When Data Science Becomes Software Engineering

Lito Perez Cruz
School of Business, Monash University, 7/271 Collins St., Melbourne Vic 3000, Australia

Keywords: Data Science, Data Analytics, Software Engineering, Education.

Abstract: Data science is strongly related to knowledge discovery. It can be said that the output of the data science work is input to the knowledge discovery process. With data science evolving as a discipline of its own, it is estimated that the U.S.A alone, needs more than 1M professionals skilled in the discipline by next year. If we include the needs of the rest of the world, then internationally, it needs more than that. Consequently, private and public educational institutions are hurriedly offering data science courses to candidates. The general emphasis of these courses understandably, is in the use of data mining and machine learning tools and methods. In this paper, we will argue that the subject of software engineering should also be taught to these candidates formally, and not haphazardly, as if it is something the would-be data scientist can pick up along the way. In this paper, we will examine the data science work process and the present state of skills training provided by data science educators. We will present warrants and arguments that software engineering as a discipline can not be taken for granted in the training of a data scientist.

1 INTRODUCTION

By now it is probably safe to say that most in the field of IT have heard of the McKinsey & Co report (Strawn, 2016) that says the USA needs 1.5M data savvy managers. If this is the number of managers in the USA who must be have insights and knowledge from data, we can only imagine the huge number needed by the world. It is predicted that in 2018, the demand for data scientists will out run the supply by 60% according to the same report Co (Strawn, 2016). If that is true, there is very little time left to supply that great demand. Consequently, even a few years before this publication, educational institutions, and even private training companies have been offering data science courses. Many are in a hurry to produce data scientists that will meet that great need. The depth of offering varies, from short courses up to masters degree level. However, in these courses, the emphasis is in the use of data mining and statistical software which is understandable. Unfortunately, the pragmatics necessitate that the style of teaching is in the cook book method approach and in such a hurried pace, there is no time to reflect if the data scientist is being equipped sufficiently for the road ahead. No time is devoted to teaching the prospective data scientist the art of properly crafting software. Meaning, the engineering of software is taken for granted in many of these courses, which we believe is short-sighted. In this position paper, we will offer evidences, critiques, and suggestions for the improvement on the present emphasis given by course providers on the use of statistical and data mining software tools. Though there is programming in the courses, software engineering principles and practices are hardly touched on. In this work, we will argue that software engineering should be a mandatory subject or unit that must be taken in some form in any data science training program. In the process, we will imply the benefits to the community when its data scientists are knowledgeable in software engineering theories, practices, and management[1].

2 THE DATA SCIENTIST'S WORK

In order to know the scope of work performed by data scientists, we need to know first the nature of data science itself. The term originated in 2008 and so it is not even a decade long, a very young discipline indeed (Cage, 2017). In the article of (Cage, 2017), we do not find the definition of what is data science but it describes what data scientists do. (Rose, 2016) for instance, agrees that it is hard to define because right

[1]For word economy in this work, we will use DS to denote data science/data scientist. We will use SE to denote software engineering.

Cruz L.
When Data Science Becomes Software Engineering.
DOI: 10.5220/0006508502260232
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD 2017)*, pages 226-232
ISBN: 978-989-758-272-1

now it exists as a practice rather than a discipline and as such a multi-disciplinary enterprise as well. Being mathematical and statistical is only part of the job but more importantly, it involves insight and the ability to see behind the data such as spotting what conclusions can be drawn from a dataset. Then there is also the ability to explain all of these findings to stakeholders. Clearly, there is a mixture of specialist skills at play but one thing is for sure, the data scientist is a professional who acts as a consultant to the people he/she serves and normally they are decision makers high up in the business organization.

One idea that will help us understand deeper the duties and responsibilities of a data scientist is to examine a typical workflow involved in this work. Various practitioners may interpret or classify their processes uniquely. Below are three examples of workflows or lifecycle phases that are usually taken by typical data scientists.

