TECHNICAL REPORT

Report No. CS2011-02
Date: May 2011

An Event-Driven Language for Cartographic Modelling of Knowledge in Software Development Organisations

Mark Micallef
Christian Colombo

Department of Computer Science
University of Malta
Msida MSD 2080
MALTA

Tel: +356-2340 2519
Fax: +356-2132 0539
`http://www.cs.um.edu.mt`

An Event-Driven Language for Cartographic Modelling of Knowledge in Software Development Organisations

Mark Micallef, Christian Colombo
Department of Computer Science
University of Malta
Msida, Malta
{mark.micallef,christian.colombo}@um.edu.mt

Abstract: *With software engineering now being considered a fully-fledged knowledge industry in which the most valuable asset to an organisation is the knowledge held by its employees [BD08], high staff turnover rates are becoming increasingly worrying. If software engineering organisations are to maintain their competitive edge, they need to ensure that their intellectual capital continues to grow and is not lost as people move in and out of their employ.*

In this paper, the authors present work involving the formalisation of a language that enables organisations to create and analyse maps of their organisational knowledge. In a more elaborate version of the traditional yellow-pages approach utilised in the cartographic school of thought, the proposed language models various relationships between knowledge assets, uses an event-driven mechanism to determine who knows what within the organisation, and finally provides metrics for detecting three types of risk related to knowledge management in modern software engineering. A three month evaluation of the language is also outlined and results discussed.

An Event-Driven Language for Cartographic Modelling of Knowledge in Software Development Organisations

Mark Micallef, Christian Colombo
Department of Computer Science
University of Malta
Msida, Malta
{mark.micallef,christian.colombo}@um.edu.mt

Abstract: *With software engineering now being considered a fully-fledged knowledge industry in which the most valuable asset to an organisation is the knowledge held by its employees [BD08], high staff turnover rates are becoming increasingly worrying. If software engineering organisations are to maintain their competitive edge, they need to ensure that their intellectual capital continues to grow and is not lost as people move in and out of their employ.*

In this paper, the authors present work involving the formalisation of a language that enables organisations to create and analyse maps of their organisational knowledge. In a more elaborate version of the traditional yellow-pages approach utilised in the cartographic school of thought, the proposed language models various relationships between knowledge assets, uses an event-driven mechanism to determine who knows what within the organisation, and finally provides metrics for detecting three types of risk related to knowledge management in modern software engineering. A three month evaluation of the language is also outlined and results discussed.

1 Introduction

Software engineering is now a fully-fledged knowledge industry. Even though practitioners have a systems-centric view of their day and think in terms of *specifying systems*, *designing systems*, *coding systems* and so on, they are in fact going to work everyday to create, store, retrieve, transfer and apply knowledge. The authors argue that what most people perceive as developers' jobs (designing, coding, etc) is in fact simply the *application* phase of an implicit knowledge management cycle. Most software development processes cater for knowledge management solely through codification strategies, which tend to focus on generating documentation at the end of each development phase. This, coupled with high staff turnover rates and the extensive role played by tacit knowledge in the industry, exposes software engineering organisations to knowledge risk.

Knowledge risk is defined as operational risk that is caused by a dependency on, loss of, unsuccessful intended or unintended transfer of knowledge assets and results in a lack of, or non-exclusivity of these assets [BM06]. When employees leave, companies lose not only human capital, but also accumulated knowledge [DH03]. As a result, information technology firms are realising that their "whole business is pretty much locked away in the minds of employees" [Kou03], yet this knowledge is rarely shared, swapped, traced and fertilised to ensure that it remains, at least in part, with the firm when employees leave [DH03]. When an employee does hand in her notice, managers usually spring to action, doing their best to capture her knowledge by means of knowledge transfer activities such as exit interviews [Bra98]. At this point however, such activities may be disruptive to other employees at best and too late to have any meaningful effect at worst. The work presented here seeks to address these issues based on insights from the cartographic school of thought [Ear01].

The cartographic school of thought is one of nine schools in knowledge management identified by Earl [Ear01]. The driving principle here is that of connectivity. That is to say, maximising the use of knowledge within the organisation by focusing on leading knowledge seekers directly to the knowledge providers who can satisfy their needs. This is usually accomplished using a yellow-pages style directory and has been shown to be useful for allocating resources, searching for competence, identifying project opportunities and upgrading skills [DDR05].

In this paper, the authors extend the yellow pages concept in order to realise a number of benefits. Firstly, it would be useful for an organisation to model various relationships between its knowledge assets, thus paving the way for structural analysis of its knowledge landscape. Secondly, the proposed approach makes it possible to not only model *who knows what* within the organisation but also *how much they know it* when compared to their colleagues. This is achieved by means of an event-driven mechanism which relates an employee's involvement in the life cycle of a knowledge asset to the likelihood of her 'knowing' the knowledge asset to some degree. Finally, as a result of the first two benefits, the proposed approach enables the definition of metrics that enable a company to detect the build up of three types knowledge risks and thus be able to take mitigating action when required.

This paper is organised as follows. In section 2, the basic concepts of knowledge map creation are introduced, as well as an example which will be used as a running example throughout the paper. Section 3 introduces an event-based mechanism for keeping knowledge maps representative of the knowledge landscape they model. Sections 4 and 5 then go on to showcase the languages features when it comes to drawing conclusions about organisational knowledge and identifying knowledge risks. Finally, a three-month evaluation of the language is outlined and results discussed in section 6.

2 Basic Concepts

Consider a typical software development organisation with a number of persons employed, and a number of knowledge assets which the organisation has identified as being valuable to it. Persons are organised into teams and knowledge assets have various attributes. For example, an organisation has identified seven knowledge assets: *Java*, *Smalltalk*, *OOP*, *SQL*, *JDBC*, *Serlvets*, and *WebDevelopment* and employs a single team of three people: *Chris*, *Mary* and *Jane*. The team members are knowledgeable of different topics and some persons are more expert than others on particular topics. For example, whilst everyone knows Java, Chris is clearly far more knowledgeable about it than Mary and Jane. Furthermore, knowledge assets are related to each other: the use of *Java* and *Smalltalk* is dependent on *OOP*, similarly the knowledge of *JDBC* depends on *SQL*; on the other hand, the knowledge of *JDBC* and *Servlets* is part of the knowledge of *Java*; and the knowledge of *Servlets* is related to *WebDevelopment*. This scenario is depicted in figure 1.

Please note that the convention for graphical representation is specified when relationships are introduced in section 2.1.

In the following subsections, we formalise the concepts necessary to specify such a scenario. This paves the way for building knowledge maps which reflect organisations' knowledge landscapes, which in turn can be analysed for various types of knowledge risks as discussed in section 5.

