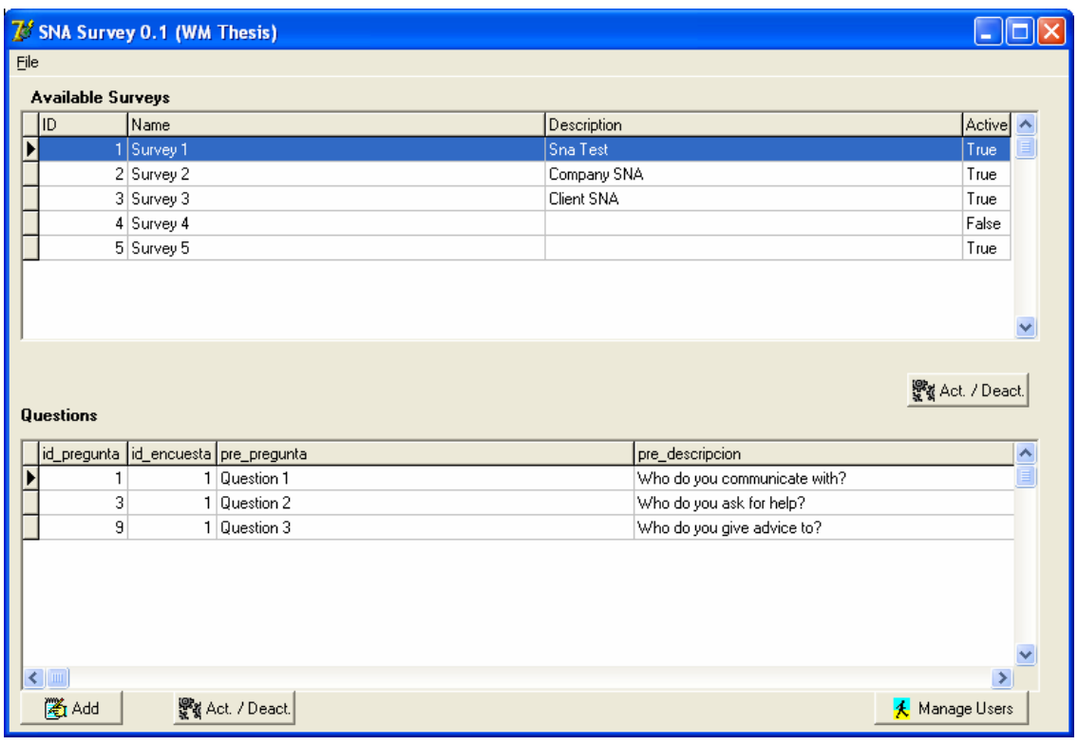


Illustration 5 SNASurvey Main Form

For the evaluation of the analysis software we considered the following products:

- **Agna** Version 2.0.7, Release: 1st May, 2003, I. M. Benta
- **Pajek** 1.03

  November 1996 - January 2005

  Copyright (c) 1996

  Vladimir Batagelj and Andrej Mrvar
- **Krackplot** Version 3.03

  David Krackhardt

  Carnegie Mellon University
- **Ucinet 6 for Windows**. Version 6.79

  Borgatti, S.P., Everett, M.G. and Freeman, L.C. 2002. Ucinet for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies.

7.2.2 Selection Process

To help select a software product from the ones we evaluated, we constructed an evaluation matrix.

The scale for the grading ranges from 2 to 10 and considers several key aspects of the products. The following table describes the key points we evaluated in each software and their weight in the final grade.

The next table describes how we graded each

Poor	2
Average	4
Good	6
Very Good	8
Superb	10

Table 11 SNA Software grading system

The following table (Table 12) shows the final grades for every system evaluated. Each parameter was assigned a weight that can be seen between parentheses next to the parameter description. The weights were assigned based on our project's requirements and should not be considered a general evaluation for SNA tools. The two last parameters (Graph Quality and Analysis Functionality) were considered as most important because they produce the key outputs . Another important consideration was user friendliness (Ease of use, Interface Quality

and Data Insertion Ease) because the methodology must use tools as simple as possible so users will willingly accept it.

SNA Software Comparison Matrix				
	Agna	Pajek	Krackplot	Ucinet
Ease of Installation (5%)	4	8	6	8
Ease of Use 15%	6	6	2	6
Interface Quality 15%	6	6	2	6
Data Insertion Ease 10%	6	6	6	10
Graph Quality 25%	6	6	2	10
Analysis Functionality 30%	6	6	6	8
Final Grade	**5,9**	**6,1**	**3,8**	**8,1**

Table 12 SNA Software evaluation results

7.2.3 Definition and Justification

Any of the evaluated software packages would be acceptable for this project, but UCINET is not just a SNA tool; it is also a suite of tools. It includes Pajek as one of the options, Mage 3d which includes 3D visualization of a network. The analysis functionality is outstanding, including the most types of analysis of any of the packages and even includes a spreadsheet editor where you can directly input data. It translates data formats for all different tools. Since Ucinet also received the highest final grade, we selected it for this project.

7.3 Part 3. Methodology Modification Process

In the following section our proposed SDM modifications will be described, and the expected results will be discussed.

7.3.1 Description of intended modification Process

General description of the process:

The work described here evaluates and analyzes our proposed modifications to the SCRUM software development methodology using SNA. We evaluate the modifications in an attempt to find weaknesses in the process. If any weaknesses are found, possible improvements to the methodology will be proposed some of which may incorporate social network analysis into the software development methodology.

After defining the set of possible improvements, limited testing will be conducted. The objective of the tests will be mainly to decide whether the improvements seem to work on a limited scale, and whether they hinder the software development process unnecessarily.

Analytical Method:

For experimentation purposes, we divide the software development process into three parts, and we will try to make improvements on each stage of the process.

1. Requirement Definition: This stage is one of the most important to any software project and where most mistakes are made. Any mistake at this stage could means huge cost increments later on in the project, or even make the project not feasible. This stage consists of determining what will be done in the project, how the users needs will be resolved and what the restrictions and possibilities are.
2. Development Process: This is the implementation of the requirement definition. This is the stage when the project is taken from paper statements of intent and brought to reality. Programmers work on turning every requirement into code that solves a problem. Two of the main risks of this stage are low quality code and late deliveries. Both are ultimately the responsibility of the project manager or team lead.

3. Deployment: This stage may seem to be the easiest if the two earlier stages have been resolved correctly. This can be misleading because no matter how good the software if it is not accepted by the client users, for any reason, the software will never be properly used and will ultimately fall short of its expectations. This stage includes releasing the software to the client, training and supporting the users and finally releasing the final product into production.

For each stage we will focus on one or two different groups of people; Development Team and Client User Group.

As an early approach to the problem and considering the information gathered so far, we can identify improvements that might be appropriate based on certain observations.

7.3.1.1 Requirement Definition

Introduction

Normally, during a normal requirement definition process, the development team interviews a set of users called the client's project team. On larger projects the interviewed group may neglect all or some of the most relevant users. This can happen because the client's actual working structure has diverged from the formal structure. Here we propose a methodology to improve the selection process for the interviewed user group selection.

Proposed Methodology

When considering a normal requirement definition process, the development company executives meet with their counterparts in the client company and decide on a group of users out of the client company that are relevant and should be considered for meetings in the definition process. At first glance this seems reasonable, but normally these groups are based on the company's formal structure. We have already discussed how SNA is used to discover the informal, but real, structure that normally underlies the formal structure of any company. If the formal and informal structures are not the same, and the proposed user group is based on the formal structure, the resulting user group may not be effective in defining requirements.

Here we propose that, before a decision is made on the composition of this group, a SNA evaluation will take place. The evaluation questionnaire should be directed toward the processes that will be studied during requirement definition. Based on the resulting network, we will be able to determine key users, groups and interfaces. Once this information is available, it can be taken into consideration while selecting the requirement definition client group.

Result Evaluation

To be able to determine the effectiveness of our proposed method, we conduct a SNA study of a company whose requirements were defined before this work was undertaken. We then analyze this network to decide if key users or interfaces were missed in the selection of the client project team. We then interview the users not included in the original client project team, searching for important factors that might cause changes in our design. Based on the changes we encounter and the time that they take to implement, we estimate what percentage of our original project effort might change.

7.3.1.2 Development Process

Introduction

The development process may vary considerably from team to team, so it becomes hard to generalize about the correct discipline for a team. Most methodologies suggest a series of best practices from which teams can choose according to their particular style. In this case we are interested in studying of the team's behavior. We shall provide the team leader with relevant information that would normally be unavailable to him throughout the development process.

A similar approach is the one described by Paul Hersey in his Situational Leadership[4] model. Here we formulate a quantitative method based on the communications within the Scrum generates to be able to place the team within on of Hersey's Readiness Levels. If this can be accomplished, the situational leadership model can be applied by the Team Leader. Usually the Readiness Level is estimated based on comparisons and the team leader's experience.
If we can ascertain quantitatively the correct readiness level of the team, it will make the manager's task easier.

We might determine that a team's readiness level is affected negatively by some members within the team, bringing the general level down. The manager might be able to focus on helping those members so that the team can benefit, or assign these members to a more appropriate team.

[4] Paul Hersey: The Situational Leader

Proposed methodology

During the evaluation, two separate and complementary processes will be conducted. The first is a SNA evaluation of the team (based on questionnaires) and the second is based on questionnaires designed to determine technical ability and experience of the Team.

It is hoped that this information can be used as an aid for the team leader to determine the Readiness Level of the team. These results should not supersede the team leader's opinion, but rather be used as additional information.

Next the method for conducting both processes will be described in general terms

SNA Method

During the Sprint, SNA evaluations will be conducted periodically; based on these observations we will be able to ascertain the shape of the social network and the flow of information within the network. Considering the nature of the Team Readiness Level classification system, and our own experience, we suspect that the analysis will provide us with a way to determine both of the needed parameters in the Readiness evaluation (Ability and Willingness).

Network Shape

A star shaped network will mean a team that is highly dependent on its leader's counsel regarding most issues concerning the project, technical or not technical.

if team members communicate heavily with their team leader and not among themselves , it means they need him for most decisions or solutions. This can be considered an unable team.

A *mesh shape* would imply a team that communicates primarily among the team members and goes to the Team Lead only when necessary. This kind of Team is proficient in solving problems and can make decisions unaided.

These team members communicate with each other constantly; they discuss problems and solutions and inform the team leader of their decisions or get new assignments from him.

This kind of team is considered an able team.

Example: During a Sprint the network shape for a 4 person team was the one shown in Illustration 5. (Here the line width is generated according to amount of communication.)

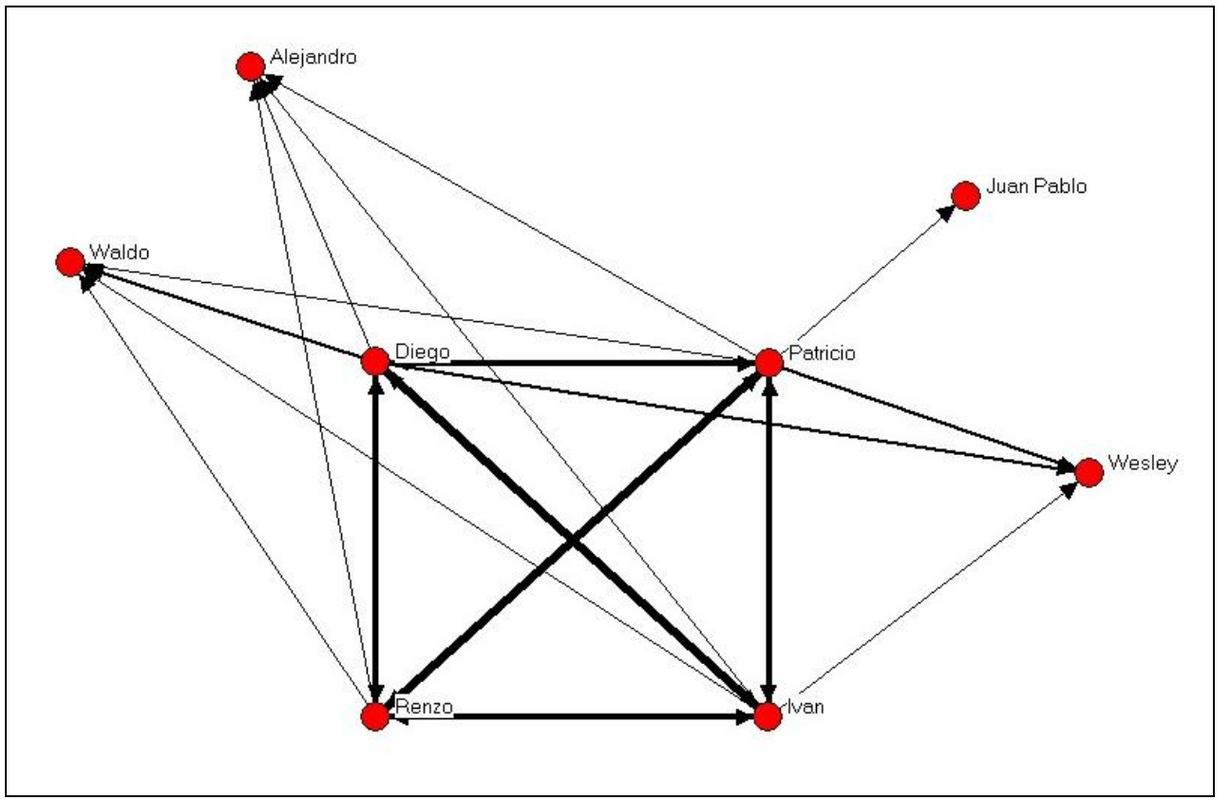


Illustration 6 Sprint Scrum Team Example

The central four with the strongest communication are the project team. The rest are management, clients or technical advisors. The team communicates strongly among themselves in a mesh shaped pattern. The Team lead is Diego, but this is not apparent from the network shape.

Communications Flow Direction

The direction of the communications can tell us something about the team's ability. There are three types of communications:

Unidirectional from the Leader toward the Team	Unwilling or Unconfident
Unidirectional from Team towards the Leader	Willing and Confident
Bidirectional between Leader and Team	Can be either, depends on the network shape

Table 13 Willingness Based on Communication Directionality

Table 13 shows that a team that receives information from the leader in an unable team that requires constant support, i.e., an R1 team.

A team that communicates bidirectionally with the Leader is either an R2 or R3 Team. The reason is that an R2 team is unable, but confident, so the members want to do more and let the team leader know of their ideas and opinions, but they lack experience and technical ability so the Team Lead has to supervise constantly so the team delivers what is expected.

An R3 team is different. The members are able, but they communicate with their team lead because they aren't confident and need reassurance of their ideas and solutions.

The R4 team only receives basic instructions from their leader. Members report back only with results or problems which they are not permitted to resolve unaided.

Result Evaluation

Based on the previous observations we can establish the following way of cataloging Teams based only on analytical quantitative methods.

Communication from Leader to Team	Star Network	Mesh Network
Unidirectional toward Team	R1	
Unidirectional toward Leader		R4
Bidirectional	R2	R3

Table 14 Readiness classification based on team communications

We expect most teams to conform to this model. However, to verify its accuracy, it needs to be applied on numerous occasions and validated statistically.

Evaluation Method

Our evaluation method is direct. We ask the team to answer a questionnaire designed to evaluate their experience level. It determines several key points:

- Team Technical Ability
- Team Experience
- Team work methodology and cohesion

The questionnaire is standard and will have a standard evaluation matrix so the process is the least arbitrary as possible.

Result Discussion

The results will be evaluated by direct interviews with team leaders to verify whether they seem appropriate to their particular experience and if they would be useful in making decisions. A more formal approach would be to try to analyze results over a long period of time and interview as many team leaders as possible. But, given the scope and time allotted for our ongoing project, this is not feasible. It could be included in further investigations about our current problem.

7.3.1.3 Deployment

For the deployment process the focus will mainly be on the client's networks. Based on the social network information, we hope to be able to direct the user instruction so they have better understanding of the product, available technical support within their company and make sure that key users have a good understanding of the system to insure that the dataflow in not interrupted. In this manner through focused training we hope to be able to increase two key elements which are important to project success.

- Maintain informal support networks
- Improve user perception and acceptance of the new systems.

