Winter | 15

A Reference Model Method to align the development of one software system with multiple Hinterland Container Terminals

Master Thesis BIT by Pim Dietz

Supervised by Dr. Klaas Sikkel and Prof. Dr. Jos van Hillegersberg

At the University of Twente.

UNIVERSITY OF TWENTE.

COFANO

Management summary

Cofano Software Solutions B.V. develops the Terminal Operating System (TOS) to support the business processes of Hinterland Container Terminals (HCTs). The system under development is to be implemented at multiple HCTs using the same codebase. Therefore, Cofano needs to find a fit between the supported business processes of the existing codebase and the adopting organisations. This raises the need to identify commonality and variability between the customers' business processes while dealing with a growing organisation.

To that end, we developed a method that makes use of a reference model. A first version of the reference model has been developed by both reverse engineering the assumptions about the supported domain from the system as well as the development team's mental model of the supported domain. The model describes the assumed processes supported by the TOS, thereby enabling the comparison with the actual business processes to find a fit. The method is tailored to fit into the agile software development process of Cofano.

According to experts' opinions, the method has merit in meeting Cofano's arising needs and even unforeseen advantages, such as the ability to make the added value of the implicit knowledge explicit to the customer and to provide process optimisation as part of its implementation process. There are, however, important challenges that may impede the advantages of this method. Sharing knowledge with the competitor may be an objection to cooperate in developing new variations for the system. In addition, required capabilities in modelling skills and abstract thinking in using the method chunk are stringent, changes to the codebase should be made with care as the existing customer base may disagree, and changes proposed to the adopting organisation are more easily said than done.

The most quantifiable advantage is estimated to be between 15-20% of the implementation project's resources in terms of programming and configuration activities and is most likely to increase as more variations are added and the customer base expands. Not including the other qualitative benefits in the equation, this implies that at least 15% of an implementation project's resources may be spent to maintain the reference model and cope with its challenges and pitfalls before a break even point is reached and further investment in the method no longer yields any advantages.

For its adoption at Cofano, a few workshops are recommended given the required capabilities of the method's users. Through these workshops, both the organisation's required capabilities in its use and the method itself further develop from the experiences in these workshops. The learning experience by means of workshops prevents confronting customer organisations with a suboptimal methodology.

As the method and the organisation's capabilities require further development, it requires investment in terms of time and effort at first before its benefits materialise. In addition, the TOS' variations and customer base are currently small but expected to expand in the future. Both imply that the benefits of its adoption come with a delay.

Table of contents

1. Introduction

Cofano is a small software development company that targets roughly two markets. One of those markets is the logistics industry. Its goal is to supply software to support their main processes and gradually improve transparency throughout the network of actors involved in the logistic chains. One of their main products for this market, the Terminal Operating System (TOS) aims to do so by delivering the Software as a Service (SaaS) application, standardising the processes it supports where possible and provide a few alternatives to these processes to fit the idiosyncratic needs of a (few) particular customer(s) through mass customisation (i.e. functionality can be enabled and disabled by the user).

The logistics industry, however, is quite old fashioned when it comes to technology and information sharing. As one of the owners of the Cofano put it: "We try to innovate **with** the industry." Indicating it is a conservative industry, which requires a lot of dialogue to develop innovative software solutions and achieve acceptance of doing things differently than before.

Roughly speaking, this dialogue has two facets. On the one hand, the dialogue has to maintain a relationship with the customer. This is the commercial side of the dialogue where it is up to the personal social experiences, social capital, sales techniques and/or ability to convince the customer while managing expectations of the persons engaging in these dialogues to maintain a healthy relationship. On the other hand, in order to actually develop new solutions, the business of the customers needs to be analysed to identify the needs of the new solutions and opportunities for (radical) improvements.

It is therefore not surprising that this dialogue with the customer is of high importance to the success of Cofano in this market. Cofano has two owners. Currently, one of which is mainly responsible for this dialogue in the logistics market; in other words, he is the product owner. He has indicated that he does not wish to continue managing the customer relationships in the future, certainly as the company and its clientele will grow.

The process is observed to be unstructured; even at later stages in the relationship with customers, the meetings with the customer resemble that of a brainstorm session and product demonstration mash up, while the main business processes and information flows to be supported remain unclear to the development team. This results in seemingly even more changes in requirements, as the proposed solutions appear unfit or not as optimal as expected, and consequently costly changes.

In the future, these processes will have a focus more on user acceptance, rather than developing new solutions: Once the main products/solutions have been developed in dialogue with the early adopters, it is to be seen how these will fit the needs of new customers that have a similar business, assuming they have similar problems and processes as the early adopters. The main innovations done with the early adopters have to be accepted by other customers (albeit with some customisation or extensions as is to be expected from any software implementation, but these are then available to any other user as well and added to the mass customisation options).

In order to not only develop the new solutions with the early adopters, but also to achieve acceptance of existing solutions, the businesses need to be analysed in order to gain insight whether existing solutions will fit the needs of the new customer, identifying idiosyncratic or new needs if any.

In essence, the dialogue with the customer is a 'solution-developing-cycle' comparable to Wieringa's design science cycle, where the problem context is analysed, or problem investigation (Wieringa, 2012). It is the analysis of the problem context, in particular its entities, relationships and events the system should 'know' about (i.e. the subject domain (Wieringa, 2003)), and business processes the system should support, that will be the focal point of this thesis for both the development of new solutions and the acceptance of existing solutions.

1.1 Challenges for Cofano

The challenge for Cofano is to consistently conduct the problem context analysis by multiple employees of Cofano in both of two scenarios:

In the first scenario, the main solutions are yet to be developed with early adopters, which are to be used later by the rest of the industry. The business analysis will explore the businesses of early adopters to 'innovate with'. Therefore, the analyst needs to be able to effectively identify needs and opportunities for radical improvements using the technological capabilities and insights of Cofano. This phase is similar to the first two steps in Steven G. Blank's 'Four Steps to Epiphany (Blank, 2006).

The second scenario is similar to the last two steps from Blank (2005). Solutions have been developed with the early adopters and an analysis of the new customer's problem context is required. The analysis identifies whether the same solutions will fit that organisation or idiosyncratic needs require customisation of the solutions. In addition, the analysis of the new customers may give insight into new opportunities for improvements that are applicable for other customers in the industry as well. For instance, recently a new customer acquired by Cofano included a business aspect unexplored yet by Cofano; an hinterland container terminal that also provides its customers with the rail modality. This instance is expected to be a very important aspect for many of Cofano's future customers. It has given rise to the opportunity to develop solutions that (radically) improve the performance of this business aspect as well.

Furthermore, the business analysis should fit into the agile software development approach the developing team is currently slowly adopting. Cofano's development team is slowly adopting practices from Scrum. Part of which is the acquisition of feedback about the proposed solutions after each short iteration called a 'sprint'. The feedback will refine the insights of the problem context for another iteration in developing fit solutions.

With the future in mind, the business analysis method should be scalable for the future growth of the organisation. More than one person will engage in dialogue with the customers of Cofano. Cofano's customers in the logistics industry can be divided into types depending on their role in the logistics chain. Hinterland container terminals and barge operators are examples of these types. The concept of Cofano is fitted in a SaaS

application with one single codebase per customer type where the 'best solution fits most'. There is a need to identify variability and commonality among business needs of different customers of the same type. Sharing information within Cofano about the problem contexts between product owners is key.

With the organisational growth also comes a more distributed character of the development team; currently there is a development team in Sliedrecht and a smaller team in Enschede. This aspect also calls for the need to effectively share information. The distributed character is a known factor to negatively influencing coordination and communication in distributed development teams. Effectively sharing the long term task knowledge (i.e. the problem contexts) will implicitly improve coordination between distributed development teams (Espinosa, Slaughter, Kraut, & Herbsleb, 2007).

In addition, required information to develop solutions is easily overlooked in an unstructured setting where conversations spur in every direction. The business analysis needs to effectively cover the required information while preserving the informal charms of the dialogue.

In summary, meeting those challenges the business analysis needs to:

- Fit the agile approach of the development team (i.e. it fits the iterative and value first approach of the development methodology).
- Part of the above but very important: Acquire feedback from the customer about the proposed solutions to refine the insights of the problem context and hence provide input for the development of a better solution.
- Be scalable for the future growth of the organisation (i.e. used by more than one product owner, able to share information within the organisation and create transactive group memory to support the distributed character of the development team).
- Efficiently and effectively acquire information about the problem contexts (i.e. entities, relationship, events and business processes).
- Support the identification of variability and commonality between problem contexts of different customers.
- Preserve unstructured practices that promote innovation and relationship building with the customer (it should support the dialogue, not lead it)

1.2 Research goal

The challenges for Cofano described above call for the development of a method chunk that focuses on the business analysis, supplementing the main development process. A method chunk is a made-to-measure development methodology element that can be instantiated for more than one software development project (Rothengatter, 2012). As the Hinterland Container Terminal (HCT) is the most prominent customer type that currently gives rise to the challenges described, this project revolves around the development of a method chunk for development projects with this customer type.

Therefore, given the method chunk as artefact to develop, the challenges described as problem context and stakeholder goals, the research goal is formulated as:

To improve the acquisition of information about the problem context of HCTs,

by developing a method chunk,

such that the information about the problem context is structured, used and shared throughout the organisation and best practice is applied to acquire that information,

in order to improve accuracy of the acquired information (effectiveness), acquire the information in less iterations (efficiency), identify commonality and variability among problem contexts and prepare Cofano for a growing organisation with more product owners and customers.

Ultimately the improved information acquisition about the problem context should result in a reduction of resources needed to develop suitable solutions by hitting the target earlier throughout the iterations of development. Solutions will fit the target organisations better due to the improved accuracy.

1.3 Method chunk

The concept of method chunks comes from the (Situational) Method Engineering ((S)ME) approach, which is an approach to select or configure the most applicable methodology for a specific IS development project. SME is a reaction to the fact that each project has its own idiosyncrasies and no traditional methodology can foresee all these idiosyncrasies. In addition, traditional methodologies are often difficult to adapt to a given situation (Harmsen, 1997). To that end, SME aims to develop a methodology that is constructed from smaller components, called *method fragments* or *method chunks*.

As there is no consensus in literature on the distinction between fragments and chunks (Ågerfalk et al., 2007), Rothengatter's definitions will be adopted for this project (Rothengatter, 2012). These definitions are:

A method fragment is an atomic methodology element, either product or process oriented.

A method chunk is a combination of at least two method fragments, one of which product oriented and one process oriented.

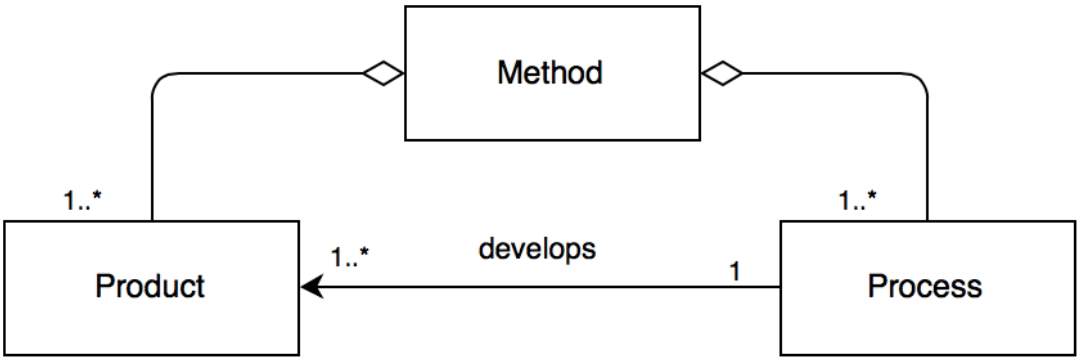

Figure 1 - Composition of a method (adopted from Rolland, Prakash, & Benjamen, 1999)

In short, an overall methodology has two interdependent aspects consisting of products and processes (Rolland et al., 1999)(see Figure 1). Therefore, a method chunk is a combination of, at least, one product oriented fragment and one process oriented fragment. In order to construct the method chunk, at least one of both is required.

There are two mainstream approaches to identify or construct a method fragment (Rothengatter, 2012). The first one is to create method fragments by reverse engineering existing methodologies. The second approach is to create method fragments from scratch. However, the reverse engineering approach poses problems, as most methodologies are not modularly built. Therefore, the fragments need to be created from scratch.

The initial design of the method chunk consists of four stages (Rothengatter, 2012): (1) the identification of the contingency variables; (2) the definition of the method chunk's scope; (3) selection of method fragments and assembly thereof into a method chunk; and (4) implementation of the method chunk into the overall methodology.

