Competence matching system
Anton Jonsson

May, 26, 2015
Master’s Thesis in Computing Science, 30 Credits
Supervisor at CS-UmU: Mikael Rännar
Examiner: Fredrik Georgsson

Umeå University
Department of Computing Science
SE-907 87 UMEÅ
SWEDEN
Abstract

Finding the most suitable consultant for an assignment is challenging and becoming more and more important in the competitive IT-market. In this thesis a system for matching consultants to assignment specifications is implemented. The system also supports a representational scheme that accounts for cascading competences. This improves the matching system by taking attributes from relating competences in consideration when calculating the most suitable consultant. People that normally assigns consultants to assignments has evaluated the system and the result from that evaluation is a system that gives good matches.
## Contents

Introduction ...................................................................................................................... 1
Partner .............................................................................................................................. 1
Problem Statement ......................................................................................................... 2
Purpose and goal .............................................................................................................. 2
In-depth study .................................................................................................................. 2
Method ............................................................................................................................. 3
Result of feasibility study .............................................................................................. 5
Consultant ....................................................................................................................... 5
Competence .................................................................................................................... 5
Interviews ......................................................................................................................... 5
Assignment ...................................................................................................................... 6
Assignment specification ............................................................................................... 6
Résumé .............................................................................................................................. 7
Comparison between résumés and assignment specification ....................................... 8
Process of assigning consultants ................................................................................ 9
Issues assigning consultants ....................................................................................... 10
Representation of competences .................................................................................... 12
Competence experience ............................................................................................... 12
Competence relationships ............................................................................................ 13
Representing relationships between competences ...................................................... 15
Assigning weights ......................................................................................................... 18
Representing consultants ............................................................................................. 19
Representing assignment specification ......................................................................... 19
Matching assignment specifications and consultants .................................................. 21
Algorithm theory ........................................................................................................... 21
Level coefficient .......................................................................................................... 22
Path coefficient ............................................................................................................ 27
Average coefficient ....................................................................................................... 27
Cost coefficient ............................................................................................................. 28
Results ........................................................................................................................... 31
System description ....................................................................................................... 31
Data layer ....................................................................................................................... 32
Business layer ............................................................................................................... 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation layer</td>
<td>39</td>
</tr>
<tr>
<td>Deployment Azure</td>
<td>43</td>
</tr>
<tr>
<td>Evaluation</td>
<td>45</td>
</tr>
<tr>
<td>Method</td>
<td>45</td>
</tr>
<tr>
<td>Result</td>
<td>45</td>
</tr>
<tr>
<td>Conclusions</td>
<td>46</td>
</tr>
<tr>
<td>Restrictions</td>
<td>47</td>
</tr>
<tr>
<td>Limitations</td>
<td>47</td>
</tr>
<tr>
<td>Future work</td>
<td>48</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>49</td>
</tr>
<tr>
<td>References</td>
<td>50</td>
</tr>
</tbody>
</table>
Introduction

The importance of the IT market in Sweden is significant. The market turnover for 2012 was about 550 billion SEK (the whole retail industry turnover was 564 billion SEK) and employs about 183 000 people (4.1 % of Sweden’s total work force) (itstatistik, 2013).

Today’s market for IT consultant firms is competitive. Considering only public procurement in Sweden for 2013, each procurement relating to the IT industry received on average 11.2 offers. The average number of offers for all industries were 4.6. In IT related procurements 37% of the offers were given a contract. The average value for contract wins in all industries were 40% (Konkurrensverket, 2014). This statistics suggests that the IT industry is more competitive than many other industries since they have more competitors and a more difficult time winning offers.

Being in a competitive market entails that offers made by consultant firms must meet the customers’ demands and requirements to an even higher extent then what might be necessary in other industries. Consultant firms that are skillful at this have an increased possibility to win offers which increases the possibility for them to stay financially viable.

XLENT is a software consultant firm located in Umeå. Their process of assigning consultants to customer assignments is today mostly done using the experience and knowledge of the people in charge of assigning. This process is challenging and also limits the variety of people that can perform the assigning process.

A tool for facilitating and improving the identification of consultants for an assignment could potentially help the company in a number of ways. The company’s ability to win offers could improve since better consultant candidates are identified. Time could be saved since the process of identifying consultants would be simplified. More people could also perform the identification task which makes the company less dependent on a few individuals.

Partner

Xlent Consulting Group is a Scandinavian strategy- and IT-consultant company. Their main focus is to help companies and organizations to increase their business value in their customer- and business relations. One of its subsidiaries is XLENT NORR AB who is the main partner used for this thesis project. They have supplied a workplace, equipment and knowledge for conducting this thesis.
Problem Statement

The primary objective of the project is to develop a competence management system for helping XLENT to assign consultants to assignments. This is done by creating a system that stores consultant and competence information while also implementing a matching engine used to match consultants to assignment specifications. The system should always be available and accessible by users who should be able to interact with the system using a graphical user interface. The main problem to solve is how to match consultants to assignments, therefore it is not required to produce a production ready system with features like user authentication. The research questions addressed in this thesis report are the following:

- How can competences be identified from an assignment specification?
- How can suitable consultants be identified from competences?
- How can a consultants cascading competence knowledge be accounted for?

Purpose and goal

The purpose of the project is to facilitate the access and management of consultant information for XLENT's consultants. Users should more easily be able to find the consultants they desire and more users should be able to search for consultants. This will lead to less time needed for finding a consultant and that better suited consultants are appointed to assignments.

The goal of the project is to design and implement a system for managing competences and consultants. Users should also be able to search for consultants and get the best suited candidates based on assignment specification.

In-depth study

The in-depth study focuses on identifying any issues XLENT has with today’s processes for assigning consultants to assignments. This includes how cascading properties of competences can be represented and how to match requested competences with suitable consultants.
Method

The project is divided into three phases: research, development and testing.

The initial step in the research phase is to do a feasibility study where the current processes of assigning consultants used by XLENT are analyzed. This is done by interviewing employees at XLENT to gain knowledge about how they assign consultants to assignments. By analyzing the result of the interviews potential issues with their current way of working can be identified. The feasibility study determines if the issues are significant enough to justify doing a whole thesis project.

The next step is developing a project plan and an initial design on how to solve the problems. The project plan specifies how to perform the project and why it is done. The initial design contains the overall approach to solving the problems identified in the feasibility study. This is done by researching literature, journals and publications to get a better understanding of the problems and identifying different methods of solving them. XLENT will also provide résumés (CVs) of their consultants and example of assignment specifications which will be help in identifying a way of matching consultants to assignments and how the competences can be represented.

After the research phase the development phase is initialized. Using the knowledge gathered from the research phase a more concrete design will be developed and implemented. Focus during this phase is using an agile approach to development, which is done using SCRUM. Development will be done using sprints where a handler at XLENT will commit to the role of product owner. Activities performed includes: sprint planning, sprint review and sprint retrospective. A backlog is used to define which tasks and stories that is to be performed and how they are prioritized. The development phase should generate a prototype that solves the problems identified in the research phase.

Lastly a testing phase will be done where some of the people responsible for assigning consultants evaluates the system and expresses their thoughts of the system. This will be used to determine if the system actually solved the problems and issues identified in the feasibility study.
Result of feasibility study

This section focuses on the result of the feasibility study. This includes identifying the problems and issues with matching consultants to assignment specifications.

Consultant

Being an IT consultant today is a complex profession. The consultant is required and expected to possess a multitude of skills such as:

- Advisory skills
- Technical skills
- Business skills
- Communication skills
- Management skills
- Advisory language skills
- Business and management language skills
- Technical language skills

The purpose of the consultant is to advise businesses on how to best use information technology to meet their business objectives. In practice this can be done by implementing or guiding businesses on how to improve IT-systems and structures. The skills listed above are all needed to be able to advice businesses. (Wikipedia, 2015a)

Competence

Consultants can have a range of competences, but what is a competence? According to Oxford Dictionaries competence is “the ability to do something successfully or efficiently” (Oxford Dictionaries, 2015). This definition refers to competences only as an ability which is too limiting considering consultants. Therefore the definition of competence will be extended to also include skills and knowledge, since these are attributes that a consultant is expected to have.

