2014 Delivery Management System

Recommendations for appropriate software development methodologies

Adam Bull

ABSTRACT

This report analyzes the variety of requirements as set out in the interview notes provided by Ms. Penny Pincher and sets to reduce ambiguity and redundancy in the original user specification by grouping functions by commonality. The report examines the leading software development mythologies in light of the main characteristics and ambiguities of the project and recommends that we adopt primarily agile methodologies in the development of the DMS, with some plan-driven benchmarking.

Table of Contents

Introduction

I have produced my report outlining the relative advantages and disadvantages of different software development life-cycle (SDLC) models. Before I begin discussing the merits of each individual SDLC in light of various aspects of the proposed 2014 Delivery Management System (DMS) outlined in the interview notes provided, I will provide you with a general introduction and definition of some common terminology. The software development lifecycle describes the life of a software product, with a particular emphasis on the internal processes, from its conception to its design, implementation, deployment and maintenance. A software process is a "set of activities that lead to the production of a software product"[1]. A methodology is a structure imposed on the development of a software product. Traditional software methodologies that you may be aware of include the waterfall and spiral models; these models are often referred to as "plan-driven" or "heavyweight" because they are highly structured and predictive by nature, and place a great onus on the development team strictly adhering to a schedule or sequence of targets. Another characteristic of heavyweight methodologies is that they attempt to garner the majority of requirements of a piece of software at the beginning of the process and do not attempt implementation until this is the case – the project is therefore completed within the bounds of one singular "cycle" or "pass".

"Agile" methodologies are a set of software development methods that stress the need for flexibility and collaboration in the development of software; Agile methodologies recognize that requirements for software change as time goes on, and that the methodologies used in the development of software must allow for these changes to happen, however last minute in the development process they are.

I will begin by discussing the merits of applying plan-driven methodologies to the development of the DMS and discuss the characteristics of the project that are most likely to be aided by a traditional approach, and then I will discuss the leading agile methodologies and how they can be applied effectively to the development of the DMS.

Discussion & Recommendations

Plan-Driven Methodologies

Before the 1970s, the most prevalent, if not only methodology used to develop software was the "ad-hoc" methodology. This is essentially the opposite of a methodology, as it has no guidelines about how software should be developed – this method led to spiraling overheads on projects, and inspired Winston Royce to develop the Waterfall model to deal with the increasing complexity of aerospace software. The waterfall model was the first of many traditional or "heavy" methodologies that emphasized a structured approach to developing software and stressed that software should be designed and implemented in clearly defined stages.

Waterfall Model

The DMS specifies various high-level user requirements like being able to "track deliveries" and "resolve access conflicts on an ad-hoc basis" but it doesn't provide much detail on how the business would like these functionalities implemented. Of course, there is still the design phase to come, where most of the user interface (UI) considerations will be mocked up, but translating abstract user requirements into quantifiable "system requirements" is key to the realization of a system that meets business requirements.

The waterfall model is concerned with gathering the majority of requirements at the beginning of the project, which in this case may not be possible. As the waterfall model includes only one "cycle" of development, it is often the case that by the time the implementation phase is nearing its end, the requirements of the busine[illegible]ave changed and the development team often need to go back and amend something, which can prove to be very costly.

[1] Khan, Qurashi & Khan – pp. 441

In the case of the DMS, I can stipulate that a majority of the requirements have yet to be discovered; the inner workings of each function (particularly the website interface) have not been comprehensively mapped out, and so therefore the waterfall model is probably not very appropriate here as adopting it will result in huge cost if new requirements come to light.

There are a multitude of reasons why the plan-driven waterfall approach became the precedential model in use in the early years of software development; perhaps the primary reason is that it is easy to explain to the business stakeholder, another is that it “gives the illusion of an orderly, accountable, and measurable process, with simple documentation-driven milestones”[2]. However, just because something is easy to understand in layman’s terms does not necessarily mean that it should be applied to the development of software, which is usually a very complex process.