3 THE DATA SCIENCE WORKFLOW

We will now offer an assessment of the present data science practice through its self-understanding of it's workflow. This will make clear that software engineering, and not just coding, is important in the job of a data scientist. By SE, we mean the systematic procedure for the analysis, design, development/implementation, testing, and maintenance of software(Leach, 2016). However, before we jump in, it is best to pause and mention the two crucial contributions of software engineering in the process of developing software. We suggest that we bear in mind the following concepts below as we go along in our discussion. Data science involves itself, perhaps even reluctantly, in the discipline of SE because the models discovered are embodied in a set of code. On the other hand software engineering ensures the product is of high:

- Quality - code should work flawlessly as expected
- Maintainability - code can be modified seamlessly

Each of the above treat code as an organization's asset that needs to be maintained and managed because it provides its owners intangible benefits in the long run. Those two basic tenets are meant to protect the client's investment and to gain as much return in utility from its software product. We will comeback to other advantageous concepts provided by software engineering that affects data science, but for now, we shall illustrate how it emerges in the data scientist's practice.

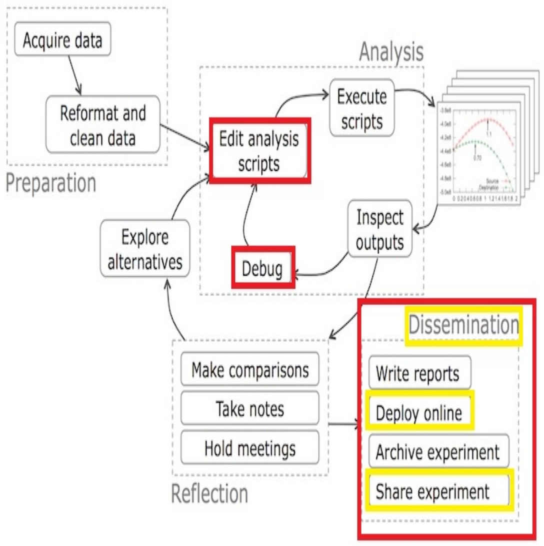

Figure 1: Typical Waterfall Process.

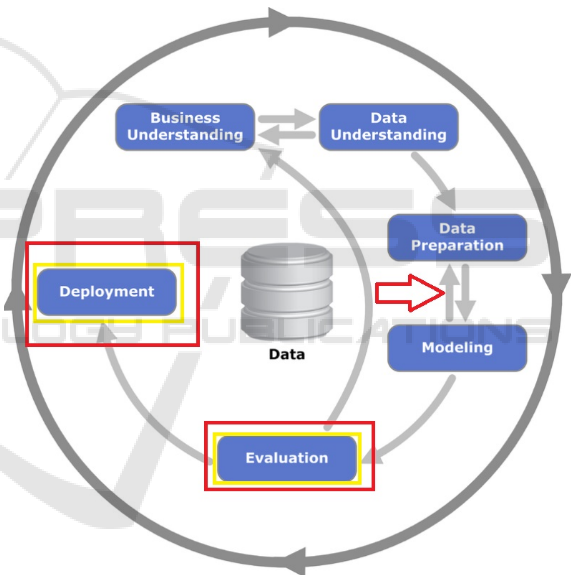

Figure 2: CRISP-DM Process.

In Figures 1 (Guo, 2013), 2 (Rose, 2016), 3 (Rose, 2016) we depict the typical lifecycle understanding that may be adopted by a DS practitioners. We have highlighted in red the part of the process relevant to our point. Figure 1 may be called the typical "waterfall" process where the output of a process becomes input to the next. Figure 2, is the famous CRISP-DM process for data mining and lastly Figure 3, is an agile method of incremental moving backwards and forwards between processes. This method is explained in (Rose, 2016) but the details of this will not invalidate the arguments we will make here, the point are the highlighted processes that are relevant to our SE argument.

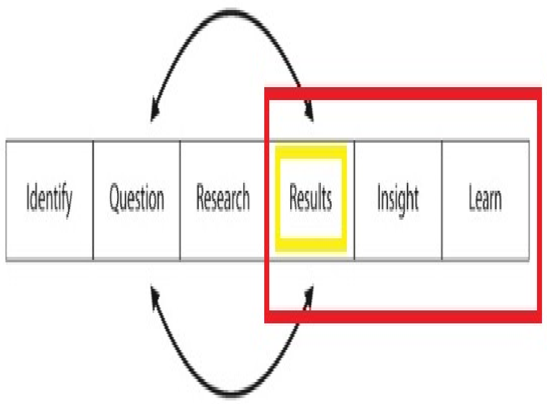

Figure 3: An Agile Process.