Interviews

The interviews were performed by telephone with employees at XLENT that normally assigns consultants to assignments. The questions asked were:

- Which steps do you take to identify which competences an assignment specification contains?
- How do you identify which consultant that is best suited for an assignment?
Assignment

From the interviews performed at XLENT, there are three types of assignments that they can acquired from businesses: projects, resource renting and commitments.

Projects are assignments where the consulting firm is responsible for the result in time and money. The firm is also a part of the steering group who is responsible for the project’s direction, scope and planning. Requirements and penalties are handled in a project directive and project plan. Assigning of positions within the project is done by the firm without customer interference.

Resource renting are assignments where the consulting firm is renting out resources (usually consultants) to a partner of the customer. The partner (or other project leader) is responsible for carrying out a project and the resource is needed to fill a role in that project. The firm is responsible to ensure that the resource is delivering by the requirements for the requested role. Usually 1 month notice.

Commitments can typically be a management commitment where the customer is purchasing a service from the consultant firm that the customer otherwise would have handled themselves. The firm is responsible for delivering the requested result to the customer. Terms and penalties are handled in a service level agreement (firm and customer agrees on a definition for the service and aspects such as: quality, responsibility and scope (Wikipedia, 2015b)). Commitments usually has a contract length longer than 12 months and typically 6 months notice.

Assignment specification

Each assignment comes with a specification which is a problem description with varying amounts of requirements. Requirements can be of two types: must and should requirements. The specifications often contain desired or required roles. A role can be seen as a label for a consultant using job titles (ex. Developer, Project leader). The definitions of the roles varies between specifications, from a definition simply stating a job title to a definition containing the tasks a role is expected to perform (ex. Unit testing, documentation). The amount of competences a role needs to master varies between specifications. Some contains detailed descriptions of methods, technologies, environments and which competences that are meritorious while others do not mention any competences and instead describes functionality and design. A competence can also contain a description of which version or edition of it that needs to be mastered by a potential consultant . Specifications can also contain a level value representing the experience a consultant should have using a competence. The level can be expressed as years (number of years working with the competence) or a value between 1 and 5. If the level is described by a value a definition is available defining each level. Usually characteristic competence for a role is described (ex. independent, good working with others). If an initial requirement list is missing or needed to be supplemented, it is done by XLENT in collaboration with the customer.
Résumé

A résumé or curriculum vitae (CV) is summarization of a person’s work experience, educational background and skills (about careers, 2015). The résumés for the consultants at XLENT are stored on the company network as text documents (Microsoft Word or PDF). These résumés are not standardized and varies between offices (even within offices) which is an obstacle to create a common basis to search for consultants. The résumés often describes roles in the same fashion as for assignment specifications but are more detailed, defining which competences a role consists of. The experience using a competence is rarely described, when it is, both years and levels are used as metrics. Résumés often includes short summarizations of previous projects that includes what a consultant did in them and what technologies and methods that were used. Every résumé contains the consultants’ education, certifications, earlier employments, language skills and personality.
Comparison between résumés and assignment specification

A part of identifying the most suitable consultant for an assignment is to map a consultant’s résumé to an assignment specification. Therefore a comparison between the two has been made that can be seen in Table 1.

<table>
<thead>
<tr>
<th>Assignment specifications</th>
<th>Consultant résumés</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles are often described in specifications. Often limited and lacking description of competences needed by role.</td>
<td>Roles are most of the time described. Extensive description of which competences a role consists of.</td>
</tr>
<tr>
<td>Level of experience for competences are sometimes described.</td>
<td>Level of experience for a competence is rarely described.</td>
</tr>
<tr>
<td>Roles described is often some sort of developer.</td>
<td>Résumés contains different roles and they and the definition of them can be different to the role in an assignment specification.</td>
</tr>
<tr>
<td>Personality of desired consultant is often described.</td>
<td>Consultants often characterize themselves, defining their personality.</td>
</tr>
<tr>
<td>Can contain description of which methods, technologies and environments that the consultant needs to master. Sometimes also which version/edition they need to master.</td>
<td>Always describes which methods, technologies and environments that the consultant masters. Rarely describes which version/edition they master.</td>
</tr>
</tbody>
</table>

Table 1. A comparison of consultant résumés and assignment specifications.

Both the specifications and the résumés uses roles to group together competences. An issue with this is that they have different definitions of which competences a role should consist of. Also, using roles to group together competences is unlikely to work in reality where it is likely that consultants masters competences from different roles but not all competences that defines a role.

Specifications does on occasion define a competence with a level attached to it. Résumés rarely does this which introduces a problem in mapping competences to consultants.

Both résumés and specifications often describes the personality of the consultant.

Methods, technologies and environments are mentioned in specifications and is always described in résumés.
Process of assigning consultants

When XLENT receives an assignment specification the needed competences are identified by a person. The responsible persons generally uses their own knowledge and experience for this process.

When the competencies needed for the assignment are identified, consultants can be assigned. This is done by estimating which consultants that would best match the identified competences for the assignment. In general the person assigning uses their own experience and contacts to perform the estimation. A common method used to find potential consultants is to ask around the local office. Other offices within the XLENT organization are also asked, especially if the person assigning knows someone at a different office. Another method used when estimating potential consultants is to search among résumés. This method is mostly used by the less experienced people. If no consultant possesses the required competences a customer is desiring, sub-consultants can be leased. If it is unclear if a consultant fulfills the requirements that the customer has, the résumé can be sent to the customer for verification. The communication between people assigning, consultants and other parties is done in direct contact, telephone or e-mail.

During procurements it might be required to format résumés to a format defined by the customer.
Issues assigning consultants

When a consultant is to be appointed for an assignment, two tasks need to be done. First the needed competences from the requirements need to be identified. Secondly, consultants that might fulfill the requirements need to be identified.

In general the process of identifying competences and consultants is not considered a major problem by the employees. This is likely due to the good experience and knowledge of the persons that identifies competences and consultants. The complaints the participants did have about the current way of identifying competences and finding consultants were:

- Difficult knowing which consultants are available.
- Difficult knowing what competences an office possesses.
- Tediously not knowing if a résumé is up to date.
- The knowledge base of who-has-what-competence is largely kept in the heads of a few.
- Time-consuming to reformat résumés to the customers format.

By analyzing the current way of identifying competences and consultants some additional points were added:

- Time-consuming to contact others when searching for consultants.
- Hard to get a base of consultants to base the assigning process on.
- Risk of missing the best suited consultant since it is difficult getting a base of consultants.
- People might get extra work delegated to them when they are helping someone to search for consultants.
- People assigning consultants needs to have good knowledge about which competences consultants have and if they are available. This should not be necessary.

Since the profession of being an IT consultant is complex, it entails that the person appointing consultants needs to have great knowledge about the competences consultants have and which competences are needed to fulfill the customers' requirements. This information combined with the issues identified above entails that the tasks of identifying competences and consultants is challenging and justifies the implementation of a system that facilitates these tasks.
Representation of competences

This chapter focuses on how competences can be represented. It involves defining what a competence should consist of and how they are related to each other.

Competence experience

Two people having the same competence does not mean that both could execute the same assignments equally well. As both the assignment specification and résumés demonstrated, it is common to attach some form of experience to competences. Considering two persons practicing the same competence but one having 10 years of experience and the other with 1 year of experience. It is reasonable to believe that the person with 10 years of experience is the more competent of the two persons. How can experience be accounted for in a competence?

Dreyfus suggested a model for how students acquire skills, the Dreyfus model. The model emphasizes that the development of a skill is more dependent on the practical experience then on more abstract learning principles (Dreyfus, 1987). A student passes through five stages when acquiring and developing a skill and is summarized by Michael Eraut in the list below (Cheetham, Graham, Chivers, Geoff, 2005).

