Applying a Test for Atomicity of Method Fragments

Ben Rogers[1] and Brian Henderson-Sellers[1]

[1]*University of Technology, Sydney, Broadway, NSW 2007, Australia*
brian.henderson-sellers@uts.edu.au, benrogers@iinet.net.au

Abstract

One aspect of the conceptual modelling of processes is their quality. Here, we examine one aspect of quality – atomicity – as evaluated for a number of method fragments. High quality fragments will increase the quality of software development process models by application of the tenets of situational method engineering. Here, we identify a number of fragments from a previously developed methodbase/repository as being potentially non-atomic and suggest possible revisions to increase their quality.

Keywords: Metrics, Method Fragments, Situational Method Engineering, Atomicity

1 Introduction

For software development processes to be successful in their enactment, they need to be modelled conceptually, primarily in terms of a process model but also with respect to the underpinning metamodel. The elements in such a design-level process model need to be conformant to a metamodel and, at the same time, provide the templates needed for the creation and enactment of processes enacted on a specific project. Here, we examine these process models in the context of situational method engineering (SME).

Situational method engineering relies on small elements of methods being available from which a complete methodological approach can be constructed e.g. Henderson-Sellers and Ralyté (2010). Although there are many 'flavours' of method parts, we focus here on method fragments e.g. Brinkkemper (1996), which are generally described as being 'atomic' parts of a method.

One approach that is based on the use of method fragments is that of the OPEN Process Framework (OPF: Graham et al. 1997, Firesmith and Henderson-Sellers 2002), which, in turn, employs the Software Engineering Metamodel for Development Methodologies: SEMDM (ISO/IEC 2007). Although the original OPF publications presented method fragments that were atomic, subsequent modifications and revisions led to some of them growing to such a size that their atomicity could be challenged. Indeed, Henderson-Sellers and Gonzalez-Perez (2011) presented an analysis based on the granularity of method fragments (Hobbs 1985, Mani 1998), a theory based in turn on abstraction theory (Giunchiglia and Walsh, 1992, Kaschek 2004, Keet 2007), and concluded that at least one Task fragment (Construct the object model) could no longer be recognized as being of an atomic nature. However, these authors did *not* use any measure to determine objectively whether this particular fragment (and others) are or are not atomic in nature. The current study aims to define some metrics that can be applied to method fragments to assess their quality. We focus on only one aspect of quality: whether or not fragments are atomic. That is, can the fragment be broken down into smaller entities? In this project only the fragments constructed from the OPF/SEMDM methodology framework will be assessed to illustrate the appropriateness of the proposed metric whilst recognizing its more global applicability i.e. to fragments from sources other than the OPF. Once some metrics have been defined, they will be applied to existing fragments in the OPF repository. This is not only to measure the quality of the fragments but also to verify that the metrics do fulfil their purpose. One of the objectives of the project is to provide a means of evaluating the quality of new fragments, so that fragment authors have a tool to help them design fragments that are of good quality. This paper describes the metrics and also the principles behind them, so that fragment designers can understand what makes a fragment atomic.

In Section 2 we describe what is meant by the atomicity of method fragments. Section 3 discusses software engineering metrics and how we might calculate appropriate metrics for determining the atomicity of method fragments. In Section 4, we present the results of applying the atomicity calculations to a large number of fragments conformant to the OPF and/or SEMDM metamodels. Section 5 concludes, together with suggestions for future research on this topic.