However, the first step is based on contingency theory. Contingency theory suggests that a number of variables influence the performance of information systems, where a better fit between the variables design and use of the system imply a better performance of that system and ultimately of the organisation (Weill & Olson, 1989). However, the causal relation between IS performance and organizational performance is often assumed (Rothengatter, 2012).

There is some criticism on this theory (Weill & Olson, 1989). The concepts of fit and performance are often ill defined and performance measures are often reduced to one single measure, making it difficult to apply. In addition, there are conflicting empirical results from studies measuring similar constructs that implicates low correlations between the variables. It is suggested that the relationship between the independent variables on organisational performance are more complex than contingency theory assumes (Schoonhoven, 1981).

In relation to this project, the theory poses difficulties to apply as it requires a longitudinal study to relate the effect of the methodology on the performance of the IS developed by Cofano and it will be near impossible to control for other variables. Therefore, this project takes a more pragmatic approach and bases the argumentation for the methodology chunk's development on the analysis of the problem context (Cofano itself in this case, not its customers) for it to affect. In other words, the method chunk will be based on Cofano's challenges as analysed and the assumptions made about the effects the method chunk has in supporting the stakeholder goals as described in the research goal. These effects will function as evaluation criteria throughout its development.

Since fragments are either product oriented or process oriented both the process model(s) and the product model(s) of the methodology have to be developed to construct the method chunk. A product model describes the concepts, the relationships between these concepts, and the constraints that the concepts have to satisfy in the product resulting from the method. A process model describes how the corresponding product is developed.

The second stage defines these models for our method chunk:

- **Product model:** Given the purpose of the method chunk, it needs to produce a product containing the information about the problem context (Cofano's customers) that tells the development team what kind of solutions or variations of those solutions to develop. Therefore, as product model for the chunk, a framework needs to be developed that structures and identifies the required information about the problem contexts to meet the knowledge sharing goal and consequently the identification of commonality and variability goal.
- **Process model:** The method chunk needs a process model describing how it is used in the overall development process of Cofano.

In the third stage, the actual models are constructed and combined after which, in the last stage, the method chunk can be applied in the overall methodology.

2 Research questions

In order to develop the method chunk and its constituents, the following questions must be answered. The main question is:

What is a suitable method chunk for analysing the problem contexts of HCTs in order to identify their needs and priorities for software solutions?

In order to answer that question, a product model and the process model need to be developed.

To develop the product model, we need to know what pieces of information about the problem context are needed such that we can develop a product model that supports its identification and documentation:

- *Which pieces of information about the problem contexts can be identified that are relevant for analysis?*
- *What is a suitable method for documenting the relevant pieces of information?*

To develop the macro process, we need a process model that describes the use of the product model in the overall development process of Cofano:

- *What is a suitable process for applying the product oriented fragment in the overall development process of Cofano?*

3 Approach

Since the fragments and the chunk is developed from scratch, it is said that theory and practice permits the initial identification of fragments and chunks (Rothengatter, 2012). After which, the newly constructed fragments are evaluated by practitioners, refined and evaluated in an iterative fashion. Therefore, this project follows a similar iterative approach to develop the method chunk.

The first version of the product model is developed based on best practice from literature. The different aspects of the current (early adopting) customers' businesses (HCTs) as seen by the software system are identified and structured in that framework. As such, the reverse engineering of the system in conjunction with the mental models of various software engineers and personal firsthand experience on the customers' problem context form the input for the first version of the product model. Throughout its development, the model is often subjected to feedback from fellow business analysts whether it is able to identify the relevant information, describes it correctly and is able to structure it parsimoniously.

Then, the process model for using the product model in the context of Cofano's development organisation is developed by analysing and extending its current development process.

After that, the product model and the process model form a complete method chunk. The method chunk is presented to a few likely end users of the chunk. Their input will form input for an iteration before the chunk is validated through expert opinion.

The approach is depicted in Figure 2. Boxes with bold text indicate the deliverable of the previous stage. All boxes in one stage function as input for the deliverable outputted by that stage.

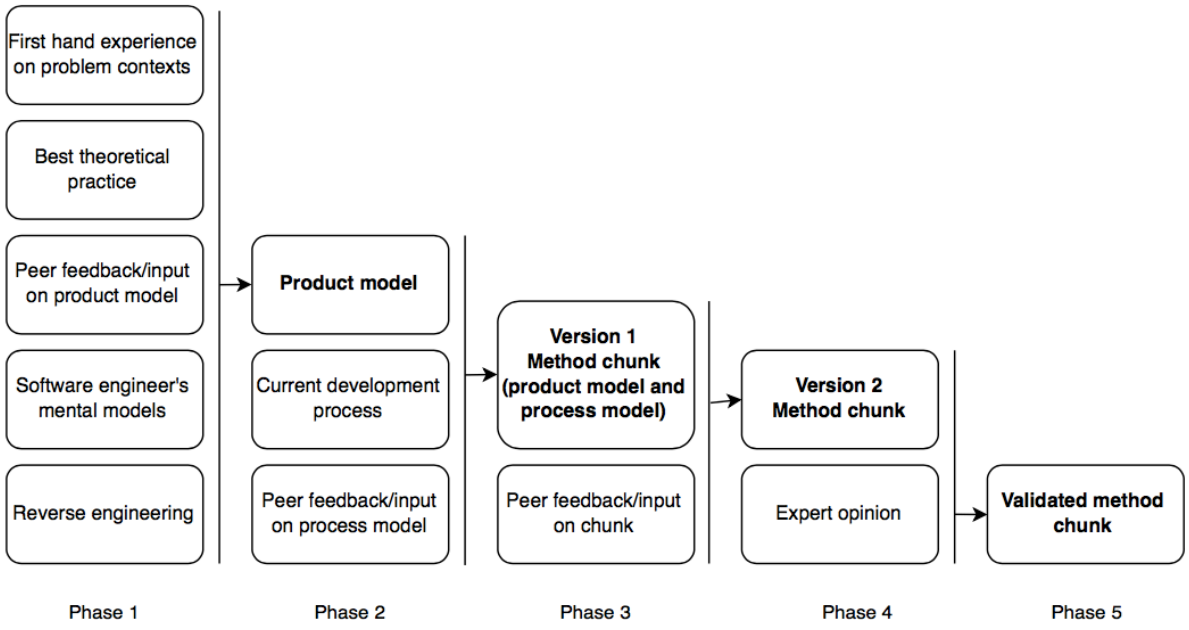

Figure 2 - Approach stages based on deliverables and their input

4 Product Model: a Reference model

4.1 In theory

The product of the method chunk needs to describe the required information about the problem contexts for the development team to develop new solutions or variations on existing solutions. Therefore, the product model only needs to acquire information about the problem context describing that what is not (yet) supported by the system. This chapter will look into reference models to lay the theoretical foundation for a product model that effectively identifies those pieces of information.

4.1.1 Reference models

The products developed by Cofano are in many ways a modern version of an Enterprise Resource Planning (ERP) system for actors in the logistics sector; it is meant to automate the majority of their business processes, while it is being delivered as a service and is connected to many other information systems in the sector (which include other systems of Cofano providing services to parties in the logistics sector) to achieve as much transparency in the logistics chain as possible.

Similar to the 'solution-developing-cycle' of Cofano, the implementation of an ERP system involves a process of finding a fit between existing solutions (the generic packages) and the organisation. If necessary, either the product can be customized for the specific needs of the business or the business processes can be adapted to fit the existing solution (Pajk, Indihar-Štemberger, & Kovačič, 2011; Soffer, Golany, & Dori, 2003). Although the latter is most preferred from Cofano's perspective, it is not always an option.

In this process, reference models (also called universal models, generic models or model patterns) may be used to describe the solutions embodied in the system, which can then be used to analyse customisations needed of the system or changes needed in the processes in the problem context (i.e. adaptations of the customer's practices) (Pajk et al., 2011).

Although there is no consensus about the definition of reference models, Peter Nes (2007) captured their salient aspects. In general, a reference model is *universal* in a certain *domain*, usually with a *recommending character*, such as when the model describes best practices supported by the ERP system. From a reference model, a model can be *derived* that is specific to a certain organisation. Derivatives are made from the possible best practices in a reference model to design a 'to be' situation of an organisation where an ERP system is implemented.

Therefore, as Soffer et al. (2003) put it, creating a reference model is a reverse engineering effort: The starting point of a reference model are the existing solutions imposed by the system and not the customer's business. The reference model contains the practices supported from which a configuration can be made for a particular organisation.

At first, this may seem to be the opposite of what the product is required to describe; namely the system and not the customer, while we need to describe the problem

context. However, the system is developed based on the problem contexts of early adopters. A reference model of TOS is, therefore, a consolidation of those problem contexts that it supports. In other words, *the reference model describes the world according to the system.* It makes all assumptions about the problem context that are part of its design explicit. This means that, in order to identify the required information about the problem context of a customer, we simply have to compare the reference model to the customer's actual problem context and look for differences. Having a model to discuss can prove to be a catalyst of establishing a solution (Svensson & Hvolby, 2012).

The identified differences can either be mitigated by the customer, changing its process to one that is already supported by the system (the recommending character of the reference model), or require the development team to develop a new solution or a variation of an existing solution. After the development of a new solution or variation, these can be added to the reference model for future reference. The process is shown in Figure 3.

In effect, this can be a very efficient way of identifying the required information about the problem context: This approach saves effort in creating a model of the problem context in that it only focuses on the pieces of information needed. The development for a fully detailed model of every problem context would become very labour-intensive. This approach makes full use of the reusability advantage of reference models (Nes, 2007), where the reference model functions as a checklist and its derivatives as checked off checklists. Therefore, the product model is the derivative of the reference model, which describes differences identified.

Overall, the reference model approach described meets the following criteria of the method chunk:

- The required information is efficiently identified by focusing on the differences with the reference model that require documentation for the development team.
- The knowledge about the problem contexts is shared in the organisation in twofold; the reference model describes the supported problem contexts and functions as a central place to document what it supports, and derivatives describe the idiosyncrasies for each customer's problem context.

Therefore, the aim is to develop a reference model such that the creation of derivatives becomes part of the method chunk.