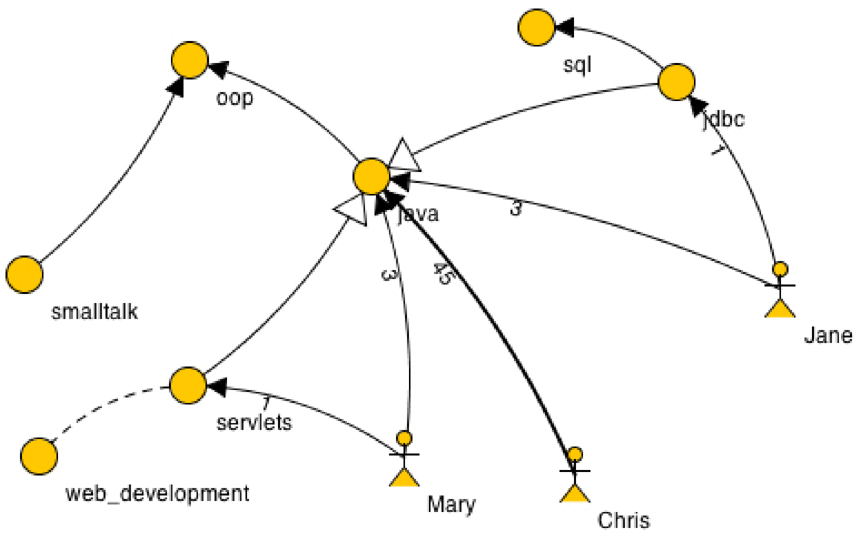

Figure 1: A graphical depiction of a scenario example.

2.1 Knowledge Assets

Knowledge assets are intangible firm-specific knowledge resources that are indispensable to create value for firms [BM06], and as such form the basis of any attempt to create a knowledge map of an organisation. For the purposes of richer knowledge modelling, it is desirable to model properties of knowledge assets. More specifically, we consider four properties of knowledge assets:

Category A property can fall under one of three categories: *technical*, *business*, or *general*. These categories are an adaptation of the those proposed by Ramal [RMA02] who proposed that software engineers know three categories of knowledge: *computer science*, *business* and *general*. The authors felt that the renaming of the "Computer Science" category to "Technical" was required because the term "Technical" knowledge provides an umbrella term for knowledge which may have otherwise been confusing given that computer science refers to a specific subset of topics in the academic world.

Visibility The second property of knowledge assets is *visibility* and refers to the widely cited *tacit* and *explicit* knowledge taxonomy of knowledge [AL01][Duf99][Szu96][Tiw00].

Sociality Another property is the *sociality* property which classifies a knowledge asset as either *individual* knowledge or *social* knowledge [Non94].

Operationality Finally, the language also models an *operationality* property which classifies a knowledge asset as *declarative* (know-about), *procedural* (know-how), *causal* (know-why), *conditional* (know-if) or *relational* (know-with)[NNI98][Zac98].

In summary and more formally, we define four corresponding types: CAT, VIS, SOC, and OPR as follows:

$$
\begin{aligned}
\text{CAT} &\stackrel{\text{def}}{=} \{\text{technical, business, general, undefined}\} \\
\text{VIS} &\stackrel{\text{def}}{=} \{\text{tacit, explicit, undefined}\} \\
\text{SOC} &\stackrel{\text{def}}{=} \{\text{individual, social, undefined}\} \\
\text{OPR} &\stackrel{\text{def}}{=} \{\text{declarative, procedural, causal,} \\
&\quad\ \text{conditional, relational, undefined}\}
\end{aligned}
$$

Given a type K of knowledge assets, we assign an attribute of each of the above types through the following four total functions:

$$
\begin{aligned}
\text{category} &: K \longrightarrow \text{CAT} \\
\text{visibility} &: K \longrightarrow \text{VIS} \\
\text{sociality} &: K \longrightarrow \text{SOC} \\
\text{operationality} &: K \longrightarrow \text{OPR}
\end{aligned}
$$

Further to assigning attribute to knowledge assets, we relate knowledge assets to each other through the following four kinds of relationships of type $\mathbb{P}(K \times K)$:

Related - A *related* relationship signifies that the two knowledge assets are related in some way. For example, in the example provided at the beginning of section 2, the *servlets* knowledge asset and *web_development* knowledge asset are related. This does not imply that one cannot exist without the other, but merely that if a person is knowledgeable about one asset, there is a likelihood that she is knowledgeable about the second. This relationship is the weakest form of relationship between two knowledge assets in the proposed model and is denoted by the symmetric relation $\xleftrightarrow{related}$. In graphical depictions of knowledge maps, the convention is adapted that two related knowledge assets are linked together by a dashed edge.

Dependency - One knowledge asset k_a is said to be dependent on a second knowledge asset k_b if in order to learn k_a, one first has to learn k_b. A typical example from the knowledge map depicted in figure 1 is *"java depends on oop"*. That is to say that before one can effectively use the Java programming language, one must understand the concepts of object oriented programming. We denote this relation by $\xrightarrow{\textit{deps on}}$. Note that $\xrightarrow{\textit{deps on}}$ is antisymmetric, i.e. if an assets k_a depends on k_b, then k_b cannot depend on k_a. In graphical representations of knowledge maps, the convention is being adapted that $k_a \xrightarrow{\textit{deps on}} k_b$ is represented by a continuous directed edge between k_a and k_b with a solid arrowhead.

Composition - A knowledge asset k_b is said to be partially composed of another knowledge asset k_a when k_a represents a subset of knowledge represented by k_b. This relation, which is also antisymmetric, is denoted by $\xrightarrow{\textit{part of}}$. Note that $\xrightarrow{\textit{part of}} \subseteq \xrightarrow{\textit{deps on}}$, i.e. if an asset k_a is a part of k_b, then k_a depends on k_b. A typical example is provided in figure 1 whereby servlets $\xrightarrow{\textit{part of}}$ java and jdbc $\xrightarrow{\textit{part of}}$ java. In graphical representations of knowledge maps, the convention is being adapted that $k_a \xrightarrow{\textit{part of}} k_b$ is represented by a continuous directed edge between k_a and k_b with a hollow arrowhead.

Having outlined the attributes of, and the relationships between knowledge assets, we next define the relationship between people and knowledge assets.

2.2 People and Knowledge

Our interest in reasoning about knowledge assets is to be able to reason about what people in an organisation know. Thus we relate people to knowledge assets by assigning a magnitude relative to how much the person knows that knowledge asset. More formally let P be the type representing people in the organisation, the relation $\xrightarrow{\textit{knows}}$ of type $P \times K \times \mathbb{N}$ is a set of triples representing a person, a knowledge asset and a magnitude. To facilitate reasoning about people, we also define a set T of teams whereby each team is a subset of people, $T \subseteq \mathbb{P}P$.

Finally, we define the whole scenario in terms of a graph structure as follows:

Definition 1 *The graph G, representing the knowledge landscape, is a triple (V, L, E) where: (i) a set of vertices, V, made up of persons and knowledge assets, $V = P \cup K$; (ii) a set of labels, L, made up of natural numbers and $L' = \{related, deps\ on, part\ of\}$, $L = \mathbb{N} \cup L'$; and iii a set of edges, E, a subset of $V \times L \times V$.*

Given relations, $\xleftrightarrow{\textit{related}}$, $\xrightarrow{\textit{deps on}}$, $\xrightarrow{\textit{part of}}$, and $\xrightarrow{\textit{knows}}$, defined over knowledge assets K and persons P, we can derive the corresponding graph edges as follows:

$$\begin{aligned}
E \overset{def}{=} & \{(k_a, related, k_b) : V \times L' \times V \mid (k_a, k_b) \in \xleftrightarrow{\textit{related}}\} \\
\cup & \{(k_a, deps\ on, k_b) : V \times L' \times V \mid (k_a, k_b) \in \xrightarrow{\textit{deps on}}\} \\
\cup & \{(k_a, part\ of, k_b) : V \times L' \times V \mid (k_a, k_b) \in \xrightarrow{\textit{part of}}\} \\
\cup & \{(p, n, k) : P \times \mathbb{N} \times V \mid (p, k, n) \in \xrightarrow{\textit{knows}}\}
\end{aligned}$$

As abbreviation we use $v \xrightarrow{label} v'$ *for* $(v, {\scriptscriptstyle label}, v') \in E$.