2 Fragment atomicity

An atomic fragment is one that is not made up of other fragments, i.e. it has fine-grained granularity (Henderson-Sellers and Gonzalez-Perez 2011). A fragment that is not atomic is more likely to be usable in a limited set of situations, as it will not have the flexibility to be reused in a wide variety of circumstances. Furthermore, if fragments are not atomic, the functionality of one fragment could overlap that of another fragment. The benefits of maximising fragment reuse include the following. The more a fragment is reused, the greater the chance it will become expert in what it does. This is because it will be used in a wide variety of situations with different demands placed on the fragment. As the fragment is widely reused, all the uses of it will benefit from the increased expertise. Furthermore, the reasons for having atomic fragments are similar to why a software class should have high cohesion. A class with high

Copyright © 2014, Australian Computer Society, Inc. This paper appeared at the 10th Asia-Pacific Conference on Conceptual Modelling (APCCM 2014), Auckland, New Zealand, 20-23 January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 154. G. Grossmann and M. Saeki, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

cohesion enjoys the property of togetherness, while one with low cohesion could be broken down into two or more classes (Henderson-Sellers 1996). The elements of a class with high cohesion work together in a consistent purpose (Booch 1993). For example, a highly cohesive Car class contains the behaviour for a car and nothing but a car. The class with high cohesion is more likely to be able to be reused because of its targeted purpose. Also a class with high cohesion can be expert in its functionality because of its tight focus. The same can be said about atomic fragments in SME. They are more reusable and they can be refined to be expert in their purpose.

3 Metrics for fragment atomicity

A metric "is the mapping of a particular characteristic of a measured entity to a numerical value" (Lanza et al. 2006, p.11). An example of an entity could be a person and the characteristic could be the height of the person. Although it is possible to measure a wide range of attributes such as height, weight, age, sex of a person, the measurements only have meaning if there is a clear purpose to the exercise. For example, the purpose may be to determine if the person would be a good athelete.

Basili and Rombach (1988) proposed the Goal/ Question/Metric paradigm to refocus on the measurement goal rather than simply a procedure to count something with no goal in mind. The first step is to define the goals that the use of the metrics will achieve. From the goals, a list of questions that need to be answered for the goals to be met are compiled. For each question, the metrics are chosen that will answer these questions. Two well-known metrics are coupling and cohesion (Stevens et al., 1974).

An atomic fragment is like a class with high cohesion and low coupling (as noted above). A class with low cohesion implies that it should be divided into multiple classes, and, conversely, one with high cohesion should not be divided (Henderson-Sellers 1996). Thus, an atomic fragment can be identified because of its high cohesion value. Secondly, a class with low coupling makes it easier to reuse and is also indicative of atomicity or near-atomicity. Coupling could mean object-to-object coupling only or could include coupling due to inheritance. For atomic fragments, the interest is restricted to object-to-object coupling, which is simply called coupling hereafter. That is, two classes are coupled if the methods in one class make use of methods or instance variables in the other class.

A class has high cohesion if its methods work towards one purpose that is easy to identify (Henderson-Sellers 1996). Cohesion in a class is often measured in terms of its complement: the 'lack of cohesion' e.g. Chidamber and Kemerer's (1991) LCOM. This metric looks at each method's use of the class instance variables. The use of instance variables by methods determines the intersections of methods. If two methods access completely different instance variables then probably they should belong to different classes. However, there has been an extensive debate on the precise definition of LCOM and the mathematics to calculate it (Henderson-Sellers et al. 1996). The problem with various formulae for the Lack of Cohesion metric is that it may not be possible to discriminate between dissimilar entities. That is, two classes with obvious differences in cohesion may be given the same score by the LCOM metric. Also, although a high LCOM score would indicate low cohesion, a score of zero does not necessary indicate high cohesion (Henderson-Sellers et al. 1996). Despite the problems with LCOM, the study of it does help in understanding the nature of class cohesion.

Another metric to measure cohesion is Tight Class Cohesion (TCC): "the relative number of method pairs of a class that access at least one common attribute of that class" (Lanza et al. 2006, p.17). Its value is between zero and one, where a low value indicates low cohesion and a high value indicates high cohesion.

There are also several existing metrics for coupling. The coupling between objects metric CBO (Chidamber and Kemerer 1994) applies to a given class. It counts the number of classes that are coupled to the given class. Fan-out for a given class measures the count of classes that the given class makes use of (Henry and Kafura 1981). Fan-in for a given class measures the count of classes that makes use of the given class. A high fan-in value is desirable because it indicates that the given class is being reused extensively, while a low fan-out value is desirable because that shows that the given class does not need too many other classes to operate.