The SNA information gathered earlier in the project during the requirement definition phase, in addition to one more questions in the survey, is used to determine key users within the client network regarding the process being studied and informal technical support network key users. We try to determine key users for every area of the company that will be using the system.
Any users with high scores on both networks are prime possibilities for in house support for the new system. In any case the key users from either network should be considered for one of the two stated purposes. What should happen is that any person determined to be a key user is someone the rest of the organization relies on for information or support, so naturally they will go to this person for support in many matters. If this person coincidentally is a user of relevance to the project it would make sense he knew as much as possible about the new systems because it would be in his own benefit. If this person is trained extensively, many problems during deployment can be avoided because it will be easy for him to solve minor problems or misunderstandings about the system within his immediate surroundings. If many of these users can be found within an organization, help desk and post-installation support could diminish significantly.

7.3.2 Methodology Modification

The methodology will consist of three stages:

- Requirement Definition
- Development Process
- Deployment

Each of these stages will have certain steps that will be added or modified in the original methodology.

7.3.2.1 Requirement Definition

The design methodology in SCRUM is very open, and allows for varying degrees of requirement specification. With SCRUM a project backlog must be established. The backlog must include at least all the main activities that the project contemplates. The project normally revolves around a certain set of problems that need to be resolved. Normally the development team leader and development company management meet with the client management to select a client project team.

Next the client project team works with the project development team on the requirement specifications. During this process key users within the client company are interviewed to determine requirements. The key users are selected based on the company's formal structure and management's opinion on who should be interviewed.

In most smaller projects this works well because process models for simple projects tend to be clear, and determining who an important player for the project is fairly easy.

A problem arises, however, with a large project that involves the client organization in more than one department. When this is the case key users may be missing from the client team. As a result, requirements may be incomplete.

Our solution is to conduct a focused survey to determine who by a social network analysis to determine who the most important players are.

This can be done by using a SNA questionnaire that should at least include the three following questions:

1. Who do you normally speak to regarding Process A?
2. Who do you ask advice for when considering problems regarding Process A?
3. Who do you give advice to regarding process A?

These questionnaires should be completed by as many relevant people as possible within the client organization and by all of the people in the organization if possible, or at least all those that work within the interviewed departments.

Subsequently, we can use the following metrics to determine user importance:

- Closeness
- Betweenness.
- Cutpoints

Working Example.

Next we will use an example to describe how the SNA metrics should be applied and considered during the development process.

We will conduct an analysis of the example network to determine the relevance of each node to the network as a whole. During a project, we suggest that the person in charge of requirement definition should use this information to decide which users should be included in the process (by becoming a member of the project team).

Once all three metrics have been studied, a consolidated report can be delivered to the team lead, and based on this information, and in conjunction with executives of the client company, he should be able to decide whether an individual should be involved in the requirement definition process.

The number of individuals included will be determined by the interviewing team's capacity the project and user availability.

The final report that will be delivered to the team leader should be formatted as in the following example.

Node	Betweenness	Closeness	Cutpoint
A	0,0037037	0,6363636	No
B	0,0037037	0,6363636	No
C	0,0000000	0,5656566	Yes

Table 15 Node Relevance report sample

Next we will explain what each metric means and how it should be considered.

Consider the following network:

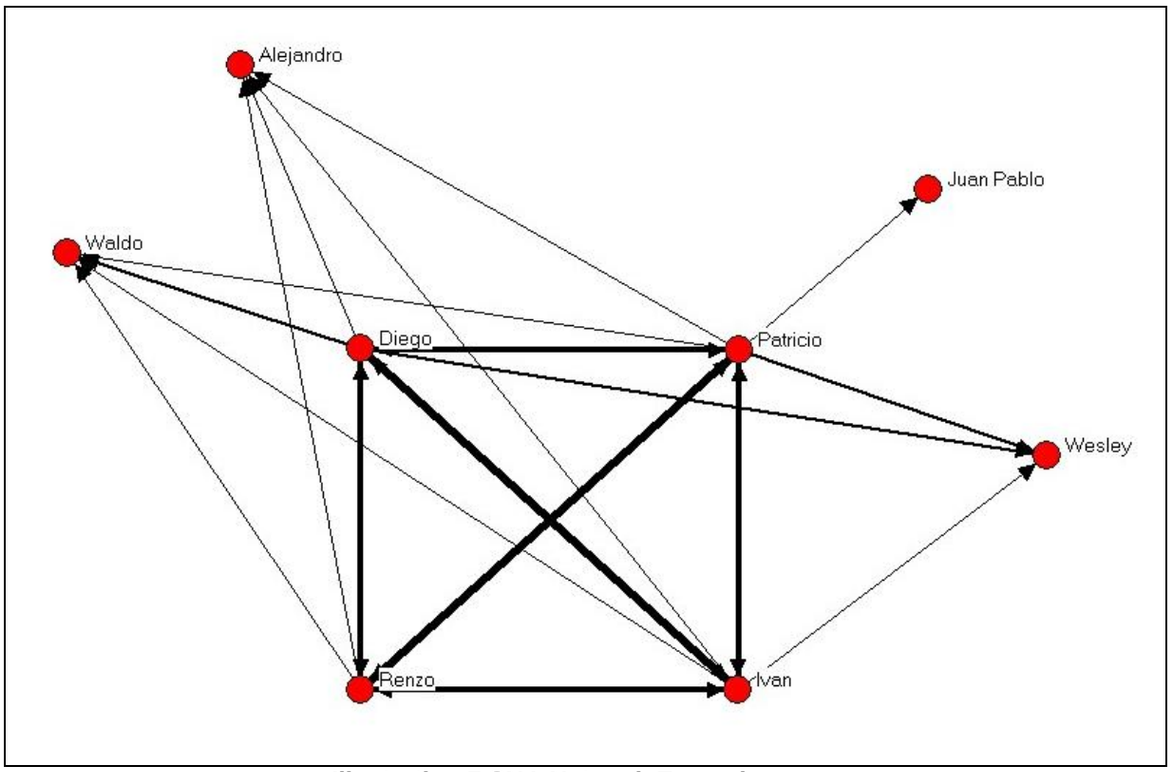


Illustration 7 SNA Network Example

The correct procedure would be to evaluate the three metrics, order them in a chart and deliver them to the project manager for consideration. Using these metrics he would be able to determine user relevance in the requirement definition process. Based on this information and his conclusions he will have a better understanding of the problem he is trying to solve.

Metric 1 Betweenness

Betweenness related to how important to keep communications flowing a node is. In our example the results calculated by UCINET are:

Betweenness	
0,0037037	Diego
0,0037037	Ivan
0,0000000	Renzo
0,0370370	Patricio
0,0000000	Wesley
0,0000000	Juan Pablo
0,0000000	Waldo
0,0000000	Alejandro

Table 16 Betweenness values for sample network

Betweenness analysis can be done using the Pajek module of Ucinet; it is located in the "Net" menu, under "Vector", "Centrality".

Betweenness can be considered similar to cutpoint in the sense that normally a node with high betweenness is also a cutpoint, but it also means that the node is in a good position to convey information or to monitor information flows within the network.

A node with high betweenness has great influence over what flows in the network.

Metric 2 Closeness

These nodes have the shortest paths to all others -- they are close to everyone else. They are in an excellent position to monitor the information flow in the network -- they have the best visibility into what is happening in the network.

In our example the results calculated by UCINET are:

Closeness	
0,6363636	Diego
0,6363636	Ivan
0,5656566	Renzo
0,7272727	Patricio
0,4628099	Wesley
0,3916084	Juan Pablo
0,5090909	Waldo
0,5090909	Alejandro

Table 17 Closeness values for sample network

Closeness analysis can be done using the Pajek module of Ucinet; it is located in the “Net” menu, under “Vector”, “Centrality”.

Metric 3 Cutpoints

This analysis may be complicated for a larger network, but it can be done automatically using Ucinet. Within the Netdraw tool in the Ucinet Suite, one of the analysis options is Blocks & Cutpoints; this analysis tool automatically detects all cutpoints within a network.

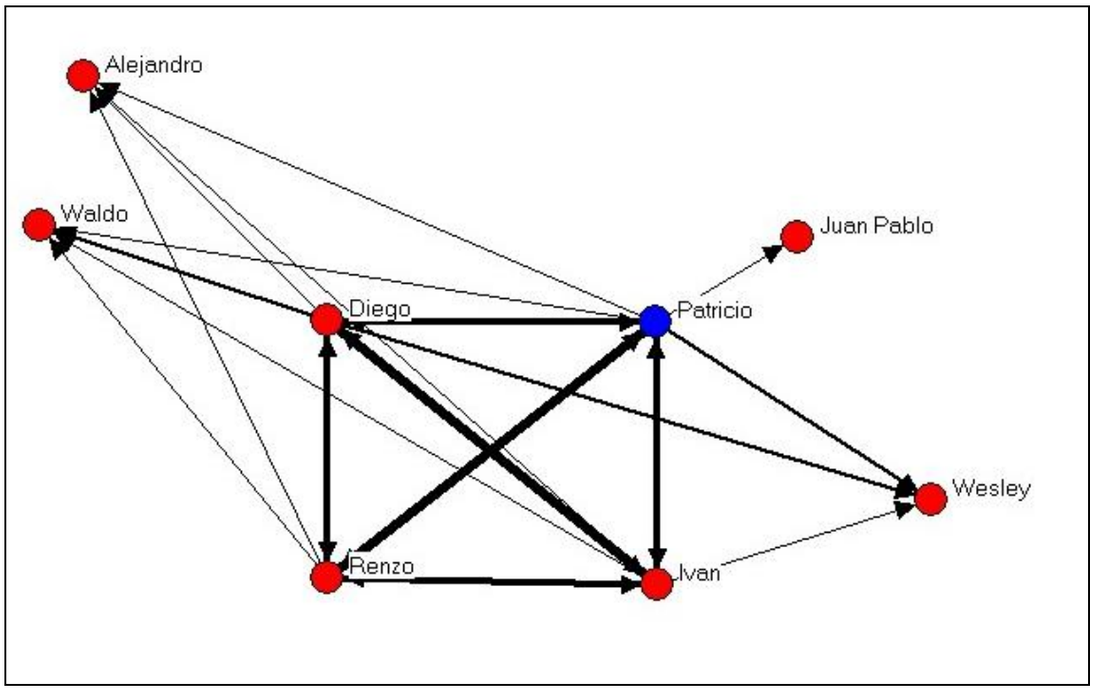


Illustration 8 SNA network cutpoint example

In the example the node colored in blue is a cutpoint because Juan Pablo would be disconnected from the network in the event that Patricio wasn't considered.

The aforementioned metrics should allow us to determine if any elements that are relevant to requirement definition haven't been considered in the focus group without having to notice them too late in the project development thus incurring in additional cost.

7.3.2.2 Development Process

The development process is somewhat harder to modify because SCRUM allows It maintains all the mandatory backlogs and structures the sprint meetings as needed.

Here we do not focus on the development process *per se*, but on one of the key aspects of any successful project, leadership. Effective leadership will almost certainly always be the key success factor in any project, software development or otherwise. We have evaluated scrum teams and have studied the SCRUM proposed leadership methodology. When taking our investigation a step further than the SCRUM methodology and studying leadership as a whole, we found that for most successful teams it is not one particular style of leadership that is most effective, but rather using the right type of leadership at the right time that is sine qua non to success**.**

This brings us to Paul Hersey's Situational Leadership[3] model that we discussed earlier. The model proposes that, for any given team, if you are able to determine a team's readiness level, you can determine the correct type of leadership required. To date we haven't found any analytical models to determine a team's readiness level. Usually it is left to the leader's experience to determine the readiness level of the team. The problem is:

- Experienced leaders need to work with a team for a while to determine how it is composed and if it is homogeneous in its composition.
- Inexperienced leaders might misjudge their teams' readiness levels and apply the wrong type of leadership and orientation.

Even though we consider Hersey's model as appropriate and relevant to our problem, we would like to facilitate the evaluation process by introducing certain metrics that might help.

With this in mind we have constructed the following model that shows different behaviors in the communication patterns depending on readiness levels.

We will consider two relevant metrics:

- Network Shape
- Communication Directionality

Network Shape

We have already discussed the levels of readiness in earlier chapters and how network shape should be relevant to determining readiness, but we will explain again the two possible network shapes within a Scrum team.

A star shaped network will mean a team that is highly dependent on their leaders counsel regarding most issues concerning the project, technical or not technical.

A *mesh shape* would imply a team that communicates primarily among the team members and goes to the Team Lead only when necessary. This kind of Team is proficient in solving problems and can make decisions unaided.

The shape of the network should always be easy to determine because scrum teams are relatively small (up to 9 people).

Network Shape Examples:

Star Shape Network Example

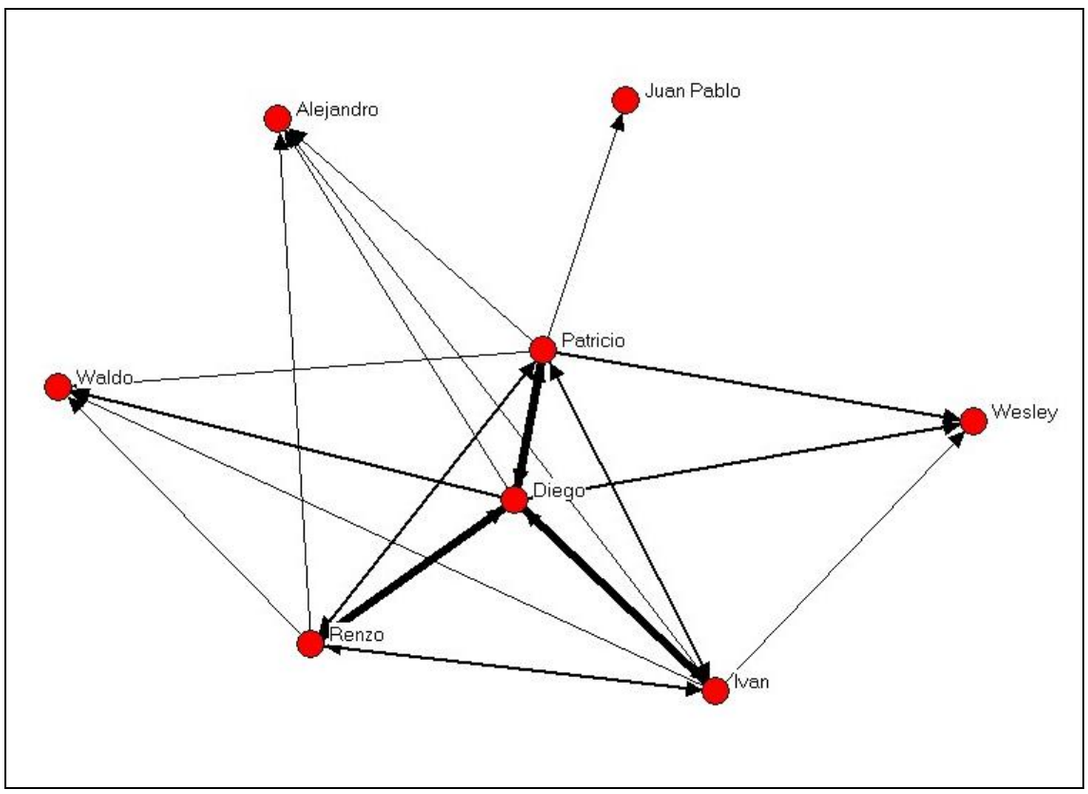


Illustration 9 SNA Star Shape Network Example

Here we can see that the four team members communicate heavily with the team leader (Diego) and not as frequently among themselves, the amount of communication

(represented through the line thickness) among the team is similar to the amount with management or clients.