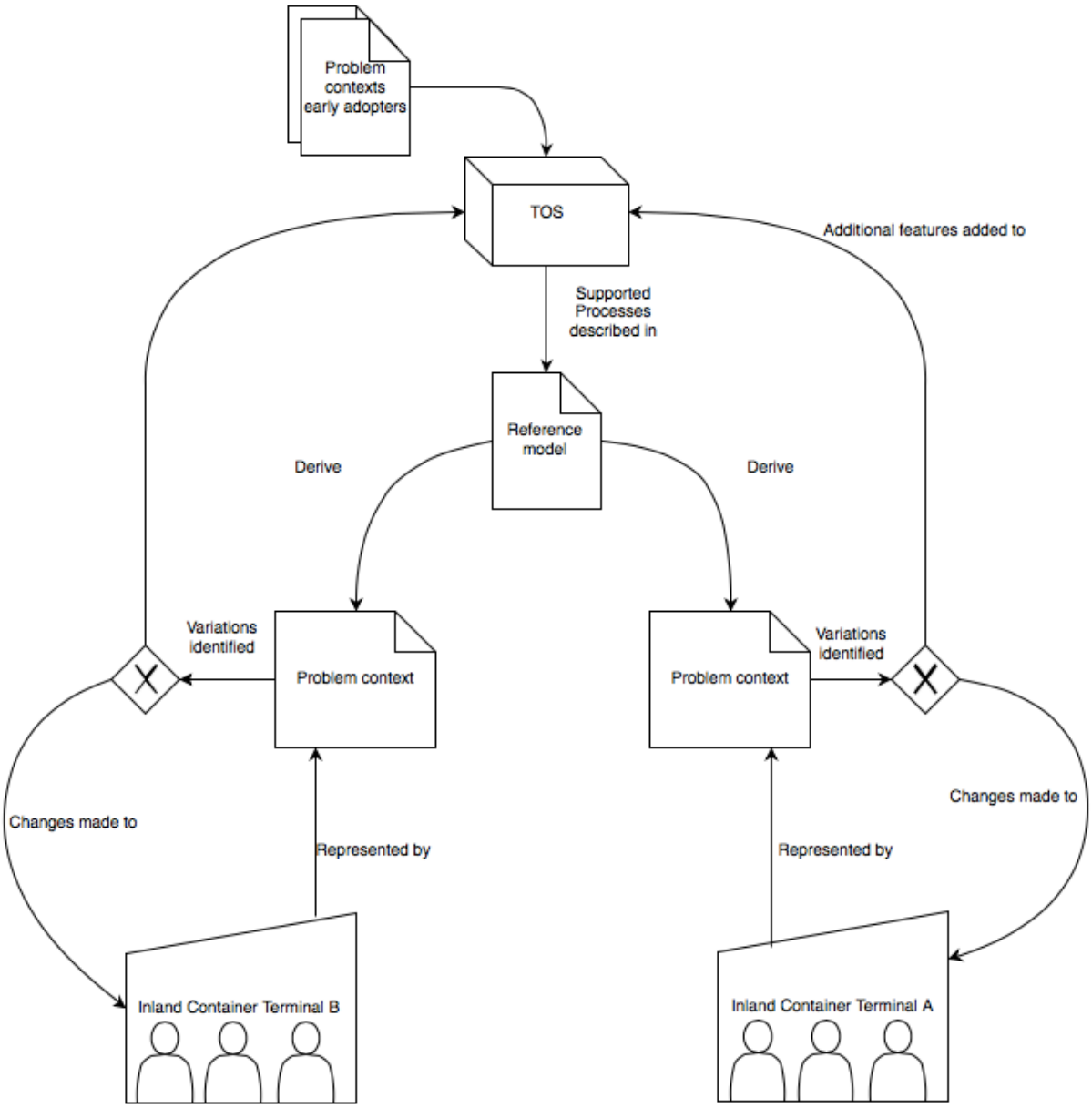

Figure 3 – Using reference models for variation identification

On a side note, Deelstra, Sinnema et al. (2004) identified problems when using reference models for addressing commonality and variability. As software systems grow, the complexity and the amount of possible variation will grow with it, making it unmanageable for the individual. Implicit properties describe the phenomenon of dependencies of possible variations in system functionality supported by the product that are undocumented and either unknown or known only by experts. Fortunately, compared to ERP systems, TOS will not hold that many variations as these are kept to a minimum.

Also note that normally reference models are used to create configurations of ERP systems. The goal is merely to identify the need for new features as the configuring of TOS is nowhere as complex as fully fletched ERP systems from one of the bigger well known vendors.

4.2 Building the reference model's skeleton

The reference model describes the processes supported by TOS, in essence describing the 'ideal' problem context for the system to support. The 'ideal' problem context will become more and more generic as the TOS supports more HCTs; supporting more HCTs implies that the reference model becomes more universal in the domain of HCTs. Hence, we will refer to this 'ideal' problem context as the Generic Hinterland Container Terminal (GHCT).

The development of the high level skeleton takes inspiration from the simple, concise, and building blocks based business model canvas from Osterwalder (Osterwalder & Pigneur, 2009). To achieve a high level building blocks structure for the reference model, a similar approach as Pajk et al. (2011) is adopted. The process of the GHCT at a high level of abstraction will form the main skeleton for the reference model. Each step in that high level process we call a *process domain*. These domains will function as building blocks of the reference model, containing *process subdomains* with detailed processes performed in that domain.

In this section the processes of Generic HCTs are explored as far that is currently supported by TOS (or will be in the near future). This means that, as the reference model is build throughout this section, it will include domains that are not yet supported by the system or that there are domains that are not yet discovered. Unsupported domains are not described in detail. Undiscovered domains and to be explored domains' details are to be added once they are explored.

To build the reference model in this chapter step by step, we will start from the most generic fundamental process of the GHCT and work to the most detailed sub processes, which will include details of data flows and relevant entities and relationships

4.2.1 The core process – the round trip

In its most distilled form, the GHCT arranges transport of shipping containers from or to the customer, via the terminal, to or from a certain other location. Often, that 'other location' is a deep sea terminal, where containers are loaded or unloaded on/from a large sea container ship. Depending on whether the container comes from a foreign country, unloaded on the deep sea terminal, transported to the GHCT and ultimately to the customer or completely the other way around, the transportation is referred to as an import or an export. However, in essence, the process remains the same; an import merely implies the opposite process of an export. In rare cases the destination or origin is not a deep sea terminal; the transportation of a container takes place from or to the customer via the GHCT to or from another location. In general, the transportation from/to a customer from/to another location is called a *round trip.*

4.2.1.1 The foundation

On a high level, the process can be divided into process domains. A process domain is a set of processes, which together realise a step of the high level process. The high level process described so far consists of the following process domains as shown in Figure 4. The Customer process domain consists of all the processes handling customer interaction. The processes in the Transportation Process area together realise the transportation step. The processes in the Terminal Management process area together realise all that is necessary to store a container for some time (examples are handling it

off and on a truck, stacking, connecting cooling containers to a power supply, weighing, etc.)

We illustrate the control flow through these process domains with an example of a round trip process. We will expand on this example in more detail as we add domains and subdomains to the reference model:

The process starts at the customer process domain, where an order at the GHCT for transportation is received and processed. Let's assume that the order is an export booking. The flow of the high level process now shifts to the transportation domain: an empty container is transported from a depot of empty containers to the terminal. The flow has now shifted to the Terminal Management process domain: the container is stored (albeit possibly for a very short amount of time) on the terminal.

Once the container is loaded on a truck for transportation to the customer, the flow shifts back to the Transportation process domain. The GHCT transports an empty container to the customer's specified location, loads it with goods and transports it back to the GHCT. Here, the flow has shifted back to the Terminal Management domain. The GHCT stores the container until it is transported to the deep sea terminal for export.

Once the terminal has loaded it on a truck, a barge or a train for transportation to the deep sea terminal, the flow shifts back to the transportation domain. Once the container is delivered at the deep sea terminal and the round trip has completed, the flow shifts back to the Customer process domain, where the invoice for the completed round trip is created and sent to the customer.

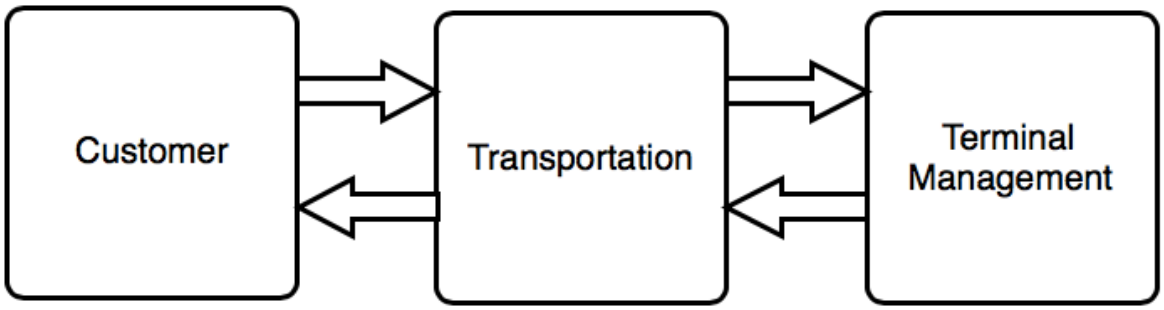

Figure 4 – Most elementary process domains for the reference model

Note that the process domains describe processes performed by the GHCT. Processes performed by other parties are scoped out since these are not part of the scope of TOS.

4.2.1.2 Adding subdomains

The process domains in Figure 4 can, at a lower level, be divided into smaller subdomains, which is also a collection of processes to achieve a certain step within the process domain it is in. An extension of Figure 4 with subdomains is shown in Figure 5.

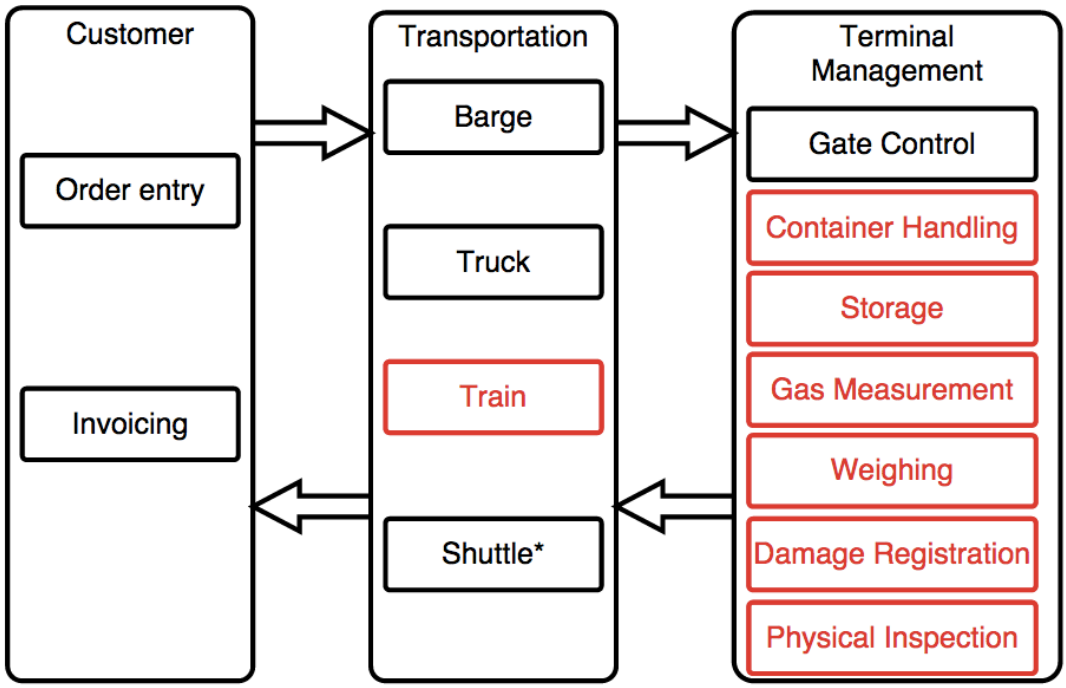

Figure 5 – The most elementary process domains extended with process subdomains

The process domain of the customer, as we have seen in the round trip example, has two process subdomains: That of Order Entry and that of Invoicing. The Transportation process domain has a subdomain for each modality: Truck, Barge, Train and Shuttle[1]. As soon as the container enters or leaves the terminal, the arrival or departure needs to be registered for the administration of the inventory of stored containers on the terminal. This process subdomain is called Gate Control. The handling of the containers by the reach truck (i.e. stacking, loading, unloading, registration of its location) is all part of the Container Handling process subdomain. Furthermore are there some miscellaneous activities that are performed on the terminal during the stay of a container. These are all separate process subdomains: Storage (the stay of the container itself), Damage Registration, Physical Inspection, Gas Measurement and Weighing.

4.2.1.3 Process (sub)domains as building blocks

The process (sub)domains function as building blocks, such that derivatives can be created from the reference model by assembly (Nes, 2007). Building process (sub) domains can be used in, or left out of, a derivative of the reference model, depending on whether the problem context has these process domains. For example, most HCTs do not have rail as a modality option for transportation. In such case, the subdomain Train could be left out entirely in the derivative.

In addition, it will be easy for the analyst to add completely new process domains in the derivative if these are non-existent in the reference model. This is the case when an unexplored process area is discovered and needs to be supported by TOS. For example, say thus far we have never heard of the possibility that a container terminal could be a depot for empty containers. We could simply add this process domain to our derived

[1] A shuttle is a truck meant for short distances only and has some restrictions. It therefore has a separate process area

model and once this process domain is supported by TOS also added to the reference model.

4.2.1.4 Adding the Depot process Domain

When a customer makes a booking at a deep sea carrier to export goods, it has to load their freight into a container of that particular carrier. Normally this implies that the GHCT has to retrieve a container from a depot holding containers for that specific carrier. However, being a depot holds that the GHCT is allowed to hold empty equipment for a certain amount of time before it has to be returned to the depot of the carrier, which allows reuse of empty containers in that time period. Therefore, the Depot process domain of the GHCT has a process subdomain for this reuse process.

In addition, the GHCT has to report so called daily moves to the carrier such that the carrier knows exactly where their equipment is. Therefore, the depot has the reporting process subdomain.

The model with the depot process and its constituting subdomains is shown in Figure 6. Now, when a customer books for an export, instead of transporting first, the control flow starts at the Depot domain looking for a reusable container and if that is the case, the flow continues at the Terminal Management domain. This would imply, however, that an arrow skips from the customer straight to the depot domain to indicate the control flow of this case. However, for simplicity sake, this has been left out.