In Figure 1, the end of the data science work, the ”product” that results in the workflow is disseminated or shared. The result of the data science work is typically embodied in a program code, that is why in the process there is the task of ”dissemination”(Guo, 2013). This is what is meant by ”deployment” in Figure 2. In Figure 3, an agile process method, the ”deployment” starts off from the Results $\longrightarrow$ Insight $\longrightarrow$Learn phase.

For once off adhoc decision making single instance of ”deployment may do, but experience says that vital solutions sought by businesses require the repeated execution of the ”product”. Note that in all these workflows, the cyclical nature of some parts of the DS process. Observe the arrows iterating from one process to the other. Let us give a concrete example from the author’s experience with one of his clients. Assume we have a transport bus company that has a fleet of buses used for moving passengers. Assume further that the fleet is composed of a couple of hundred buses operating 24×7 and these vehicles are getting on in age. The data scientist may be tasked of producing a system that reports to management which vehicles are worthy of being repaired or which buses need to be retired/replaced by brand new ones because it is more economical to do so. Such a system will not only run once but will have to run at a minimum, once a month. Such running obviously involves mathematical and statistical analysis for sure. In this regard, there is an important reality embedded in those highlighted processes where SE is lurking at the core of those steps.

All have code iteration editing. If we notice, all of the lifecycle models have an iteration process where code is edited and refined. This is true for all of them (see steps highlighted in red). There is continued testing and refinement going on in all of the processes. However, in such refinement work, this is where knowledge of SE principles will help. If the code is written right with SE best practices in mind, it will help the DS trap defects easily, extensively and efficiently. It will also make the code easier to modify and so, easier to edit.

A model has a life span and is valid only for a certain time At a certain point in time, the model has to be updated because things change in time and the parameters of the model may have to be checked or revalidated. Thus, the model may simply need to be updated and the code made current as well. It may not be necessary to altogether discard the code, indeed it may be economical to retain parts of the it for after all, the client paid for it by way of time and money already. So for this reason, since it is sometimes prudent to maintain the code, a knowledge of SE will help the DS produce code geared for maintainability and perhaps even portability etc. Such knowledge can only help rather than harm. The fact that code is deployed does not mean the code ends there.

Some have the idea that once the DS gets into the deployment stage, he/she then passes the result to an SE who will then integrate it into the production system. In an organization which is large and employs an army of SEs and DS personnel, this might fly. Though indeed, this setup might work, in general this is not suitable or not an ideal situation for the client. We can surely surmise that whether a client is from a small or large organization, the executive using the DS will prefer to have an integrated final solution from the data scientist.

Some businesses want a one stop shop of services. We mean that often, clients do not wish to deal with several skill providers. Dealing with various mix of people to get a solution implemented is just too much of an administrative burden for small or medium business operators. Such a scheme will only dissuade executives in embracing the benefits of data science. It will not work for them because they do NOT enjoy big project funding, unlike bigger organizations.

We mentioned in the previous section that the DS is a consultant. As such, small to medium organizations most likely will hire a DS for a project with the intention of obtaining a finished and complete decision system. It will be impractical, if not cost inefficient, if the client obtains a half-finished product which still has to be turned into industrial grade/quality code going forward. For this reason, the DS will give added value to the client if he/she is knowledgeable of SE principles, i.e., he/she can at least hand over the code engineered in such a way that it follows sound SE practices.

4 THE SOFTWARE ENGINEERING TERRITORY

As we have seen in the sample of workflows that we examined in the data science discipline, it is inevitable that the work of a data scientist gets wind up into a set of code. Thus, into a product that will produce reports that business management/executives will use for decision making. In this regard, once something is put into code, that program becomes a property or an object that has value to its owners, i.e. the clients who commissioned the project, a ”property” if you like. Because the product is in the form of a code, in that respect, the software engineering discipline has a lot to contribute to the code’s lasting benefits.