3.1 Metrics for fragment relationships

There are, at least, three methodological approaches that have a common semantic core: the OPEN Process Framework (OPF) (Graham et al. 1997, Firesmith and Henderson-Sellers 2002), SEMDM (ISO/IEC 2007) and the Software Process Engineering metamodel (SPEM) (OMG 2008). All three identify three critical areas: work units, work products and producers – although here we focus on examples of fragments conformant primarily to SEMDM.

Henderson-Sellers and Gonzalez-Perez (2011, p56) have argued that the "Construct the object model" task fragment described in the OPEN Process Specification is not atomic because more than one technique had to be chosen out of thirty seven techniques. Given this precedent, of counting the number of techniques for a task fragment to determine whether a fragment is atomic or not, we postulate that other relationships could also be used. Genero et al. (2005) used metrics to measure the complexity of a Software Process Model conformant to SPEM. For activities there are work product and role counts, while for work products there are activity counts where the work product is consumed or produced, and for roles there are activity counts for which they are responsible. Genero et al. (2005) carried out some experiments to check the validity of the metrics thus defined. In the first experiment, ratings made by students and professors on a set of software process models were compared with the results of the measurements performed on the software process models. The null hypothesis was that there was no significant correlation between the structural complexity metrics and the students' and professors' ratings. The ratings included understandability, analysability and modifiability of the software process models. For some of the metrics that

applied to the software process model, the null hypothesis was rejected (Genero et al. 2005).

It can be the case that a number of measurements are needed for answering only one question (Lanza et al. 2006). That is the case for this research. For example, to answer the question of whether a given Task fragment is atomic there are the Technique Count, Work Product Create Count, Work Product Update Count, Work Product Read Count, Role Count and Team Count metrics. Looking at relationships between fragments not only gives a view on coupling, it also indicates cohesion – does the fragment have more than one purpose?

Just as different fragment types have different relationships with other fragments, different metrics are applicable. In the SEMDM metamodel, producer fragments have relationships with work unit and work product fragments; so, for producer fragments, metrics that count related work units and work products would be used. In contrast, work unit fragments have relationships with producer and work product fragments; so, for work unit fragments, metrics that count related producer and work products would be used. To facilitate identification of pairings, a matrix was constructed, with the fragment types listed vertically and the metric types listed horizontally.

All the metrics used here are counts of distinct related entities. For example, the role count is the number of distinct producer roles related to the fragment type. The relationship between work unit fragments and work product fragments is defined by the use of Actions, which define whether work products are created, updated or read only. The proposed metrics are divided by these action types. For instance, there is not just a Work Product Count, there is also a Create Work Product Count, Update Work Product Count and Read Work Product Count.

On producer fragment counts, only roles and teams are included. This is because, at the repository level, persons would not be defined. A role is only included in a role count if it participates directly in a Work Performance; as opposed to roles that are in teams and the teams that are in the Work Performance.

As a metric for atomicity, we propose here counts of relationships between each fragment and roles, teams etc..

3.2 Calculating the atomicity metric

It is important that measurements for metrics are easy to calculate. If possible, it would be good for these to be calculated automatically. To achieve this, a database has been created to capture the repository of fragments and their relationships. Then, SQL queries calculate the measurements of metrics for all of the fragments. The database schema and measurement SQL queries have been constructed to cover all the elements in the SEMDM metamodel.