1. Novice
   1. "rigid adherence to taught rules or plans"
   2. no exercise of "discretionary judgment"
2. Advanced beginner
   1. limited "situational perception"
   2. all aspects of work treated separately with equal importance
3. Competent
   1. "coping with crowdedness" (multiple activities, accumulation of information)
   2. some perception of actions in relation to goals
   3. deliberate planning
   4. formulates routines
4. Proficient
   1. holistic view of situation
   2. prioritizes importance of aspects
   3. "perceives deviations from the normal pattern"
   4. employs maxims for guidance, with meanings that adapt to the situation at hand
5. Expert
   1. transcends reliance on rules, guidelines, and maxims
   2. "intuitive grasp of situations based on deep, tacit understanding"
   3. has "vision of what is possible"
   4. uses "analytical approaches" in new situations or in case of problems
Important to note is that when a person stops developing a competence, the level will descend. This is especially true in an IT environment where tools and technology quickly changes. Take Java for example. There has been nine major releases of Java during the last 18 years (1.0, 1.1, 1.2, 1.3, 1.4, 5.0, 6, 7, 8), one release every second year on average (Wikipedia, 2015c). Every new release contains new features, consider someone not using Java for four years and missing two releases. That person will have missed multiple new features and will likely need to decrease their level of experience.

Since different levels are often described in the assignment specifications it is important that consultants can express which levels their competences are. Assignments often uses five levels which the Dreyfus model also does, therefore it is reasonable to use that scale. The Dreyfus model also gives a definition of what each level represents, which makes it easier for consultants to identify which levels their competences should be at.

A problem with using levels is that customers that have defined levels for the desired competences uses their own definitions of what each level represents. Their definitions need to be mapped to the definitions in the Dreyfus model. To limit the problem scope of this project, this is an issue that is not addressed by the system. Instead the people appointing consultants need to map the customers level definitions to the Dreyfus model.

Competence relationships

Dreyfus tells us that a competence held by a consultant is not static but continually evolving and needed to be maintained. A question to ask is if competences are isolated to their own domain or if competences can share attributes among themselves. Considering programming languages, it is obvious that techniques learned in one language can be applied in others. Examples can be how to express iterations, variable declaration and how functions work. Do I know how a for-loop is expressed in C++, then I know how to express it in Java. That being said, a Java developer might not be aware that manually allocated memory in C++, also needs to be manually de-allocated (no garbage collector in standard C++). This means that knowing a competence can imply that some of the attributes within a competence can be applied on other competences. This recognition raises two more questions, which competences are related to which and how much of the attributes within one competence can be applied to another?

Let’s consider a set of competences. Researching how these competences are related to each other and representing it using a model could be done. However, it would probably be difficult, time consuming and expensive since someone needs to analyze competences, identify which competences are related to which and define how related they are. Another issue is that it is not flexible enough. As described previously the IT industry is quick to adopt new technologies and tools, i.e. adopting new competences. Flexibility is needed to be able to add new competences, such as a new programming language. Competences also continually evolve which implies that the relationships between competences changes since more or less attributes from a competence can be applied to another. This is also something the flexibility of the model needs to handle.
Still, a model for representing how competences are related to each other is needed but with some degree of flexibility to cope with new competences and the changing relationships between them. The flexibility of the model could be improved by letting users themselves edit the relationships between competences. This would allow multiple users to define relationships which makes the process of keeping the relationships current less resource demanding. The system would also benefit from the combined experience and knowledge from a group of users.
Representing relationships between competences

What is the best approach to create the model that represent competences and the relationships between them? A few options are available and are described in this chapter.

**Hierarchy**

The concept of hierarchies are inherited from Aristotle who believed that all of nature was part of a unified whole that could be subdivided into different classes and subclasses (Wikipedia, 2015d). Each class can have a subclass that inherits all properties from the parent. A subclass can then be further subdivided into new subclasses (Kwasnik, Barbara H, 1999).

![Hierarchy of animals](image)

*Figure 1. A hierarchy with classes of animals where a characteristic property of a class is in parenthesis.*

Figure 1 describes a hierarchy with animals where each class of animal inherits the properties of the parent. For example: Human has inherited the properties “Multicellular”, “Warm-blooded” and “Large brain” since its parent is Ape who inherits properties from Mammal that inherits properties from Animal.

A problem using this approach when representing competences is the domain that competences are part of. Hierarchies assumes that all properties from a parent is inherited by the subclass. To do this, the domain that the competences are part of needs to be known, i.e. knowing which properties a class consist of. Defining all properties of a competence is difficult and then being able to classify competences that inherits all those properties is even more difficult. For example, C++ could be considered a subclass of object oriented programming. A property of object oriented programming could be the use of a garbage collector. By default C++ is lacking a garbage collector and is therefore not able to inherit all properties of the object oriented programming class.

Some competences could also inherit competences from different branches of the hierarchy which makes the classification even harder. For example it can be argued that C++ share properties with both object oriented programming and procedural programming. There could also be competences that share attributes in both directions which makes it difficult to create a hierarchy.
Another issue is how to represent the degree of similarity two classes share. As mentioned before competences share different amount of attributes between them. Meaning that competences are related to each other with different degrees of similarity. Using a hierarchy makes it impossible to define the degree of similarity between classes.

**Tree**

A similar method for creating a model for competence representation is using a tree. The major difference between a hierarchy and a tree is that trees subdivides classes based on rules and does not assume inheritance between classes (Kwasnik, Barbara H, 1999).

![Tree Diagram](image)

*Figure 2. A tree of classes describing the chain of command.*

Figure 2 describes a tree where each class represents an army rank. The relationships between the classes describes the chain of command. General reports to Colonel, Colonel reports to Sergeant, who reports to Private. Since inheritance is not assumed, a subclass does not inherit any of the properties from the parent. Therefore a Private is not a kind of Sergeant.

Using a tree solves some of the problems a hierarchy suffers from. Since it is possible to specify what a relationship between competences is, the issue of inheriting all the parent's properties can be solved. Using the example of object oriented programming and C++ it is possible in a tree to define a number of properties that the subclass should have, excluding garbage collector.

Still, the problem of classes sharing properties from other branches exists and it is impossible to define how much of the attributes a class shares with another. Neither are competences sharing attributes in both directions handled.
Graph

A third option for representing competences and the relationships between them is to use a graph.

![Graph Diagram](image)

*Figure 3. A directed graph where each node represents a competence and each vertex represents a relationship between the competences.*

Figure 3 represents a directed graph where the nodes represent the competences and the vertices represent the relationships between them. Since a graph is not a hierarchy, there is no assumed inheritance. Each node can also create an arbitrarily number of relationships to other nodes which solves the problem of nodes sharing properties with multiple nodes. To define how many of the attributes in a node that can be applied on another, a weight value can be added to each vertex.

![Graph Diagram with Weights](image)

*Figure 4. A directed graph where each node represents a competence and each vertex represents a relationship between competences. The weight each vertex has represents the proportion of attributes from a competence that can be applied on another.*
Figure 4 shows the directed graph where the weight associated with each vertex represents the proportion of a competence that can be applied on another competence. For example: 70% of C can be applied on C++, while 50% of C++ can be applied on C.

Using a weighted directed graph makes it possible to specify how many of the attributes in a competence that can be applied on another. It is also possible to define an arbitrary number of relationships between nodes and a unique value for each relationship.

Because of this, a directed weighted graph is chosen as the medium to represent the relationships between competences.

Assigning weights

Using a directed graph it is possible to create relationships between nodes. Assigning weights to the relationships defines how many of the attributes that one node can apply to another. Choosing the values for these weights is a challenge. As described before it is too difficult, time consuming and expensive to research the relationships between competences. The flexibility of adjusting the relationship model is also somewhat lost since a resource demanding research is needed to identify the new relationships.

The preferred way of assigning weights to relationships would be to automate the process. Richardson et al. (1994) proposes the usage of WordNet to create a knowledge basis for information retrieval. WordNet is a lexical database that groups English words into sets of synonyms called synsets. Synsets contains descriptions, usage examples and relationships to other synsets (Wordnet, 2015). Using the relationships between synsets it might be possible to get a conception of which competences that are related to which and what the appropriate weights should be. Creating a system that automatically identifies relationships and weights is likely a comprehensive task and will therefore be subjected to the limitations of this thesis.