4.1 Goals

Perhaps the quickest way to convince data scientists and their educators the help provided by software engineering is to list its goals. This list is taken from Software Engineering Book of Knowledge (SWEBOK) (Leach, 2016). Again, we are dealing with this because the data scientists effort is enclosed in code and that is where software engineering becomes a necessity:

1. Correctness
2. Reliability
3. Modifiability
4. Testability
5. Reusability
6. Maintainability
7. Efficiency
8. Usability
9. Portability
10. Interoperability

It is hard to believe that the above goals do not affect the code produced by data scientists. Portability and interoperability are none issues if we speak about R or Python as the languages of choice. These all run in all present existing operating system platforms. However, points 1-7 are extremely relevant to the work produced by data scientists.

For example, the subject of correctness is crucial in the DSs code. The old principle of garbage-in and garbage-out (GIGO) is true in DS as it is true in computer science. Reusability, modifiability and maintainability are related goals and imply each other. Why would we not code in such a way that our functions are self-contained and can be called, or be used in the new programs we might code? Surely this is less work for the DS and an increase in productivity. Or, would it not be a great turn of events if by just a slight modification of an old code, the DS arrives at a useful version of the program applicable to current problem case? All of these imply that the code has to be written in such a way that it is easily maintainable.

4.2 Testing

The DS employs data mining or machine learning algorithms used for modeling. These are embodied in tools but crafted in code. Because of this the DS invariably will go through a cycle of coding and testing until he/she settles on the version that is satisfactory for the problem. For this reason we give special mention to testing here because it is vital in DS work. It pertains to job quality. Testing then touches upon the quality of the model and so the quality of code produced by the DS professional. If we go back to the workflow, both of them mentioned ’debugging’ and ’validation’ which pertains to the testing of software and the model together. In DS, not only is the code tested but effectively the model is tested as well because the code is the ”physical” object that reflects the model. The question is two fold. First is the model accurate enough to be useful? Then is the code written correctly to the specification of the model? In SE, testing covers a huge aspect of the discipline and there have been extensive methods and techniques developed by SE practitioners to help pinpoint for the DS where the code could be made robust or secure from obvious errors.

For example in big data processing, the calculation could take some hours to complete. What will happen if the DS’s code encounters an unexpected data that makes it abnormally terminate? This spells wastage of resources and can be frustrating to stakeholders as well as to the DS him/herself. SE practitioners have gathered recommendations that can anticipate such incidences, by developing guidelines for the testing regime. For example, SE has a method for directing code testing based on data partitioning, boundary value testing, path testing, data flow testing, etc. Another one, which is becoming popular is the so-called ”test driven development”. Here the coder builds up the code in small portions and tests these portions as the code is grown. The programmer then does not insert additional new code unless the existing once are clean. As the coder extends the code, so is the testing as well (Jorgensen, 2014).

4.3 Packages

We cannot leave this discussion without saying something about the importance of goals 1-7 in the business of writing *packages* or *libraries*. Whether one is using R, Python, SAS or SPSS in the DS enterprise, we can expect that these languages or platforms support the creation and activation of common code routines in the form of functions housed in packages or libraries. Functions of course are reusable. An outside program can call functions external to itself. This aspect in the language use, by default, necessitates the need to use SE design principles and goals. Hence, another encroaching of the SE discipline into the DS enterprise. When a DS deploys code in a form of a package, especially in open source environments, the code achieves lasting value and mature usage as far as it adheres to best practice ideas found in SE. Consequently, SE has something to say in the way code should be structured or formed.

5 THE NEGLECT IN DATA SCIENCE EDUCATION

Unfortunately. most DS courses are wanting in covering issues where SE can provide good contribution to the DS field. We can expect this lack in short courses because there is not enough time to cover these topics in detail. There should be plenty of opportunities though, to address these issues in the bachelors and graduate level DS degree programs.

Below are a couple of recent research works on the subject of DS education.

5.1 A Survey on DS Education

Doing a survey on the present state of DS education at the moment is not easy. The reason is because the terminology is not settled. For example, some course providers use the term Data Science while others use the term Business Analytics for the same discipline. Because people are in a hurry to produce DS, it is expected that those entering the field will likely avail of on-line based education for their master's degrees in DS. To find out if course providers are offering SE subjects in DS training, our first impression is to look at the OCR (The Online Course Report - http://www.onlinecoursereport.com/), who published an article in their site - the "Best Online Masters in Data Science". The report classified programs according to affordability, flexibility, and student support.