Example 1 *Referring back to the example given at the beginning of section 2 and depicted in figure 1, we now give the formal definition of the knowledge landscape, g, by instantiating the graph structure we have just defined:*

$$\begin{aligned}
K &\overset{def}{=} \{\mathit{Java},\ \mathit{Smalltalk},\ \mathit{OOP},\ \mathit{SQL},\\
&\qquad \mathit{JDBC},\ \mathit{Serlvets},\ \mathit{WebDevelopment}\}\\
P &\overset{def}{=} \{\mathit{Chris},\ \mathit{Jane},\ \mathit{Mary}\}\\
E &\overset{def}{=} \{\mathit{Servlets} \xrightarrow{related} \mathit{WebDevelopment},\\
&\mathit{WebDevelopment} \xrightarrow{related} \mathit{Servlets}, \mathit{JDBC} \xrightarrow{part\ of} \mathit{Java},\\
&\mathit{Java} \xrightarrow{deps\ on} \mathit{OOP}, \mathit{Smalltalk} \xrightarrow{deps\ on} \mathit{OOP},\\
&\mathit{JDBC} \xrightarrow{deps\ on} \mathit{SQL}, \mathit{Mary} \xrightarrow{1} \mathit{Servlets},\\
&\mathit{Mary} \xrightarrow{3} \mathit{Java}, \mathit{Chris} \xrightarrow{45} \mathit{Java},\\
&\mathit{Jane} \xrightarrow{3} \mathit{Java}, \mathit{Jane} \xrightarrow{1} \mathit{JDBC}\}\\
g &\overset{def}{=} (P \cup K, L, E)
\end{aligned}$$

One can note that the example defines certain knowledge relationships and their magnitudes explicitly. This is indeed useful when one first starts building a knowledge map. However, it is desirable for knowledge relationships to be created and adjusted automatically over time based on some mechanism. This is the subject of section 3.

3 Maintaining the Knowledge Landscape

Knowledge is never static in an organisation. People learn new knowledge and forget other knowledge, other people join and leave the organisation bringing/taking knowledge with them, and so on. Therefore it is important to model such changes so that the knowledge landscape remains up to date and representative of the organisation's knowledge landscape. We model such changes in terms of events. For example, if an event is modelled whereby a person p is applying a knowledge asset k_a over a period of time, then it can be inferred that the longer the period of time over which p applies k_a, the more p knows k_a. Conversely, if there are no events in the model which link p to a second knowledge asset k_b, then it can be inferred that p does not know k_b.

The events we consider here fall under one of three categories: *Asset-Changing Events*, *Relationship-Changing Events*, and *Time Events*. In the following subsections, we introduce the different event types and explain the changes that occur to the underlying knowledge map upon the triggering of an event. To this end, we define a function *Prog*, of type $(G \times \mathit{Events}) \to G$, which takes a graph and an event (the type *Events* is defined to be the union of the events defined below) and returns a modified graph. This new graph represents the organisation's updated knowledge landscape.

3.1 Asset-Changing Events

Resource events deal with the changes in the personnel and knowledge assets of the organisation. For simplicity we parametrise events over persons and knowledge assets, virtually having an event for each possible person/knowledge asset.

- Upon a person p joining the organisation, we simply add a vertex to the graph: $\mathit{Prog}((V, L, E), \texttt{p\_left\_org}_{(p)}) \overset{\text{def}}{=} (V \cup \{p\}, L, E)$
- Upon a person p leaving the organisation, we simply remove the vertex from the graph and remove the edges concerned: $\mathit{Prog}((V, L, E), \texttt{p\_left\_org}_{(p)}) \overset{\text{def}}{=} (V \backslash \{p\}, L, \{(v, l, v') \in E \mid v \neq p \wedge v' \neq p\})$
- When a knowledge asset k is identified, a vertex is added to the graph: $\mathit{Prog}((V, L, E), \texttt{k\_ident}_{(k)}) \overset{\text{def}}{=} (V \cup \{k\}, L, E)$
- When a knowledge asset k is discarded, the vertex and the edges concerned are removed: $\mathit{Prog}((V, L, E), \texttt{k\_disc}_{(k)}) \overset{\text{def}}{=} (V \backslash \{k\}, L, \{(v, l, v') \in E \mid v \neq k \wedge v' \neq k\})$

3.2 Relationship-Changing Events

Several researchers within the knowledge-based perspective of the firm have come to see organisations as knowledge systems consisting of a set of socially enacted knowledge processes[?][?]. Much in the same way that software systems have a life cycle or development process, so do knowledge assets. Although various, more detailed delineations of knowledge processes exist, these can all be generalised to four primary elements: (i) knowledge creation, (ii) knowledee storage/retrieval, (iii) knowledge transfer, and (iv) knowledge application [?][?][?][?]. When these processes are enacted, the organisation's knowledge landscape changes and peoples' knowledge of particular knowledge assets can grow or shrink. The events in this section are thus modelled on these four basic knowledge processes.

A considerable challenge here is the calculation of how much to influence the magnitude of individual knowledge relationships based on some event. It is arguably impossible to model learning and forgetting with a simple formula and achieve 100% accuracy. The amount one learns or forgets depends on a number of factors such as the individual's age, background and cognitive ability. For this reason, the language outsources these calculations to an oracle. This provides us with the ability to plug in oracles based on the organisation's context and/or the state of understanding of human learning (and forgetting) over time.

- When knowledge is created, the likelihood is that the person p who creates knowledge about knowledge asset k has increased his knowledge about k. Thus the magnitude of the relationship between p and k increases and a relationship is created if it does not already exist. Note the use of oracle-decided m (of type $\mathbb{N}$) which signifies the new magnitude of the relationship.
  $\mathit{Prog}((V, L, E), \mathtt{create}_{(p,k,m)}) \stackrel{\text{def}}{=} (V, L, E' \cup \{p, m, k\})$
  where $E' = \{(v, l, v') \in E \mid v \neq p \wedge v' \neq k\}$.