XLENT possesses consultants with knowledge about a wide range of competences. Letting the consultants themselves adjust the relationship model is a less resource demanding and more flexible approach. Initially the model will be lacking but as time goes by it will become more comprehensive. An issue that might occur is that consultants might have different views on how competences are related to each other. Someone might argue that a specific relationship should have a certain weight value while someone else argues for another value. To limit the times this issue might occur it might be better to limit the number of consultants that are allowed to adjust the relationships. Considering the benefits of using this option it is acceptable that conflicts between consultants might occur.
Representing consultants

Besides storing the model for how competences are related to each other the system should also store the consultants that holds the competences. A consultant can be considered being a knowledge container, containing a variety of competences. As described previously consultants not only have different types of competences but also have different levels of experience using the competences. Representing competences for consultants will therefore contain two elements: description of skill or knowledge and the experience using them. The description is simply a keyword representing a skill or knowledge. The experience using a competence is expressed as a numeric value between 1 and 5, where 5 is the highest experience. Table 2 describes how a consultant’s competence is represented.

<table>
<thead>
<tr>
<th>Competence</th>
<th>Level</th>
</tr>
</thead>
</table>

*Table 2. Describes how a consultant’s competence is represented.*

The reason for choosing a level representation that is numerical is that it makes it easier for consultants to identify which level of a competence they have. Using a system of years does not necessarily give a good indication of the experience the consultant has using the competence. A consultant with 10 years of experience using a competence but who has not used it for 5 years will likely be missing some of the knowledge that is expected by a consultant with that type of experience. Using levels makes it easier for consultants to fit their competence based on the knowledge they have today.

Representing assignment specification

By analyzing the assignment specifications there are three features that defines a desired competence: description of skill or knowledge, experience using the competence and importance of mastering it. The description is once again a keyword representing the skill or knowledge. The experience is also expressed as a numerical value between 1 and 5. Importance of using a competence can have two states: *must* (consultant *must* know C++) and *should* (consultant *should* know C++). A *must* requirement defines that a competence must be among a consultant’s competences, it is obligatory. A *should* requirement defines that a competence is desirable among a consultant’s competences, but not obligatory.

Having two states representing the importance of a competence comes from the client specifications. In those specifications, the importance of having a competence is described using *must* requirements and *should* requirements. Table 3 describes how an assignment specification is represented.

<table>
<thead>
<tr>
<th>Competence</th>
<th>Level</th>
<th>Requirement</th>
</tr>
</thead>
</table>

*Table 3. Describes how an assignment specification is represented.*
Matching assignment specifications and consultants

This chapter focuses on the algorithms used to match assignment specifications and consultants.

Algorithm theory

To match an assignment specification and a consultant, a matchmaking algorithm is needed. One approach is to use a vector space model where each request (containing competences) and consultant would be represented as a multidimensional vector. Each vector represents a competence and by using different vector operations, requests and consultants can be compared (Salton et al., 1975). Another approach is to use an ontology-based matchmaker (Colucci et al., 2003). Skills are considered a commodity that are available in a virtual market where the customers are entities that need the commodity. An example of a customer is a project. To calculate similarities between assignment specifications and consultants, a semantics based matching algorithm can be used (Zhong et al. 2002).

Matchmaking strategies can be divided into two groups: logic deductive and similarity based. Logic deductive methods matches entities based on pure logic and is considered being of high precision and recall, but of low flexibility. This means that it is good at finding consultants that matches a request perfectly but lacks flexibility to identify consultants that almost matches. It has also lacking possibility to rank how well consultants matches an assignment. Similarity based methods of matchmaking are the opposite of logic deductive, limited precision and recall but with high flexibility. Meaning that similarity based methods are better at identifying consultants that are similar to a request and rank how well they match, but has limited capabilities in identifying perfect matches (Bianchini et al, 2008).

The algorithm that the system will use is a combination of the two strategies to benefit from the strengths that they have and limiting the weaknesses they have. Initially a logic deductive system matchmaking will be performed to gather all consultants that has the requested competences. After this the similarity based matching will be performed where the properties of the requested competences will be matched against the properties of the corresponding competences that the consultant has.

The simplest way to perform the initial logical deductive matching would be to match the competences from the request with the competences that consultants have. The consultants with all the competences that are in the request will be considered a match. A flaw with this method is that only perfect matches will be considered being a match. A consultant that has 9 out of 10 competences will still be considered inappropriate although in a real situation that consultant might have been a good candidate.

To give the consultant with 9 out of 10 competences a chance to be among the consultants that are considered being matches, the initial method will be supplemented. By investigating if the consultant has a competence with a relationship to the requested competence that the consultant is missing, the consultant can still be considered being a match. One thing to consider is the importance of the requested competence. Only competences with should requirement can be handled using the graph.
In practice the logical deductive method is performed by mapping each of the competence in the request to the best matching competence a consultant has. If a requested competence is not among a consultant's competences and the competence has a *should* requirement, the relationship model is used (the graph). Initially a group of competences with relationships to the desired competences are attained. The group of competences includes competences which need to pass multiple competences to reach the requested competence. For a relating competence to be considered relevant, the weight value cannot be too small and it can neither be allowed to pass through too many competences, to reach the requested one. If a consultant has multiple competences with a relationship to a desired competence, the competence with the highest weight is chosen. If a consultant lacks all of the competences that have a relationship to the desired competence, including the competence itself, then the consultant is not considered being a match for the request.

The second part of the matching algorithm, the similarity based, is performed on the group of consultants that are considered matches. The purpose of the similarity based matching is to rank the consultants that are matching the request. This is done by matching the properties of the competences from the request with the properties that the consultants' competences have. The properties that are of interest are the levels of experience and the weights that are associated with the relating competences.

**Level coefficient**

The level coefficient represents how experienced a consultant is using a competence in relation to a desired experience.

Levels can be between 1 and 5 for a competence and by calculating the differences between two of them, nine possible states can be identified. Each difference represents a state and the smallest state is -4 (1 - 5 = -4) while the largest states is 4 (5 - 1 = 4), see Table 4.

<table>
<thead>
<tr>
<th>State</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>

*Table 4. Table displaying the different states.*

The ambition is to normalize the states on the interval [0, 1]. By normalizing the states the values in the interval can represent more meaningful data. For example, by normalizing on the interval [0, 1] we know that 0.5 will be the point where two states are equal.
The initial step to normalize the states is to convert the states to ranks, from 1 to 9.

\[ Level_{Max} = 5 \]

\[ Rank = Level_{Max} - (Level_{Requested Competence} - Level_{Consultant Competence}) \] (0)

Where

\( Level_{max} \) is the maximum possible level of experience a competence can have. 
\( Level_{Requested Competence} \) is the level of experience the competence from the request has. 
\( Level_{Consultant Competence} \) is the level of experience the competence from the consultant has. 

\( Rank \) is the value assigned to the state of the two competences.

Using Equation 0 the ranks for all the states can be calculated, see Table 5.

<table>
<thead>
<tr>
<th>State</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

*Table 5. The states and their respective ranks.*

The next step is to normalize the Rank to the interval \([0, 1]\).

\[ Rank_{Max} = (Level_{Max} * 2) - 1 = 9 \]

\[ Rank_{Normalized} = \frac{Rank - 1}{Rank_{Max} - 1} \] (1)

Where

\( Rank_{Max} \) is the the highest possible rank. 
\( Rank_{Normalized} \) is the normalized rank.

Using Equation 1 it is possible to normalize the rank which can be seen in Table 6.