3.3 Setting thresholds

Metrics themselves give no information on model quality. They can be used indicatively if appropriate thresholds are included (e.g. Szentes and Gras 1986, Kitchenham and Linkman 1990). Thresholds can be used to specify regions, so that conclusions can be made from the data with respect to an appropriate threshold regarding atomicity/non-atomicity. Although these can only be statistical conclusions, the usefulness is increased if the underlying statistical distribution is known or can be assumed (Haynes and Henderson-Sellers 1997). Setting two thresholds is useful: the first to indicate a "watch" if exceeded and the second (higher) one to flag as statistically likely to be an outlier. Explicable thresholds are set based on good arguments and are rarely perfect (Lanza et al. 2006). Some thresholds can be determined by generally accepted knowledge. For example, people expect to eat three meals a day. Another way to set thresholds is through statistical measurements. For example, is ten thousand hairs on a head a lot of hair or is the person balding? The number of hairs could be counted on a large population of people and an average calculated. If the average is between eighty thousand and one hundred and twenty thousand then we can conclude the person is balding (Lanza et al. 2006).

For the case of a normal distribution, the average and standard deviation are useful values. However, it is expected that the distributions for the proposed metrics will not only be discrete (i.e. the normality of the distribution can only be approximate) but are also likely to be skewed to the right since there will be no values less than zero and most of the values will be in a low range with only a few high values. This means that the numerical summary needs to be resistant (Sullivan 2011). That is, the value of the numerical summary does not change much if an extreme value is added. The median is resistant while the mean is not (Sullivan 2011). For example, say there are five observations: 179, 201, 206, 208 and 217. The median is 206 and the mean is 202.2. Now if an observation of 1000 is added the mean increases substantially to 335.1666 while the median is 207. Since the underlying data here are discrete, the median is the more useful.

The median is the fiftieth percentile. The percentile gives the position of an observation within a set of data. Quartiles are special cases of percentiles. They break the set of data into four pieces. Like the median, quartiles and percentiles are resistant numerical summaries. The interquartile range, IQR, measures the range of values between the first quartile and the third quartile; that is, the range of the middle fifty percent of values. Outliers are unusually very low or very high values. To identify outliers, fences are calculated. Fences are thresholds used as boundaries so that outliers can be found. If a value is lower than the lower fence or higher than the upper fence then it is deemed to be an outlier (Sullivan 2011). The formulae for the fences are

Lower fence = Q1 – 1.5 x IQR
Upper fence = Q3 + 1.5 x IQR
(Sullivan 2011, p.160)

In this research, since the distribution of the measurements will be right skewed with no negative values, the outliers will only be unusually high values. It is important to determine whether a measurement is an outlier or not, because measurements that are outliers will indicate that the relevant fragment is probably not atomic. That is, the upper fence of each metric's distribution will be used as the threshold for fragment atomicity.

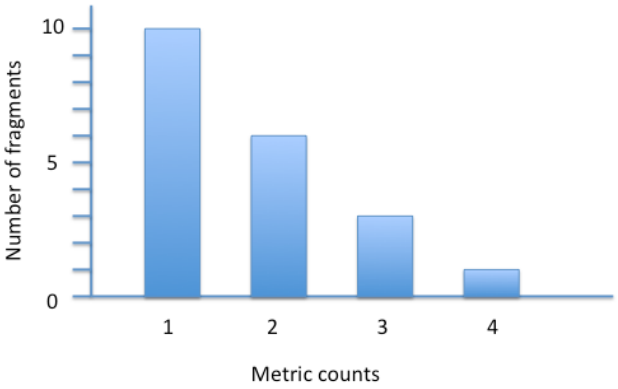

Figure 1: Sample fragment distribution using 20 fragments

Figure 1 provides an example of an artificially constructed distribution of fragments over some metric counts. Each fragment has an integer metric count value. This is the case for all metrics proposed in this paper. In this example, there are twenty fragments that have measurements between one and four. The distribution is right skewed. The median is given by the value at the (n+1)/2 position when n is odd or by the average of the values at the n/2 and the (n+1)/2 positions when n is even. Thus, for n=20, the median is the average of the 10th and 11th values = (1+2)/2 = 1.5.