Mesh Shape Network Example

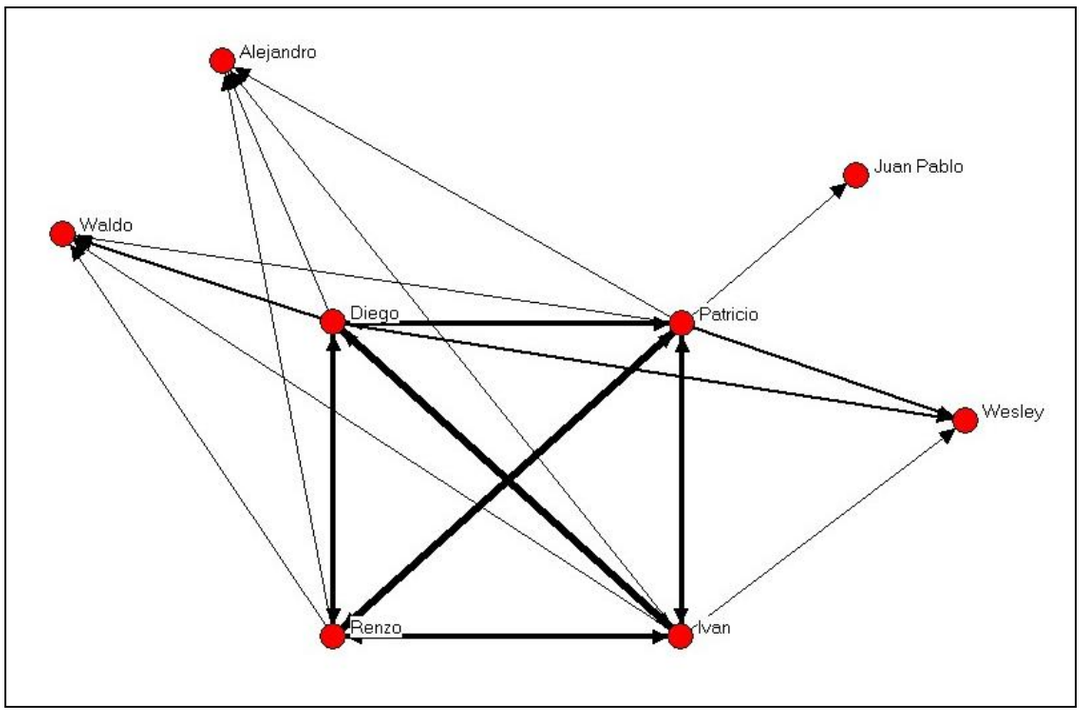


Illustration 10 SNA Mesh Shape Network Example

Here we can see that all four team members communicate heavily among each other to the point where the team leader (Diego) is no longer recognizable as such only from the network shape.

A metric for how much a network conforms to mesh communications is:

Meshness

$$i = \text{Originating node}$$

$$j = \text{Target node}$$

$$W_{i,j} = \text{Weight of edge (communication) from i to j}$$

$$M = \frac{\sum_{i=2}^{n}\sum_{j=2}^{n} W_{ij}}{\sum_{i=2}^{n} W_{i1} + \sum_{j=2}^{n} W_{1j}}$$

Where:

$1 =$ Team Leader

$2..n =$ Team Members

This metric can be used to determine a ratio which shows if a network is mesh shaped. We define the following:

$M > 1$ then Network is a Mesh

$M < 1$ then Network is a Star

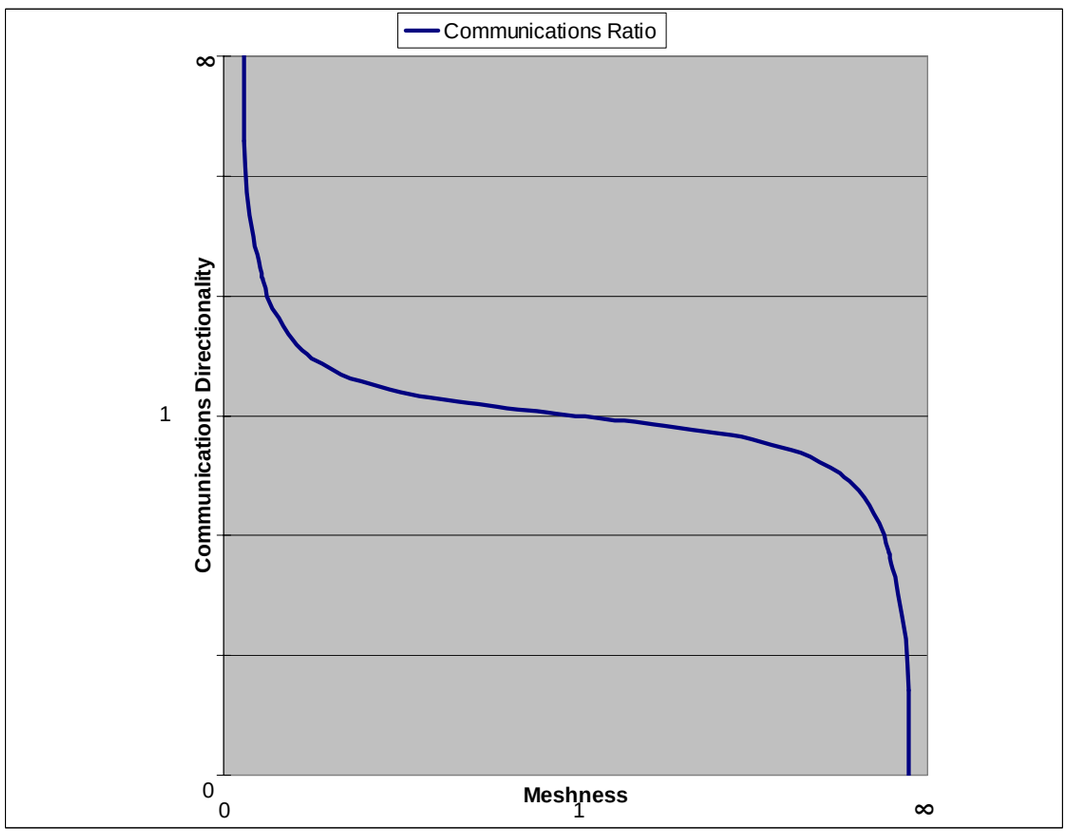
The threshold should ultimately be determined empirically but for experimentation purposes we will use a threshold of one.

Communication Directionality

The second metric is harder to evaluate because it requires definition of certain thresholds that will be considered. We at this time don't have enough data to determine the appropriate thresholds, since they must be determined by statistical evaluation, but for experimentation purposes the model should still be valid. The thresholds should be determined in the future through additional work in a continuing project.

Our model is based upon our experience and consultation and corroboration with other experienced project developers.

Readiness vs. Communication Directionality Model.



Graph 5 Communication Directionality vs. Meshness

Graph 5 shows how a normal evolution pattern for a team's communications Directionality and Meshness should look.

To determine the thresholds for directionality (D) we used the following equation:

Using the SNA graph information we can extract the following variables:

$i = \text{Originating node}$

$j = \text{Target node}$

$W_{i,j} = \text{Weight of edge (communication) from i to j}$

Then:

$$D = \frac{\sum_{j=2}^{n} W_{1j}}{\sum_{i=2}^{n} W_{i1}}$$

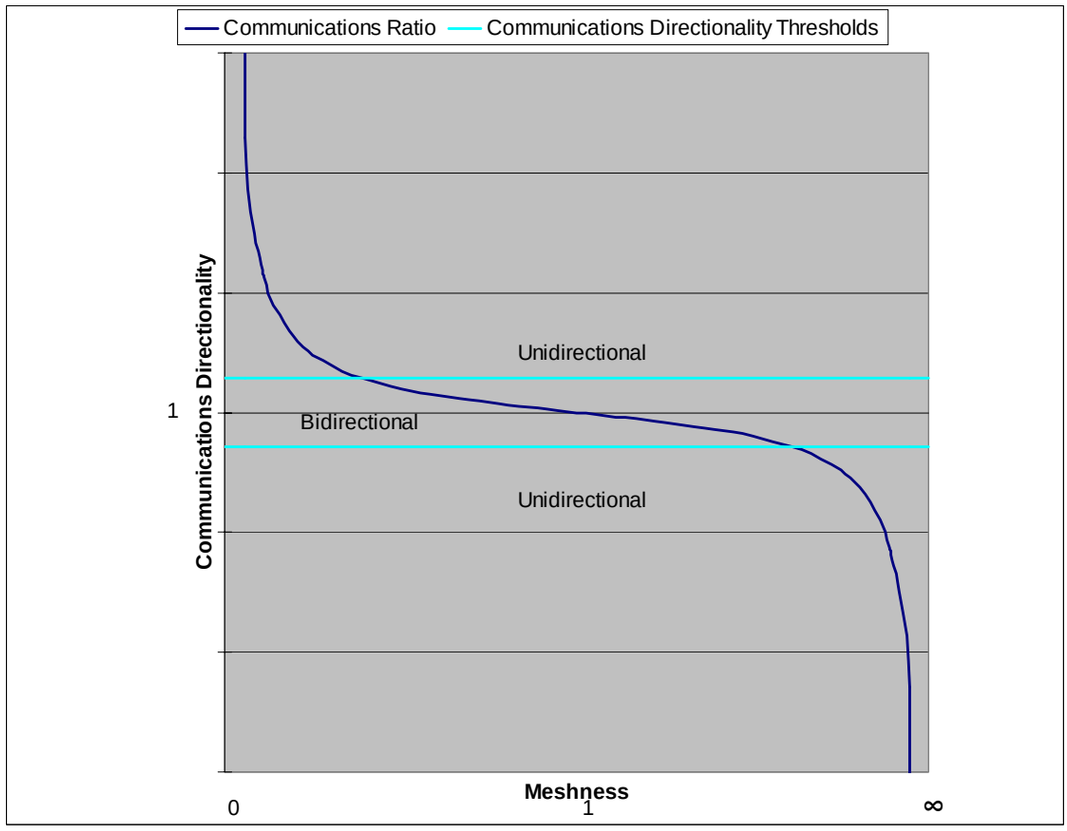
Where:

$1 = \text{Team Leader}$

$2..n = \text{Team Members}$

Our graph at this point may look like the one on the following page.

The curve goes from mostly unidirectional communications from the leader toward a star shaped team, trough bidirectional star and mesh, and finally a unidirectional mesh communicating from the team towards the leader. As we have seen earlier in our model bidirectionality is representative of R2 and R3 teams, and unidirectionality is found in R1 (leader to team) and R4 (team to leader).



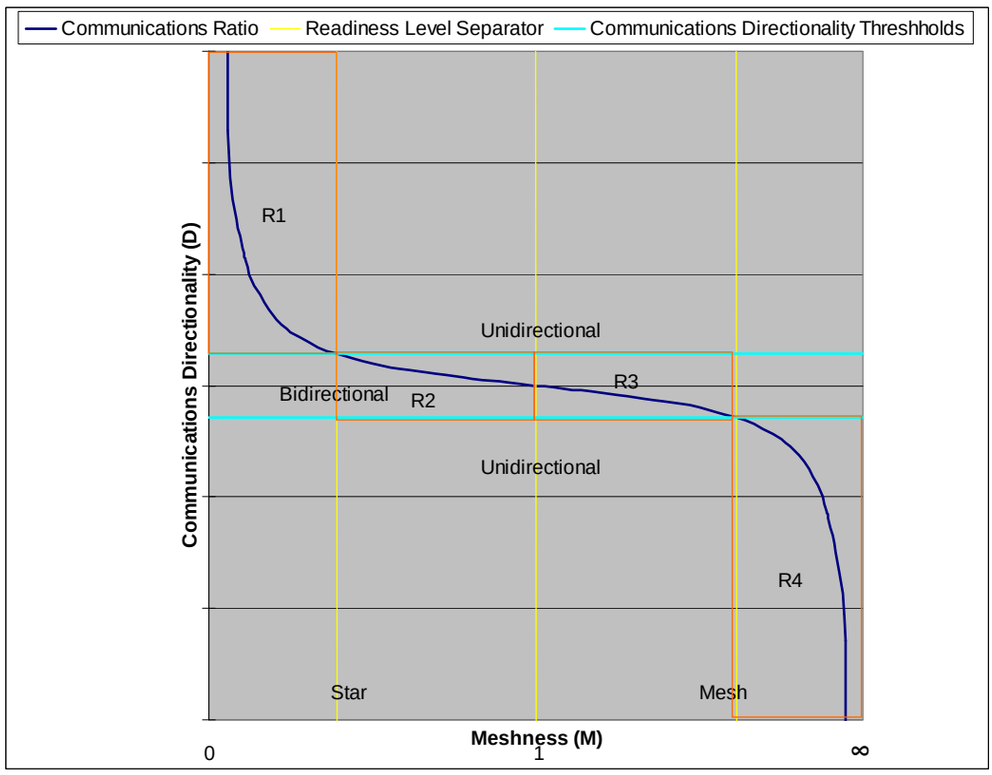
Graph 6 Communication directionality thresholds

This model has not yet been validated empirically. Our purpose here is not to establish a final model, but to propose a model that is ready to be validated empirically.

If we combine the network shapes with communications directionality we have a complete model that could provide easy assessment and visualization of a team's readiness level.

The metric we have determined for this purpose is Meshness, which means the degree of communication that is flowing among team members vs communications from team members toward the team leader. This would signify that the network is Mesh shaped instead of star shaped. In a Star shape network the center of the star is the leader. For a mesh shaped network, the leader communicates with the team as any other team member would.

In our graph that would be represented as follows.



Graph 7 Final Communications vs. Meshness Model

The final model shows how Meshness and directionality can be indicators of team readiness. Directionality differentiates between R1, R4 and R2, R3 and finally meshness allows differentiating between R1, R2 and R3, R4. This shows us in the final model how a teams communications would evolve to be representative of their readiness level.

Deployment

For the deployment stage of the process the analysis will be centered on two aspects:

1. User Support
2. Key User comprehension

For this the same survey as in the requirement definition stage will be used, except we will add one additional element to the questionnaire. This element will help us determine which users play an informal support role within the organization and enable us to have them inadvertently continue to do so. This element is one additional question:

- Who do you ask for help regarding information systems outside the help desk?

This will allow us to determine key users for the two aspects mentioned above. Once they have been identified using the same metrics as were used during requirement definition (betweenness, closeness, cutpoints), the following actions can take place:
During training, exceptional care should be taken to ensure proper system comprehension by key users to increase the chances that the system will:

- Be used properly by key users of the process, thus ensuring that the correct data is being introduced into the system at an appropriate time and that it is flowing correctly among users.
- Be supported by users that were supporters of earlier systems, providing a better user acceptance of the changes it may introduce, and also allow help desk to be less strained because the users with better knowledge of the system will train or explain the system to users with less comprehension.

8 TESTING

In the following chapter we describe how the modified methodology was tested to determine its effectiveness. This model is intended as a starting point in a perfectible process methodology. These tests are only meant as an initial set of observations on the methodology, because more definitive conclusions will require more testing and analysis. Here we attempt only to determine preliminarily, at an early stage of testing, if the proposed modifications appear to be effective. Further testing and analysis are required to determine the methodology's efficiency and measurable benefits. This will be discussed further in our conclusions.

8.1 Testing Process

The testing process consisted of analyzing all three stages identified in the development process (Requirement Definition, Development and Deployment) in a project that used the modified methodology. This project had several modules that needed to be developed, was transversal to the client company's procedures and had a high impact on the company. It was developed in sprints by a team that varied in size during the project, fluctuating between 4 and 7 members, depending on the stage of the project. The core of the development team (4 members) remained for the whole project and was reinforced with additional developers at various times. The team leader is an Ingeniero Civil en Informática with 4 years experience developing similar applications. The development stage was supervised by the software development company's upper management and an external consultant with over 10 years experience in software development.

The team developed the software using SCRUM as the development methodology and, during this process, additional SNA information was collected which would later be used to provide information based on the new modified SCRUM methodology. (Social SCRUM or S-SCRUM). In some cases this information was used only to determine whether some key factors could have been established earlier in the project, and, in others, S-SCRUM was used to increase the project's chance of success.

8.2 Description of testing environment

The testing was done during a project developed for a company whose name is undisclosed. The software under development was a complex materials administration system. The final product will be used transversally throughout the company. Our project team consisted of a Team Leader and up to 6 developers depending on the stage of the project. The project team included an external consultant in the requirement definition and design phases. It was supervised closely throughout the development by upper management.

Since the work was done using a variant of SCRUM, all the normal Scrum elements were present. The project was developed in 2 week sprints, during which SNA Surveys were taken once a week. There were eight sprints, four of which were done using S-SCRUM survey and analysis work was done regarding all three development stages. Next all three testing environments will be described in detail including the testing methods and expected results. The structure of the formal organizational chart for the client company will be presented, using numbers assigned to each position to show which the organizations key users are.