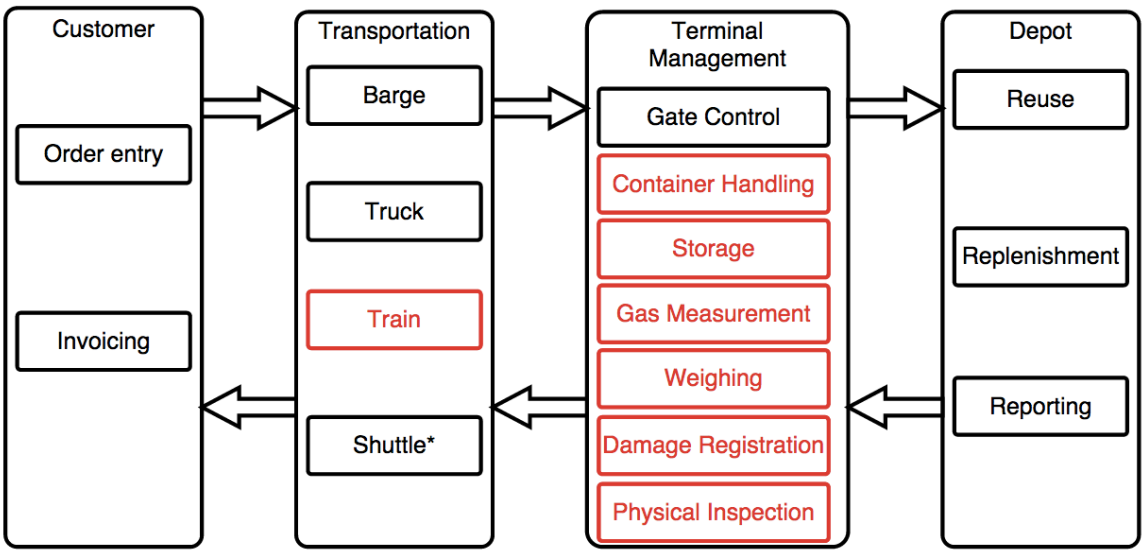

Figure 6 - High level view of reference model with depot process domain

4.2.1.5 Adding the Support process Domain

In terms of porters value chain (Porter, 2008), so far the domains in the reference model cover the primary activities of the GHCT. The GHCT too, has some supporting activities. So far, only the need for visibility is identified. First, there is a need for aggregated data reporting (such as a dashboard). Second, tracking and tracing is not part of the primary process but is necessary for monitoring as well. Therefore, we add these two subdomains to the model, as shown in Figure 7.

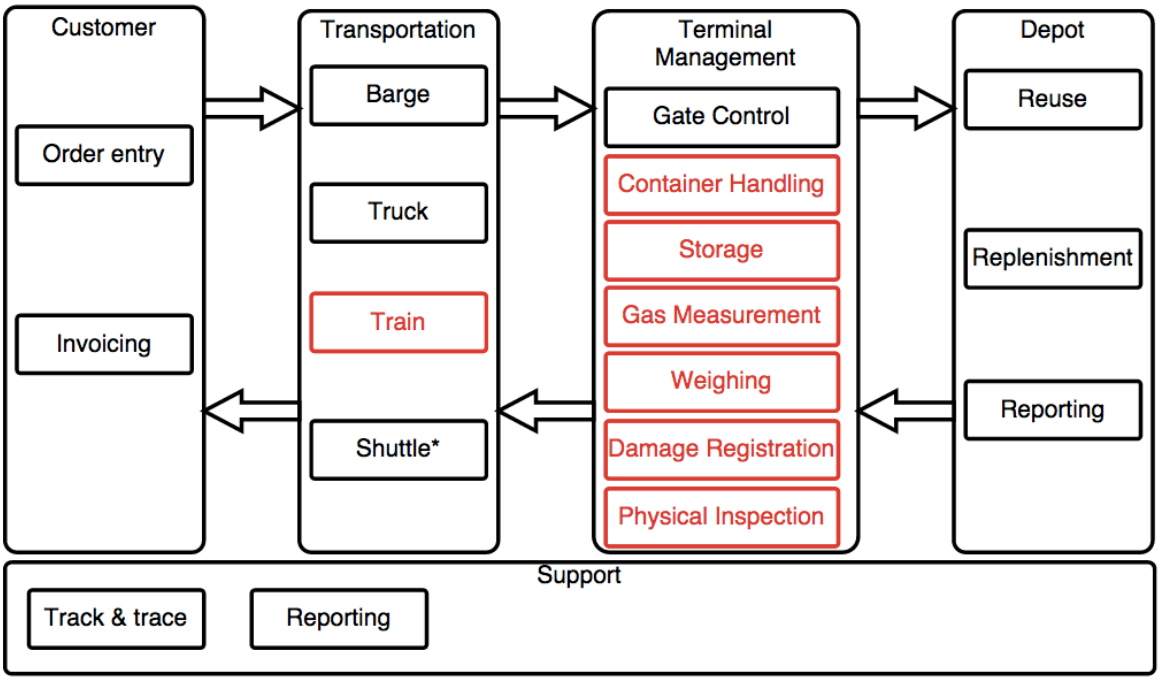

Figure 7 - High level view of the complete reference model

Note however, that the focus of TOS lies at supporting the primary functions at the moment and tracking and tracing (one of its selling points). Therefore, there are currently no known processes in detail of reporting and it is expected that other subdomains will be added to the support process domain.

4.3 Adding detailed domain descriptions to the process subdomains

Now that we have a high level skeleton, the details for each process subdomain can be added. Note, however, that some domains are identified but not yet supported or even analysed for its process (domain in red in the skeleton models). These are therefore not covered by the reference model and thus not described in detail in this section.

For each (sub)domain, one or more of the following aspects are covered in detail:

- The **process** (by means of a process model and/or a short description)
- The **data flows** related to that process (possibly supported with pictures from real life documents).
- An **Entity Relationship** Diagram (ERD). The ERD in the reference model will simply be that what the system currently uses to describe the domain (abstracting away implementation details), since this will help the analyst spot whether it can describe the situation at any future customer (hence spotting differences from prior adopters that are implicitly consolidated by the ERD). In other words, the ERD diagrams are almost the same to that implemented in the system, with very few implementation details omitted (i.e. internal identifiers) as we want the reference model to reflect the assumptions made by the TOS.
- **State** diagrams. Sometimes the focal point of the domain is the **events**, and a process model will not suffice. (For example, the Gate Control subdomain centres around the registration of so called Gate In and Gate Out events at different locations.)

The Business Process Modelling Notation (BPMN) (Weske, 2007) is used for the process and data flows, as it supports to model both in one diagram. Data flow documents in the diagram may refer to an example document from the domain or further documentation. The ERD diagrams follow a notation similar to that proposed by Wieringa (2003) and supplemented with a dictionary defining non self-explanatory entities and attributes. The state transitions are described in Mealy diagrams (Wieringa, 2003).

4.3.1 Order Entry process subdomain

In essence, Order Entry is simply the task of registering the orders of a customer for the transportation of containers, such that other processes can start working on the delivery of that order (i.e. making the round trip happen). There exists a lexical entity (Wieringa, 2003) for each individual container that needs to be shipped, which we call ContainerBooking.

There are, however, a few optional paths of this process that needs to be considered. A ContainerBooking may be of three different types; import, export or export of which the Carrierbooking (the booking details at the deep sea carrier by the customer) is unknown. Depending on these three types, the information related to the ContainerBooking may be entered at different times. These three different flows are depicted in the process diagram in Figure 9.

After the initial insertion of the order details, the Actions to be executed to complete the order are determined. The Actions represent the activities that have to take place to complete the order. (See the description of the Transportation process domain in general, paragraph 4.3.2, for a more elaborate explanation on Actions). Thereafter, details can be added or changed, upon which the actions are determined again. This cycle may repeat until all the Actions have been completed.

The focal point of the overall process of the GHCT is the ContainerBooking. As such, the ContainerBooking goes through several states in its life cycle. The state diagram is presented in Figure 8. (Note that it will help the reader to comprehend the description that follows next, by reading the descriptions in the Transportation process domain, paragraph 4.3.2, and the Invoicing subdomain, paragraph 4.3.10, first.)

Once the ContainerBooking is created, it is in the Draft state. As soon as the chain of Actions is created, the ContainerBooking is in the Ready state: Ready to have its TransportActions planned. If one or more TransportActions are planned, the Unplanned state is entered. As soon as all TransportActions are planned, the Planned state is entered. Once the Actions in the Action chain are completed, the ContainerBooking transitions to the Finished state. From there, ContainerInvoiceItemGroups can be created based on the ContainerBooking. Once all the InvoiceContainerGroups have been placed on an invoice, the ContainerBooking enters the Invoiced state.