- Similarly, when knowledge about a knowledge asset k is stored, retrieved, transferred of applied, the people involved in those events gain more knowledge about k. Please note that we are splitting knowledge transfer events into two such that the "giving" and "receiving" ends of a knowledge transfer activity are modelled as separate events. This is mathematically more elegant and also facilitates the modelling of one-to-many and many-to-many transfer activities. Therefore the definitions of *Prog* for events $\mathtt{store}_{(p,k,\delta)}$, $\mathtt{retrieve}_{(p,k,\delta)}$, $\mathtt{apply}_{(p,k,\delta)}$, $\mathtt{given}_{(p,k,\delta)}$, and $\mathtt{recvd}_{(p,k,\delta)}$ are all identical and are defined as follows (we use the event $\mathtt{store}_{(p,k,\delta)}$ as an example):
  $\mathit{Prog}((V, L, E), \mathtt{store}_{(p,k,\delta)}) \stackrel{\text{def}}{=} (V, L, E')$
  where
  $E' = \{(p, l, k) \in E \bullet (v, l + \delta, v')\}$
  $\quad \cup \{(v, l, v') \in E \mid v \neq p \vee v' \neq k \bullet (v, l, v')\}$

- The authors also propose the modelling of events whereby existing knowledge is modified. This is perceived as being important for two reasons. Firstly, since the knowledge map clearly shows who knows a particular knowledge asset, this information can be used to notify all knowers when it changes. Also, the fact that knowledge has been modified is likely to result in an increase in the knowledge of the person who modified it but also a decrease in the knowledge of all other persons who knew the knowledge asset before it was modified. Thus, the modification event is parametrised over a person p, a knowledge asset k,

an upward change δ and a downward change δ': $\mathit{Prog}((V, L, E), \texttt{mod}_{(p,k,\delta,\delta')}) \overset{\text{def}}{=} (V, L, E')$
where
$E' = \{(p, l, k) \in E \bullet (v, l + \delta, v')\}$
$\quad \cup \{(p', l, k) \in E \mid p' \in P \bullet (v, \mathit{max}(0, l - \delta'), v')\}$
$\quad \cup \{(v, l, v') \in E \mid v \notin P\}$

3.3 Time Events

Knowledge decreases over time if not used. Thus, to keep the graph realistic, we propose a time event which decreases all the knowledge magnitudes as follows:

$\mathit{Prog}((V, L, E), \texttt{time}_{(\delta')}) \overset{\text{def}}{=} (V, L, E')$
where
$E' = \{(v, l, v') \in E \mid v \in P \wedge v' \in K \bullet (v, \mathit{max}(0, l - \delta'), v')\}$
$\quad \cup \{(v, l, v') \in E \mid v \notin P\}$

3.4 Bringing it all Together

Starting from an initial graph g_1, which may either be the empty graph $(\emptyset, L, \emptyset)$ or an initialised graph (V, L, E), one can apply a number of events through the function *Prog* to keep the graph updated. Each application of *Prog* would then result in a new version of the graph $g_2, g_3, \ldots$. By representing $\mathit{Prog}(g_i, \texttt{ev}) = g_{i+1}$ as $g_i \xrightarrow{\texttt{ev}} g_{i+1}$, we get the following:

$$g_1 \xrightarrow{\texttt{ev}_1} g_2 \xrightarrow{\texttt{ev}_2} g_3 \quad \cdots \quad \xrightarrow{\texttt{ev}_n} g_{n+1}$$

Example 2 *Building on example 1, consider a scenario whereby on the following events occur: (i) Mary reads a book about Smalltalk and considers herself knowledgeable on the subject, (ii) Mary applies her knowledge of Smalltalk for 5 days, (iii) Mary teaches Chris about Smalltalk. Assuming g is a graph representing the knowledge map from example 1, the language can be used to model these events as follows:*

$$\begin{array}{lclcl}
g & \xrightarrow{\texttt{create}_{(mary,smalltalk,1)}} & g_1 & \xrightarrow{\texttt{apply}_{(mary,smalltalk,1)}} & g_2 \\
g_2 & \xrightarrow{\texttt{time}_{(0)}} & g_3 & \xrightarrow{\texttt{apply}_{(mary,smalltalk,1)}} & g_4 \\
g_4 & \xrightarrow{\texttt{time}_{(0)}} & g_5 & \xrightarrow{\texttt{apply}_{(mary,smalltalk,1)}} & g_6 \\
g_6 & \xrightarrow{\texttt{time}_{(0)}} & g_7 & \xrightarrow{\texttt{apply}_{(mary,smalltalk,1)}} & g_8 \\
g_8 & \xrightarrow{\texttt{time}_{(0)}} & g_9 & \xrightarrow{\texttt{apply}_{(mary,smalltalk,1)}} & g_{10} \\
g_{10} & \xrightarrow{\texttt{time}_{(0)}} & g_{11} & \xrightarrow{\texttt{given}_{(mary,smalltalk,1)}} & g_{11} \\
g_{11} & \xrightarrow{\texttt{recvd}_{(chris,smalltalk,1)}} & g_{12} & &
\end{array}$$

In the name of simplicity, this example assumes that the oracle being utilised always increases knowledge relationship magnitudes by 1 and time has no negative affect on knowledge relationships in the short timeframe of 5 days being considered here. At this point, the graph g_{12} is an update version of the original graph g such that it has been updated relevant knowledge relationships as depicted in in figure 2.

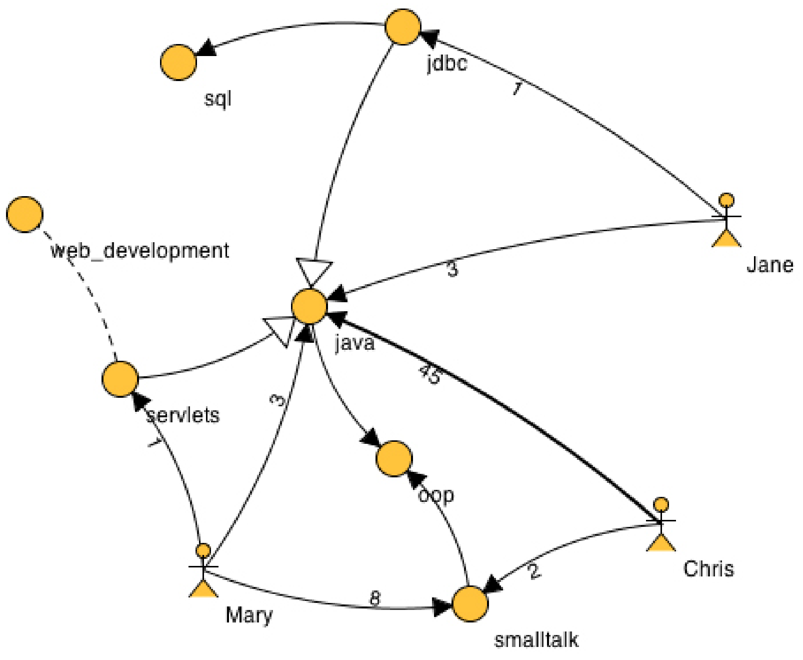

Figure 2: The resulting graph after events from example 2 have been logged.