Spiral Model

Barry Boehm first outlined the Spiral model in the 1986 paper “A Spiral Model of Software Development and Enhancement”; he described a new software methodology that would combine elements of the waterfall model and of another model called the *prototyping model*. The prototyping model places the onus on creating many prototypes through the lifecycle of the software, each time going back to the customer and reviewing it with them. In a way the prototyping model is a form of insurance; it allows the development team to capture a high proportion of potential issues with a design before developing the full application, without incurring the cost of building a full scale version of a piece of software. The prototyping model shares some similarities with Rapid Application Development (RAD), which I will remark upon later on in the report. The advantages of using the Spiral model in the development of the DMS are many – if the business / stakeholder is willing to fully participate in the development process, then the Spiral model can be very valuable in assuring that all of the business’s requirements are fully realized. The Spiral model can be very valuable if the stakeholder does not fully know their needs.

If we examine the process involved in the Spiral model in relation to a test case within the DMS we see how useful it could be; the Administration / Monitoring functionality described in the interview notes could be efficiently streamlined with the Spiral model. The first step in the Spiral model would be to garner the main requirements of the Monitoring/Administration functionality (these include the prescription of a “Roles” system, as well as several individual functions to monitor and perform administrative functions on the data in the DMS). The steps would be as follows: firstly the main requirements above would be established, and then a design would be made for the new system. A first prototype would be constructed, usually with a scaled down set of features. The first prototype would then be sent to the client for review, allowing them to evaluate the first prototype for any obvious usability problems as well as any strengths and weaknesses. A second prototype would then be made, building on the lessons learnt from the issues in the first prototype. This process would continue recursively until the client is happy with the full-scale software system.

There are disadvantages associated with the Spiral model, however; the Spiral model is often associated with high overheads and the high risk of project failure. Often a client may lose heart, or the requirements change so much that the prototypes are unable to keep up with the rate of change. If we examine the DMS in light of these disadvantages, there are actually a lot of unknowns and the prototyping method may indeed be too costly and slow-paced to adopt for all areas of the DMS. However I recommend that some key parts of the system could be built according to the Spiral model – particularly the website aspect of the DMS and the Administrative and Monitoring functionalities.

Rapid Application Development (RAD)

Although many consider RAD to be a subset of Agile, there are several key reasons why Rapid Application Development (RAD) should not be viewed as an agile methodology; perhaps the most important one is that Agile does not allow prototypes; RAD is concerned with designing prototypes and then re-engineering them into the production code. RAD would be useful in the development of the DMS because it allows the development team to produce low cost interactive mockups of the full system. RAD would be particularly useful for allowing the client to highlight any UI issues that may arise or may not have been accommodated for.

[2] Larman, C – ‘Iterative and Incremental Development: A Brief History’ – p. 10

This element of RAD is particularly useful for user-facing elements of the DMS such as the website interface and the Administration & Monitoring functionality that is to be accessible through a standard PC interface at the storage facilities.

Adopting RAD does have some disadvantages however; it is noted in the interview notes that the lead-in time for the project is quite short, which indicates that the client would ideally like to see a full-scale piece of software fairly promptly. In light of this, the RAD methodology could be seen as being too ponderous.

In the email briefing, it is noted that Ms. Penny Pincher is very busy and may or may not be available to assist in the development process – RAD, along with other lightweight, collaborative methodologies, require that the business stakeholder is fully immersed in the development process, so potentially if Ms. Pincher, on behalf of the client, is unwilling to involve herself in the day to day prototyping process, this methodology may not be entirely appropriate.

“Agile” Methodologies

To be ‘agile’ is to be “able to change or be changed rapidly in response to customer needs and market forces”[3]. This definition highlights the fleeting nature of requirements in most modern software projects. It is likely the implementation phase will unearth new requirements and that in the development of the DMS, particularly the user-facing elements of the system, that the business stakeholder will discover that they can implement a better feature than previously imagined once they have signed off on the design. This causes problems if development teams adopt fixed plan-driven models.

Iterative and incremental development are not all-encompassing methodologies like plan-driven models tend to be; they are terms used to describe the manner by which development teams conduct their builds. The concepts of iterative and incremental development can be combined or the methods can be used separately; the iterative method involves developing a system through repeated cycles, whilst the incremental method is concerned with developing a system in small modular “fragments” through each build. Iterative and incremental methods are key tenets of agile methods, so in a sense they are interchangeable with the term agile. In an article on the aforementioned methods, Craig Larman notes “some view iterative, evolutionary, and incremental software development—a cornerstone of [agile methodologies]”[4].