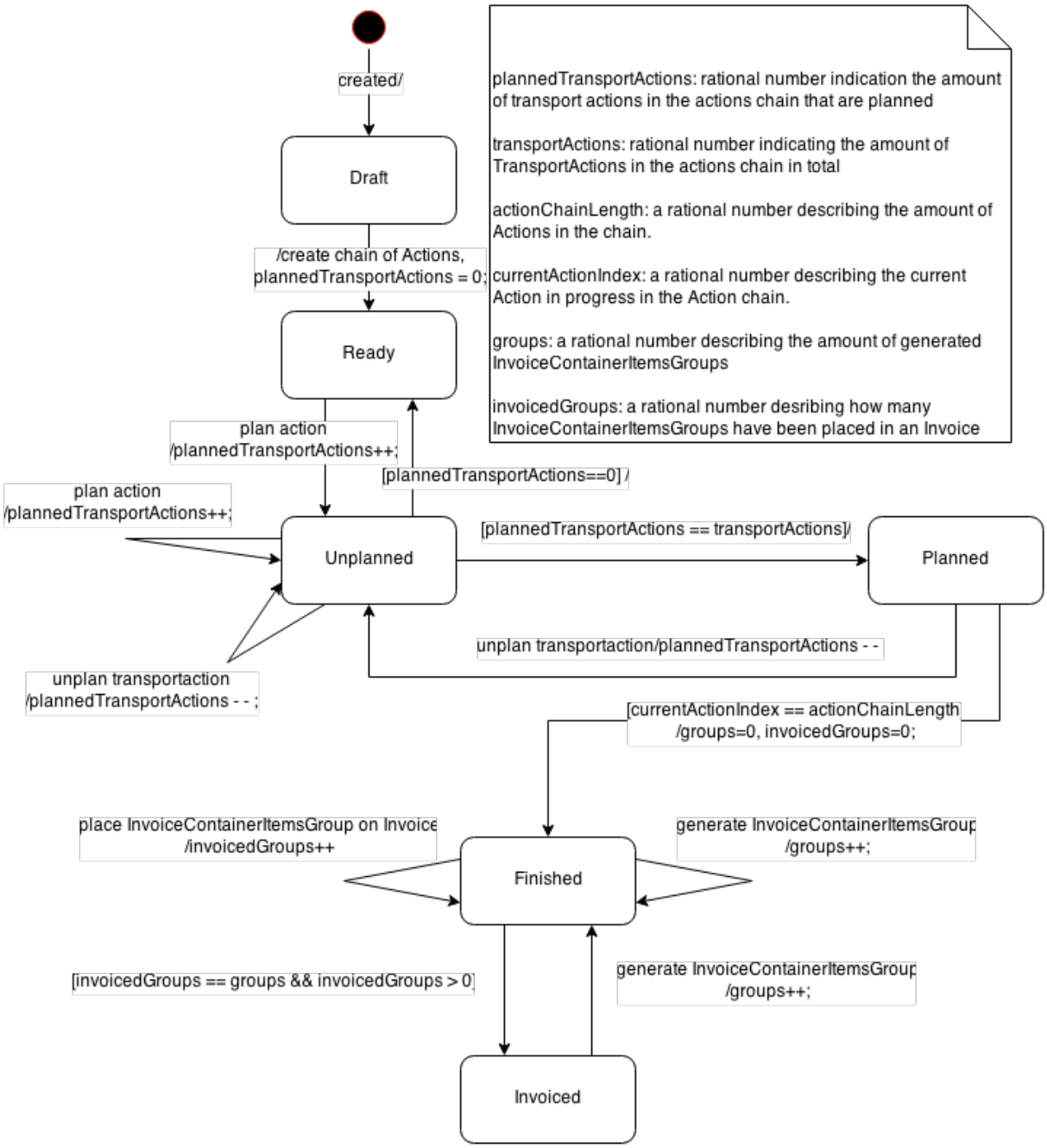

Figure 8 - State diagram of a ContainerBooking's lifecycl

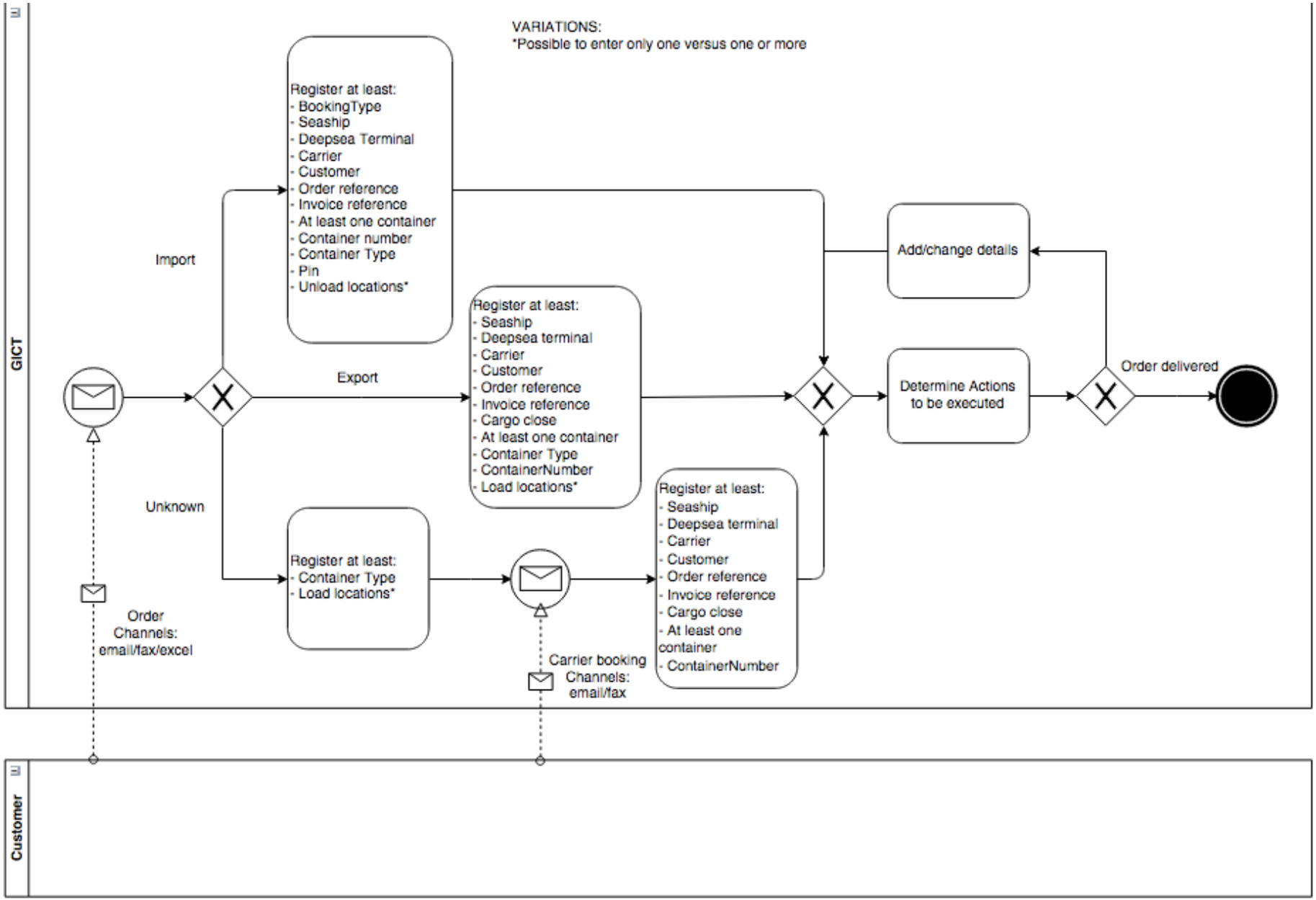

Figure 9 - Order Entry process diagram

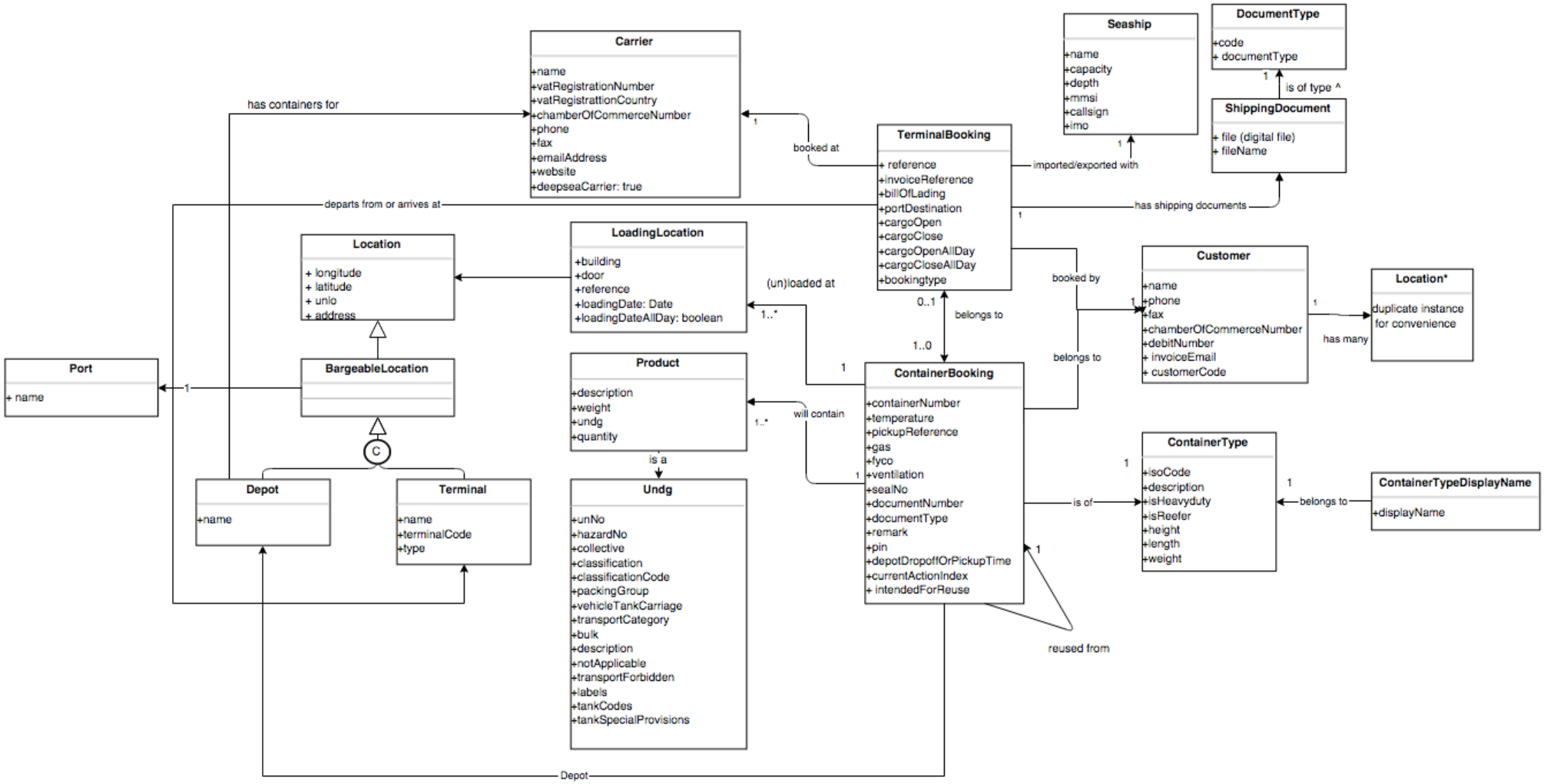

Figure 10 - ERD of Order Entry subdomain

The data related to the order is shown in the ERD diagram of the Order Entry process subdomain shown in Figure 10. A short dictionary (Wieringa, 2003) is presented here for the non-self-explanatory elements in the ERD:

4.3.1.1 Entities

ContainerBooking. Entity type name. The agreement between the customer and the GHCT to transport one container.

TerminalBooking. Entity type name. The booking a customer has made at a deep sea carrier to transport the container over international waters.

Carrier. Entity type name. The carrier that provides the international transport.

ContainerType. Entity type name. The type of the container, indicating dimensions and certain features.

ContainerTypeDisplayName. Entity type name. The name a user (left out of scope) has given to a ContainerType. (Most users will not use the ISO standard and prefer to use their own naming.)

Depot. Entity type name. A Location where equipment for certain carriers are stored.

Terminal. Entity type name. A Location where there is either an HCT or a deep sea terminal or a combination of both.

Undg. Entity type name. A hazardous material a Product could be, as defined by the United Nations. Its characteristics are used to determine what regulations apply to handling the Product.

ShippingDocument. Entity type name. A digital document accompanying the TerminalBooking.

DocumentType. Entity type name. The type of a ShippingDocument.

4.3.1.2 Attributes

terminalCode (t:Terminal). Attribute. A shorthand identifying a terminal used in the industry.

type (t:Terminal). Attribute. Type of the terminal, either hinterland, deep sea or both.

unlo (l:Location). Attribute. United Nations Code for Trade and Transport Locations that assigns codes to locations used in trade and transport, such as deep sea terminals, hinterland terminals and airports.

reference(ll:LoadingLocation). Attribute. The reference assigned to the (un)loading of the container by the customer, such that the customer knows what to (un)load in/from the container upon delivery.

undg (p:Product). Attribute. Is a four-digit number created by the United Nations Development Group that identifies hazardous substances, and articles (such as

explosives, flammable liquids, toxic substances, etc.). When the undg is present, it implies that the product is a hazardous material.

quantity (p:Product). Attribute. The amount of units the hazardous material is packaged. When quantity is present, it implies that the hazardous material is of limited quantity, meaning that it is packaged in such a small amount per packaging, different regulations apply.

gas (c:ContainerBooking). Attribute. A Boolean that indicates that the container will contain gas and a gas measurement is required while the container is on the GHCT before further shipment.

fyco (c:ContainerBooking). Attribute. A Boolean that indicates that the container requires physical inspection by a customs official while on the GHCT before further transportation.

ventilation (c:ContainerBooking). Attribute. A Boolean that indicates that the container will require to be ventilated while on the GHCT before further shipment.

sealNo (c:ContainerBooking). Attribute. The number of the security seal that seals the doors of the loaded container. Security seals are mechanisms used to seal shipping containers in a way that provides tamper evidence.

pin (c:ContainerBooking). Attribute. A container at the deep sea terminal is released by means of a pin code.

depotDropoffOrPickupTime (c:ContainerBooking). Attribute. Date and time indicating when the container should be dropped off/picked up at the depot related to c.

currentActionIndex (c:ContainerBooking). Attribute. Index of the first Action in the chain of Actions that has not yet been completed. In other words, the index indicating the Action that is currently in execution. (See ERD and dictionary of the Transportation domain for an explanation on Actions and the Action chain.)

intendedForReuse (c:ContainerBooking). Attribute. Boolean value indicating whether the container is allowed to be reused after import (i.e. the GHCT is allowed to detain the container for an agreed amount of days before returning it to the depot in which time it should be reused for another booking) or whether a container may be reused in case of an export booking.

isoCode (ct: containerType). Attribute. Is an ISO 6346 code, that is an international standard for the unique identification and marking of specifications of containers.

billOfLading (t:TerminalBooking). Attribute. A string of numbers and digits that refer to a certain booking at a carrier.

reference (t:TerminalBooking). Attribute. The internal reference of the customer to t.

invoiceReference (t:TerminalBooking). Attribute. Reference to t that the customer wishes the invoice of the GHCT uses to refer to t.

CargoOpen (t:TerminalBooking). Attribute. Start time and date of the time window within which the container needs to be handed in or picked up at the deep sea terminal.