One can note that Mary now knows Smalltalk with a magnitude of 8 whilst Christ *knows Smalltalk with a magnitude of 2. Mary's knowledge comes from reading a book (creating knowledge), applying her newly acquired knowledge for 5 days and transferring it to Chris. Chris on the other hand, only interacted with Smalltalk during the knowledge transfer activity. Hence his knowledge is considerably less than Mary's.*

4 Cartographic Queries

Much in the same way that the traditional yellow-pages style knowledge maps can be used for allocating resources, searching for competence, identifying project opportunities and upgrading skills, so can the language presented in this paper. However, since the technique presented here maintains knowledge relationship magnitudes, it allows for a number of interesting queries. In this section, a number of cartographic queries which operate on knowledge maps are outlined. Please note that due to length restrictions, it is impractical to give a mathematical definition of these functions here. Consequently, in this section name a number of functions and describe their purpose but will not provide a mathematical definition.

knows is a function which given a graph g, a person p and a knowledge asset k, returns the magnitude of $p \xrightarrow{knows} k$ in the context of g. This is a simple query function which is utilised in the definition of other functions and the metrics presented in section 5.

who_knows is a function which given a graph and a knowledge asset, returns an ordered tuple of persons who know the knowledge asset. The tuple is sorted in descending order of the magnitude of the knowledge relationships. This query enables organisations to analyse their knowledge of particular assets and can thus prove useful in project sourcing.

knows_what is a function which given a graph and a person, returns an ordered tuple of knowledge assets which the person knows. The tuple is sorted in descending order of how much the person knows the asset. This query is useful for team formation and planning of training for individual staff members.

experts is a function which given a graph and a knowledge asset, returns an ordered list of people whose knowledge of the asset is statistically much stronger than that of their colleagues and can thus be considered to be the experts on the topic. This list is sorted in descending order of knowledge magnitude. This query is similar to the *who_knows* query but since it only returns people who are statistically considered to be experts would prove more useful when the expertise of such people is being sought.

person_similarity is a function that given a graph g, a person p and a set of knowledge assets K_a, returns an ordered tuple of persons whose knowledge of the assets in K_a is similar to that of p. The list is sorted in descending order of similarity. This query enables organisations to find suitable candidates during team formation or in cases where for example, a person needs to be transferred to a different team and thus needs to be replaced.

team_knows is a function which given a graph and a team assignment, returns the knowledge assets which the team knows, in descending order of knowledge magnitudes. This enables organisations to reason about the knowledge of groups of people rather than individuals.

assets_with_attribute is a function which given a graph and a knowledge attribute assignment function, returns a set of assets which exhibit the attribute. Such queries can be useful when analysing a knowledge landscape from different points of view. For example, it might be useful for an organisation to analyse its knowledge landscape from the point of view of the Operational Classification of it's knowledge assets (see section 2). In such cases, it could detect that for example, the organisation has very little causal knowledge about what it does and this might present problems in future.

5 Knowledge Metrics

This section defines a number of metrics which can be used to analyse a model with a view to understand the health of the organisation's knowledge landscape. Please note that due to length contraints, only a few salient metrics could be defined in this paper.

5.1 Knowledge Mobility Risk

The term *knowledge mobility risk* refers to the chance of the company loosing a valued knowledge asset as a result of a particular person or persons leaving. In general, the greater the number of people that know a knowledge asset, the less mobility risk that knowledge asset exhibits. However, one must consider 'how much' each person knows the knowledge asset. If (for example) four people in a team are vaguely familiar with a critical part of a system but a fifth person is an expert about it, then with respect to that particular knowledge asset, loosing that one expert will probably hurt more than loosing any number of the other four members.

Two mobility risk metrics are defined:

Knowledge Asset Mobility Risk (KMR) - Assesses the risk of individual knowledge assets being lost should certain personnel movements occur. Any risky assets identified by this metric should lead to the organisation taking risk-mitigation action such as knowledge transfer activities.

Person Mobility Risk (PMR) - Assesses the risk posed by individual persons should they suddenly leave the company. This metric operates in the context of a subset of knowledge assets chosen by the user so as to provide provide more meaningful information. For example, one could calculate PMR in the context of the subset of knowledge assets related to database technologies in the organisation. This localises the metric and provides more useful information. If a person is identified as being risky by this metric, it implies that if the person leaves, the organisation's knowledge of one or more knowledge assets is likely to be severely damaged. In such a case, the organisation should take risk mitigation and transfer the person's knowledge to other people in the organisation, thus spreading the risk.

Both metrics are adaptations of centrality metrics presented by Botafogo et al [BRS92] which measure the social importance of vertices in a graph. However, Botafogo's work calculated centrality based on the length of paths between pairs of vertices. This is not suitable in our context because (a) the maximum length of a knowledge path between a person and a vertex is 1 and (b) knowledge relationships exhibit a *magnitude* property which must be taken into account.

Definition 2 *Given a graph g with $k_1, ..., k_n$ knowledge asset vertices and $p_1, ..., p_m$ person person vertices, then the* knowledge distance matrix *M for the graph is defined as a $n \times m$ matrix in which:*

$$M_{n',m'} = \begin{cases} K & \textit{if } knows(g, p_{m'}, k_{n'}) = 0 \\ \frac{maxKM}{knows(g, p_{m'}, k_{n'})} & \textit{otherwise} \end{cases}$$

Where:

K is a constant which represents infinity in the graph. Throughout this paper, K will be $maxKM + 1$.

$maxKM$ is the highest knowledge magnitude assigned to knowledge relationships in the graph. This is used as a normalising value whereby a simulated path distance of 1 is assumed for knowledge relationships of $maxKM$ magnitude. All smaller values of magnitude will be translated to a proportionately larger path size. This allows us to use centrality metrics.

Definition 3 *Given a knowledge distance matrix M representing $k_1, ..., k_n$ knowledge asset vertices and $p_1, ..., p_m$ person person vertices, we define the* knowledge in distance (KID) *and the* knowledge out distance (KOD) *for each person vertex as follows:*

$$\begin{aligned} KID(k_{n'}) &= \textstyle\sum_{i=1}^{m} M_{n',i} \\ KOD(p_{m'}) &= \textstyle\sum_{i=1}^{n} M_{m',i} \end{aligned}$$

We also define the concept of knowledge distance (KD) *as:*

$$KD = \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} M_{i,j}$$

Definition 4 *At this point we can define the metrics for* knowledge asset mobility risk (KMR) *and* person mobility risk (PMR) *as:*

$$\begin{aligned} KMR(k_{n'}) &= \tfrac{KD}{KID(k_{n'})} \\ PMR(p_{m'}) &= \tfrac{KOD(p_{m'})}{KD} \end{aligned}$$

The higher the value of KMR for a particular knowledge asset, the more at risk it is of being lost due to personnel movements. Similarly, the higher the value of PMR for a particular person, the more knowledge the company stands to loose if the person leaves.

Example 3 *Consider the subgraph of a model depicted in figure 3. In this example, the organisation is analysing whether any knowledge risks exist when considering its knowledge of dynamic web technologies (jsp and servlets). The calculations for KMR and PMR would be carried out as follows:*