We looked at the first in the list, the most affordable one – Dakota State University's program, but as can be expected there is a subject on programming but not SE as such. We looked also at big name universities, like University of California's offering but we noticed none of them dealing specifically in the art of SE. Of course, there are subjects about algorithms and so forth, but SE is broader than that (see SE goals mentioned above).

The recent work of (Song and Zhu, 2015), seems to be the most recent survey conducted on the subjects taught in DS education in the USA. Due to its shear volume of programs and degrees delivered, we can for now, look at this study as representative of DS education around the globe. The study surveyed DS offerings in universities and have been published in 2015. We repeat here their findings.

Table 1: Subjects offered in Bachelor's level.

Course	No. of unis offering the subject
Probability and Statistics	7
Data Mining	7
Programming	5
Discrete Mathematics	4
Data Structures and Algorithms	4
Database	4
Machine Learning	4
Statistical Modelling	3
Data Visualization	3
Introduction to Data Science	2
Artificial intelligence	2
Computer Security	2

We can see in Tables 1 and 2, none of them mention SE education formally speaking. Now granted there are those who offer data structures and algorithms and they are computer science topics indeed, but as we mentioned before they are a small part of SE . A course in programming language teaches a student how to use the syntax of the language and its semantics. Coverage of philosophical issues on how to architect modules and code, document them, how to design, test and package code are topics too large to be covered in a programming subject and properly should be covered in a separate SE course. What we are talking about are the considerations for the systematic designing, developing and debugging of the software product produced by the DS effort.

To illustrate this, we take the case of deciding to split a block of code into a function. Normally the programmer makes a chunk of code to a function be-

Table 2: Subjects offered in Master's level.

Course	No. of unis offering the subject
Exploratory Data Analysis	10
Database	10
Data Mining	9
Data Visualization	8
Statistical Modeling	8
Machine Learning	6
Information Retrieval	5
Information and Social Network Analysis	4
Data Warehouse	4
Introduction to Data Science	3
Research Methods	3
Social Aspects of Data Science	3
Algorithms	2
Data Cleaning	2
Text Mining	2
Healthcare Analytics	2

cause the coder sees that such a sub-routine can be used by some other parts of a program. However, that is not the only decision to farm out code into a function. We know some programmers do this because the work done by the function stands alone and from a reading perspective, doing so makes the code easier to handle and understand. Thus writing a function involves some reflection on the programmer's part. Discussion such as these and theories about function coupling and cohesion are topics found in SE courses and can be a very informative and revealing. Yet these are topics in SE and not just a topic in the use of a programming language.

5.2 DS Education by Transdisciplinary Approach

(Topi, 2015) reported a workshop conducted on DS education by the ACM Education Board and Council. The workshop brought twenty-five academic experts from various disciplines not only from the STEM group but even those academics from arts and social sciences. There were also representatives of professional societies like IEEE-CS, ASA(American Statistical Association), etc. The objective of this workshop is for academics and professionals in the field to have a respectful conversation on the future of DS education. It also explored the feasibility of developing curriculum guidance for DS degree programs.

Below is a list of identified competencies a DS education should provide (Topi, 2015):

- Broad areas of statistics
- Research design
- Predictive modeling
- Visual Analytics
- Computational and algorithmic thinking
- Programming
- Machine learning
- Data and database management
- Distributed and parallel technologies
- Data mining
- Domain knowledge
- Ethics and DS implications

The group recognized that there are varying competencies for those doing the analytic algorithms and complex statistical models versus those who acquire, prepare and transform data versus again on those who effect the results of the analytics work to the domain of practice.