CargoClose(t:TerminalBooking). Attribute. End time and date of the time window within which the container needs to be handed in or picked up at the deep sea terminal.

bookingType (t:TerminalBooking). Attribute. Type of the booking made at the carrier, either importing a container or exporting a container overseas.

mmsi (ss:Seaship). Attribute. A Maritime Mobile Service Identity (MMSI) is a series of nine digits which are sent in digital form over a radio frequency channel in order to uniquely identify ship radio stations.

imo (ss:Seaship). Attribute. International Maritime Organization (IMO) numbers are a unique reference for ships and for registered ship owners and management companies.

debitNumber (cs: Customer). Attribute. Internal number identifying a debtor in the accounting system.

customerCode (cs: Customer). Attribute. Shorthand for the customer based on its name.

unNo (un:Undg). Attribute. The unique identification number of the material as defined by the United Nations.

hazardNo (un:Undg). Attribute. An identification code indicating the hazards of the material (i.e. like poisonous or flammable).

collective (un:Undg). Attribute. A true or false value indicating if the category entails a group of materials. When true, the chemical or technical name of the transported material has to be added in addition to the information in the Undg entity.

classification (un:Undg). Attribute. The ADR class the material is placed in (i.e. 1 for explosive materials, 2 for gasses, 3 for flammable materials, 7 for radioactive, etc.)

classificationCode (un:Undg). Attribute. The subclass of the ADR classification code.

packingGroup (un:Undg). Attribute. Its values either being I, II or III, the packingGroup indicates certain quality requirements to the material's packaging.

vehicleTankCarriage (un:Undg). Attribute. A code indicating the type of vehicle required to transport the material.

transportCategory (un:Undg). Attribute. Also known as 'tunnel code'. A code indicating through which tunnels the material is not allowed to be transported.

bulk (un:Undg). Attribute. A special 'VV' code referencing to certain regulations when transporting the material in bulk.

notApplicable (un:Undg). This attribute's meaning is unknown, it was part of the UN hazardous materials table adopted for the Undg entity.

labels (un:Undg). Attribute. Numbers referring to certain models of labels required on the packaging for transport.

tankCodes (un:Undg). Attribute. Alphanumerical codes referring to the least stringent regulations regarding the tanks used for transport.

tankSpecialProvisions (un:Undg). Attribute. Alphanumerical codes referring to additional regulations for the tanks used for transport.

code (dt:DocumentType). Attribute. A shorthand for the DocumentType.

documentType (dt:DocumentType). Attribute. A description of the DocumentType.

4.3.2 Transportation process domain in general

The transportation process domain consists of subdomains of different modalities. However, every transportation subdomain process consists of roughly two stages. First the transportation needs to be planned, then executed.

As a prerequisite, each transportation process can only start at the event that the end time of the delivery window and the location is known (one of the details filled in at the Order Entry process subdomain). Once a booking is registered and these required details are registered, it is assumed that the transport from/to the GHCT from/to the other location that is not the customer's site will be executed by barge, which will start the process in the Barge subdomain. Just like every other transportation subdomain process, the barge subdomain process also allows to select a different modality, after which the subdomain process of that domain starts.

In addition we defined one ERD for transportation domain as a whole, as the entities are cohesive and are better understood together than apart. The ERD is shown in Figure 11. The transportation of containers is described as a *chain of Actions*. Therefore the Containerbooking entity has a list of Actions to be performed and an index of the current Action indicating its state of where it is in the process. How a ContainerBooking transitions between these states is described in the Gate Control subdomain.

A short dictionary (Wieringa, 2003) is presented here for the non-self-explanatory elements in the ERD:

4.3.2.1 Entities

ActionCollection. Entity type name. The chain of actions that have to be executed to complete the ContainerBooking or DepotBooking.

TransportAction. Entity type name. The act of transporting a container from one Location to another.

StopoverAction. Entity type name. The Action of (shortly) storing the container and possibly perform some Tasks. For example, the stop a container makes on the GHCT. Then, the GHCT may perform Tasks, such as the VentilationTask and the StorageTask. (See TerminalManagement ERD for more details.)

PreAction. Entity type name. The last Action performed in the logistics chain outside and before the GHCT's scope.

PostAction. Entity type name. The first Action performed in the logistics chain outside and after the GHCT's scope.

Trip. Entity type name. A Trip is an execution of a Line. Similar to a bus line, a certain line planned for, and performed on, a certain time and date.

Lading. Entity type name. An object that is transported by the TransportAction. Its attributes are based on the ContainerBooking or DepotBooking the TransportAction's ActionCollection belongs to.

4.3.2.2 Attributes

currentStepIndex (ac: ActionCollection). Attribute. Indicating which action in the ac is currently in progress.

sta (ta:TransportAction). Attribute. Time of arrival of ta.

std (ta:TransportAction). Attribute. Time of departure of ta.

window (sa: StopoverAction). Attribute. Window of time indicating how long the container will be at the location of the sa.

gateIn (ta:TransportAction). Attribute. The actual time the container is unloaded from the ship onto the GHCT.

gateOut (ta:TransportAction). Attribute. The actual time the container is loaded onto the ship from the GHCT.

locked (tp:Trip). Attribute. A Boolean value indicating whether the planned BargeActions to be transported by tp is changeable or not. If locked, the planning can no longer change.

mmsi (s:Ship). Attribute. Same as mmsi of SeaShip (See Order Entry sub domain dictionary).

eni (s:Ship). Attribute. An ENI number (European Number of Identification or European Vessel Identification Number) is a unique, eight-digit registered identification number for ships capable of navigating on hinterland European waters.

capacity (s:Ship). Attribute. Capacity of the loading bay of s, expressed in TUE[2].

shuttle (tk:truck). Attribute. Boolean indicating whether the truck is a shuttle with limitations or a normal truck.

4.3.3 Truck process subdomain

One of two cases starts the Truck process. The first is the transportation from and to the customer's site. It is assumed that the transportation of containers for the (un)loading at the customer's site is done by truck. That requires two trips by truck to be planned, to and from the customer to bring and retrieve the container. Second, the main transport could be done by truck when barging does not meet the requirements of the planner. In that case the planning stage of the barge process changes the modality of that transport to be done by truck.

In theory, it is also possible to retrieve or bring empty containers from or to depots. In practice, however, this is almost always done by barge.

The planning stage is a continuous process of planning and checking changing factors, such as pickup windows (check ETD ok), delivery window (check ETA ok), the presence of shipping documents and whether the container is at the pickup location. Based on these changing factors, the planning is revisited, until it is time to actually execute the transportation. In case of a main transport of an export round trip, the container has to be 'announced' at the harbour 12 hours before the ETA at the deep sea terminal.

When starting the transport at the GHCT, the CMR[3] waybill and the transport order is printed for the trucker. The CMR is given to the trucker to be signed by the recipient, or, vice versa; the GHCT must sign the CMR when receiving a container from a trucker. The CMR documents must be hardcopies by law. An example CMR is shown in Appendix B.

The transport order is a note with details about the transportation itself (i.e. times and locations) but in particular the customer's reference. The reference is given to the customer upon delivery of a container such that the customer knows from its own administration what to do with it.

The process is shown in Figure 12.

[2] Twenty-foot Equivalent Unit (TEU) is an inexact unit of cargo capacity often used to describe the capacity of container ships and container terminals based on the volume of one 20-foot-long (6.1 m) standard sized intermodal container.
[3] The United Nations Convention on the Contract for the International Carriage of Goods by Road