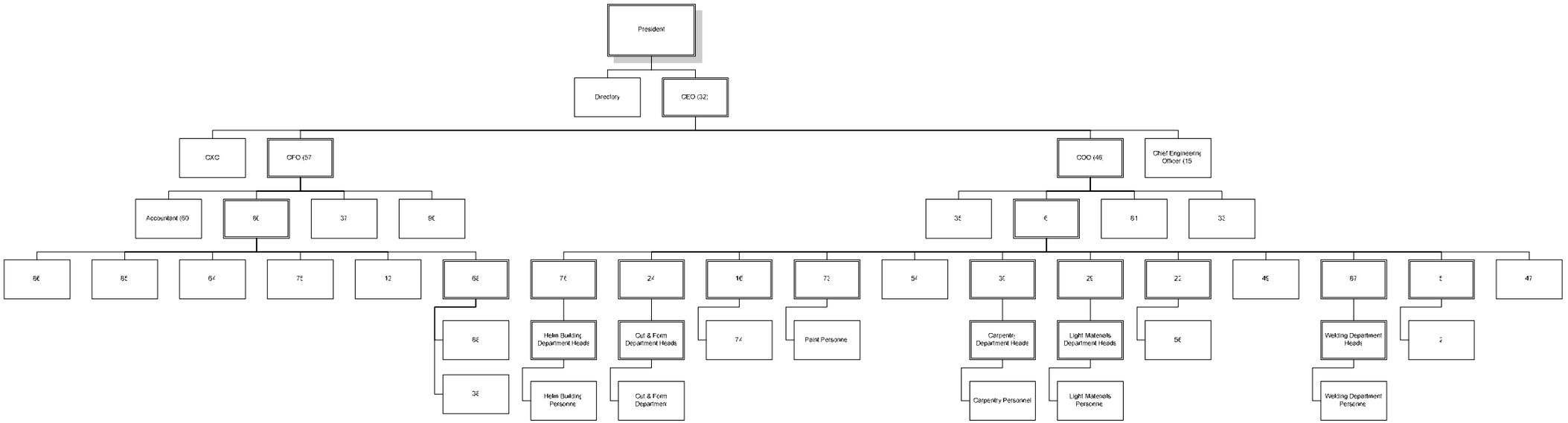


Illustration 11 Client formal Organizational Chart

Illustration 11 shows the formal organizational chart for the client company. The user list was used in the survey without including title or position. Later when showing the key user list, this chart can be used to determine a key user's position in the organization

Requirement definition:

For this stage we conducted an SNA analysis focused on the relevant set of processes. The system under development was for managing all materials requirement for a large manufacturing company. The manufacturing process was for extremely large and complex products and the materials made up approximately 70% of the total cost of the final product. The materials involved in the process were several thousand different items for each project, and every item could consist of one or more parts and in large quantities themselves. The process at hand was extremely complex and critical to the client's core business, affecting profits and losses directly.

In this case a different approach was used than the one that normally would be used in the modified methodology. The project was started before this work began and the requirement definition, design and prototyping were done as in any normal project. Subsequently, we ran a SNA survey regarding the materials administration process. The objective of this was to determine if any key users had been left out of the original requirement definition. It also became evident during design and prototyping that some users were missing from the requirement definition.

Our survey consisted of a Matrix where every user marked any other person in the organization with whom they communicated at least once a week regarding materials.

The survey was answered by the users and then the results were formatted for UCINET.

Development:

The testing during development was done using the same project team as in the requirement definition process. Periodic surveys were conducted to determine the evolution of the team's network during a project. The objective was to provide the team leader with information regarding his team's situation to help him better control the team's productivity. The intent was to help the leader foresee communication lapses or communication increases so that he could provide appropriate guidance. The surveys were conducted weekly and then formatted for analysis using UCINET.

Deployment:

For the deployment stage of the project the communications network was analyzed to determine key users regarding two issues important to a successful deployment of the system. The first one was the network concerning the materials administration process. Here our objective was to determine which key users concentrate a large amount of information traffic and therefore are key communicators for the network. These users are especially important during user instruction. One should ensure that these users are trained thoroughly because their opinion and use of the system will influence other users. If they do not use it correctly, given the amount of information they convey, the deployment could suffer immensely.

The second aspect is the analysis of the informal user support network. This information can show which users provide informal first hand technical or systems support to other users. The idea is to ensure that these users are extremely well trained, because they already enjoy the status of technical advisor for their workgroups, and they will probably feel comfortable continuing this role. This can be done explicitly with the users knowing or in a more discreet manner, in which the user won't feel this responsibility as a burden. This can be achieved in the second case is by paying extra attention to these users during training and just letting them continue in the role they already have assumed.

8.3 Result collection process

The results were collected in the way described below for each stage.

Requirement definition

For the requirement definition phase data was collected using direct SNA surveys within the client organization. The survey was focused on the materials administration process. The objective was to determine key users for the process in question; therefore a simple survey was constructed. The once per week survey measured users communication regarding materials.

The survey consisted of one question:

1. Who do you communicate with at least once a week regarding materials administration?

The possible choices included every individual in the client organization. The survey was answered by a relevant percentage of the organization, making sure that that proportion included all department heads and a similar proportion of every department (For example if 40% of the users was be surveyed, one should observe that 40% of each department was surveyed as well, as no to distort the results by having excessive or not enough information from one particular department.)

The results were gathered and logged into an Excel file, formatted for input into UCINET.

Development

During each sprint the development team was surveyed periodically to determine communication tendencies within the team and the effectiveness of integration of new team members. The following survey questions were used:

1 In respect to the project, who have you communicated with during this sprint and with what frequency?

2 Respect project requirements, who have you communicated with during this sprint and with what frequency?

3 Who have you communicated with during this sprint regarding technical issues and with what frequency?

In each question the possible answers included every other member of the development team, and every member of the project team (Client counterparts and development company management.) The results were collected and formatted for UCINET in an Excel file.

Deployment

For the deployment process the results were gathered using a simple SNA survey. There were two questions for this stage, both concerning the client organization:

1 Whom do you communicate with at least once a week regarding materials administration?

2 Who do you consult when in need of advice concerning information systems or their use?

The first question was the same as in the original survey for the requirement definition phase so that information was available. The second question was asked in the client organization. The possible answers for the question were every individual in the organization. The results were formatted for UCINET in preparation of the result analysis.

9 RESULTS AND ANALYSIS

9.1 Results

The results collected at each stage of the project were organized into three different sets of data (one for each stage) to allow easier analysis. Next some sample results are presented.

9.1.1 Requirement Definition:

After the data from the survey was formatted for Ucinet in an NxN matrix (where N is the number of users that were available as answers on our survey) Netdraw was used to visualize the resulting network.

Illustration 12 shows the social network for the client company regarding the process that was analyzed. It appears that several nodes are highly connected (e.g. 21, 22, 24, 29, 49, 51, 70) and might initially be candidates for consideration during the requirement definition phase.

Some of these are also interesting because of their position in the company;

21 Design engineer

22 Chief hydraulics and pipelines engineer

24 Chief cut and form engineer

29 Light Hull Chief Engineer

49 Chief metal workshop engineer

51 Engineering assistant

70 Head of acquisitions

Some of the people on this list were even before the SNA analysis considered key users, for example number 70. But some were real surprises, because no one, not even the client company considered them key to the process until they were pointed out. (Example: 51, engineering assistant.)

Nevertheless we will determine the true relevance analytically, the observation of the network is interesting, but it is hard to determine user relevance based solely on visual perception..

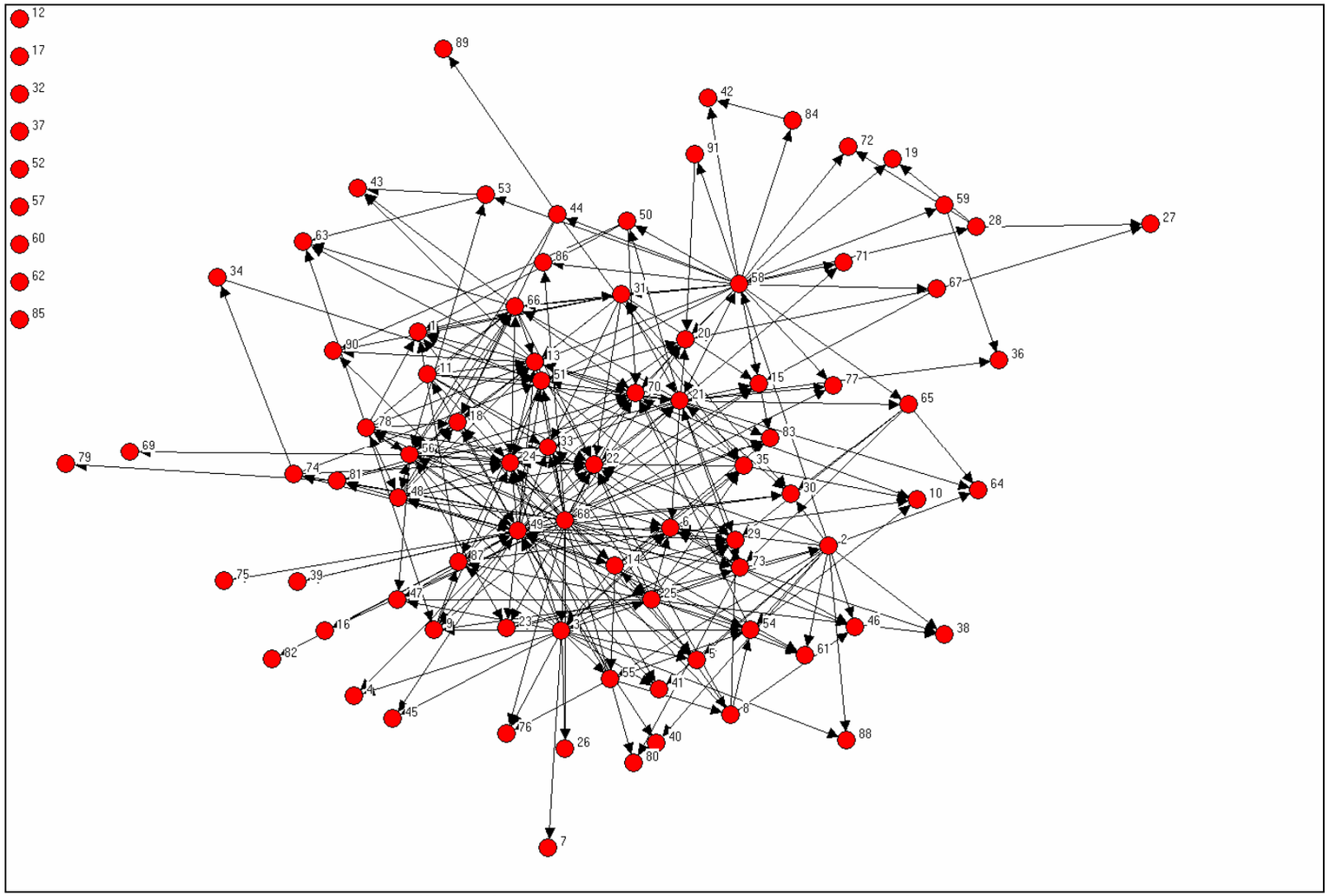


Illustration 12 Test Result Process SNA Network

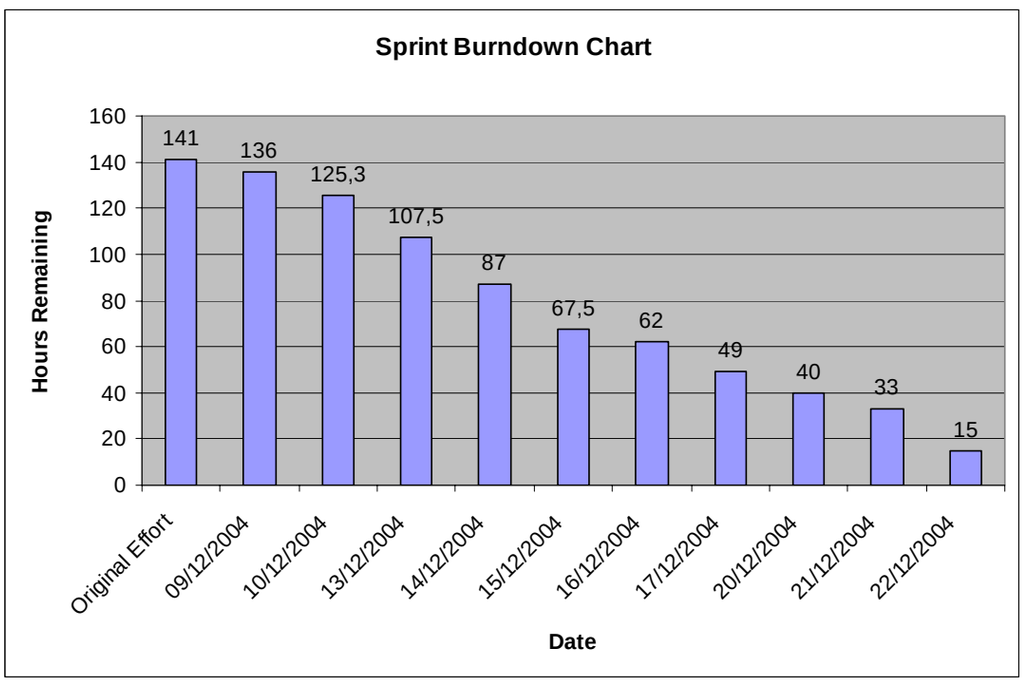
9.1.2 Development process

During this stage multiple sprints (4 with SNA) were conducted, first we will discuss a sample sprint: Table 18 shows a sample extract from a sprint backlog:

No.	Module	Description	Duration (Hours)
VERY HIGH PRIORITY			
1	Warehouse	Manage Warehouse movements	7
2	Warehouse	Input Deliveries to Warehouse	16
HIGH PRIORITY			
3	Warehouse	Tools Report	6
4	Warehouse	Project Consumption Report	6
MEDIUM PRIORITY			
5	Warehouse	Tools Reception Voucher printing	2
6	Warehouse	Print Tool delivery voucher	2
7	Warehouse	Screen for movement canceling among WH	2
8	Warehouse	Screen for canceling tool delivery	2
9	Warehouse	Edit material reception screen	6
10	Warehouse	Edit material delivery screen	6
LOW PRIORITY			

Table 18 Sample Sprint Backlog

Graph 8 shows the same sprint's Burndown chart.



Graph 8 Sprint Burndown Chart

The results for the development process were collected through multiple surveys that took place during the project development. The team was asked to answer the survey and the data was prepared for UCINET input. After several surveys at different periods of three different sprints we consolidated the data into one matrix simply by adding the values recorded for each survey and dividing by the number of surveys.

This was done because the basic network for the team almost didn't vary during the project. The only survey where it varied was one taken during a part of a sprint during which one of the developers was on vacation for a week. This particular survey was analyzed separately to determine the effect of the absence of a team member. Also during that same sprint an inexperienced developer was incorporated into the team for a short period of time to help with documentation. This will be considered in the analysis as well.

The results were weighted according to the following matrix

No. of Communications	Weight
<3 per week	1
>=3 per week	2
<4 per day	4
>=4 per day	8
>9 per day	12

Our resulting networks were formatted and put into UCINET, the network diagrams can be seen on the next page.