Q1 (25th percentile) is the average of the 5th and 6th values = (1+1)/2 = 1

Q3 (75th percentile) is the average of the 15th and 16th values = (2+2)/2 = 2

Hence the IQR = Q3 – Q1 = 2-1 = 1
Then the upper fence = Q3+1.5*IQR = 1+1.5x1 = 3.5
Outliers, in this case, are thus metric counts of four or above.

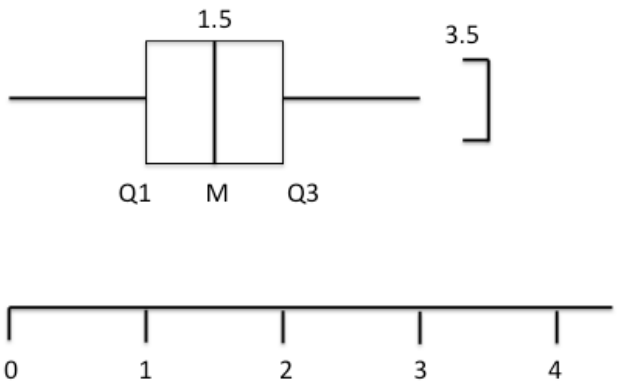

Figure 2: Box plot for sample fragment distribution of Figure 1.

Figure 2 shows the box plot diagram for the Sample Fragment Distribution of Figure 1. The first quartile, median and third quartile are shown on the box. The square bracket indicates the upper fence, and the horizontal line either side of the box indicates values that are not outliers.

4 Results and discussion

For the first assessment, fragments were sourced from Tran et al. (2009a) for Producer and Work Product and from Tran et al. (2009b) for Work Unit fragments, supplemented by fragments from the sample repository provided with the software tool MethodMate, supplied by Cesar Gonzalez-Perez.

Table 1 lists Tasks wherein the Technique Count for the Task: Software Coding in Agile is an outlier (a value of 4; upper fence value of 3.5). What is interesting about Agile software coding is that it involves more than traditional coding; for instance, Pair Programming involves two developers working together on the same task at the same computer (Beck 2000).

The Create Work Product Count for the Initiation Process is also an outlier (Table 2). It is reasonable that the Initiation process creates a number of artefacts, because of the planning involved in the process.

We also examined counts for Stage Types, Work Product Types and Producer Types but were unable to identify any other potentially non-atomic types.

The metrics do not *prove* atomicity, they only flag the possibility that the fragments may not be atomic. For the candidates identified above (Tables 1 and 2), one must determine what action to take to ensure atomicity of these tasks. For example, for the Task: Software Coding in Agile, one solution may be to split the task into three: 1) Coding plus Pair Programming and Collective Ownership (as two techniques); 2) Integration Coding as a task together with an associated technique.

What is encouraging about the metrics is that the results in this paper have identified fragments that may not be atomic. However, the metrics would be more powerful if there was a larger repository of fragments in the database. Furthermore, the fragments in the database need to have a rich assignment of relationship fragments such as WorkPerfomance, Action and TaskTechniqueMapping. Also it would be useful for the fragments to make use of the aggregation relationships specified in the metamodel. The more information there is, the greater the chance of resultant outliers correctly identifying fragments that are not atomic.

It proved not to be too onerous to load fragments into the database and then run SQL queries to calculate the measurements. That is, the metrics were easy to measure once the fragments had been loaded into the database. Furthermore, there is scope to introduce new metrics, simply by crafting new SQL queries.

5. Conclusions and future work

5.1 Conclusions

In the context of situational method engineering and process models, we have examined the atomicity of a large number of method fragments that are conformant to the SEMDM metamodel. Whilst most are found to be atomic, our metric counts and thresholds employed suggest that a small number of these fragments require detailed scrutiny since there is a high likelihood that they are not truly atomic in nature. Consideration is therefore to be given to either splitting tasks into more than one

task or converting multiple techniques to subtasks plus associated techniques such that tasks and subtasks are all atomic in nature.