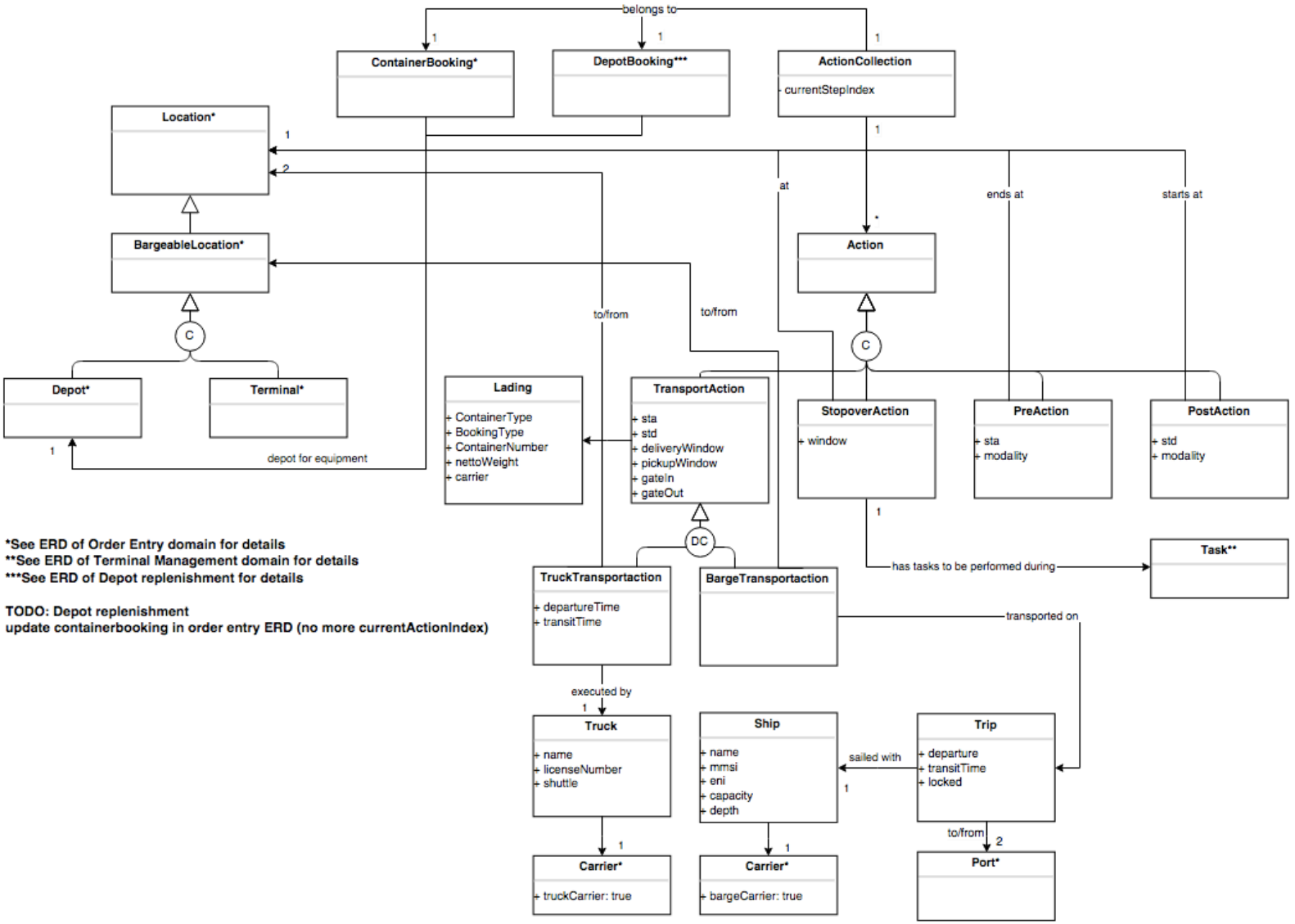

Figure 11 - ERD of the Transportation domain

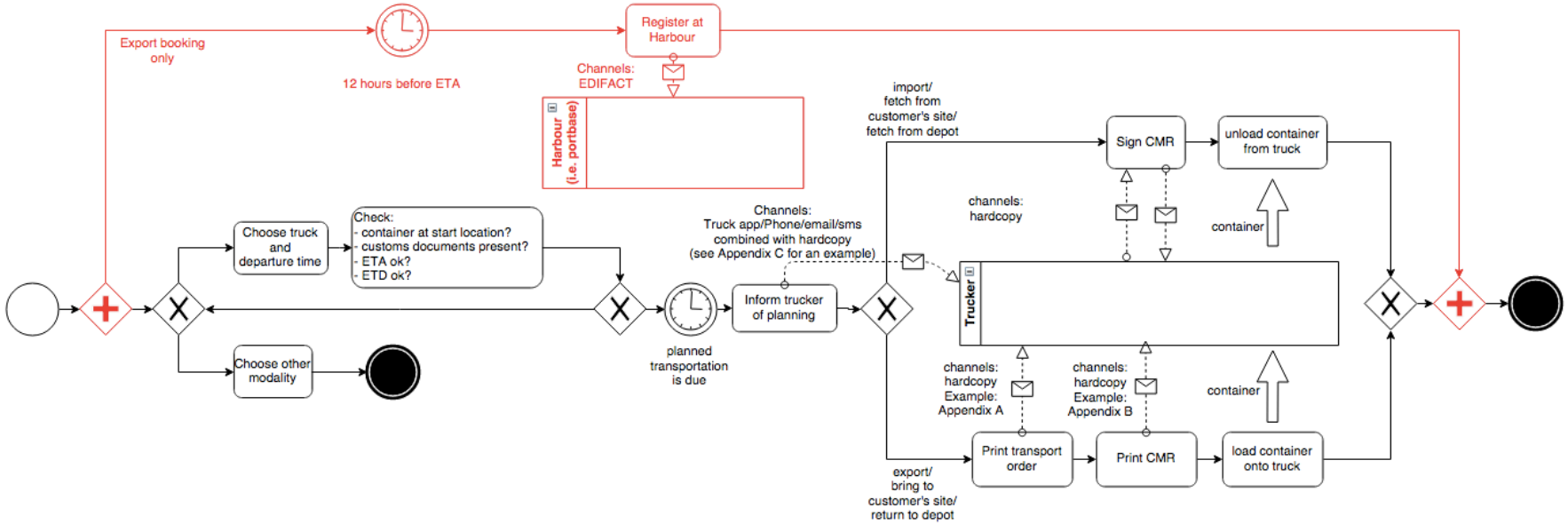

Figure 12 - Truck transportation process

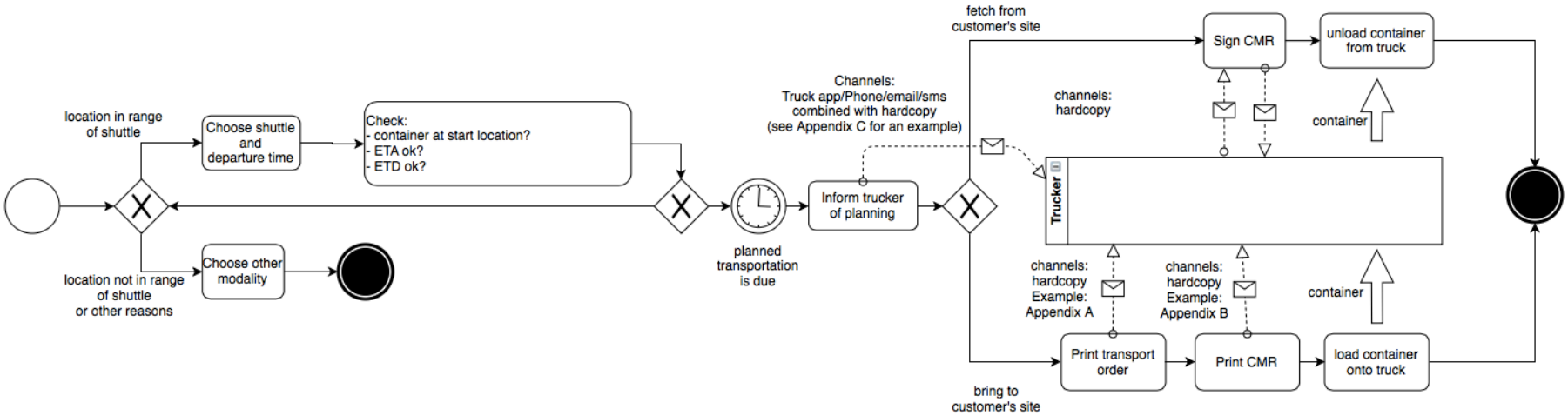

Figure 13 - Shuttle transportation process

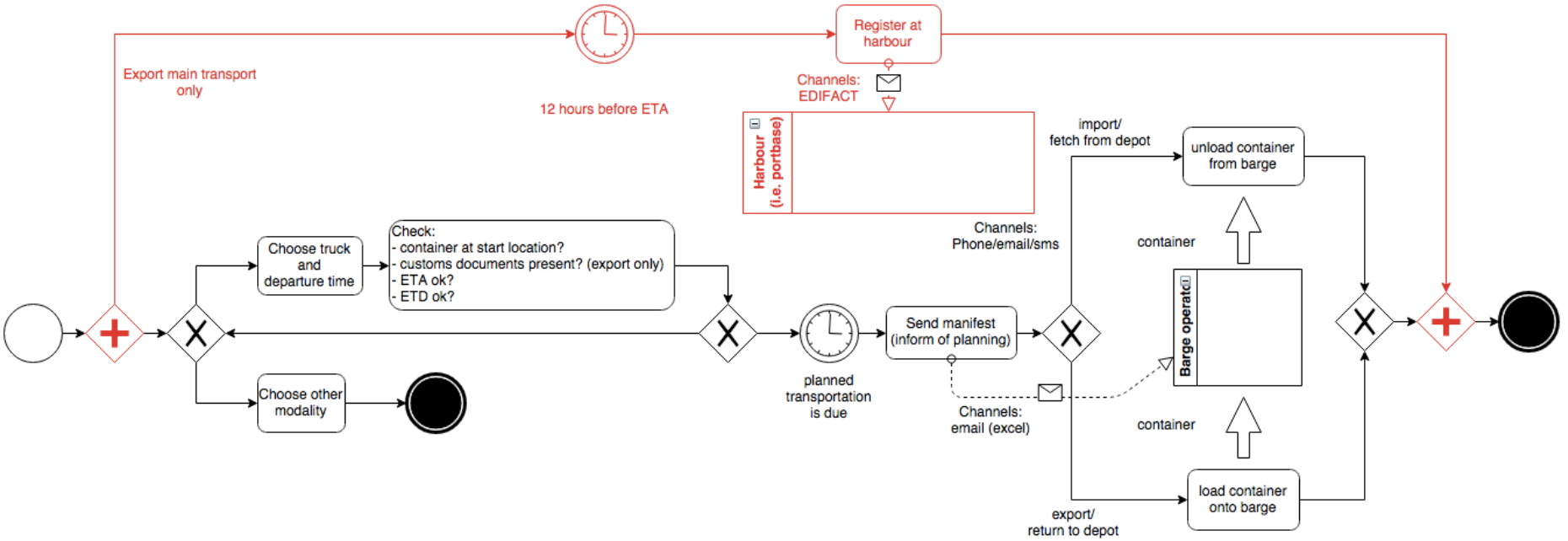

Figure 14 – Barge transportation process

4.3.4 Shuttle process subdomain

The Shuttle process is nearly the same as the Truck process, with the exception that it cannot be used for the main transport. Therefore, the process can only be started for loading or unloading containers at the customer's site, given it is in the limited range of the shuttle (may differ per terminal). The process does not include the registration of a container delivery at the harbour as it is only for main transports of export round trips to the deep sea terminal. Other jobs done by shuttle are executed ad hoc and not (yet) inside the subject domain of TOS. The process is shown in Figure 13.

4.3.5 Barge process subdomain

The barging process is similar to the trucking process with the exception that it is only able to perform main transports of round trips or retrieve/return containers from/to a depot. Another difference is that the paperwork for the barge is not provided by GHCT, but by the operator. The GHCT hands over the planning by means of a manifest in advance, but unlike the truck process, no (hardcopy) documents have to be handed over. The process is shown in Figure 14.

4.3.6 Gate Control subdomain

Gate Control is the registration of Gate In and Gate Out events. A Gate In event is when a container enters a location and Gate Out when it leaves the location. For example when a barge moors to the shore of the GHCT, than it is a Gate In event for each of the containers it unloads onto the GHCT.

The registration of the Gate In and Gate Out events has two purposes. Firstly, it is used to register where the containers are in the main process. For example, after a Gate In event, we know that the container in question is on the GHCT's terrain and the next action for the process can be executed (for example another transport of some task on the terminal). In other words, the currentStepIndex attribute of the ActionCollection entity is changed to reflect the new state in the chain of Actions for a DepotBooking or ContainerBooking.

Secondly, this information is also used to report the so called 'daily moves' to the carrier in case the GHCT is also a Depot (see the Reporting process subdomain, paragraph 4.3.9).

As the focal point of Gate Control centres on events (Gate In, Gate Out) and states of where in the main process the container is, this subdomain is described by means of a state diagram of the ContainerBooking as shown in Figure 15. The states correspond with the Actions in the ActionCollection with the exception of Reuse. (Reuse will be explained in the Reuse subdomain in the Depot domain).

As the model does not completely adhere to a standard, since it is hard to describe certain transitions otherwise, we will shortly walk through the diagram:

Once the booking is created, the currentActionIndex is 0. Then, there are two options:

- If the ContainerBooking is meant to import goods, the Action Chain starts with a PreAction of the deep sea carrier transporting it to the deep sea terminal. Once it arrives, that will be a Gate In event at the deep sea terminal's location 'l': the Action Chain's current index is updated and the ContainerBooking is now at the StopOverAction state at the deep sea terminal.

- If the ContainerBooking is meant to export goods, the Action Chain can start at the Depot where the empty container is to be retrieved and therefore transitions to the StopOverAction state.

After the necessary transitions between the StopoverAction and the TransportAction states, the chain of Actions can end in three states (note that the entity is never destroyed, it just reaches the end of the chain):

- When the export booking is at the deep sea terminal awaiting further transport by the deep sea carrier, which is called the PostAction state.
- When the empty container has been returned to the Depot, ending in the StopOverAction state.
- When the empty container is reused, either still being at the previous customer's unloading location or at the GHCT itself (hence the guard that the Actions chain current state has to be either 2 transitions away from reaching the end or 4), the ContainerBooking may end up in the Reuse state.

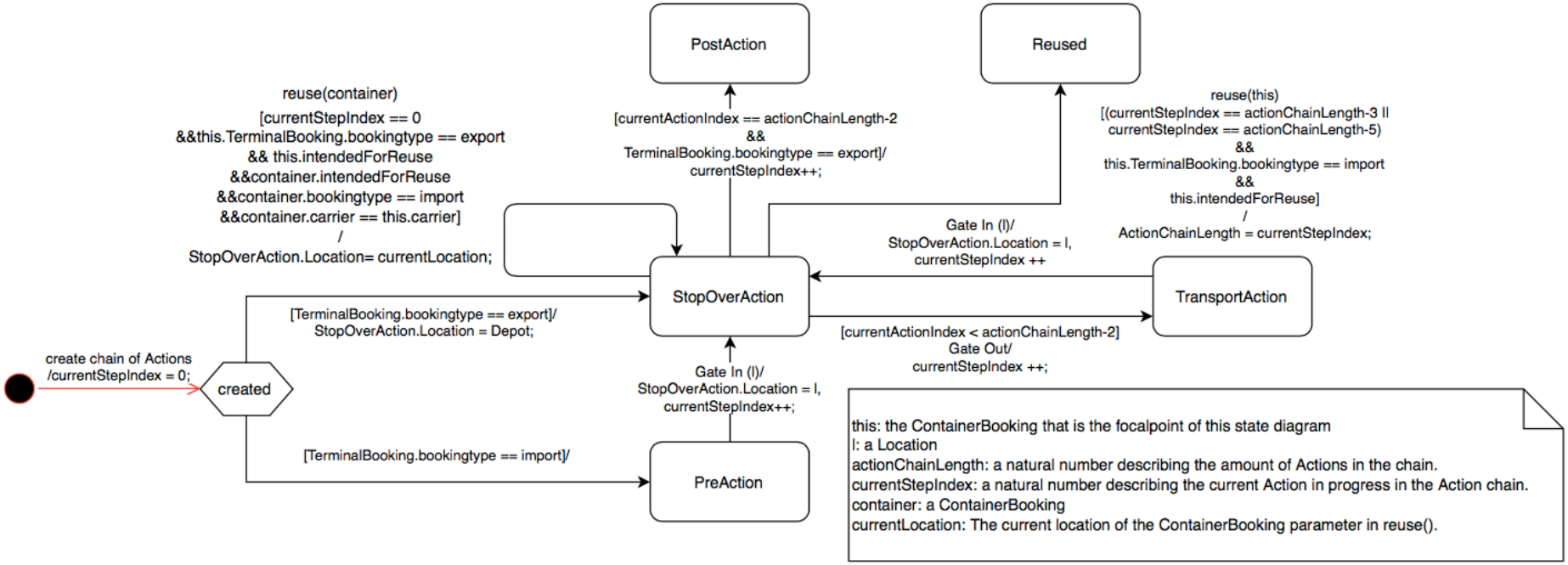

Figure 15 - Mealy diagram Gate Control subdomain

4.3.7 Task subdomains

There are several tasks that may be performed by the GHCT while the container is on the terrain awaiting further transport. So far, only a few have been identified for which there is minimum support by the TOS; it supports merely the registration of the order at the order entry, such that it is known that these tasks have to be performed during the StopoverAction at the GHCT. The process of executing the tasks itself are not (yet) supported by the TOS. They are, however, already part of the Entities and Relationships of the system:

The tasks have to be performed as part of the StopoverAction at the GHCT, which is one of the Actions in the chain of Actions to be performed for the order to be completed successfully. For these subdomains taken together, we present the ERD diagram of this relationship to the Action chain in Figure 16. The meaning of each task corresponds with the meaning of the attributes in ContainerBooking that have the same name. At the time of writing, the TOS currently has no functionality yet depending on these entities. Order Entry is dependent on the attributes in ContainerBooking for registering the order of tasks, not the Task entities presented here.