A few things worth noting about the results of the workshop. Firstly, again we do not see SE being mentioned as a competency required of a DS education. We do see programming and other computer science topics like concurrent or parallel processing and so on, making the list (something we have also seen in the OCR article). However, as we mentioned before, SE now is a recognized discipline just like Statistics and Computer Science, and should be part of the DS's education. The direct benefit of this is not only to the DS in expanding and making his/her education whole but the clients that will be served by the DS. They too will get a holistic solution to their decision support problem. Secondly, we are not sure if it is a good idea for the competencies of the entire DS work to be split into compartments as the workshop seems to suggest. This does not bode well with the idea that clients of DS want an integrated solution. Splitting the competencies into different DS roles mean that some will become specialists in a skill but that means that the DS client will have to deal with various people and coordinate the management of them. Experience suggests that this adds consulting cost and puts an overhead into the coordination of these activities, not to mention the problem of dealing with personalities in those roles.

5.3 One Course Alluding to SE

In our finding, the closest course program that deals with SE as a lesson by itself, though again barely touching upon SE is the session ”What is software engineering in data science” found in the Coursera course ”A Crash Course in Data Science” given by Dr. Roger Pang and produced by John Hopkins University. It lasts for 6 minutes, which clearly is just an informative video that intends for the listener to dig in deeper into the subject on their own. As can be expected, the session does not elaborate greatly on systematic addressing of SE topics. However, the fact that there is a lesson devoted to this question, is by default a recognition of what we are saying here, that SE can not be avoided in the practice of DS and should be learned as well. Also, this is a welcome insight and the course developers should be encouraged for the topic. We hope others will follow that lead but in an expanded way.

6 THE WAY FORWARD

First, we need to state what we are *not* advocating. We definitely are *not* implying that DS reduces to SE, no, not at all. We are not saying they should be SE with DS knowledge either. Rather what we are saying is that SE should be taught in some small but meaningful way to DS students. We are advocating that SE fundamentals should be discussed if not in a very substantial way, at least sufficient enough for the DS to ensure the resulting code produced in the effort are manageable and reliable.

We propose additionally the following:

- The continued conversation as reported in (Topi, 2015) should be promoted. Indeed, it is good that such a discussion amongst academics and professionals is happening at this stage. The question as to what competencies a DS should have is a worthy question to mull over as it will only help rather than hinder DS as a separate discipline and profession. They can serve as a guide in forming he elements of DS education. Rather than keeping this local to USA, its composition should be extended internationally.
- If time permits in a degree or course program, then definitely at least a half-course, if not a full one, in SE should be included. Below is a core list of SE areas we believe should be taught in some way to people entering the DS world.
  - Software Architecture and Design
  - Software Engineering Process and Management
  - Software Quality and Testing

  The course supplier can embed the above as components in a broad course like Data Science Engineering Issues. We believe such a course will do well in addressing the importance of SE and will add value to the DS person’s capability and professionalism.

7 CONCLUSION

In this paper, we examined the work performed by the DS practitioner and we noted that SE issues crop up in the DS work. The DS has to write code and by that, relevant SE issues enters the DS field and we have proven knowledge of SE principles will greatly help the DS in his/her work and indirectly benefits the clients as well. We have shown the importance of SE and that the DS should have not just a cursory understanding of this skill. We then examined the courses offered right now by DS educators and demonstrated that there is a present neglect for the skill. We can perhaps theorize that the skill is taken for granted by DS education providers except for one who alluded to its importance gained from their experience as DS practitioners. Finally, we suggested some solutions for addressing this need that matches SE importance to the DS work. We hope that through this work, more and more DS education providers might incorporate SE fundamentals into their programs so that the profession and its consumers might benefit from the gains already accumulated by the SE discipline.

REFERENCES

Cage, D. (2017). What is a data scientist, anyway? *The Wall Street Journal.*

Guo, P. (2013). Data science workflow: Overview and challenges. *Communications of the ACM, blog@CACM.*

Jorgensen, P. C. (2014). *Software Testing, 4/ed.* CRC Press.

Leach, R. J. (2016). *Introduction to Software Engineering, 2/ed.* CRC Press.

Rose, D. (2016). *Data Science: Create Teams That Ask the Right Questions and Deliver Real Value.* Apress.

Song, I.-Y. and Zhu, Y. (2015). Big data and data science: What should we teach. *Expert Systems.*

Strawn, G. (2016). Data scientist. *IT Professional*, 18:55–57.

Topi, H. (2015). IS EDUCATION : Advancing data science education through a transdisciplinary conversation. *ACM Inroads*, 7:26–27.