In February 2001, a group of 17 experts representing a range of lightweight methodologies met in Utah to discuss common ground. They established the Agile Alliance, and later wrote the *Agile Manifesto,* which sets out the main tenets of agile methodologies: to be agile means to welcome changing requirements, as well as placing emphasizing the importance of collaborating with the client at every stage of the lifecycle. Agile methodologies tend to focus on delivering small-scale working fragments of software on much more frequent schedules. In a sense you could compare being agile to the prototyping model described earlier on in this report. The two market leading agile concepts are Scrum and Extreme Programming (XP). Scrum is a methodology, meaning that it prescribes the process of developing a system from a project management perspective, whilst XP is a programming practice that prescribes best practices to software developers.

XP and Scrum are definitely very aligned, and share a lot of common ground, but there are some key subtle differences that in relation to the development of the DMS may prove the difference between fully meeting the client’s requirements or only proving to be satisfactory. Both XP and Scrum are based upon the iterative model, but the spans of time in which the iterations take place differ between the two.

Scrum

[3] Oxford English Dictionary Online [http://www.oed.com/view/Entry/3979?redirectedFrom=agile#eid]

[4] Larman, C. Basilli, V – *Iterative and Incremental Development: A Brief History* – p. 47

The Scrum methodology revolves around roles; the main project management role is that of the *scrum master.* The client is called the *product owner*, and developers are known as *team members*. The scrum master's role is to ensure that there is cohesion between development team members and the project owner. The *product owner*, which in the case of the DMS will be a representative of the client company, will be in charge of prioritizing requirements, and the development team will work on the requirements in order of prioritization. Scrum typically works on two week to one month iterations called "sprints" which are concerned with clearing a sprint "backlog" item; the DMS may benefit from the short iteration cycles of the Scrum methodology, particularly with the view to altering the requirements process, however, unlike XP, once a sprint has begun, the requirements "set" has to remain unchanged until the end of the run.

If we were the adopt this methodology for the DMS system, it would allow the client to have full control over the course of the development process, as it would allow them to specify what functionality gets built first. However, there are associated risks with handing over control of prioritization to the product owner; often a development team works best if they control their own workflow and decide what should be prioritized based on their own technical expertise. Scrum does allow for the development team to determine how they will break down the backlog into tasks, so there is a degree of autonomy on both sides. Other risks of Scrum include the risk of *scope creep*, meaning that the project owner may be tempted to keep demanding new functionality whilst the build is in process.

Extreme Programming (XP)

Extreme Programming is fairly similar to Scrum, except that in XP teams are more open to change within iterations. Another key difference is that whilst in the Scrum methodology the *product owner* decides the backlog and the Scrum team decides how the backlog will be broken down into tasks, in XP the order of prioritization is decided completely by the product owner. The appropriateness of implementing XP over Scrum is entirely dependent on how often the client intends for the collaborative iterative process to take place.

Conclusion

In conclusion, choosing the right software development methodology is not a black and white process; in most modern software projects, many combinations of methodologies are used, and often it is the case that the most appropriate elements of vastly different methodologies are adopted that best suit the specific project. There is no "silver bullet" solution to choosing the right software methodology; however most modern software projects make use of an iterative-based approach so that requirements can be tweaked at the end of every iteration. The development of the Delivery Management System would benefit from an iterative approach as there are still a lot of unknowns.

Most of all, I think what would be most pertinent to glean from this report is that blindly adhering to one methodology is a recipe for disaster; every project is different and the DMS could benefit from both the benchmarks set out in a plan-driven approach (i.e. deadlines and due dates could be set for the various functionalities described in the interview notes), but it could also benefit from the collaborative, iterative nature of an agile approach. I will leave you to decide which elements of the methodologies I have discussed above are most agreeable with your business requirements.