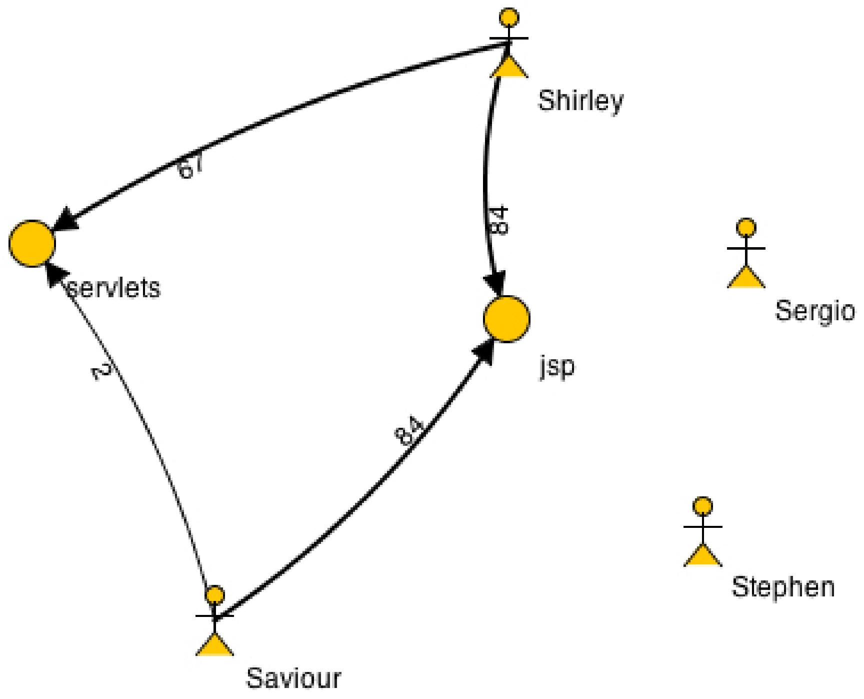

Figure 3: An example for use in knowledge mobility risk definition.

	jsp	$servlets$	KOD	PMR
$Saviour$	1	42	43	8.96
$Sergio$	85	85	170	2.27
$Shirley$	1	1.25	2.25	171.22
$Stephen$	85	85	170	2.27
KID	172	213.25	385.25	
KMR	0.44	0.55		

One can make a number of observations after examining the results. Firstly, Sergio and Stephen present no risk in this particular context. This is because according to the model, they have no knowledge of the knowledge assets in question. So if any of them leave, the organisation's knowledge of servlets and jsp will not be affected. The reason why they have a PMR value greater than 0 is related to our choice of the constant K. Secondly, Shirley has an astronomical PMR value when compared to other people in the organisation. This means that if Shirley leaves tomorrow, the organisation stand to loose substantial knowledge about servlets and jsp. In fact, Shirley has strong knowledge of both when compared to Saviour's knowledge magnitude *of 2 when considering servlets. Finally, jsp and servlets exhibit a KMR of 0.44 and 055 respectively. This essentially means that the servlets knowledge asset exhibist 25% more risk than jsp, due to the fact that even though two people know servlets, Saviour's knowledge is minuscule compared to Shirley's.*

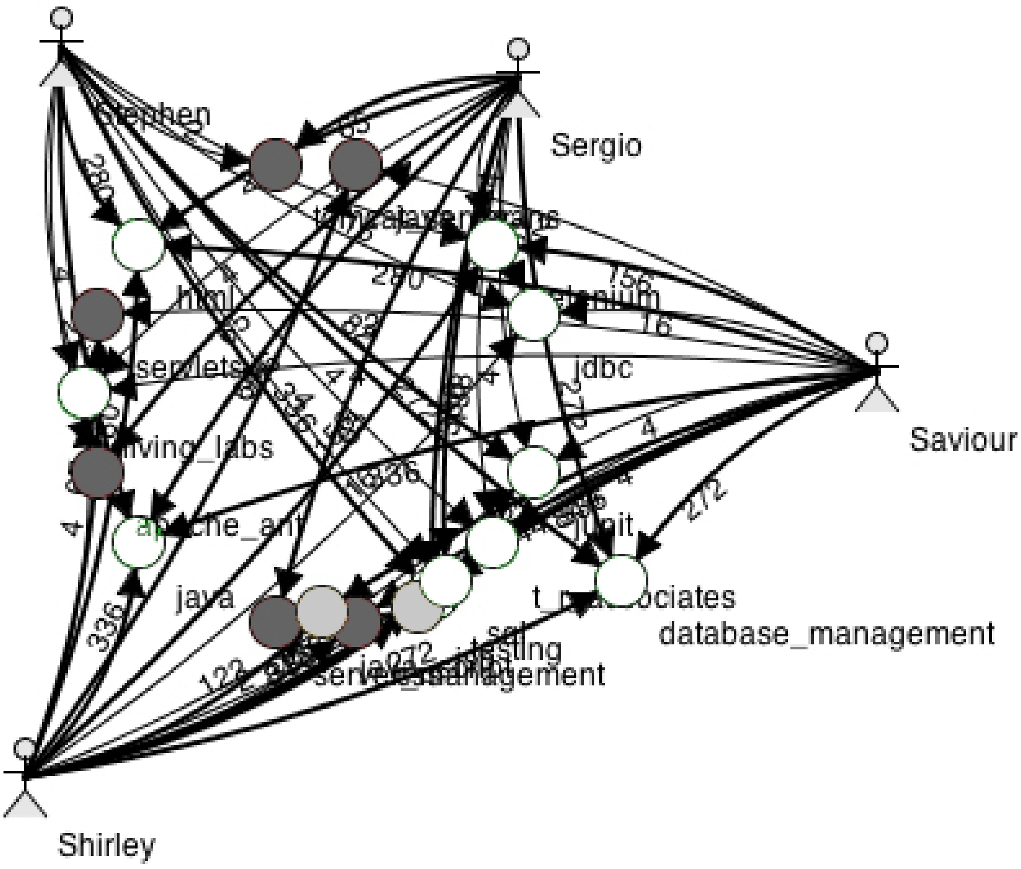

Figure 4: A knowledge map resulting from a three month evaluation exercise, which is coloured according to KMR values.

Generally speaking, KMR is useful for identifying high risk knowledge assets from all (or a large selection of) your model whilst PMR provides more useful information when you limit your context to a small subset of knowledge assets of interest. Figure 4 depicts a larger knowledge map which was constructed by a team of four people during a three-month evaluation exercise. The vertices in this map are coloured according to the following criteria:

1. Let σ be the standard deviation of the KMR values for all knowledge asset vertices in the graph
2. Let μ be the average of the KMR values for all knowledge asset vertices in the graph
3. Colour all nodes with $KMR \leq \mu$ green (represented as white in this paper)
4. Colour all nodes with $\mu > KMR \leq (\mu + \sigma)$ orange (represented as light-grey in this paper)
5. Colour all nodes with $KMR > (\mu + \sigma)$ red (represented as dark-grey in this paper)

Filtering the graph to red (dark grey) vertices results in figure 5. This demonstrates how the metrics in question can take all the clutter out of a complex knowledge map and provide a much clearer view of where the problem areas might be.

5.2 Knowledge Transfer Risks

Knowledge transfer risks refer to situations which might compromise the likelihood of a knowledge transfer activity to be successful. Knowledge transfer activities are amongst the most important activities which enable organisations to hold on to organisational knowledge despite staff turnover. Although research carried out seems to indicate that most knowledge transfer success factors can only be detected and influenced by company culture and management practices, Cummings and Teng [Cum03] developed a research model consisting of nine key factors which affect knowledge transfer, two of which can be detected using the event-based cartographic approach proposed in this paper. These are *knowledge embeddedness* and *knowledge distance*. The former refers to the extent to which a particular knowledge asset is linked to other knowledge assets within the organisational knowledge landscape. Cummings and Teng found that the more embedded a knowledge asset is, the more difficult it is to transfer. *Knowledge Distance* refers to the difference between two people's knowledge in relation to the knowledge asset being transferred. If two people share a relevant common basis of knowledge, their knowledge distance is said to be small. This makes a knowledge transfer exercise between such persons more likely to succeed than if they had a large knowledge distance separating them.