Team Project Communications

Normal Sprint Network

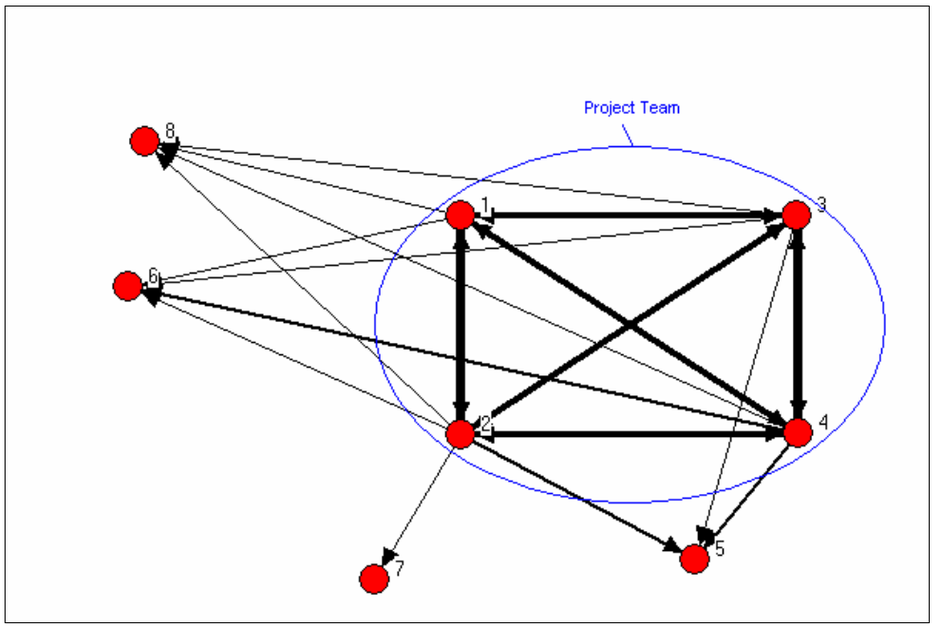


Illustration 13 Project Development Result Network (Whole Team)

Sprint Network with one developer missing and one new developer.

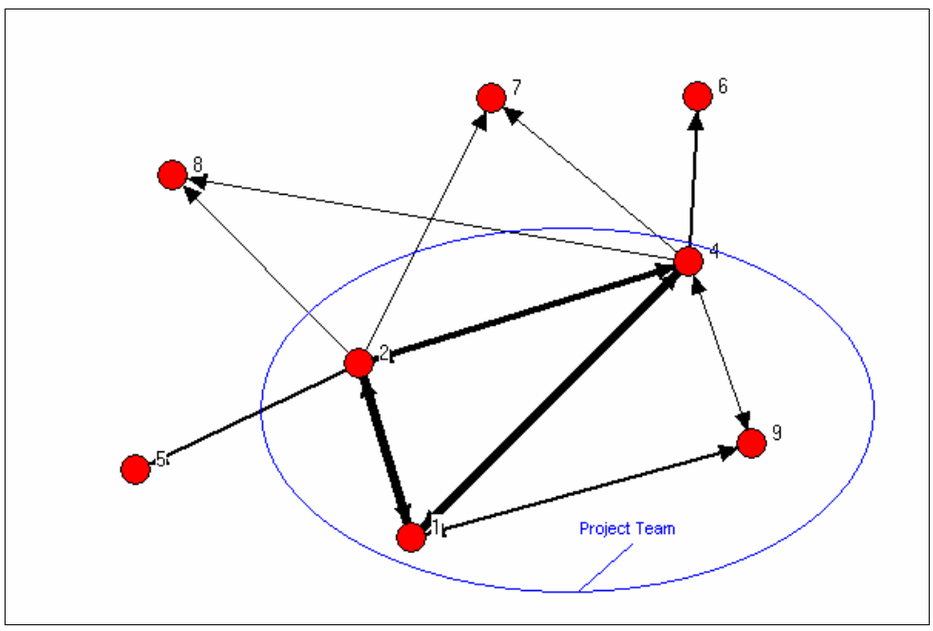


Illustration 14 Project Development Result Network (Modified Team)

9.1.3 Deployment

The deployment results are based on two separate sections of the survey, one is intended to determine key users regarding the problem being solved by the software implementation, the other is meant to be used to identify key users in the client company's informal technical support network. Based on this information the key users for consideration during testing and training will be identified.

Once the data from our survey was formatted for Ucinet in a NxN matrix (where N is the number of users that were available as answers on our survey) Netdraw was used to visualize the resulting network.

In the next pages both network diagrams will be shown. Illustration 15 is the network for the process under study, it can be seen that some nodes are heavily connected (e.g. 21, 22, 24, 29, 49, 51, 70) and others appear as not even part of the network (12, 17, 32, etc).

Their positions in the company are.

12 Accounting

17 Executive Director

32 CEO

21 Design engineer

22 Chief hydraulics and pipelines engineer

24 Chief cut and form engineer

29 Light Hull Chief Engineer

49 Chief metal workshop engineer

51 Engineering assistant

70 Head of acquisitions

Here it can be seen that not all the positions may seem relevant at first glance, but it all depends on the process under study. In this particular case the CEO was not relevant to the requirement definition whereas the engineering assistant seems to be. Even so, we will not use this information unless it is corroborated by out metric analysis.

In the second network the diagram (Illustration 16) shows the informal technical support network. As in the first network, some users are evidently key to maintaining the network. Nevertheless we will abstain from graphical analysis because of its inaccuracy.

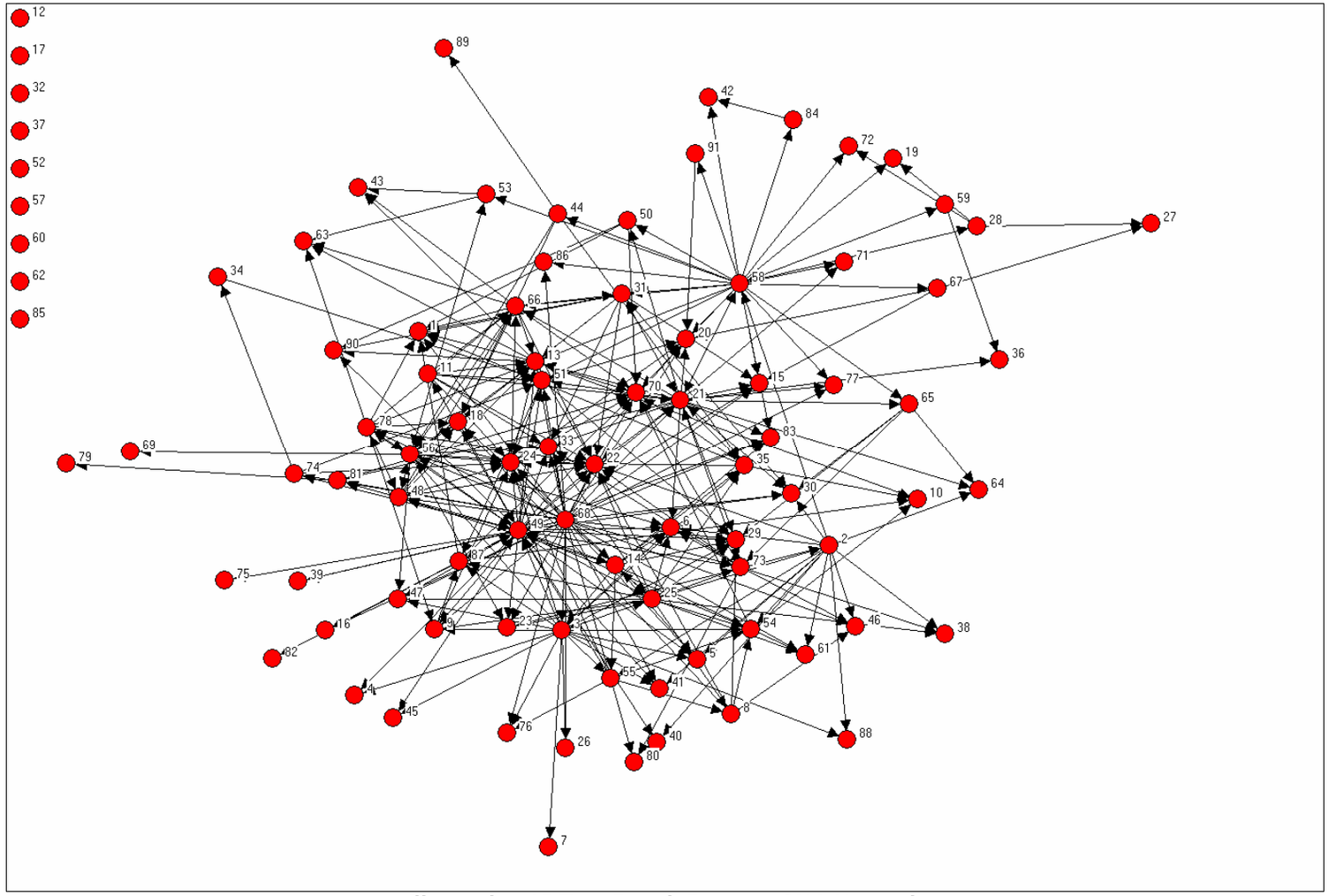


Illustration 15 Test Result Process SNA Network

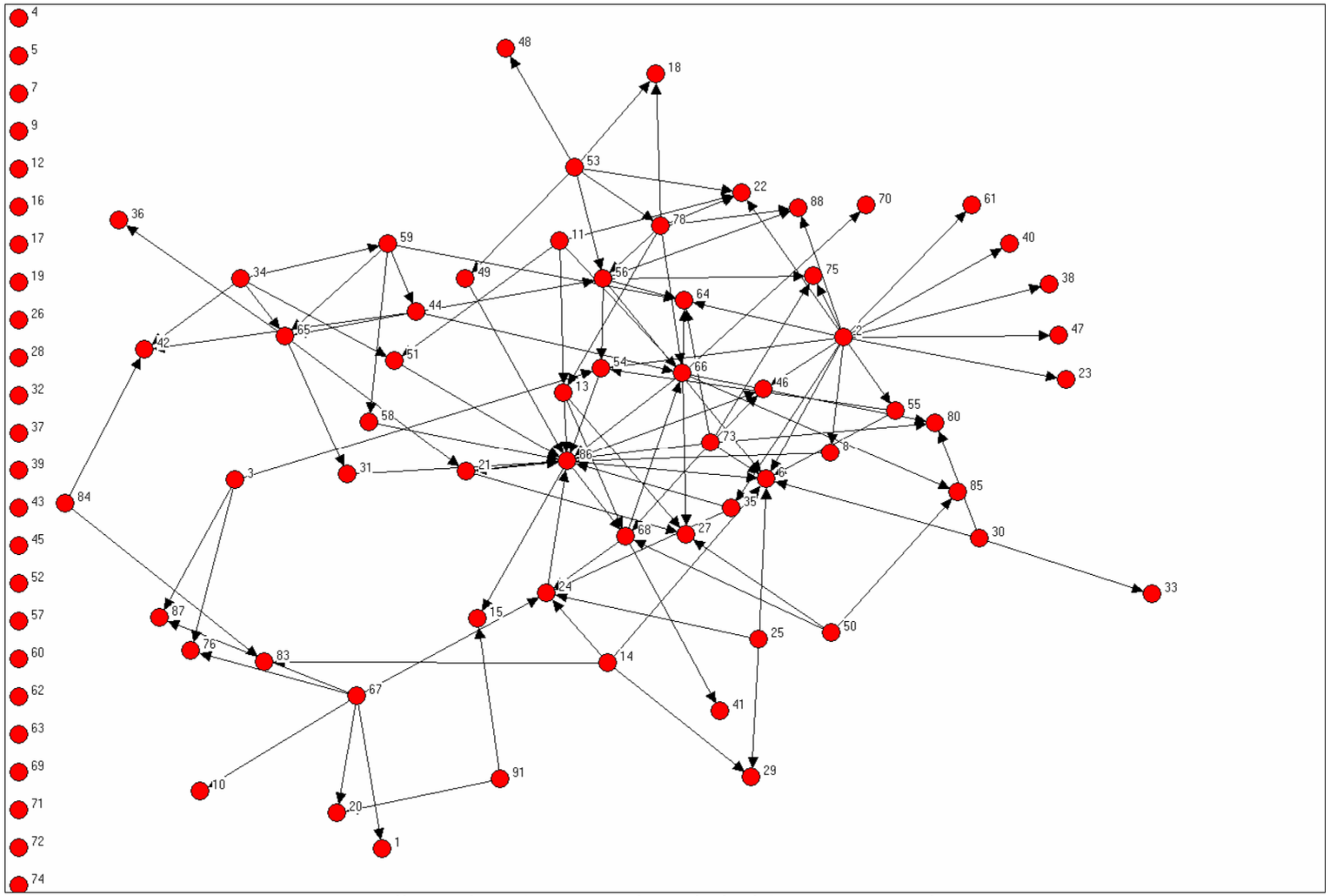


Illustration 16 Test result Technical Support Network

9.2 Analysis

9.2.1 Requirement Definition

According to our methodology, closeness and betweenness were calculated, and cutpoints were found. The data was formatted in a according to the next example:

	A	B	C	D
A	0	1	0	1
B	1	0	0	0
C	1	1	0	0
D	0	1	0	0

Table 19 Sample UCINET input format

The data was entered into UCINET and converted to Pajek format using the UCINET data format conversion tool.

Betweenness	Node
0.1378916	49
0.0835078	56
0.0804205	21
0.0538272	58
0.0377274	66
0.0348624	13
0.0315775	24
0.0206380	31
0.0202217	25
0.0128185	14
0.0127521	73
0.0127498	3
0.0116022	51
0.0059521	30
0.0058272	44
0.0051769	2
0.0042075	55
0.0036955	81
0.0033708	74
0.0025189	53
0.0020787	48
0.0018102	28
0.0018102	67
0.0015940	35
0.0015876	87
0.0014161	78
0.0009197	54
0.0008739	65
0.0005339	8
0.0003622	50

Table 20 Result Network Betweenness

Closeness	Node
0.5614539	49
0.5529471	68
0.5140071	21
0.4865934	13
0.4739546	58
0.4590504	24
0.4561813	51
0.4505495	22
0.4477853	25
0.4477853	56
0.4450549	20
0.4450549	70
0.4370599	3
0.4370599	15
0.4344584	66
0.4318876	2
0.4318876	14
0.4318876	83
0.4293471	78
0.4243547	6
0.4243547	18
0.4243547	29
0.4243547	33
0.4243547	73
0.4219018	48
0.4100506	30
0.4100506	35
0.4077598	11
0.4077598	23
0.4054945	77

Table 21 Result Network Closeness

Note: the cells colored in blue are network cutpoints. They can be identified in the graph on the following page.

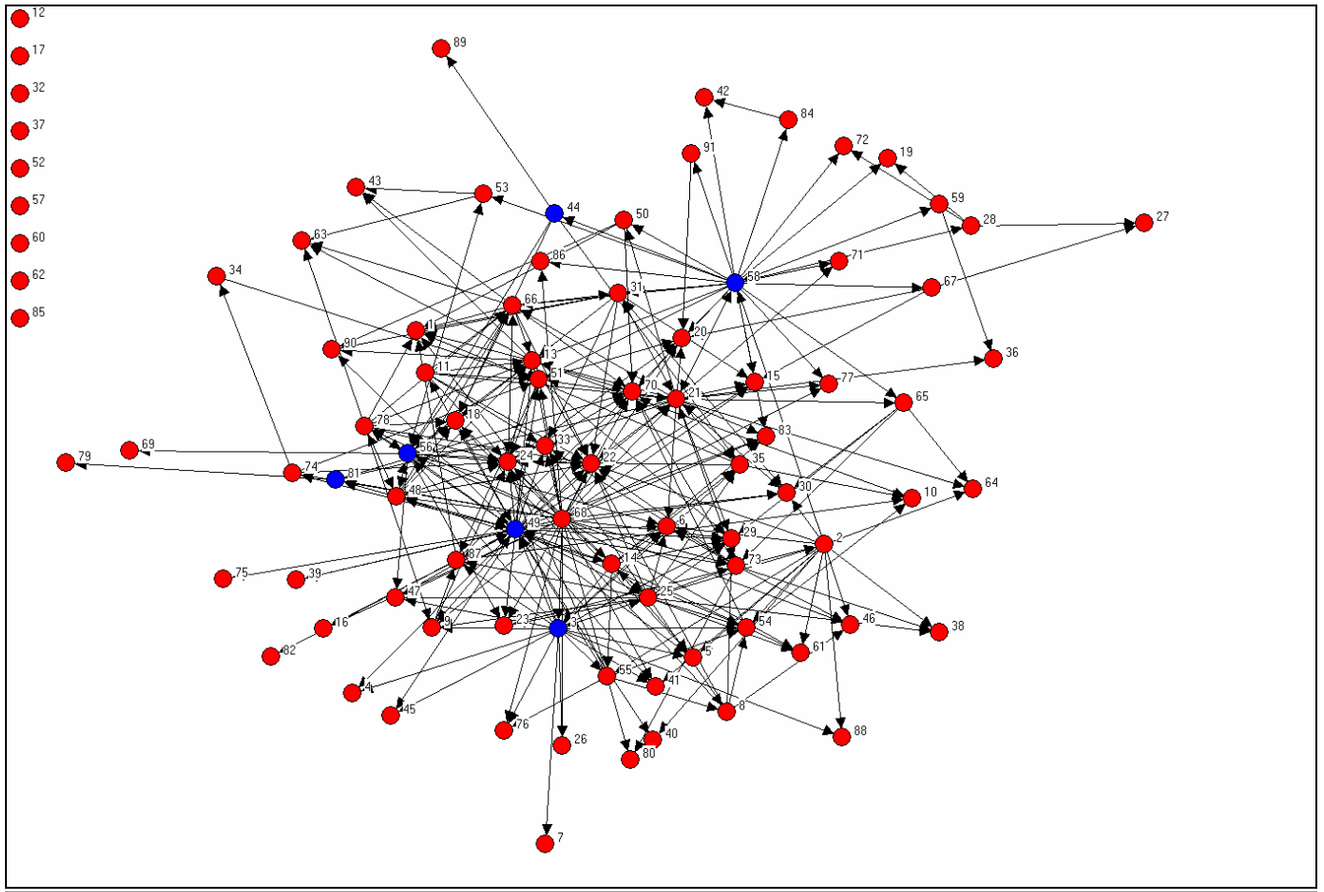


Illustration 17 Result Network Cutpoint Analysis

In this illustration we can see the client's complete network; all the points marked in blue are cutpoints for the network. This means that if they were to be removed from the network some parts of it would be disconnected.