Bibliography

- Sommerville, Ian (2011). *Software Engineering*. 9th ed. Harlow, United Kingdom: Pearson Education Ltd. p3-111.
- Robertson, James. Robertson, Suzanne (2009). *Requirements Specification Template*. 14th ed. Atlantic Systems Guild Ltd.
- Khan, IA, Qurashi, JR & Khan, AU 2011, 'A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies', *IJCSI International Journal of Computer Science Issues*, vol 8, no. 2, pp. 441-450.
- Larman, C. Basilli, V. (2003). Iterative and Incremental Development: A Brief History. *Computer Magazine (IEEE)*. 36 (6), p47-56.

Appendix

Disclaimer

As Ian Sommerville notes, the term “requirement” is not used consistently in the software industry; sometimes it is used to refer to high-level, abstract “statement of a service”[5] (Sommerville calls these form of requirements “user requirements”[6]) whilst at other times it is used to refer to a detailed technical description of an implementation of the service itself (these are “system requirements”[7] in Sommerville’s view). I have attempted to translate the “user requirements” set out in the interview notes into system requirements (although in my belief the technical detail on how each requirement is going to be implemented is still lacking, although this may be something that is decided at a latter stage of the development process). The requirements I have chosen below are in my opinion the ones that will be most visible to the user if they are missing from the DMS-P. For a full list of requirements (including stipulated ones), I have produced a mind map that can be found attached with this report (see DMS-P Requirements.pdf).

Please find attached the completed Volere shells for 5 critical functional requirements and 3 non-functional requirements:

Functional Requirements

Requirement #:	1	**Requirement Type:**	9a (Functional Req.)	**Event/ BUC/ PUC #:**	N/A
Description:	The system should incorporate a function for the reception role so that they can manually enter data concerning vehicle entries and exits to simulate the data that will eventually be provided by the barrier entry and vehicle registration recognition system (out of scope).				
Rationale:	As the barrier entry and vehicle registration recognition systems will be developed at a latter date, provision of an alternative method to log vehicle entries and exits is key to the overall success of the DMS prototype, otherwise there will be no data on the system by which other functionality can act upon.				
Originator:	Miss Penny Pincher				
Fit Criterion:	The system fulfils this requirement if the data stored by the above function agrees with the data manually entered by the receptionists at the reception office.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, sections 5.3 (paragraph 1), section 6			
History:	None				
Comments:	Section 5.1 and 5.2 describe the need for some form of functionality to collect and store driver and delivery details when the driver/delivery arrives at the storage facility, and yet section 5.3 notes that the collection of these details will take place through the DMS website – so there is definitely some redundancy here – for the purposes of the DMS-P, I have assumed that the website will be the primary method for collecting this data, and therefore have not included the requirements as set out in sections 5.1 and 5.2 of the interview notes in the Volere shells here.				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Requirement #:	2	**Requirement Type:**	9a (Functional Req.)	**Event/ BUC/ PUC #:**	N/A

[5] Sommerville, pp. 83.
[6] Ibid.
[7] Ibid.

Description:	The system should have a web interface (termed “DMS website”) that allows delivery company employees to view availability of delivery slots at any given storage facility.		
Rationale:	This is so that delivery company employees can see which slots are available and book their deliveries in for these timeslots if they desire.		
Originator:	Miss Penny Pincher		
Fit Criterion:	The system fulfils this requirement if it shows the website user all of the available slots that are available at a storage facility.		
Customer Satisfaction:	5	**Customer Dissatisfaction:**	5
Priority:	High	**Conflicts:**	None
Supporting Materials	Interview notes, section 5.3		
History:	None		
Comments:	This requirement is one of the 3 main requirements given for the DMS web interface – the others being the ability to book a delivery slot if it is free and it satisfies the preconditions (i.e. less than 24 hours notice), and also registration of driver and delivery details. I decided to split these 3 requirements out into separate sub-requirements, as they are three separate pieces of functionality. Does the online booking system require that the user registers their details first or can guests book a delivery without registering?		
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited		**acknowledgement to Atlantic Systems Guild**	