Both *knowledge embeddedness* and *knowledge distance* can be inferred from knowledge models constructed using the language presented in this paper.

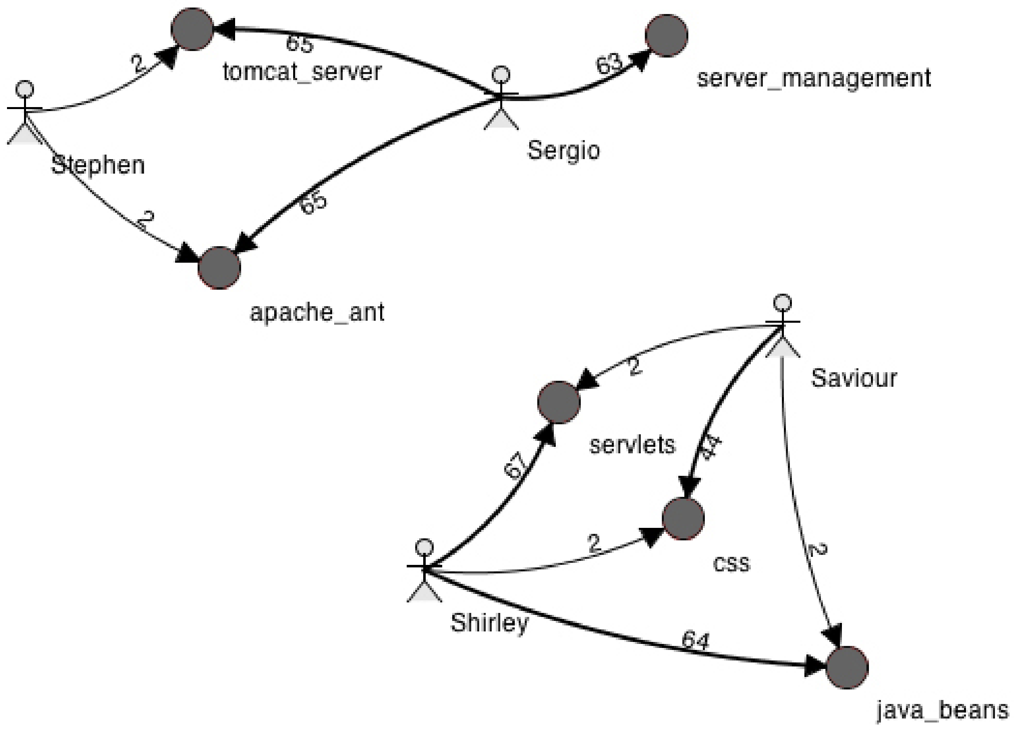

Figure 5: A filtered version of the knowledge map in figure 4 showing only high knowledge mobility risk assets.

5.2.1 Embeddedness

The following measure of embeddedness is propopsed:

$$embeddedness(\mathrm{k}) \quad = \quad |\mathrm{dep}(k)|$$

Where **dep(*k*)** is a function that given a knowledge asset k, returns the set of all knowledge assets which k *depends on*. Please note that since dependency is transitive, this set will also include knowledge assets which are depended on by direct dependencies of k. Also, since composition is a form of dependency (see section 2), **dep(*k*)** will also return elements which k forms part of.

For any knowledge asset k, a higher value of embeddedness(k) implies a higher level of knowledge transfer risk exhibited by k.

5.2.2 Knowledge Distance

Consider a knowledge transfer exercise whereby a person who is a knowledge source p_{src} is transferring a knowledge asset k to a second person who is a knowledge receiver p_{rec}. The knowledge distance measure takes into account p_{rec}'s lack of knowledge of all assets in $\mathbf{dep}(k)$ (see section 5.2.1):

$$\text{distance}(p_{src},\, p_{rec},\, k) \;=\; \sum_{\forall k' \in dep(k)} \text{dist}_{abs}(p_{src}, p_{rec}, k')$$

Where:

$\text{dist}_{abs}(p_{src}, p_{rec}, k) =$

$$\begin{cases} 0 & \text{if } knows(g, p_{src}, k) - knows(g, p_{rec}, k) < 0 \\ knows(g, p_{src}, k) - knows(g, p_{rec}, k) & \text{otherwise} \end{cases}$$

6 Evaluation

An evaluation exercise was carried out involving two four-member undergraduate student teams participating in a three-month development project for which they received academic credit. The project required students to work for an actual industry client and delivery software using agile methodologies. A number of workshops about agile development and knowledge management were delivered to all students participating in the exercise. They were also tasked with creating and maintaining a knowledge map using the techniques presented in this paper. In order to facilitate this, a GUI tool for creating and analysing knowledge maps was developed and made available to the students.

The exercise was targeted at evaluating participants' perceived accuracy of conclusions drawn from their knowledge maps with regards to (i) knowledge relationships regardless of magnitudes, (ii) knowledge relationship magnitudes and (iii) risk metrics. At the end of the three month exercise, each team's knowledge map was analysed by researchers and a custom survey was designed for each team. Each survey consisted of two sections: section one was to be answered by the team as a whole and section two consisted of a personalised question for each team member. In section one, participants were asked to rate their agreement with a number of statements such as *"All team members can write unit tests"*, *"Elise knows Javascript more than Ian"* and *"if Joseph leaves the team, we will have no knowledge of photoshop"*. Each question was designed to assess the team's perceived accuracy of various aspects of the knowledge map as discussed above. Team members discussed each statement and jointly provided a rating on a scale of 1 (strongly disagree) to 5 (strongly agree). Section two consisted of one personalised question for each team member whereby they were provided with five knowledge assets and asked to rank them in descending order of their familiarity with them. This made it possible to compare each team member's ranking with the ranking provided by the knowledge map.

6.1 Evaluation Results

Table 1 summarises the average scores for questions designed to measure the perceived accuracy of conclusions drawn from the knowledge maps. With regards to the questions designed to gauge participants' perceived accuracy of the model's knowledge relationships regardless of magnitude, the results are highly encouraging. Teams score a high level of agreement with statements like "Sergio is the only person who knows JDBC" with the average score being **4.3** (agree). This essentially means that the proposed technique effectively models who knows what within the organisation.

Perceived accuracy of...	**Avg Score**	**Comment**
Knowledge relationships	4.3	Agree
Knowledge rel magnitudes	3.6	Weak Agree
Knowledge metrics	3.9	Agree

Table 1: Results of perceived accuracy questions

The next area of interest involved investigating whether or not a knowledge map accurately represents knowledge magnitudes. That is to say, not only "what" people know but also "how much" they know it. This was approached from two angles. Firstly, in section one of the survey, a number of statements were specifically designed to query participants about conclusions involving magnitudes. Typical statements include "Jacqueline and Elise are the team's experts on Javascript", "Jacqueline, Elise and Ian are similarly knowledgeable about Liferay", and so on. The results in this case were slightly disappointinted with this category of questions scoring an average of **3.6** (weak agree). However, upon further investigation, it was discovered that the results reflected errors by participants when building the knowledge map. For example, the knowledge map for one team indicated that there was one expert on a knowledge asset labelled as *Liferay.* In fact, it transpired that a second person was also highly knowledgeable but his activities with regard to the *Liferay* knowledge asset were not properly logged. This skewed the results towards disagreement. The second angle from which this question was approached involved providing each participant with a list of five knowledge assets and asking them to rank them in descending order of familiarity. These rankings were then compared with rankings derived directly from the knowledge maps using the Spearman Rank Correlation Coefficient. This measure provides a value between -1 and 1 where a value close to -1 indicates that your data is negatively correlated, a value close to 0 indicates no linear correlation of your data whilst a value close to 1 indicates a positive correlation of your data. The Spearman Rank Coefficient was calculated for each of the eight team members' rankings and the results plotted on a frequency distribution graph (see figure 6). The results indicate that there is a positive correlation between the rankings provided by participants and those derived from the knowledge maps. Even though the level of correlation varies, one should keep in mind that the participants made some errors when logging events throughout the exercise. If these mistakes are corrected, correlation is likely to improve.