<table>
<thead>
<tr>
<th>State</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Rank_{Normalized}</td>
<td>0</td>
<td>0.125</td>
<td>0.250</td>
<td>0.375</td>
<td>0.500</td>
<td>0.625</td>
<td>0.750</td>
<td>0.875</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 6. States, ranks and normalized rank.*

The purpose of normalizing the ranks is to get coefficients that will reflect the relations between a consultant’s experience using a competence compared with the requested experience of a competence. The normalized rank represents those coefficients.
Let’s say that level 3 is requested for a competence and a consultant has the same level for the corresponding competence. The state would be 0, Rank 5 and normalized rank 0.5. Thus the coefficient would be 0.5 which can be considered a penalty, a desired coefficient would be 1 which is neither a penalty nor a reward. When consultants’ levels are above the requested levels, the coefficient value should be above 1 and when consultant levels are below the requested levels, the coefficient value should be below 1. Also, to limit the impact that difference in level will have on matching consultants it is necessary to optimize the range from where coefficients can be generated from. Therefore a function is needed that generates coefficients between two values, defined by the users. These two values, $Rank_{max}$ and $Rank_{min}$ represents the maximum and minimum coefficient values that can be generated. Another assumption made is that having a level below the requested is more severe than the beneficial part of having a level above the requested. For example, if a request contains a competence with level 3, consultant A has level 2 for the competence, consultant B has level 4 for the competence. The penalty for consultant A should be larger than the reward for consultant B.

Some of the values the function should generate can be defined. The lowest value should be with a normalized rank of 0.

$$f(0) = Rank_{min}$$

With a normalized rank of 0.5 the value should be 1, since having the same level as in the request should not generate a penalty or reward.

$$f\left(\frac{1}{2}\right) = 1$$

The maximum value should be with a normalized rank of 1.

$$f(1) = Rank_{max}$$

An approximation of the function that generated these values can be identified by using curve fitting. Curve fitting is used to construct functions that best fit a series of data points. The method used for this is polynomial curves (Wikipedia, 2015e).

Below is the definition of the function which creates a system of linear equations. $C_i$ are the constants that need to be solved.

$$f(x) = C_0 + (C_1 \times x) + (C_2 \times x^2)$$

$$f(0) = Rank_{min} = C_0 + 0 + 0 = C_0$$

$$f\left(\frac{1}{2}\right) = 1 = C_0 + \frac{1}{2}C_1 + \frac{1}{4}C_2$$

$$f(1) = Rank_{max} = C_0 + C_1 + C_2$$
Solving the system of linear equations using Gauss elimination.

\[
\begin{bmatrix}
1 & 0 & 0 & | & \text{Rank}_{\text{min}} \\
1 & 1 & 1 & | & 1 \\
\frac{1}{2} & \frac{1}{4} & | & \text{Rank}_{\text{max}}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0 & 0 & | & \text{Rank}_{\text{min}} \\
0 & 1 & 1 & | & 1 - \text{Rank}_{\text{min}} \\
0 & 0 & 1 & | & \text{Rank}_{\text{max}} - \text{Rank}_{\text{min}}
\end{bmatrix}
\]

\[
\Rightarrow
\begin{bmatrix}
1 & 0 & 0 & | & \text{Rank}_{\text{max}} - \text{Rank}_{\text{min}} \\
0 & 1 & \frac{1}{2} & | & 2 - 2 \cdot \text{Rank}_{\text{min}} \\
0 & 0 & 1 & | & \text{Rank}_{\text{max}} - \text{Rank}_{\text{min}}
\end{bmatrix}
\]

\[
\Rightarrow
\begin{bmatrix}
1 & 0 & 0 & | & \text{Rank}_{\text{max}} - \text{Rank}_{\text{min}} \\
0 & 1 & \frac{1}{2} & | & 2 - 2 \cdot \text{Rank}_{\text{min}} \\
0 & 0 & 1 & | & -2 + \text{Rank}_{\text{max}} + \text{Rank}_{\text{min}}
\end{bmatrix}
\]

The matrix is now in triangular format, and using back substitution, the following constants are calculated.

\[C_2 = -4 + 2 \cdot \text{Rank}_{\text{max}} + 2 \cdot \text{Rank}_{\text{min}}\]
\[C_1 = 2 - 2 \cdot \text{Rank}_{\text{min}} - \frac{1}{2} \cdot (-4 + 2 \cdot \text{Rank}_{\text{max}} + 2 \cdot \text{Rank}_{\text{min}}) = 4 - \text{Rank}_{\text{max}} - 3 \cdot \text{Rank}_{\text{min}}\]
\[C_0 = \text{Rank}_{\text{min}}\]

Using the calculated constants the following function is approximated:

\[f(x) = \text{Rank}_{\text{min}} + (4 - \text{Rank}_{\text{max}} - 3 \cdot \text{Rank}_{\text{min}}) \cdot x + (-4 + 2 \cdot \text{Rank}_{\text{max}} + 2 \cdot \text{Rank}_{\text{min}}) \cdot x^2\]
Figure 5 shows a graph of the function using the x-values that the normalized rank can have.

![Graph](image)

*Figure 5. The level coefficient function displayed with max constant 0.6 and min constant 1.*

Using the function shown in Figure 5 a level coefficient can be calculated. To ensure that the functions minimum value is at 0 and maximum values is at 1, the definition of \( f(x) \) is extended. See Equation 2.

\[
F(x) = \begin{cases} 
\text{Rank}_\text{min}, & f(x) < \text{Rank}_\text{min} \\
\text{Rank}_\text{max}, & f(x) > \text{Rank}_\text{max} 
\end{cases}
\]  

(2)

*Where*

\( F(x) \) *is the redefinition of* \( f(x) \)
Path coefficient

The next step is to multiply the path weight to the level coefficient. Is the requested competence and the consultants corresponding competence a perfect match the weight will be 1. This is done using Equation 3.

\[
\text{Coefficient}_{\text{Competence}} = \text{Coefficient}_{\text{Level}} \times \text{Weight}
\]  
\[\text{(3)}\]

Where

\( \text{Weight} \) is the path weight.

\( \text{Coefficient}_{\text{Level}} \) is the coefficient value calculated by the function.

\( \text{Coefficient}_{\text{Competence}} \) is the complete coefficient for the consultants competence in relation to the requested competence.

Average coefficient

The next step is to calculate the average value of the coefficients among the set of competences. This is done using Equation 4.

\[
\text{Coefficient}_{\text{Average}} = \frac{\sum_{i=0}^{n} \text{Coefficient}_{\text{Competence} i}}{n}
\]
\[\text{(4)}\]

Where

\( \text{Coefficient}_{\text{Average}} \) is the average coefficient for a consultant’s competences.

\( n \) is the number of competences.
Cost coefficient

The average coefficient for a consultant describes how well suited a consultant is in comparison to an assignment specification. The final step to perform is to account for the cost of hiring a consultant. This is an important attribute since the price of a consultant is many times what determines if a consultant gets hired. The same principle as with the difference in level is applied here. Having a consultant that is more expensive than what is requested is a larger penalty than the benefit of a consultant being cheaper than what is requested. Also, having a consultant with the same cost as what is requested should not generate a penalty or reward. A difference compared to calculating level coefficients is that the possible x-values are not in a fixed interval, meaning that only one data point is which makes a polynomial curve fit futile. Instead a utility function is used to calculate the cost coefficient. Utility functions are mostly used in economics to determine how well something satisfies a need (Wikipedia, 2015f). Consider a person wanting an apple, the first apple the person takes will generate satisfaction. The second apple the person takes will generate less satisfaction than the first one and for each new apple consumed, the generated satisfaction will be less than the one before. The same reasoning can be applied to the cost of consultants. The cheaper the consultant is in relation to the requested cost, the less it adds to the consultant’s overall value. Also, the appeal of a consultant will greatly decrease the more expensive a consultant is in relation to a requested cost.