Requirement #:	3	**Requirement Type:**	9a (Functional Req.)	**Event/ BUC/ PUC #:**	N/A
Description:	The system’s DMS web interface (“DMS website”) should allow users to book deliveries for a storage facility.				
Rationale:	So that delivery company employees can give the storage facility staff prior notice to their arrival, and also provide the storage facility staff with details of the deliveries they are carrying out (including type of delivery, what is being delivered, their contact details etc).				
Originator:	Miss Penny Pincher				
Fit Criterion:	The system fulfils this requirement if the booking made on the DMS website is stored correctly, and is available to view on the main DMS system.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 5.3			
History:	None				
Comments:	The technical details of how the DMS website will connect through with the main DMS system is not described in the interview notes, but it is presumed that the web UI and the DMS system as accessed by those in the Reception and Administrative roles will share the same database.				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Requirement #:	4	**Requirement Type:**	9a (Functional Req.)	**Event/ BUC/ PUC #:**	N/A
Description:	The system should incorporate a view whereby those in the administration role can view the status of access to the facility, and resolve access conflicts and other problems on an ad-hoc basis.				
Rationale:	So that storage facility staff can be made aware of any problems that arise, and act on them by resolving conflicts manually on an ad-hoc basis.				
Originator:	Miss Penny Pincher				
Fit Criterion:	The system fulfils this requirement if staff are alerted of problems that arise by the DMS (on screen warning), and can resolve the problems easily using the DMS ad-hoc resolution system.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 6.1			
History:	None				
Comments:	This requirement is a combination of the requirement given in section 5.4 of the interview notes (“access conflicts”) and the requirement given in section				

	6.1. I decided to combine the two requirements into one as they both share commonality. One could say that they are two separate requirements (1. To see the status of access at a storage facility and 2. To resolve a conflict/problem if one arises), but they are concerned with a common task (that is, the problem of “Access conflicts”), so I decided that they are technically one requirement.
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited	**acknowledgement to Atlantic Systems Guild**

Requirement #:	5	**Requirement Type:**	9a (Functional Req.)	**Event/ BUC/ PUC #:**	N/A
Description:	The system should provide a function which allows those in the administration role to maintain competitor and delivery contractor information.				
Rationale:	This is so that administrative staff can amend personal/business details of contractors and competitors if the information changes to ensure that the data is kept up to date.				
Originator:	Miss Penny Pincher				
Fit Criterion:	The system fulfils this requirement if the administrative staff are able to edit and update data through the DMS to reflect changes in that data.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 6.1			
History:	None				
Comments:	This is another sub-functionality of the Monitoring and Administrative function – the ability to “maintain” contractor and competitor data I stipulated as being something that those in the reception role were not authorized to do – so therefore it is a different requirement to that of the requirements set out in sections 5.1 and 5.2 (collectively to collect personal and contact details of delivery contractors). The ability to **edit** said data once it is in the system is an entirely different requirement. N.B The requirements set out in sections 5.1 and 5.2 above are not listed here because they are redundant due to the fact that the system will collect driver and delivery details through the DMS website, and therefore there isn’t a need to do so upon arrival at the storage facility.				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Non-Functional Requirements