9.2.2 Development Process

The results for the development process should focus on determining what information is useful for determining what readiness level is appropriate for each team. The results were condensed for this purpose and some of the collected data was not used. For example, tech support information is not immediately applicable to our model.

The data resulted in the following diagram (Illustration 18).

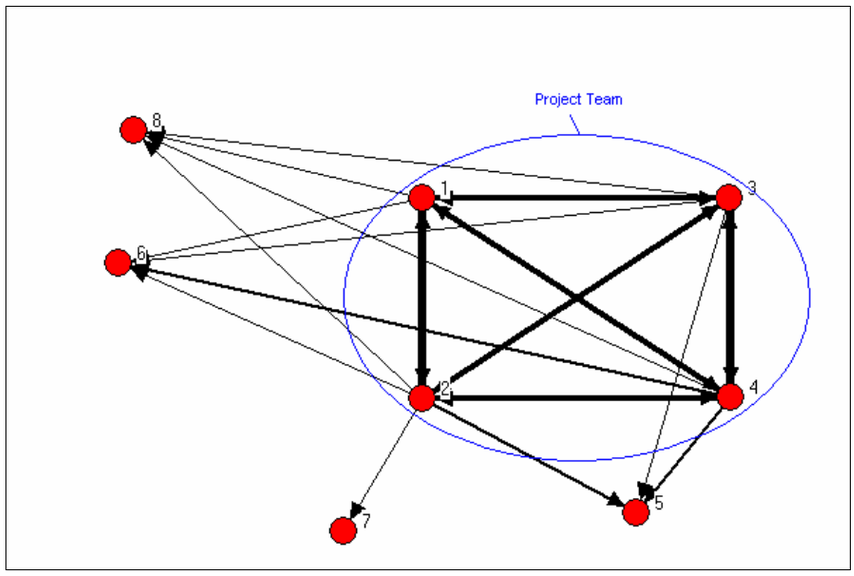


Illustration 18 Project Development Result Network (Whole Team)

This network appears at first hand to be a mesh, our model was used to determine its Meshness and Directionality values.

The resulting values are:

Meshness	0,89361702
Directionality	0,95833333

Next (Illustration 19) the values for our second network were calculated. In this network one of the developers was absent for a sprint and a newly recruited developer was incorporated into the team.

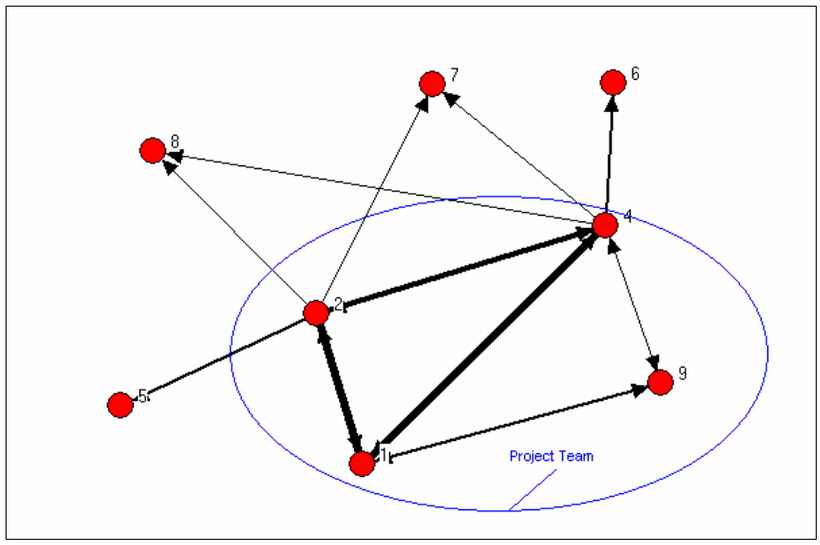
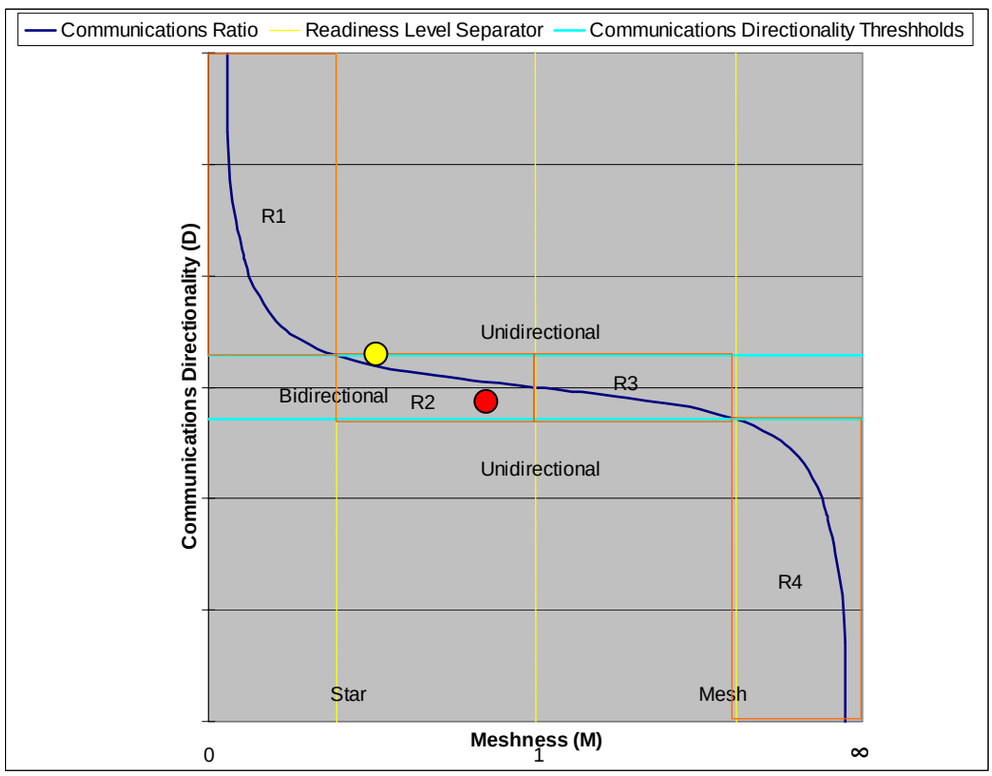


Illustration 19 Project Development Result Network (Modified Team)

The resulting values for Meshness and Directionality for this network are:

Meshness	0,66666667
Directionality	1,29411765

When placing both points in our model and using our untested thresholds, it can be seen that they both fall into the R2 quadrant. The red dot corresponds to the original Team and the yellow one represents the modified team.



Graph 9 Test result evaluation using experimental model

9.2.3 Deployment

The data was entered into UCINET and converted to Pajek format using the UCINET data format conversion tool.

As we stated earlier thee data for the first network is the same as for the requirement definition process, so the same results are applicable.

Betweenness	Node
0.1378916	49
0.0835078	56
0.0804205	21
0.0538272	58
0.0377274	66
0.0348624	13
0.0315775	24
0.0206380	31
0.0202217	25
0.0128185	14
0.0127521	73
0.0127498	3
0.0116022	51
0.0059521	30
0.0058272	44
0.0051769	2
0.0042075	55
0.0036955	81
0.0033708	74
0.0025189	53
0.0020787	48
0.0018102	28
0.0018102	67
0.0015940	35
0.0015876	87
0.0014161	78
0.0009197	54
0.0008739	65
0.0005339	8
0.0003622	50

Table 22 Result Network Betweenness

Betweenness	Node
0.5614539	49
0.5529471	68
0.5140071	21
0.4865934	13
0.4739546	58
0.4590504	24
0.4561813	51
0.4505495	22
0.4477853	25
0.4477853	56
0.4450549	20
0.4450549	70
0.4370599	3
0.4370599	15
0.4344584	66
0.4318876	2
0.4318876	14
0.4318876	83
0.4293471	78
0.4243547	6
0.4243547	18
0.4243547	29
0.4243547	33
0.4243547	73
0.4219018	48
0.4100506	30
0.4100506	35
0.4077598	11
0.4077598	23
0.4054945	77

Table 23 Result Network Closeness

Betweenness	Node
0.0285893	86
0.0212235	68
0.0169580	66
0.0048689	54
0.0047441	24
0.0023720	56
0.0022472	44
0.0014773	13
0.0014357	21
0.0013733	59
0.0013109	65
0.0010404	51
0.0009988	78
0.0008115	58
0.0007491	31
0.0005618	49
0.0004994	35
0.0003745	8
0.0000000	1
0.0000000	2
0.0000000	3
0.0000000	4
0.0000000	5
0.0000000	6
0.0000000	7
0.0000000	9
0.0000000	10
0.0000000	11
0.0000000	12
0.0000000	14

Table 24 SNA Analysis Tech Support Network Betweenness

Betweenness	Node
0.3166912	86
0.2872841	6
0.2852467	66
0.2754779	2
0.2681319	54
0.2681319	24
0.2646038	68
0.2611674	35
0.2578191	64
0.2561769	56
0.2482702	73
0.2467471	46
0.2422878	13
0.2422878	8
0.2408370	78
0.2408370	44
0.2394035	21
0.2324843	51
0.2311482	49
0.2311482	14
0.2298273	53
0.2285215	55
0.2272304	75
0.2272304	58
0.2272304	31
0.2272304	22
0.2246915	88
0.2246915	25
0.2209878	15
0.2209878	11

Table 25 SNA Analysis Tech Support Network Closeness

Note: the cells colored in blue are network cutpoints. They can be identified in the graphs depicting the cutpoint analysis results on the following page.

As can be seen from the data several nodes have high scores for both betweenness and closeness.

In the following illustrations (Illustrations 20, 21), all the points marked in blue are cutpoints for the network. This means that if they were to be removed from the network some parts of it would be disconnected.

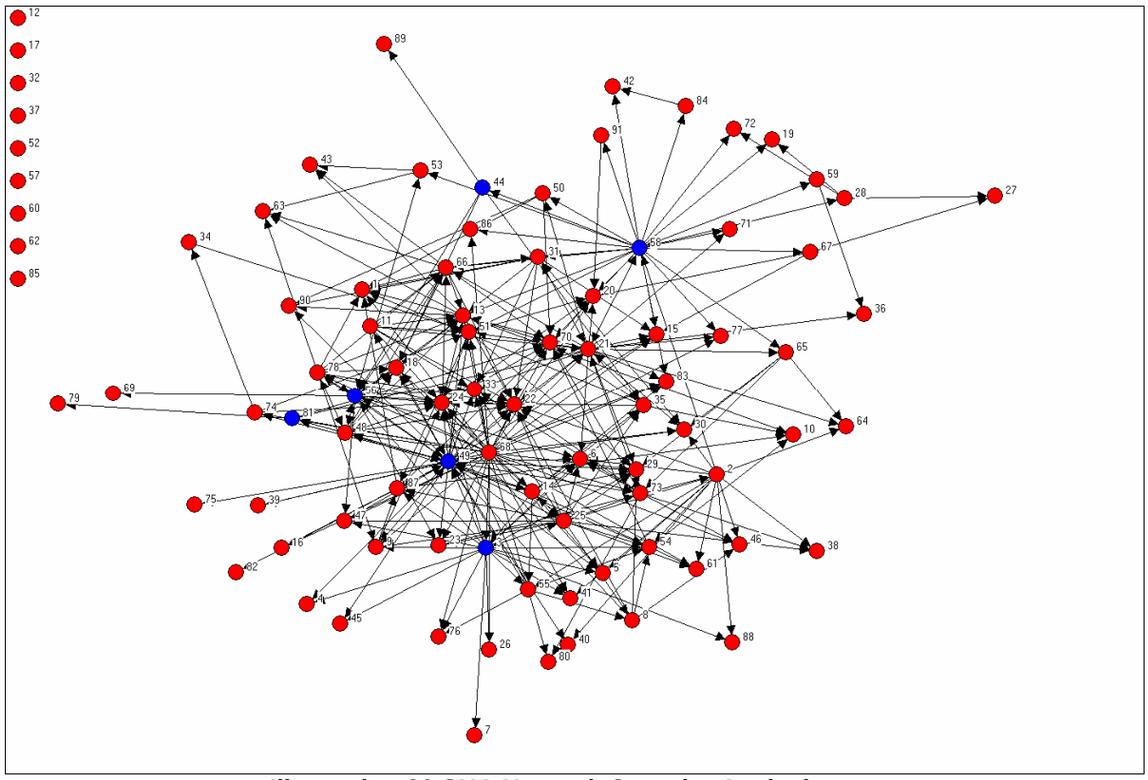


Illustration 20 SNA Network Cutpoint Analysis

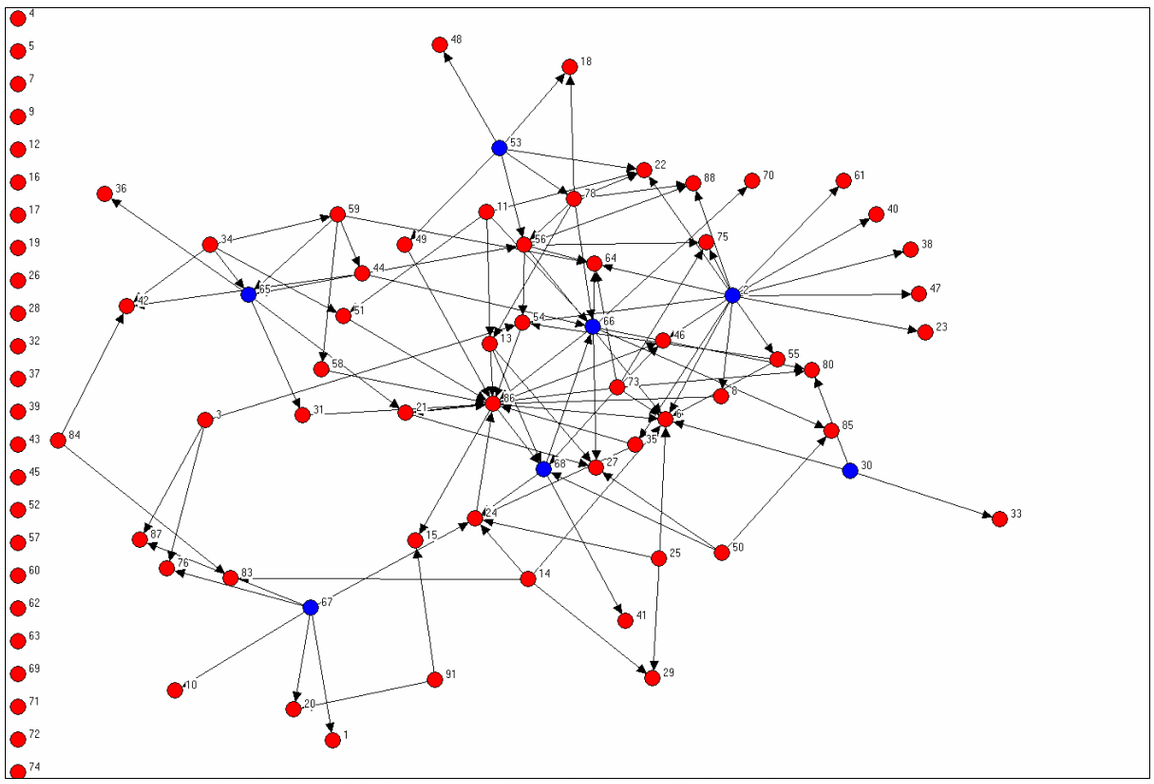


Illustration 21 Technical SNA Network Cutpoint Analysis

9.3 Result Interpretation and conclusions about the experiment

In this section our conclusions about this experiment are discussed. Each stage of the development process ise analyzed and conclusions are made.