Finally the effectiveness of the metrics presented in this paper was investigated. This was done by drawing certain conclusions based on metrics and then soliciting participants to score their agreement with the conclusions. Typical conclusions include "if Joseph leaves the team, our knowledge of photoshop will suffer", "it would be difficult to transfer your knowledge about servlets to someone who does not know Java or HTML", and so on. These conclusions were drawn from values of KMR, PMR, Embeddedness and Knowledge Distance metrics. Statements designed to gauge the validity of such conclusions scored an average of **3.9** (agree). Although this is positive, further investigation into individual results revealed an unexpected problem. There were knowledge assets used by the team which were easy to learn, apply and remember. Usually this was because the knowledge assets were 'small' in size. Even though such knowledge assets were not necessarily applied very often, participants still felt they had a strong knowledge of them. This was contrary to what the event-based mechanism of the knowledge map concluded. This development gives rise to the possibility of modifying the work presented here to reflect these types of situations.

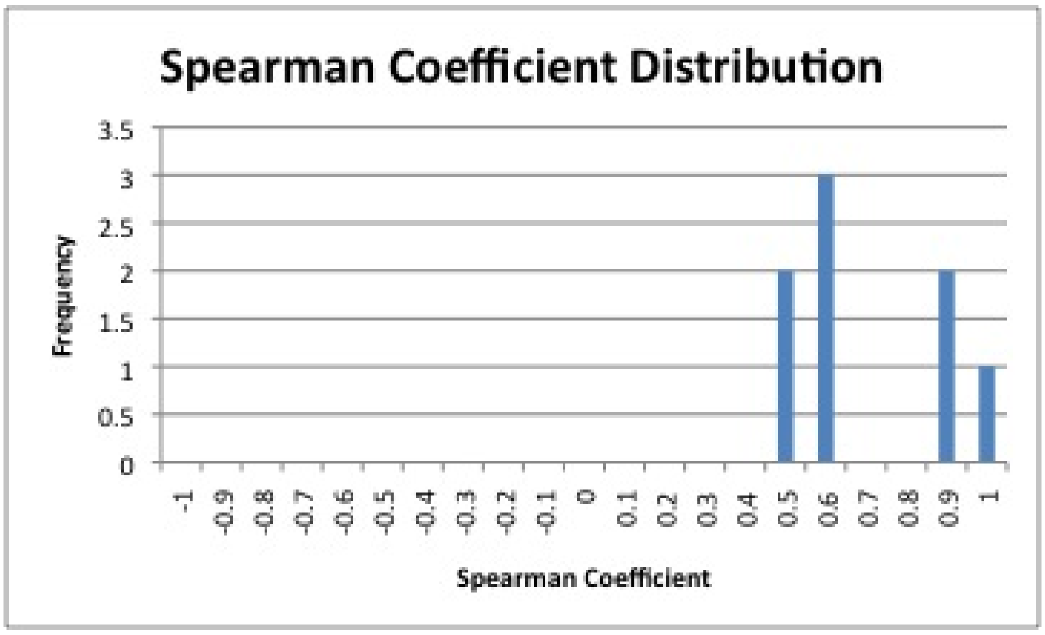

Figure 6: Distribution graph of Spearman Rank Coefficients for the knowledge magnitude ranking test

In summary, although the evaluation has its limitations, it does give encouraging results. It did however uncover two potential issues. The first is a question of incorporating the use of the language into a process so that knowledge maps are correctly created and maintained. This may sound simple but could be a challenge considering the tendency of knowledge workers to resist routine tasks. The second issue refers to the fact that the language does not differentiate between knowledge assets of different sizes. In some cases, this can skew conclusions derived from a knowledge map.

References

[AL01] Maryam Alavi and Dorothy E. Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 25(1):107–136, 2001.

[BD08] F. Bjørnson and T. Dingsøyr. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology*, 50(11):1055–1068, October 2008.

[BM06] Florian Bayer and Ronald Maier. Knowledge risks in inter-organizational knowledge transfer. In *Proceedings of the International Conference on Knowledge Management and Knowledge Technologies*. ACM ICPS, 2006.

[Bra98] S. Branch. You hired 'em. but cab you keep 'em? *Fast Company*, September 1998.

[BRS92] Rodrigo A. Botafogo, Ehud Rivlin, and Ben Shneiderman. Structural analysis of hypertexts: identifying hierarchies and useful metrics. *ACM Trans. Inf. Syst.*, 10:142–180, April 1992.

[Cum03] J. Cummings. Transferring r&d knowledge: the key factors affecting knowledge transfer success. *Journal of Engineering and Technology Management*, 20(1-2):39–68, June 2003.

[DDR05] Torgeir Dingsøyr, Hans Karim Djarraya, and Emil Røyrvik. Practical knowledge management tool use in a software consulting company. *Commun. ACM*, 48:96–100, December 2005.

[DH03] Scott B. Droege and Jenny M. Hoobler. Employee Turnover And Tacit Knowledge Diffusion: A Network Perspective. *Journal of Managerial Issues*, 15(1):50+, 2003.

[Duf99] N. Duffy. Benchmarking knowledge strategy. *Leveraging Knowledge for Business Performance 1999: Knowledge In Action*, 1999.

[Ear01] Michael Earl. Knowledge management strategies: Toward a taxonomy. *Journal of Management Information Systems*, 18(1):215–233, May 2001.

[Kou03] S. Koudsi. Actually, it is brain surgery. *Fortune*, March 2003.

[NNI98] The Nolan Norton Institute. *"Putting the knowing organization to value" white paper.* Nolan Norton Institute, 1998.

[Non94] I. Nonaka. A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1):14–37, 1994.

[RMA02] M. F. Ramal, R. de Moura Meneses, and N. Anquetil. A disturbing result on the knowledge used during software maintenance. In *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)*, pages 277–, Washington, DC, USA, 2002. IEEE Computer Society.

[Szu96] Gabriel Szulanski. Exploring internal stickiness: Impediments to the transfer of best practice within the firm. *Strategic Management Journal*, 17:27–43, 1996.

[Tiw00] A. Tiwana. *The Knowledge Management Toolkit: Practical Techniques For Building A Knowledge Management System.* Prentice Hall, 2000.

[Zac98] M. Zack. What knowledge-problems can information technology help to solve. In *Proceedings of the Fourth Americas Conference on Information Systems*, pages 644–646, 1998.