Requirement #:	6	**Requirement Type:**	5b (Business rules)	**Event/ BUC/ PUC #:**	N/A
Description:	The system should not allow users of the DMS website to book a delivery for a date that is less than 24 hours away.				
Rationale:	It is a constraint of a business that deliveries (and authorization thereof) must be booked with 24 hours notice, and this requirement satisfies the business constraint.				
Originator:	Miss Penny Pincher				
Fit Criterion:	The system fulfils this requirement if it rejects any bookings that are made on the DMS website that take place in less than 24 hours from the date of booking.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 5.3			
History:	None				
Comments:	The fit criterion uses the term “reject” to mean the website will not allow the submission of invalid deliveries, and therefore invalid delivery bookings will not enter the main DMS system. Some form of validation will be in place here (an error message indicating to the user that the booking is not valid).				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Requirement #:	7	**Requirement Type:**	5b (Business rules)	**Event/ BUC/ PUC #:**	N/A
Description:	The system's booking website should not allow deliveries to be booked in for the hours of 10pm to 7am.				
Rationale:	This is due to a business constraint described as "delivery issues" – overnight deliveries are permitted, but only on an ad-hoc basis (see comments for more details).				
Originator:	Miss Penny Pincher				
Fit Criterion:	If a user tries to book a delivery through the website for a time that lies between 10pm and 7am, the booking will be rejected.				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 5.5			
History:	None				
Comments:	In the interview notes, it is noted that deliveries to the storage facility between the hours of 10pm and 7am is permitted, but only on an ad-hoc basis. I have stipulated that an "ad-hoc basis" means that a delivery within these hours must be authorized by a member of the Administration team (which is an action that lies outside of the remit of the DMS). The main channel in the DMS system for booking deliveries is the website, and its standard functionality it is noted is that it should disallow bookings for these times.				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Requirement #:	8	**Requirement Type:**	15a (Access Requirements)	**Event/ BUC/ PUC #:**	N/A
Description:	The system should incorporate a "Roles" system whereby users can only access functionality that the role prescribes (i.e. users in the "Reception" role cannot gain access to the functionality described for the "Administration" role)				
Rationale:	This is to prevent unauthorized users accessing functionality that is not within their remit or authorization level, and to stop them modifying business data if they are not authorized.				
Originator:	Miss Penny Pincher				
Fit Criterion:	Users in the "Reception" role should not be able to access functionality associated with the "Administration" role (i.e. the monitoring function).				
Customer Satisfaction:		5	**Customer Dissatisfaction:**		5
Priority:	High		**Conflicts:**	None	
Supporting Materials		Interview notes, section 6.1 / Volere Requirements Specification Template (Edition 14)			
History:	None				
Comments:	I had originally considered this to be a functional requirement as it involves provisioning some form of authorization system and involves some development work, but after having read the Volere template (v14), section 15a seems to describe this type of requirement, and it is listed as being a non-functional requirement. Example from the Volere documentation: "Only direct managers can see the personnel records of their staff." (p. 44 of Volere documentation)				
Volere Shell © 1995 - 2010 the Atlantic Systems Guild Limited			**acknowledgement to Atlantic Systems Guild**		

Full Requirements Mind Map

- DMS-P Requirements
  - Functional
    - **The system should incorporate a function for the reception role so that they can manually enter data about vehicle entries and exits (data is auth code, date/time, vehicle reg) to simulate the data that is to be sourced from the barrier entry system (out of scope)**
    - The system should incorporate a function whereby those in the administration role can monitor usage of a facility over a fixed time period.
    - The system should incorporate a function to process and store data on vehicle entries and exits
    - The system should incorporate a view whereby staff can view the status of access to the facility, including details and error messages if necessary about failed access attempts, failures in (hypothetical at this stage) barrier systems, etc + wrongful delivery
      - **SUB REQUIREMENT:** The system should incorporate a function that enables staff to resolve these form of access conflicts on an ad-hoc basis.
    - The system should incorporate a roles system whereby only authorised staff members have access to the specific "view" (the two views are the "Admin" function and the "Monitoring" function)
    - The system should incorporate a function whereby those in the administration role can present the facility usage data in a variety of ways, specifically graphically
    - **The system should provide a facility for those in the administration role that allows them to manually enter competitor and contractor data, and to be able to edit it at a later date.**
    - **The system should have a Web Interface where delivery company employees can book deliveries (termed "DMS Website")**
    - **The system's Web Interface (termed "DMS Website") should also allow the user to register relevant details (inc. their vehicle's details, their contact details)**
    - **The web UI should allow user to search for availability of space for delivery**
    - If a single driver has multiple deliveries, the system must issue create separate records in the database for each delivery (even if the driver's details are held on a singular record)
    - The system should incorporate a facility that alerts users to deliveries that are nearing their validity expiration times
    - The system should save the details provided by the delivery contractors on the website in a database for retrieval at a later date.
    - The system should record details of what is being delivered as provided through the DMS website
  - Non Functional
    - The system should not allow users of the DMS website to book a delivery for a date with less than 24 hours notice
    - The system should be able to recall previous vehicle details such as reg number if a delivery contractor has more than one delivery within a small timescale
    - The system must provide some form of feedback message if a user tries to access a piece of functionality that they are not authorised to see.
    - The system should incorporate a colour coding system to identify deliveries that are valid, and those which have run into problems.
    - The system should not allow deliveries to be booked in the hours of 10pm - 7am (due to business constraints)