One of the first conclusions is that unnecessary data was recorded; only some of the data was necessary for the proposed modification to work. We recorded extra data during testing because the model was untried and it was not clear what problems might be encountered that might require additional information.

9.3.1 Requirement definition

Result interpretation

As can be observed in the results, the network has several users with high betweenness and closeness ratings and also 5 different cutpoints. Based on this information a list of key users was selected, the selection process was done by checking which users were listed in the top 30 for both lists or was in the top 30 for one list and also was a cutpoint. It was decided that 30 was an appropriate number because it was roughly 32% of the users and it was considered that interviewing 30% of them would be more than enough. If all of the 30 users were key users then the list would have to be extended. The resulting key user list was the following:

Key User List	B	C	CUT
49	0.1378916	0.5614539	X
56	0.0835078	0.4477853	X
58	0.0538272	0.4739546	X
44	0.0058272		X
81	0.0036955		X
13	0.0348624	0.4865934	
14	0.0128185	0.4318876	
21	0.0804205	0.5140071	
24	0.0315775	0.4590504	
25	0.0202217	0.4477853	
30	0.0059521	0.4100506	
35	0.0015940	0.4100506	
48	0.0020787	0.4219018	
51	0.0116022	0.4561813	
66	0.0377274	0.4344584	
68	0.0018102	0.5529471	
70	0.0015876	0.4450549	
73	0.0127521	0.4243547	
78	0.0014161	0.4293471	

Table 26 SNA Process results key users

The nodes are ordered as they were in the alphabetical list they were extracted from. The order in which they were extracted was;

2. Nodes which are cutpoints and are on the top 30 for either closeness or betweenness.
3. Nodes which are in the top 30 for both closeness and betweenness.

The colored rows are coded according to the following table.

Color Code	
User included in original interview	
User Interviewed at later time	
User not interviewed	

Conclusions

Some of the users were interviewed originally and were defined as key users by the client organization. Nevertheless, several key users were discovered later in the project and some of the key users in the list were never interviewed. Based on these observations we can conclude that the new method allows key users to effectively be determined early on in the project at a low cost. This may result in fewer errors in requirement definition and overall lower project costs. It cannot be said that it will include all relevant users, or that all users included are relevant, but it provides a good approximation to what an appropriate set of key users should be and is valuable information to discuss to ensure that requirements are properly defined.

9.3.2 Development Process

Result interpretation

The results were shown to the team leader, who disagrees slightly with the results. In his view the original unmodified team should be categorized as an R3 and the modified team could be R2 because the new developer was inexperienced. Given the team's relatively small size, any changes affect the overall results highly because they aren't diluted into a larger team's overall performance. Nevertheless the results seem to be relevant because the thresholds for directionality and meshness have not yet been established. In general the results seem to be consistent with our model. The results fit into the model closely, and the tendency curve seems to work appropriately. The results were useful for the team leader, and, after discussing the model and the results among several other team leaders, there was consensus that the l model is useful. Also the situational leadership model is consistent with their view of how a team should be led.

Conclusions

When considering the results, and the team leader's interpretation, the model generally seems to be correct. The thresholds need to be determined empirically so that the model is complete, but the general model (tendency line, classification method, formulae) appears to be useful.

With more work and additional data, other interesting aspects could be studied, for example the impact on a teams Readiness level when a team member is changed, someone leaves the team or two teams are merged. This could lead to further work on team behavior and management.

9.3.3 Deployment

Result interpretation

The same methodology was used to determine key users as in the requirement definition phase. This resulted in two lists of key users, one for each question in the survey. The first list is of users who are important to the process, and the second is a list of users that are relevant to the informal technical support network.

The nodes are ordered as they were in the alphabetical list they were extracted from. The order in which they were extracted was;

4. Nodes which are cutpoints and are on the top 30 for either closeness or betweenness.
5. Nodes which are in the top 30 for both closeness and betweenness.

Key User List	B	C	CUT
49	0.1378916	0.5614539	X
56	0.0835078	0.4477853	X
58	0.0538272	0.4739546	X
44	0.0058272		X
81	0.0036955		X
13	0.0348624	0.4865934	
14	0.0128185	0.4318876	
21	0.0804205	0.5140071	
24	0.0315775	0.4590504	
25	0.0202217	0.4477853	
30	0.0059521	0.4100506	
35	0.0015940	0.4100506	
48	0.0020787	0.4219018	
51	0.0116022	0.4561813	
66	0.0377274	0.4344584	
68	0.0018102	0.5529471	
70	0.0015876	0.4450549	
73	0.0127521	0.4243547	
78	0.0014161	0.4293471	

Table 27 SNA Process results key users

Key User List	B	C	CUT
2	0.0000000	0.2754779	**X**
53		0.2298273	**X**
65	0.0013109		**X**
66	0.0169580	0.2852467	**X**
68	0.0212235	0.2646038	**X**
8	0.0003745	0.2422878	
11	0.0000000	0.2209878	
13	0.0014773	0.2422878	
14	0.0000000	0.2311482	
21	0.0014357	0.2394035	
24	0.0047441	0.2681319	
31	0.0007491	0.2272304	
35	0.0004994	0.2611674	
44	0.0022472	0.2408370	
49	0.0005618	0.2311482	
51	0.0010404	0.2324843	
54	0.0048689	0.2681319	
56	0.0023720	0.2561769	
58	0.0008115	0.2272304	
78	0.0009988	0.2408370	
86	0.0285893	0.3166912	

Table 28 SNA results technical network key users

Based on these results a deployment plan could be formulated contemplating extra attention for the users on these lists, thus providing two benefits:

I. The users on the first list if properly trained will contribute to a better system acceptance and understanding. And also will ensure that key information flows correctly throughout the system.

II. The users on the second list, if properly trained will help reduce system learning cost for the other users and reduce help desk or retraining efforts by serving as an informal first hand support network.

Conclusions

Even though this part of the methodology could not be fully tested, after discussing it among other software developers, they thought it was a very reasonable approach and could reduce effort and costs in the deployment process, especially by improving the clients' perception of the new systems and reducing help desk effort. As in the two previous stages, further testing are needed to draw final conclusions about the proposed methodology modifications.

10 CONCLUSIONS

This work was intended as a way to improve software development and provide analytical tools to support decision making in the process. The resulting findings and methodology may be able do so fairly effectively. Much further work is needed before the methodology is completed and proved, but the initial models seem to be correct.

In the general for this project the following intentions were stated:

1. Generate an improvement to a specific software development methodology (SDM) that allows a better definition of the environment of the design process, better control and earlier warning of critical situations during development and a smoother and more cost-effective implementation process for both developers and clients.

2. Through a better and more informed design process, obtain systems that are more in accord with an organization's work method, thus enabling better adoption of the new system and a longer system life span.

Through the modification of SCRUM and the development of S-Scrum both objectives have been addressed and resolved effectively. The modified methodology although untried is a step forward in providing project managers with effective real time information about different stages of project development. Through this new additional information systems will be designed more in accord to the client's real requirements, the development process will be controlled more effectively and implantation will be much easier and less costly for the development company and the client organization.

Our specific objectives

1. Select an adequate methodology for the incorporation of Social Network Analysis (SNA) into the different stages of development.

   After careful review and analysis of all our options SCRUM was selected, the justification is clear in previous chapters, but at this point it is clear that our choice was correct. SCRUM is a flexible, efficient and easy to use methodology which was ideally suited for our purposes.

Scrum is also extremely interesting because of the possibility of using it as an external control wrapper for any other methodology, thus the selection of SCRUM as our platform allows S-Scrum techniques to be used in conjunction with any other methodology.

2. Structure the SNA process to optimize it for incorporation into the SDM.
3. Modify the SDM to incorporate SNA

It was possible to structure and integrate SNA into the methodology almost seamlessly and the resulting method is effective and unobtrusive in the overall methodology.

4. Evaluate its efficiency through testing

The test results, even though not conclusive are a clear indication that the methodology has potential and could be developed with some fine tuning into an effective process.

5. Analyze the test results and document the entire new SDM.

In our final chapters this final objective is resolved completely.

10.1 Possibilities for future work

One of the main concerns with the new development model is that it needs to be fine tuned extensively, all the results are within the expected parameters but for the model to be of use all the thresholds need to be determined, the model needs to be tested, and final conclusions need to be made concerning all issues still standing.

A possible line of experimentation would be to take the process as a whole and break it down into its three separate stages and experiment extensively with each isolatedly. Each stage needs to be verified against certain hypothesis that can only be considered as true if proven through experimentation.

The required work for each stage that can be foreseen would be:

Requirement definition:

- Standardize the survey process
- Determine percentages of users that should be included in key user lists
- Research other metrics that could complement the ones used until now.

A possible hypothesis to corroborate: The use of S-SCRUM key user definition lowers the risk of wrong requirement definition and because of this lowers project development cost overruns.

Development process:

- Empirically determine all the required thresholds
- Test the mathematical model extensively with other projects.
- Discuss results with as many team leaders as possible.

A possible hypothesis would be that the model can be used effectively to determine readiness levels. If this can be proven the model can be used to understand what happens to teams under different situations.

Deployment:

- Standardize the survey process
- Determine percentages of users that should be included in key user lists
- Research other metrics that could complement the ones used until now.
- Determine the effect of using this methodology on deployment costs and client perception of project success.

Hypothesis proposal: S-Scrum lowers deployment costs and enhances user perception of product quality.

For each stage a detailed experimentation model would have to be defined, and extensive test would have to take place for the model to be finished, but the work so far is interesting and would be useful if the models were complete.

SNA and Situational Leadership seem to be of extreme use when applied to software development. The results show that once these processes are proven and the models are debugged, S-SCUM or even only the models could possibly be used effectively in any type of project. In particular the SNA/Situational Leadership model proposed for the development process could be applied for any type of project team, not only software development.

The work done so far completes the objectives of this thesis, but for the methodology to be able to be used much more work is required. It is an interesting stepping stone in an effort to improve software development, but so far it is far from finished.

11 BIBLIOGRAPHY

1. **Fowler, Martin**. “Who Needs an Architect?” *Software, IEEE* Volume: 20, Issue: 5, Sept.-Oct. 2003
2. **Takeuchi, Hirotaka and Nonaka, Ikujiro**. “The New Product Development Game” *Harvard Business Review.* Jan/Feb 1986
3. **Hersey, Paul; Blanchard Kenneth; Johnson, Dewey**, 2001 "Management of Organizational Behavior" *Prentice Hall*
4. **Fowler, Martin**. ”The New Methodology” *Unpublished paper, Martinfowler.com*, July 2000.
5. **Ogunnaike, Babatunde**. “Process Dynamics, Modeling, and Control”*. Oxford University Press*, 1994
6. **Khan, Ali and Balbo, Sandrine**, “A Tale of two Methodologies: Heavyweight versus Agile” Presented at: AusWeb04. The Tenth Australian World Wide Web Conference.
7. **Krebs, Valdis** Introduction to Social Network Analysis *Unpublished paper* Orgnet
8. **Schwaber, Ken and Beedle, Mike**, 2002 "Agile Sofware Development with Scrum" *Prentice Hall*
9. **Warsta, Juhani et al**. 2002 “Agile Software Development Methods: Review and Analysis.” *VTT Publications*
10. **Cohn, mike Ford, D** “Introducing an Agile Process to an Organization” *IEEE Computer* Volume Issue 6), June 2003 p p. 74-78
11. **Highsmith,Jim.** 2002 “Agile Software Development Ecosystems” *Addison Wesley*
12. **Cohn, Mike** “Selecting an Agile Process: Choosing Among the Leading Alternatives” SD Best Practices Conference and Expo 2004, September 21, 2004
13. **Cohn, Mike** “Toward a Catalog of Scrum Smells” *Unpublished paper* Mountain Goat Software
14. **Krackhardt, David; Hanson, JR** “Informal networks: The company behind the chart“ *Harvard Business Review,* Volume 71, Jul-Aug 1993 pp 104-11.
15. **Zack, Michael H.** “Researching Organizational Systems using Social Network Analysis” Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000

16. **Cohn, Mike** “Situational Leadership for Agile Software Development” *Cutter IT Journal*, June 2004
17. **Noyes, Brian** Rugby, Anyone? *Unpublished paper*
18. **Freeman, Linton** “Centrality in Social Networks Conceptual Clarification” *Social Networks*, Volume 1 (1978/79) pp 215-239

12 GLOSSARY OF TERMS

A
Agile Methodology: Software development methodology that is defined al light or easy to implement and use

E
ERP: Enterprise Resource Planning generally refers to an ERP System.

F
FDD: Feature Driven Development

H
Heavyweight Methodology: Software development methodology that is heavily structures and normally is related to a high overhead.

P
PSP: Personal Software Process

S
SCRUM: Agile Software Development Methodology by Ken Schwaber among others
Scrum Master: Scrum development team leader.
SDM: Software development Methodology
SNA: Social Network Analysis

T
TSP: Team Software Process

U
UCINET: SNA Software Suite.
UML: Unified Modeling Language

X
XP: Extreme Programming

13 ANNEX 1 Social Scrum Methodology

Social Scrum or S-Scrum is a modified version of SCRUM all the normal process elements in Scrum are present. These include Backlog, Sprints, Scrum Meetings, and Demos. Additionally there are elements that are inserted into the methodology. The following document focuses on explaining the basics of SCRUM and the new elements introduced by S-SCRUM, for more detailed information on SCRUM please refer to original methodology documentation, for example “Agile Software Development With Scrum” by Ken Schwaber and Mike Beedle.

- Problem Process SNA Analysis
- Team Sprint SNA Analysis
- Technical Support Network SNA Analysis

Next we will describe these elements (Scrum and S-Scrum) briefly.

Backlog consists of the list of prioritized requirements or features the customer wants the team to build. Backlog is a dynamically changing list that management constantly reassesses. The customer, marketing department, and developers can all add new items to the Backlog, but only the product manager is allowed to change the items' priorities.

Sprints are the basic units of scheduling, typically 30 days or less of development activity. A Sprint consists of a preallocated work unit from the Backlog that composes the work and features to be completed during the Sprint. During a Sprint, the Backlog that the Sprint addresses is static—no new items may be added and the prioritization remains fixed for that portion of the Backlog.

Scrum Meetings are short daily meetings (targeted at 15 minutes) held without fail by the Scrum team. Each team member is expected to answer three standard questions: What did you do since the last team meeting? What obstacles are you encountering? What do you plan to accomplish by the next team meeting? A Scrum Master leads the meetings and tracks the responses from all team members. This information is the primary source of process measurement, monitoring, and documentation. The key idea

behind Scrum Meetings is: If you have a function or process that is difficult to predict, sample more frequently to determine where things are going. This provides better feedback for both managers and developers, and it keeps people synchronized with the true progress of the whole team.

Finally, a **Demo** is the culmination of a Sprint and results in the functionality delivered to the customer. This might be an informal or formal delivery, or it might simply be a demonstration of the work the team has achieved to date. The key here is that the work in a Sprint must be organized so the end result is a piece of functioning software that shows the new capabilities added.