5.2 Future Work

A useful contribution to this research would be to revise the existing fragments by associating them with relationship fragments. Increasing the sample size in the database will improve the reliability of the results (Sullivan 2011). That is the determination of the upper fence will be more accurate.

Also useful would be to undertake further research on the atomicity of fragments that have a whole-part relationship with other fragments. This impacts the construction of the SQL queries that perform the measurement calculations.

A useful enhancement to systems like MethodMate, which builds and uses fragment repositories, would be to store fragment information in a relational database. This is so that the system can automate the measurement of the metrics proposed in this paper.

References

Basili, V. R. & Rombach, H. D. (1988), 'The TAME project: towards improvement-oriented software environments', *IEEE Transactions on Software Engineering* **14(6),** 758-773.

Beck, K. (2000), *Extreme programming explained: embrace change,* Addison-Wesley.

Booch, G. (1993), *Object-oriented analysis and design with applications,* 2nd edn., Addison-Wesley.

Brinkkemper, S. (1996), 'Method engineering: engineering of information systems development methods and tools', *Inf. Software Technol.* **38(4),** 275–280.

Chidamber, S. R. & Kemerer, C. F. (1991), Towards a metrics suite for object oriented design, *in* 'OOPSLA '91 Conference proceedings on Object-oriented programming systems, languages, and applications', Vol. 26(11), pp. 197-211.

Chidamber, S. R. & Kemerer, C. F. (1994), 'A metrics suite for object oriented design', *IEEE Transactions on Software Engineering* **20(6),** 476-493.

Firesmith, D. G. & Henderson-Sellers, B. (2002), *The OPEN Process Framework. An introduction,* Addison-Wesley.

Genero, M., Piatinni, M., Calero, C. & Ebrary, I. (2005), *Metrics for software conceptual models*, Imperial College Press.

Giunchiglia, F. & Walsh, T. (1992), 'A theory of abstraction', *Artificial Intelligence* **57(2-3),** 323–390.

Graham, I., Henderson-Sellers, B. & Younessi, H. (1997), *The OPEN process specification,* Addison-Wesley.

Haynes, P. & Henderson-Sellers, B. (1997), 'Bringing OO projects under quantitative control: an output-, cash- and time-driven approach', *American Programmer* **10(11),** 23-31.

Henderson-Sellers, B. (1996), *Object-oriented metrics: measures of complexity,* Prentice Hall.

Henderson-Sellers, B. & Gonzalez-Perez, C. (2011), Towards the use of granularity theory for determining the size of atomic method fragments for use in situational method engineering, *in* J. Ralyté, I. Mirbel & R. Deneckère. eds., 'Engineering Methods in the Service-Oriented Context. 4th IFIP WG8.1 Working Conference on Method Engineering, ME 2011, Paris France, April 2011, Proceedings', Springer, Heidelberg, pp. 49-63.

Henderson-Sellers, B. & Ralyte, J. (2010), 'Situational method engineering: state-of-the-art review', *Journal of Universal Computer Science* **16(3)**, 424-478.

Henderson-Sellers, B., Constantine, L. L. & Graham, I. M. (1996), 'Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design)', *Object Oriented Systems* **3,** 143-158.

Henry, S. & Kafura, D. (1981), 'Software structure metrics based on information flow', *IEEE Trans. Software Eng.* **7(5),** 510-518.

Hobbs, J. (1985), Granularity, *in* 'Procs. Int. Joint Conf. on Artificial Intelligence, IJCAI 1985'

ISO/IEC (2007), Software Engineering – Metamodel for Software Development. ISO/IEC 24744, Geneva, Switzerland

Kaschek, R. (2004), A little theory of abstraction, *in* B. Rumpe & W. Hesse, eds., 'Modellierung 2004. Proceedings zur Tagung in Marburg', LNI, vol. 45, Gesellschaft für Informatik, Bonn, pp. 75–92.

Keet, M. (2007), Enhancing comprehension of ontologies and conceptual models through abstractions, *in* R. Basili & M. T. Pazienza, eds., 'AI*IA 2007',. LNCS (LNAI), vol. 4733, Springer, pp. 813–821.