Since the interval that x-values are defined in is not fixed, it is not possible to normalize the differences in cost. Instead the ratio between the requested cost and the consultant’s cost will be used as x-values. If the ratio is 1, i.e. the consultant and the request has the same cost, the cost coefficient should be 1. To make consultants that are too expensive not even considered matches, the user can define from which x-value and below that the function will generate a value of 0. The user can also define the maximum value the function can generate which will limit the reward a consultant can get for being cheaper than the requested cost. To do this an exponential utility function will be used, defined in Equation 5.

\[
f(x) = \text{Coefficient}_{\text{max}} - e^{-C_0 x + C_1}
\]  

(5)

The predefined values are defined below:

\[
f(1) = 1
\]

\[
f(x_{\text{min}}) = 0
\]
Calculating the constants $C_1$ and $C_0$ is done below:

$$f(1) = 1 = \text{Coefficient}_{max} - e^{-C_0 + C_1}$$

$$-C_0 + C_1 = \ln(\text{Coefficient}_{max} - 1)$$

$$C_0 = C_1 - \ln(\text{Coefficient}_{max} - 1)$$

$$f(x_{\min}) = 0 = \text{Coefficient}_{max} - e^{-C_0 x_{\min} + C_1}$$

$$-C_0 x_{\min} + C_1 = \ln(\text{Coefficient}_{max})$$

$$-(C_1 - \ln(\text{Coefficient}_{max} - 1)) x_{\min} + C_1 = \ln(\text{Coefficient}_{max})$$

$$-C_1 x_{\min} + \ln(\text{Coefficient}_{max} - 1) x_{\min} + C_1 = \ln(\text{Coefficient}_{max})$$

$$C_1 * (-x_{\min} + 1) = \ln(\text{Coefficient}_{max}) - \ln(\text{Coefficient}_{max} - 1) * x_{\min}$$

$$C_1 = \frac{\ln(\text{Coefficient}_{max}) - \ln(\text{Coefficient}_{max} - 1) * x_{\min}}{(-x_{\min} + 1)}$$

$$C_0 = \frac{\ln(\text{Coefficient}_{max}) - \ln(\text{Coefficient}_{max} - 1) * x_{\min}}{(-x_{\min} + 1)} - \ln(\text{Coefficient}_{max} - 1)$$

The cost coefficient function is shown in Figure 6.

![Figure 6. Describes the function used to calculate the cost coefficient.](image)

A consultant’s total matching coefficients are obtained by multiplying the average coefficient with the cost coefficient.
Results

This chapter describes the system used to implement the algorithms from the chapter Matching assignment specifications and consultants.

System description

The system is deployed in the cloud using Azure, which is Microsoft's cloud computing platform. Architecture of the system is split into three layers: presentation, business and data. The presentation layer represents the client's user interface and is a single page application that communicates with the business layer. The Main logic of the system is handled in the business layer where requests from the presentation layer are handled and suitable actions are performed based on the request. The matching engine is also located in the business layer which is used for identifying the most suitable consultant based on desired competences. The data layer handles store and retrieve procedures and is utilized by the business layer. Figure 7 shows an overall description of the system.

Figure 7. An overall description of the system consisting of a presentation layer, business layer and a data layer.
Data layer

The data layer contains entities that stores data. See Figure 8 for an overall description of the system architecture for the layer.

The data layer is used by the business layer to insert, update or remove data from any of the storage entities. Instead of communicating directly with the storage entities, the business layer uses client modules for the communication. These clients are part of a sub layer inside the data layer called the data access layer. This sub layer contains interfaces with methods used for manipulating the storage entities. Three clients in the data access layer exists: Database, Graph database and a Blob storage.

The database client exposes methods for manipulating data in a relational database. The technology used for communication with the database is Entity framework. Entity framework is a .NET technology used for mapping objects and their relationships to an underlying database. By defining domain specific objects, Entity Framework maps objects to the tables and columns in the database, without concerning the developer. Entity Framework offers a higher level of abstraction which makes development faster and less code heavy (Wikipedia, 2015h). Another feature of Entity Framework is the ability to create and update table schemes based on the models that are represented. This approach is referred to as Code First and means that a developer can define some models and the relationships between them. Entity Framework can then model the tables and columns in the database based on the developer defined models. Code First is used in the system.
Since the system is hosted in the cloud it is appropriate to use the services available there, Azure SQL is one of these services. Azure SQL is a relational database build using Microsoft SQL Server and is a database used by the data layer (Microsoft Azure, 2015b). The database scheme is shown in Figure 9.

![Figure 9. An entity relationship diagram representing the consultant and competence entities and their relationship.](image)

There are two main entities, consultant and competence. Consultant can possess multiple competences with a specific level for each of those, hence the “has” relationship with the level attribute. The purpose of the relational database is to store all consultants, all competences and the competences that each consultant masters.

The graph database client exposes methods for manipulating data in a graph database. In the chapter Competence relationships, it is described how competences are related to each other and how they can be represented. The best approach for representing relationships was reasoned to be in a weighted directed graph. This is why a graph database is used, to store the graph where the relationships between competences are.

Neo4j is today the most used graph database and has arguably the largest community supporting it (db-engines, 2015). At the bottom, Neo4j is a NoSQL database and implements graphs using Java and Scala. To communicate with the database a REST API is available, making it possible to use Neo4j in many different environments, including .NET. Three entities exists in the database: edge, node and attribute. Nodes are used to symbolize entities in the graph and uses edges to create relationships between them. Attributes can be attached to both nodes and edges can be seen as a key-value property (Neo4j, 2015).

Requesting and sending data to Neo4j is done using queries similar to SQL. The query language used is called Cypher and lets developers describe what to insert, update, remove or select. Describing how the data is manipulated is not required in the same extent as in SQL. In Cypher a query consist of nodes and edges with attached attributes.

Nodes are defined as follows:

\[(\text{label}: \text{Type} \{\text{attribute: value}\})\]

Edges are defined as follows:

\[-(\text{label}: \text{Type} \{\text{attribute: value}\})-\]
Table 7 compares a SQL and Cypher query that is performing the same tasks and returning the same data. Query 1 fetches the classes that Anton is studying and query 2 fetches all students studying any of those classes.

**Sql query 1**

```
SELECT Class.name
FROM Student
INNER JOIN Study ON Student.id = studyId
INNER JOIN Class ON Class.id = classId
WHERE Student.name = "Anton";
```

**Sql query 2**

```
SELECT Class.name
FROM Student
INNER JOIN Study ON Student.id = studyId
INNER JOIN Class ON Class.id = classId
WHERE Class.id IN (
    SELECT Class.id
    FROM Student
    INNER JOIN Study ON Student.id = studyId
    INNER JOIN Class ON Class.id = classId
    WHERE Student.name = "Anton")
```

**Cypher query 1**

```
MATCH (:Student {name: "Anton"}) - [:Study] -> (class:Class)
RETURN class.name;
```

**Cypher query 2**

```
MATCH (:Student {name: "Anton"}) - [:Study] ->
  (class:Class) < - [:Study] - (student:Student)
RETURN student.name;
```

Consider two tables storing students and classes. A third table (Study) is used to map students to classes. Query 1 retrieves all classes that Anton is studying. Query 2 retrieves all students that are studying any of classes that Anton is studying.

The strength of the Cypher query language is displayed in Table 7. Query 1, finding all the classes a student is studying is compacter and arguably more easy to understand in the Cypher query compared to the SQL query. In Query 2 the Cypher query only needs to add a second relationship to the previous query. The SQL query is quickly becoming more complex by the need of adding a second Select statement.

---

Table 7. A table comparing a SQL and Cypher query performing the same task.
The Neo4j database is run in Azure, using a virtual machine and access to the database is done using the REST API. The graph database client used to communicate with the Neo4j database is build using an underlying .NET Neo4j client framework, called Neo4jClient. This framework supports basic CRUD operations and streamlines the process of writing Cypher queries (Neo4jClient, 2015).

The third client in the data access layer is the Blob storage client that offers methods for storing and retrieving data from the Azure blob storage. A blob is a collection of binary data that is stored in a single object. Azure includes a blob storage that is a service for storing unstructured data such as images and text files (Microsoft Azure, 2015a). This storage entity is used for storing images of consultants and configuration files. Communication with the storage is done using Microsoft's own blob storage client, CloudBlobClient.
Business layer

The business layer is a REST API service that handles requests from the presentation layer and performs suitable actions based on the request. REST (Representational State Transfer) is a design architecture that defines guideline on how to expose and transfer data using a web service. REST emphasizes a decoupled and lightweight architecture between client and server (IBM, 2015). Figure 10. An overall description of the business layer.