Problem Process SNA Analysis is a SNA analysis focused on the process or processes that are being studied and implemented as software. The objective is to determine key users as early in the project as possible so that the requirement definition can consider the best possible choices for interviewing. If this is done correctly it considerably reduces the chances that key users won't be detected when the project team is being defined. If the team is defined correctly, and all the key users are interviewed chances are, project requirements will be more stable later in the project.

Team Sprint SNA Analysis is a periodical analysis of the team's communications; it can be used to determine the team's readiness level. Using the readiness level as a parameter for performance it could possibly also be used to determine problems in the team with time to correct them before they generate losses or performance deterioration within the project. This technique also can be used on a long term basis to illustrate the effects of any kind of change in the team's environment on the performance levels.

The **Technical Support Network SNA Analysis** will be used to determine key users within the informal systems support group. This means users that normally help others with technical issues regarding the Information Systems and who are not part of the official support staff. These candidates will later be trained with more care to ensure that they will continue to do so.

Normal Scrum Process

The following diagram shows a normal scrum development lifecycle.

1. The development starts with a defined Product Backlog. This Backlog has been defined by management and the development team. It is comprised of a set of loose definitions of every requirement or task for the whole project.
2. The next stage is to define a subset of work that can be finished within the defined time period assigned for the sprint (normally 1 month).
3. Following the definition of the Sprint Backlog the team on goes on to expand the backlog, so that every item is comprehended in its entirety. Each individual task is discussed, defined and written down in more detail at the beginning of the sprint. This will allow the team to determine the effort needed to complete each task. Every task is also prioritized so that they can be resolved in that order. Here is an example excerpt from a Sprint Backlog.

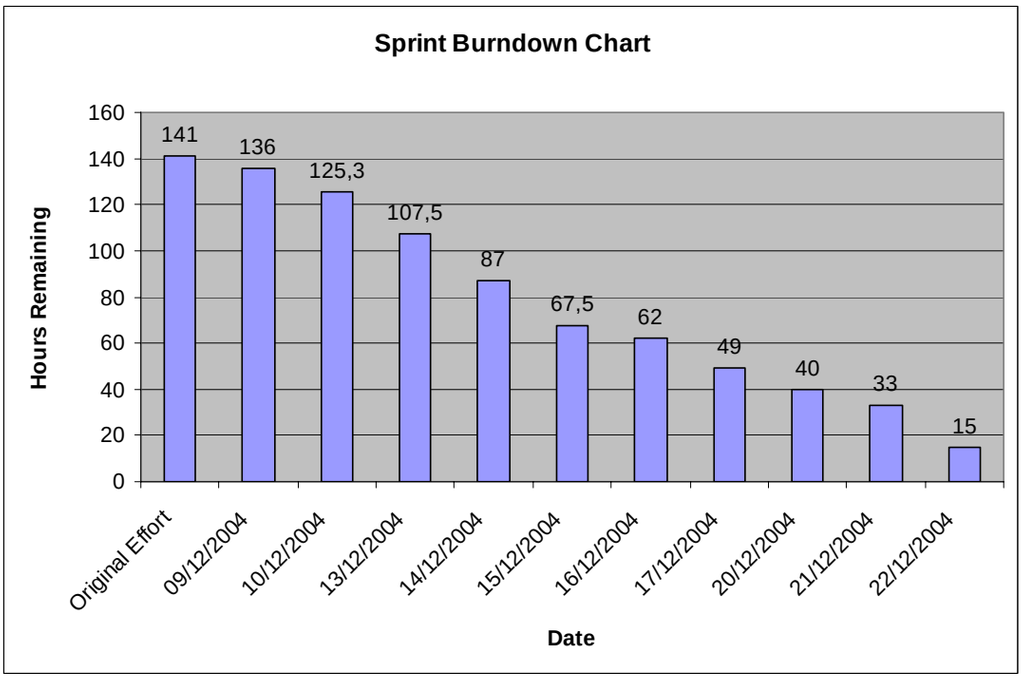
No.	Module	Description	Duration (Hours)	
VERY HIGH PRIORITY				
1	Warehouse	Manage Warehouse movements	7	
2	Warehouse	Input Deliveries to Warehouse	16	
HIGH PRIORITY				
3	Warehouse	Tools Report	6	
4	Warehouse	Project Consumption Report	6	
MEDIUM PRIORITY				
5	Warehouse	Tools Reception Voucher printing	2	
6	Warehouse	Print Tool delivery voucher	2	
7	Warehouse	Screen for movement canceling among WH	2	
8	Warehouse	Screen for canceling tool delivery	2	
9	Warehouse	Edit material reception screen	6	
10	Warehouse	Edit material delivery screen	6	
LOW PRIORITY				

Table 29 Sample Sprint Backlog

4. The next step in the process is to start the SPRINT. The sprint starts with a Scrum in which all the team comes together, decides on who will do what particular task and each individual proposes some work that will be his responsibility and which he will report on

during the following scrum meetings. It is essential that the team is allowed to self organize.

5. At each daily meeting each developer reports to the team what percentage of the work he has completed, what work he will complete by the next meeting and what obstacles are in his way. It is the Scrum Masters responsibility to track progress and remove obstacles. The progress should be introduced into a Sprint Burndown Chart, which includes the whole team's progress and can be used to determine a Burndown Tendency that will allow to see if the team is going to finish on time.



Graph 10 Sample Sprint Burndown Chart

At the end of the sprint a DEMO is held and the final product of the sprint is presented to the client. If changes are required they are incorporated into the Product Backlog. After that is done the next sprint is planned. This cycle is repeated any numbers of times until all the tasks in the Product Backlog are finished.

A Typical representation of the scrum process is the one shown in Illustration 22. Here the whole process described above can be seen in sequential form.

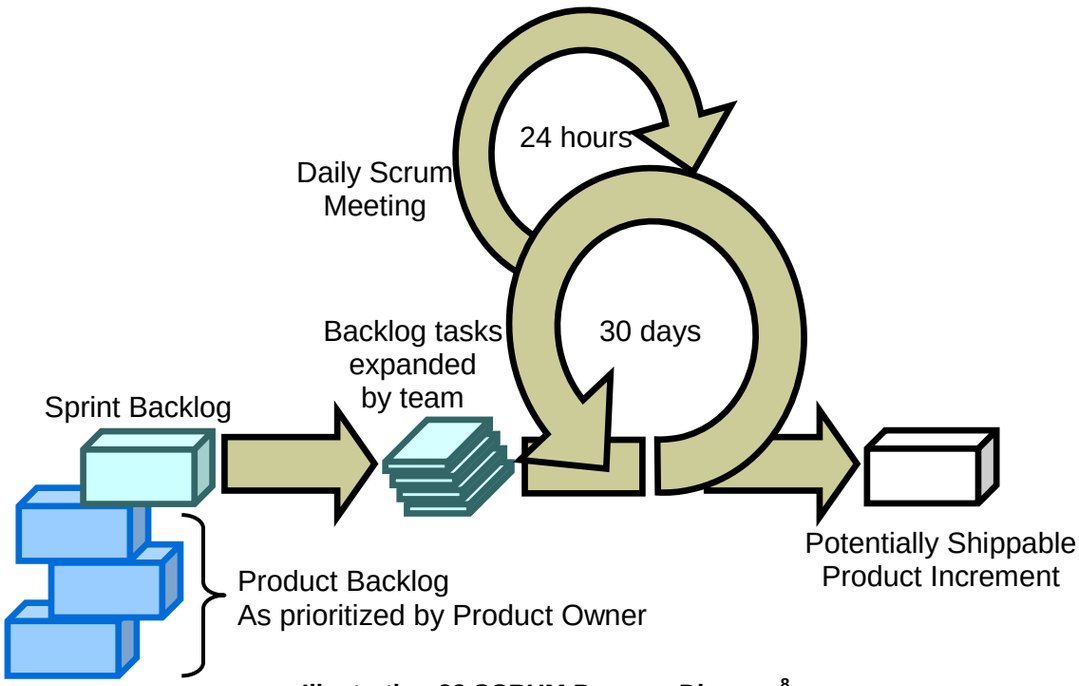


Illustration 22 SCRUM Process Diagram[8]

S-Scrum adds several steps to this process.

1. Before the product requirements are defined a Problem Process SNA Analysis is conducted. This process should consist of a SURVEY that asks the following question
   - Who do you communicate with regarding “INSERT PROBLEM BEING ANALYZED” and how often?

   The possible answers for this question should be every other user in the organization. Using the information in from the survey, the data must be weighted and formatted for input into a SNA analysis tool (for example UCINET). A possible way to do this is discussed thoroughly in the Thesis of which this Annex is part of.

   The data must be analyzed for Betweenness, Closeness and Cutpoints. Once finished a list of key users can be determined simply by taking the top 30% from each ordered list and selecting any users who are included into at least 2 of the lists.

   These users are key users and should be considered during requirement definition.
2. During each sprint a weekly Team Sprint SNA Analysis is conducted, the resulting network’s Meshness and Directionality must be calculated and located in the proposed model for team readiness. The formulae for this purpose are:

$i = \text{Originating node}$

$j = \text{Target node}$

$W_{i,j} = \text{Weight of edge (communication) from i to j}$

$$M = \frac{\sum_{i=2}^{n}\sum_{j=2}^{n} W_{ij}}{\sum_{i=2}^{n} W_{i1} + \sum_{j=2}^{n} W_{1j}}$$

Where:

$1 = \text{Team Leader}$

$2..n = \text{Team Members}$

for meshness, and;

$i = \text{Originating node}$

$j = \text{Target node}$

$W_{i,j} = \text{Weight of edge (communication) from i to j}$

Then:

$$D = \frac{\sum_{j=2}^{n} W_{1j}}{\sum_{i=2}^{n} W_{i1}}$$

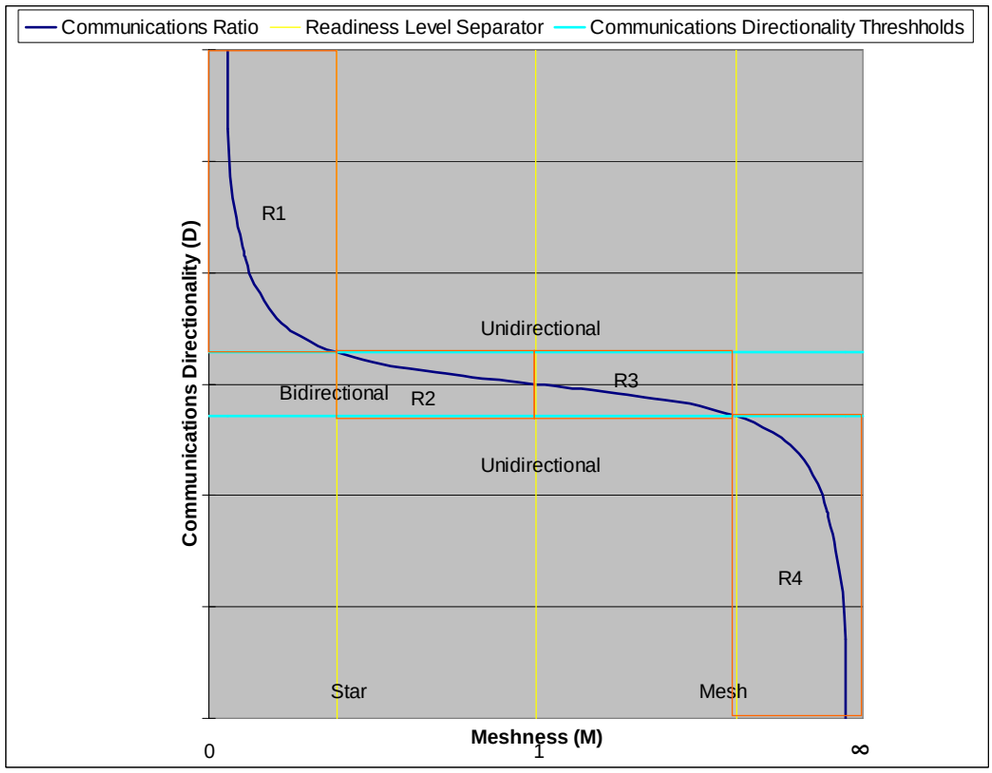
Where:

$1 = \text{Team Leader}$

$2..n = \text{Team Members}$

for directionality.

After calculating both metrics, we can determine the team's readiness level using the following diagram (Graph 11).



Graph 11 Communications vs. Meshness Model

The resulting readiness level will help the team leader decide which kind of leadership (According to the Situational Leadership Model) should be used.

3. Finally the last additional step is to use the Technical Support Network SNA Analysis to determine key users to ensure that the implantation process will go as smoothly as possible. The correct procedure is the same as for the a Problem Process SNA Analysis, except the new question is.
    - Who do you go to for advice or support regarding the use of Information Systems outside the formal help desk?

All three additional steps are inserted into the normal SCRUM process. The resulting process diagram is the one on the following page (Illustration 23).

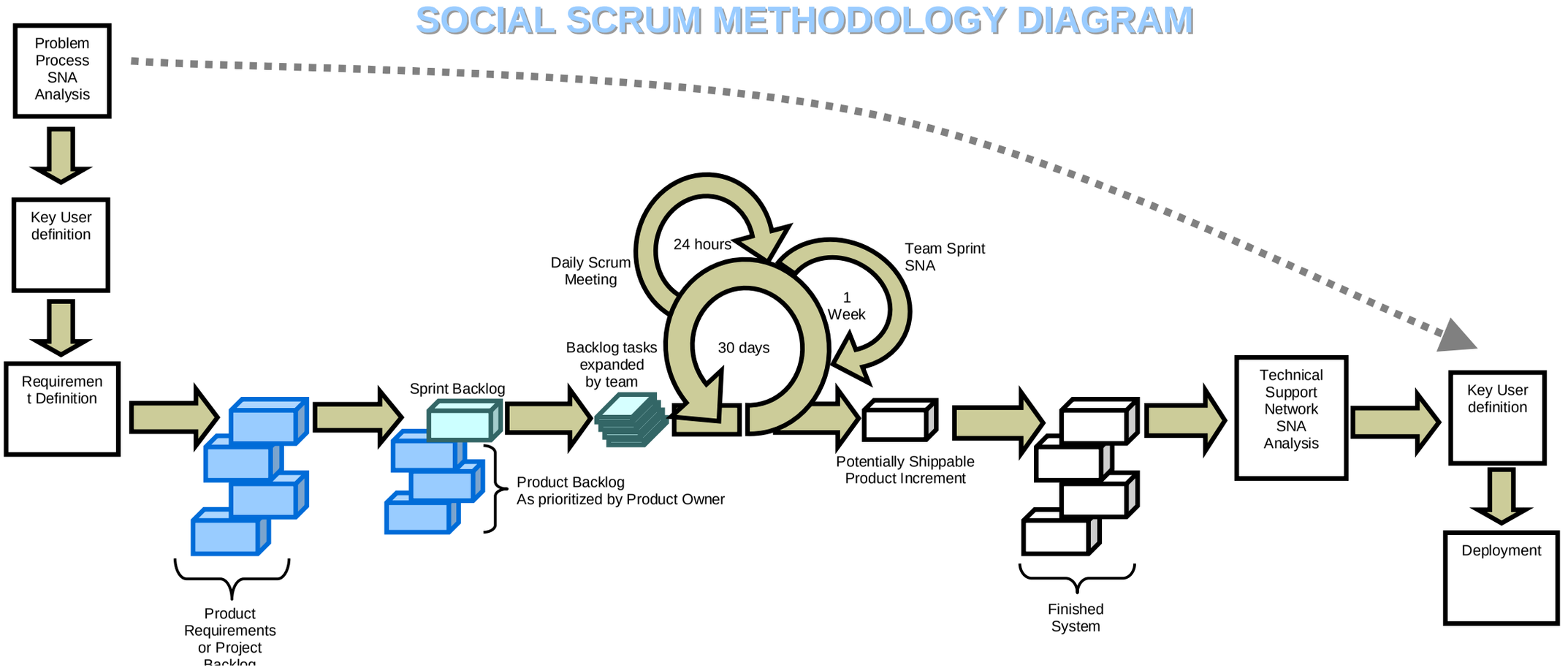


Illustration 23 S-SCRUM process diagram