Kitchenham, B. A. & Linkman, S .J. (1990), 'Design metrics in practice', *Inf. Software Technol*. **32(4),** 304-310.

Lanza, M., Marinescu, R. & Ducasse, S. (2006), *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-orientated systems,* Springer.

Mani, I. (1998), A theory of granularity and its application to problems of polysemy and underspecification of meaning, *in* A. G. Cohn, L. K. Schubert & S. C. Shapiro, eds., 'Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR 1998)', Morgan Kaufmann, pp. 245-257.

OMG (2008), Software Process Engineering Metamodel Specification; adopted specification, version 2.0. Object Management Group. April. Available in http://www.omg.org/spec/SPEM/2.0/

Stevens, W.P., Myers, G.J. and Constantine, L.L., 1974, Structured design, *IBM Syst. J.,* **13(2),** 115-139

Sullivan, M. (2011), *Fundamentals of statistics,* 3rd edn. Prentice Hall.

Szentes, J. & Gras, J. (1986), Some practical views of software complexity metrics and a universal measurement tool, *in* 'First Australian Software Engineering Conference (ASWEC), Canberra, May 14-16'

Tran, Q.-N. N., Henderson-Sellers, B. & Hawryszkiewycz, I. (2009a), Some method fragments for agile software development, *in* 'Handbook of Research on Modern Systems Analysis and Design Technologies and Applications', IGI Global, pp. 223-242.

Tran, Q.-N. N., Henderson-Sellers, B. & Hawryszkiewycz, I. (2009b), Agile method fragments and construction validation, *in* 'Handbook of Research on Modern Systems Analysis and Design Technologies and Applications', IGI Global, pp. 243-270.

Table 1: Counts for a number of SEMDM-conformant Tasks

Task	Source	Technique Count	Create Work Product Count	Modify Work Product Count	Role Count
Analyse requirements	MethodMate			1	2
Coding	MethodMate		1		1
Develop class models	MethodMate		1		2
Document requirements	MethodMate			1	1
Elicit requirements	MethodMate		1		2
Perform peer review	MethodMate			3	2
Unit-test code	MethodMate			1	1
Write user stories	Agile Method	1	1		1
Explore architectural possibilities	Agile Method				
Analyze technologies	Agile Method				
Describe application	Agile Method	1			
Prototype the architecture	Agile Method				
Develop release plan	Agile Method	1	1		1
Monitor Work Products	Agile Method				
Develop iteration plan	Agile Method	2	2		
Software Coding in Agile	Agile Method	4	1		
Design agile code	Agile Method	1			1
Refactor	Agile Method				1
Testing tasks	Agile Method				3
Integrate software	Agile Method				1
Write Manuals	Agile Method				
Manage Shared Artefacts	Agile Method				
Identify shared artefacts	Agile Method				
Allocate shared artefacts	Agile Method				
Specify Permissions to Shared Artefacts	Agile Method		1		
Mediate/monitor the performance of team's tasks	Agile Method				
Meditate/monitor team's interactions	Agile Method				
Conflict management	Agile Method				
Monitor members' performance	Agile Method				
Member motivation	Agile Method				
Ensure workload balance	Agile Method				
Specify team policies	Agile Method				
Specify team structure	Agile Method				1
Upper fence value		**3.5**	**2.5**	**3.5**	**3.5**

Table 2: Counts for a number of SEMDM-conformant Processes

Process	Source	Create Work Product Count	Modify Work Product Count
High-Level Modelling	MethodMate	1	
Implementation	MethodMate	1	1
Low-Level Modelling	MethodMate	1	
Quality Assurance	MethodMate		3
Requirements Engineering	MethodMate	1	1
Initiation	Agile Method	4	
Construction	Agile Method	1	
Delivery	Agile Method	1	
Usage	Agile Method		
Team Management	Agile Method		
Upper fence value		**2.5**	**4**