![Diagram of the business layer](image)

Figure 10. An overall description of the business layer.

The business layer is built using .NET technology, more precise ASP.NET Web API 2. Web API is a framework for exposing data and web services that are build using the .NET framework. The layer is also built with MVC in mind.

MVC (Model-View-Controller) is a design pattern that separates the processing of data from the representation of the data. This makes modification of the business logic and user interface easier. The pattern consists of a model that represents the data, a view that is the visual representation of the data and a controller that links view and model and decides which view to render based on user input and business logic (Yener et al., 2015).

The business layer follows the design pattern of MVC by separating the logic to controllers, representing data in models and returning response data based on user input.

Controllers handles calls and can manipulate models and views. Models represents the data and views is the outputting data that clients will receive (ASP.NET, 2015). The exposure of the data and services are performed using a Restful interface which accepts HTTP requests. The calls are done using AJAX, which is a technique for sending to and receiving from a server, asynchronously. This technology enables the client to dynamically change the content of a website (Wikipedia, 2015g).
From the presentation layer the client makes AJAX calls to the business layer. The call is initially routed to a controller object that performs suitable actions. How the routing should be performed is specified in a configuration file where the format of the URL is mapped to a controller. In the controllers the format of the methods maps to a specific HTTP method. For example: a controller named StudentController with a method named GetStudents will be invoked when a GET request is performed using the URL <root_api>/Student/. From the URL, Student maps to the StudentController and the GET request maps to the method declaration starting with Get.

Usually the controller needs to fetch or alter data in the data layer, this data is then placed in a model. When the controller is done manipulating the models the suitable model is sent to a media type converter that converts it to a JSON object that is sent to the presentation layer.

The two most important models used by the business layer are the consultant and competence models. The competence model represents a competence and the consultant model represents a consultant including the held competences.

Requests from the presentation layer is handled in the four controllers: Consultant, Competence, Config and Matching.

**ConsultantController**

Handles all actions related to consultants such as creating, updating or removing them. From the data access layer the database client is used for manipulating consultants and the blob storage client is used for uploading consultant images.

**CompetenceController**

Handles all actions related to competences such as creating updating or removing them. Also relationships between competences are handled in this controller. From the data access layer, the database client and the graph database clients are used for manipulating competences.

**ConfigController**

Handles the configuration and retrieval of the constants used in the matching algorithm. Uses the blob storage client in the data access layer to perform its actions.

**MatchingController**

Controller that handles the matching aspects of the system, i.e. matching a request of competences with the consultants and returning the most appropriate candidates. From the data access layer the database, graph database and blob storage clients are used to fetch consultants, relationships between competences and the configuration file for matching. Along with the requested competences all this data is passed to the matching engine that performs the matchmaking.
The matching engine identifies which consultants that have competences that matches the requested competences. The requested competences includes a value for how important a competence is (must be mastered by a consultant, should be mastered by the consultant) and which level a consultant with the competence should have. The algorithm is summarized below:

1. The requested competences are split into two lists based on importance.
2. The graph is used to calculate which competences that has the highest weight to the mustCompetences, resulting competences are stored in a list, pathCompetences.
3. Foreach consultant:
   a. Collect all competences the consultant has from the mustCompetences and shouldCompetences.
   b. If consultant does not have all mustCompetences:
      i. Go to 3.
   c. If consultant does not have all shouldCompetences:
      i. Use the pathCompetences list to find other competences that the consultant might have that is relating to the competences that the consultant is missing.
      ii. If consultant does not have relating competences to all missing competences:
         1. Go to 3.
      iii. Add the relating competences to the collection of matching competences.
   d. Use algorithm from section "Matching assignment specifications and consultants" to calculate score for the consultant based on the competences that are collected. The constants used in the algorithm is obtained from the blob storage client.
   e. Add consultant to list of matching consultants.
Presentation layer

The presentation layer is built as a single page application. The purpose of using this is to provide a more fluid user experience than desktop applications. This is done by retrieving the necessary resources needed by the application during a single page load. Based on user actions new resources can be dynamically loaded without the need of a new page load. To dynamically load these resources, AJAX is used. AJAX consists of a group of technologies that can make HTTP requests to a webserver and retrieve the response without reloading the page (Wikipedia, 2015i).

To streamline the development of the presentation layer AngularJS is used. AngularJS is a JavaScript framework that provides a MVC architecture and other components used in single page applications. One of the prominent features of AngularJS is the two way data binding between the model and view. Any changes in the view will be reflected in the model and any changes in the model will be reflected in the view (AngularJS, 2015). Figure 11 shows an overview for how the applications architecture is.

![Presentation layer diagram]

**Figure 11.** An overview of the presentation layer.

The user interacts using the view which consists of HTML files styled using CSS. Using AngularJS, objects in the model are bonded to components in the HTML files. Any changes in the view will be reflected in the model. If a user performs an action such as posting data, the event is handled in the controller. Since the view and model uses two way data binding, the controller can use any model without worrying about any changes in the view. Requests sent to the business layer is done using AJAX and the responses are JSON objects. JSON objects received from the business layer are sent to the model and thanks to the data binding between model and view, the view is also updated.

The presentation layer consist of four pages: Search, Consultants, Competences and Algorithm.

Search is the page where users can search for consultants by specifying competences with level and requirement. See Figure 12 for a screenshot of the search form.
Figure 12. Screenshot shows the search form for finding consultants.

Three competences are specified where two of them must be mastered by consultants to be considered matches. One competence should be mastered by consultants which entails that a relating competence can be used. Consultant cost specifies how expensive consultants can be without getting a penalty. The result of performing the search can be seen in Figure 13.

Figure 13. Screenshot showing the result from doing a search.
Consultants matching the requested competences are presented in the table to the right, “Matching consultants”. Each row represents a matching consultant. The “Score” column represents how suitable a consultant is based on the requested competence. A value around 100 is considered a good match. “Cost” column represents the cost of the consultant and the “Requested” column contains the competences that were requested. The value in parenthesis behind a competence represents what level it has. In the “Matching” column a symbol indicates how a requested competence is matching with the consultant’s competence. A green “Check” symbol indicates that the consultant masters the requested competences while a red “Cross” symbol indicates that the consultant does not have the required competence but a related one. On the “Check” symbols a “Plus” or “Minus” symbol can be attached that indicates whether the consultants level of the competences is higher or lower than the requested level. If a consultant has a higher level, the “Plus” symbol is used, if a consultant has a lower level, the “Minus” symbol is used.

A consultant with a relating competence will present a path to the requested competences. In Figure 13 the consultant Georg Sandholm has a relating competence to one of the requested competences. The requested competence is “python” which Georg does not master, instead he masters “c++”, which has a relationship to “c”, which has a relationship to “python”. The total path cost is 0.15 which implies that 15% of Georgs “c++” knowledge can be applied to “python”.

Below the search form is a graph showing the consultants score in relation to the cost limit for consultants. The score values are calculated for the same search parameters but with different “Consultant cost” values. The red vertical line indicates the cost that is currently used. The graph is helpful since it shows how the consultants’ suitability changes based on cost. Example: from the cost of 1100, Daniel will be the most suitable consultant.

From the “Consultant” page, consultants can be created, edited, removed and displayed. From the “Competence” page, competences can be created, edited, removed and displayed. This page can also be used to create, edit and remove relationships between competences.

The “Algorithm” page is used for editing the constants used in the matching algorithm. See Figure 14.
The following constants can be altered:

- “Max level competence”. Represents the highest possible level a competences can have. An assumption made is that the lowest level is 1.
- “Max coefficient level”. The maximum bonus a consultant can get for having a higher level than what is requested. This maximum value is obtained when a consultant masters a competence with the maximum possible level and the requested competence is on the lowest level.
- “Min coefficient level”. The maximum penalty a consultant can get for having a lower level than what is requested. This minimum value is obtained when a consultant masters a competence on the lowest level and the requested competence is on the maximum level.
- “Max coefficient cost”. The maximum bonus a consultant can get for being cheaper than the requested cost.
- “Ratio threshold”. The point from where a smaller ratio between requested cost and consultant cost will generate a 0 value, i.e. the point from where a consultant is considered too expensive to be considered a matching consultant.

The two graphs on the page describes the functions for generating level coefficients and cost coefficients based on the constants on the page.
Deployment Azure

The whole system is deployed in Azure, Microsoft's cloud platform. The reason for deploying in the cloud is to minimize the administration needed to run an internet based application and ensuring that it is available and accessible from anywhere. Choosing Azure as the cloud provider is based on the close integration with .NET that is offering good client libraries for communication with Azure services.
Evaluation

This chapter will focus on what the people at XLENT think of the system.

Method

To evaluate the system and determine if it simplifies the process of identifying the most suitable consultant, acceptance testing were performed. The tests were conducted by inserting some competences (programming languages) into the system and creating some realistic relationships between them. Some consultants at XLENT were also inserted with competences reflecting their real life competences. The evaluation were performed by some employees at XLENT who were asked to use the system.

Result

By observing the evaluators and letting them comment on how they experienced the system the following understandings were identified.

- Overall the system recommended good consultant candidates. The ranking among them were also good but did favor the cheaper consultants.
- Cost impacts the score too much. A consultant's cost is not static but can be adjusted to ensure that XLENT is getting an assignment.
- A consultant’s cost should reflect the minimum cost that XLENT can lend out the consultant, without losing money.
- Useful to get all the candidates next to each other since it makes it easier to compare them.
- The symbols representing how a consultant’s competences matches requested competences is useful and gives a quick overview on how well suited the consultants are. The additional symbols (plus and minus) representing the difference in level were confusing.
- The graph displaying the consultant's scores for different costs were confusing and the usefulness of it was questioned.
- Algorithm page was confusing.
Conclusions

This work has presented an approach for solving the problem of identifying the most suitable consultants based on an assignment specification. The prototype simplifies the process of identifying consultants and improves the possibility that the most suitable consultant is chosen for an assignment. This is done by providing a larger basis of consultants to consider when a consultant is to be assigned. Consultants are also ranked to indicate how suitable they are. XLENT is also getting a better overview of the knowledge that the company’s consultants masters.

To identify suitable consultants a matching algorithm has been developed which has been presented in the section Matching assignment specifications and consultants. Both the benefits from the logic deductive and similar based approaches are utilized in the algorithm. The logic deductive part of the algorithm identifies potential candidates and the similar based ranks the consultants. The rank of the consultants consists of a number of coefficients that are merged to a single one, reflecting the suitability of the consultant.

This works has also identified which properties a customer uses in an assignment specification for defining a desired consultant. Knowing this helps the people in charge of identifying the best suited consultants since it makes it easier to filter out the desired competences.

A conclusion drawn from the in-depth study is that competences can have attributes that can be applied to other competences. An approach for handling this has been proposed in the form of a weighted directed graph. The prototype has shown that the graph can be implemented using a graph database, Neo4j.

The whole system with its different parts are located in Azure. This makes the system available and accessible for everyone with an internet connection and a browser.

From the evaluation there are some conclusion that can be drawn. Identifying consultants from competences does work and gives good candidates. The ranking among them is also decent, although some adjustments is needed to reduce the benefit cheaper consultants get. Using symbols that represents how a consultant’s competence matches against a requested competence made it easier for users to get a quick view on how well suited a consultant is. Getting all candidates next to each other also helped the user comparing them and identifying who they think is the most suitable one.

An issue identified is that the cost coefficient is influencing the score too much. Consultants being more expensive than the cost in the request gets a significant penalty. To some degree this penalty can be adjusted by changing the constant used in the matching algorithm. Still, the penalty is too severe and a new cost algorithm is needed. Also, a consultant’s cost is not static but adjustable to fit a customer’s requirements. Using the system in its current state would mean that a consultant’s cost might be needed to be adjusted for each new assignment which is undesirable. Changing the cost to represent a minimum cost that XLENT can lend out the consultant, without losing money, would be a better approach and was also suggested by evaluators of the system.
The users did find the user interface confusing and not intuitive enough. Considering this thesis, this issue is not as serious as the cost issue. This is because the purpose of the thesis is to create a system that can identify suitable consultants to assignments. The cost issue is affecting the matching procedure directly since it is influencing the score. An unintuitive user interface does make it more difficult to see consultant suitability but does not affect the matching procedure directly. However, creating a production ready system would require that the user interface would be more intuitive.

Restrictions

One of the original problem statements was how to make the system identify which competences an assignment would require. This statement was excluded since it introduced a number of problems that would make the problem scope too big. For the system being able to identify competences, it needs to understand what type of system the customer is desiring which would likely need some type of AI. How the user is going to feed the system a specification is another problem that would arise. Most importantly thou, from the in-depth study is that the people assigning consultants to assignments did not consider the identification of competences a major issue. Therefore this problem statement was excluded.

Another idea to improve the matching of consultants was to introduce an AI aspect to the system. The idea was that users searching for consultants would be able to give feedback to the system based on which candidates the system produced and how they were ranked. If a user thought that a candidate should be ranked higher or lower, this would be noted by the system and incorporated in future similar searches. Considering the limited time doing the project this was skipped. A form of compromise was implemented in the system where the user can adjust the constants used in the matching algorithm.

Limitations

Some limitations to the system is the strong connection to Microsoft environments. Although .NET has recently become open source, the support in other environments is limited. The services also needs to be run in Azure, converting the project to be run on other cloud services could be challenging. Using Neo4j creates the same conundrum, a strong connection to a technology. Neo4j is supported in many environments and can be run on different platforms. But replacing Neo4j with another graph database is an obstacle since it is using the Cypher query language and a REST API service which other graph database might not.
Future work

In the section Assigning weights an approach for automatically creating relationships between competences and also weighting them was proposed. This is a feature that could facilitate the maintenance of the competences.

The matching algorithm could also be improved by weighting competences based on their importance. Competences that the user has specified as must should be more important to be fulfilled then a should competence.

Making the system production ready involves creating user roles in the system. Currently anyone can search, manipulate consultants and competences and adjust the constants used for the matching engine. Three proposed roles:

- User – Can access the search page and search for consultants.
- Consultant – Consultants can adjust which competences they master and at which level.
- Administrator – Administrators can create and remove consultants and competences. They can adjust the relationships between competences and the constants used in the matching engine.

Implementing these roles would ensure that only authorized persons would be able to do potentially harmful actions such as removing a consultant.

Another feature that could be useful is the ability to generate printable résumés. An observation when comparing the résumés at XLENT was that they lacked a common format. Since résumés are still commonly used when offering a consultant to a customer, it would be convenient being able to get a consultants competences and other traits generated to a document. This would also simplify the consultants need to update their résumés since they would only need to update it in the system. The format for the résumés would also be the same for all.

Since all competence XLENT masters is located in a system it would be possible to implement features that gives an overview of the competences that XLENT masters. This information could indicate what strengths the company has and which fields they can improve.

The current system only handles consultants. Making other types of person’s part of the searchable entities could make the system applicable on other businesses.
Acknowledgements

I like to thank my supervisor Peter Johansson at XLENT for supporting and advising me throughout the project and coming up with the idea of this thesis. I would also like to thank my technical supervisor Hanna Öhman at XLENT for advising me and answering questions I have had. Thanks to my supervisor at Umeå University, Mikael Rännar for feedback and support during the project.
References


Bianchini, D, De Antonellis, V, Melchiori, M, 2008, Flexible Semantic-BasedService Matchmaking and Discovery,

Cheetham, Grham, Chivers, Geoff, 2005, Professions, Competence and Informal Learning, http://books.google.se/books?id=xwyqLmG4cC&pg=PA162&dq=%27%27%27Dreyfus+Model+of+Skill+Acquisition%27%27%27 &hl=sv#v=onepage&q="Dreyfus%20Model%20of%20Skill%20Acquisition"&f=false, 2015-04-23