Once functional support has been added, the supported process is made explicit and can be modelled for the Task subdomains.

4.3.8 Track & Trace subdomain

Currently, the system supports the lookup of two things:

1. The current state of the ContainerBooking in terms of its state in the Actions chain described in the Gate Control.
2. The current position of a vehicle in transport (i.e. the current position of a specific truck or barge)

There is no specific process to be defined that is supported by this functionality. It is an analytical tool that gives insight into the process for many different purposes (even purposes that are yet to be identified.)

4.3.9 Reporting (Support) subdomain

Currently there is just one report known to the system, which is the so called Environment List ('Milieulijst' in Dutch). The GHCT handling hazardous materials is required by law to present a list of all hazardous materials currently present on the terminal. An example is present in Appendix C.

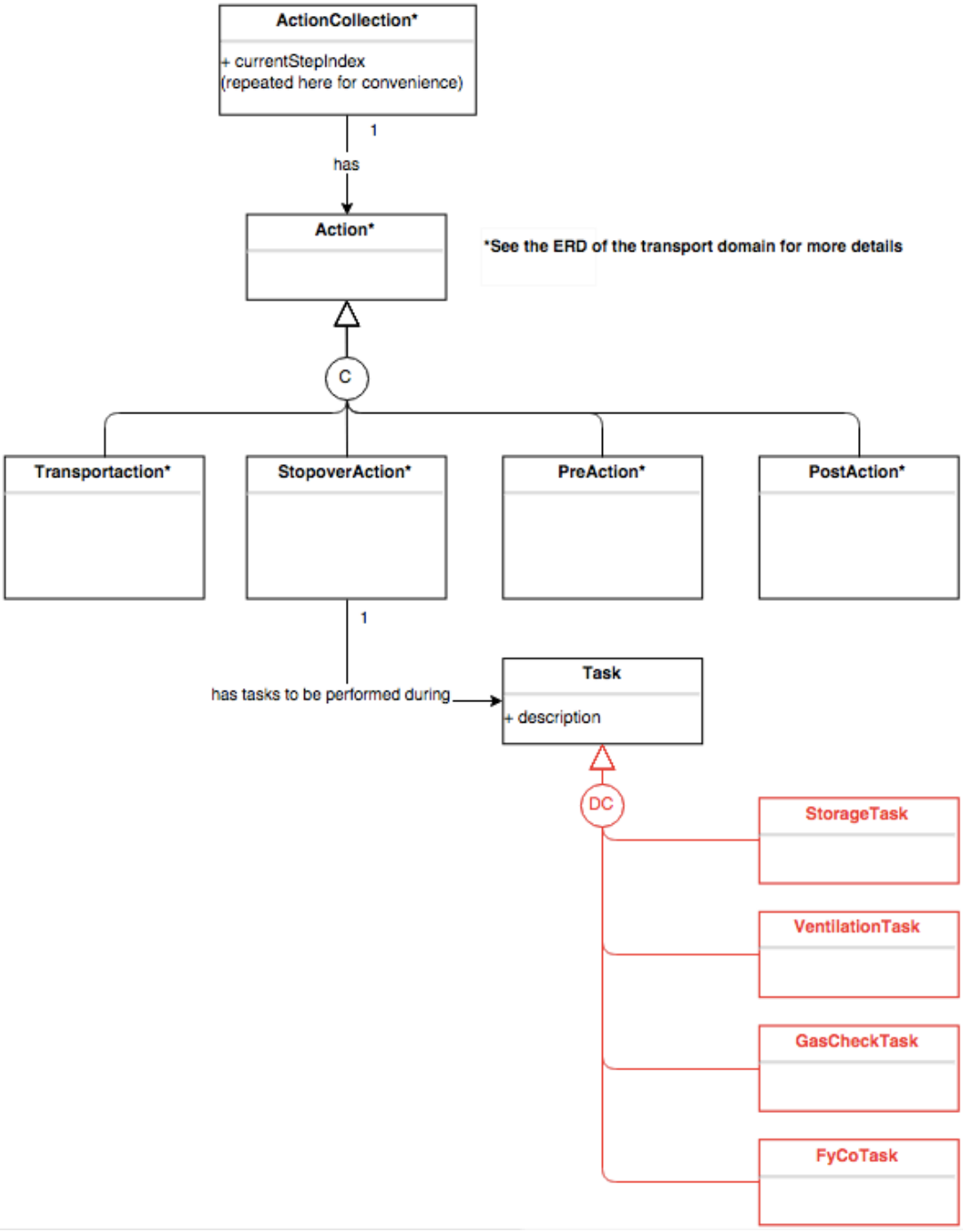

Figure 16 - ERD Tasks in the Action chain

4.3.10 Invoicing subdomain

After the services have been completed (sometimes even before), the GHCT creates an Invoice for its customer.

We first present the Entities and Relationships in the domain since we will be using these to explain the process. The ERD is shown in Figure 17.

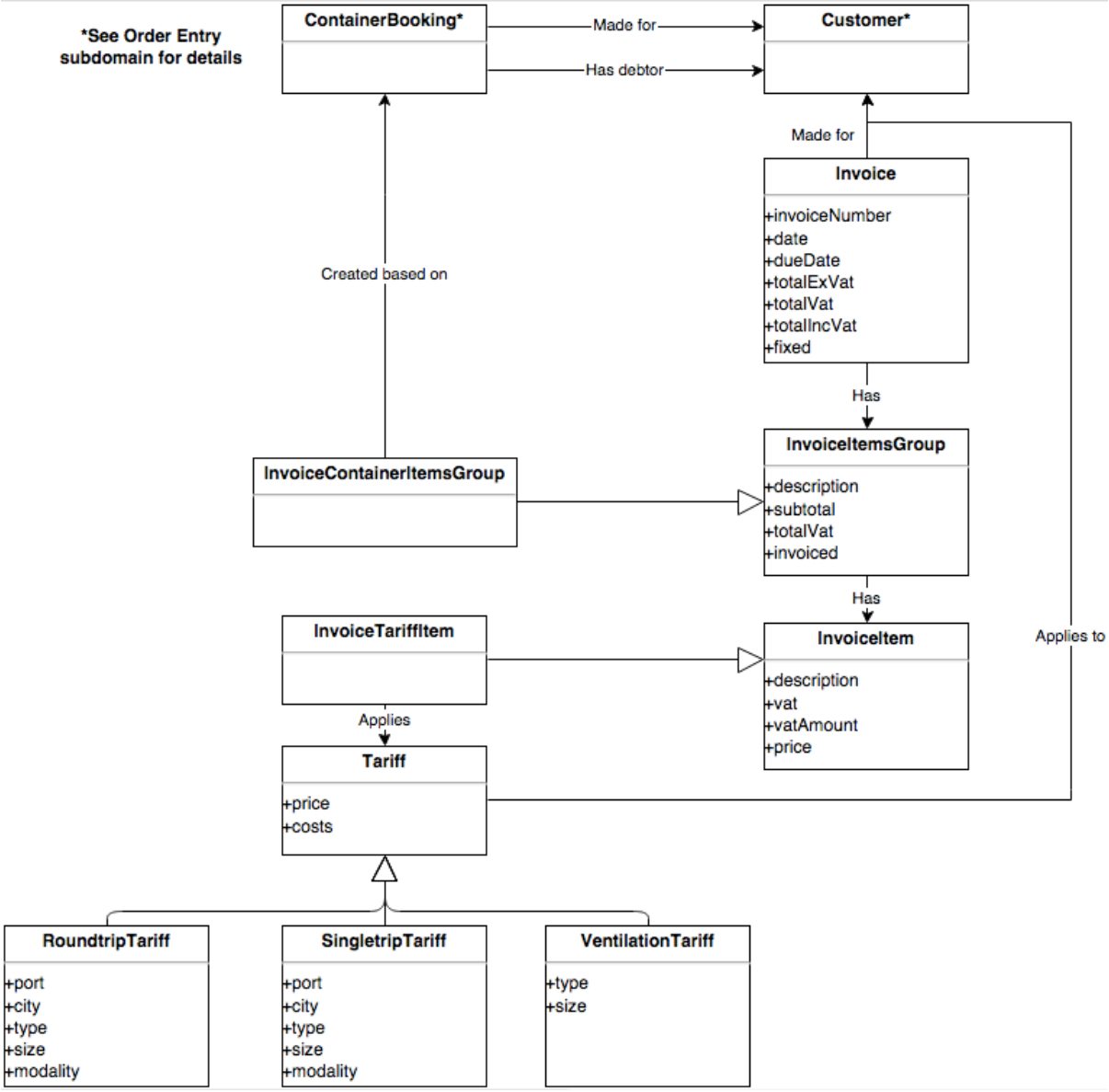

Figure 17 - Invoicing ERD

The Entities and Attributes are explained in the dictionary:

4.3.10.1 Entities

Invoice. Entity type name. Invoice specifying payment due by the customer to the GHCT. Consists of InvoiceItemsGroups.

InvoiceItemsGroup. Entity type name. A named group of InvoiceItems.

InvoiceItem. Entity type name. A description of a specific entry on the invoice together with its price to be paid by the customer (or credited in case of a credit invoice).

InvoiceContainerItemsGroup. Entity type name. An InvoiceItemsGroup created by applying Tariffs onto the ContainerBooking. Contains InvoiceTariffItems.

Tariff. Entity type name. A rule that is either applicable on a ContainerBooking or not. When an InvoiceContainerItemsGroup has to be created for a ContainerBooking, all the applicable Tariffs result in an InvoiceTariffItems each, placed in the

InvoiceContainerItemsGroup for that ContainerBooking. Tariffs can have limited applicability to a particular customer (i.e. deals made with certain customers).

RoundtripTariff. Entity type name. A type of Tariff that is applicable when the ContainerBooking is a roundtrip and is of the same type, size, modality and from/to the same city and port as is specified in the RoundtripTariff.

SingletripTariff. Entity type name. A type of Tariff that is applicable when the ContainerBooking is a singletrip (the container is not return or retrieved from a depot but reused) and is of the same type, size, modality and from/to the same city and port as is specified in the SingletripTariff.

VentilationTariff. Entity type name. A type of Tariff that is applicable when the ContainerBooking has an order for ventilation (i.e. ventilation attribute in the ContainerBooking is true) and is of the same type and size.

InvoiceTariffItem. Entity type name. An InvoiceItem that is created by applying a Tariff and placed in an InvoiceContainerItemsGroup.

4.3.10.2 Attributes

invoiceNumber (i: Invoice). Attribute. A number uniquely identifying an invoice, numbered sequentially after one another in the same year.

fixed (i: Invoice). Attribute. Boolean indicating whether the Invoice has been sent. If so, the Invoice can no longer be changed.

subtotal (iig: InvoiceItemsGroup). Attribute. The sum of all price attributes of each InvoiceItem in the InvoiceItemsGroup.

invoiced (iig: InvoiceItemsGroup). Attribute. Boolean indicating whether the InvoiceItemsGroup has been put onto an Invoice (InvoiceItemsGroups can exist without an Invoice.)

vat (ii: InvoiceItem). Attribute. A number expressing the percentage of VAT to apply on the price multiplied by 100 (so '21' for 21% vat).

price (ii: InvoiceItem). Attribute. A number expressing the price in Eurocents, excluding vat.

costs (tf: Tariff). Attribute. The costs that are assumed to be made for a ContainerBooking when the tariff is applicable for the ContainerBooking.

4.3.10.3 The process

There are two options when creating invoices. Either create a complete customized invoice that is not based on ContainerBooking(s) (for example, when the GHCT lends a reach truck for other purposes, or other exceptions), or compose an invoice from InvoiceContainerItemsGroups, which are based on ContainerBookings.

In the first case, the Invoice consists of plain InvoiceItemsGroups and InvoiceItems. However, in the second scenario, the Invoice consists of InvoiceContainerItemsGroups, which have to be created based on Tariffs that are applicable for that particular ContainerBooking and Customer.

After composing an Invoice from InvoiceContainerItemsGroups, the Invoice can be customized (i.e. plain InvoiceItemsGroups and InvoiceItems can be added, InvoiceContainerItemsGroups and InvoiceTariffItems can be edited.)

Once an InvoiceContainerItemsGroup has been created for a ContainerBooking, a new correcting InvoiceContainerItemsGroup is created each time a ContainerBooking changes, as those changes imply a different Tariff to be applied then before. (For example, a credit Invoice may be issued when a container was reused, which implies a SingletripTariff instead of a RoundtripTariff.) InvoiceContainerItemsGroup have to be put onto an Invoice again. The process is shown in Figure 18.

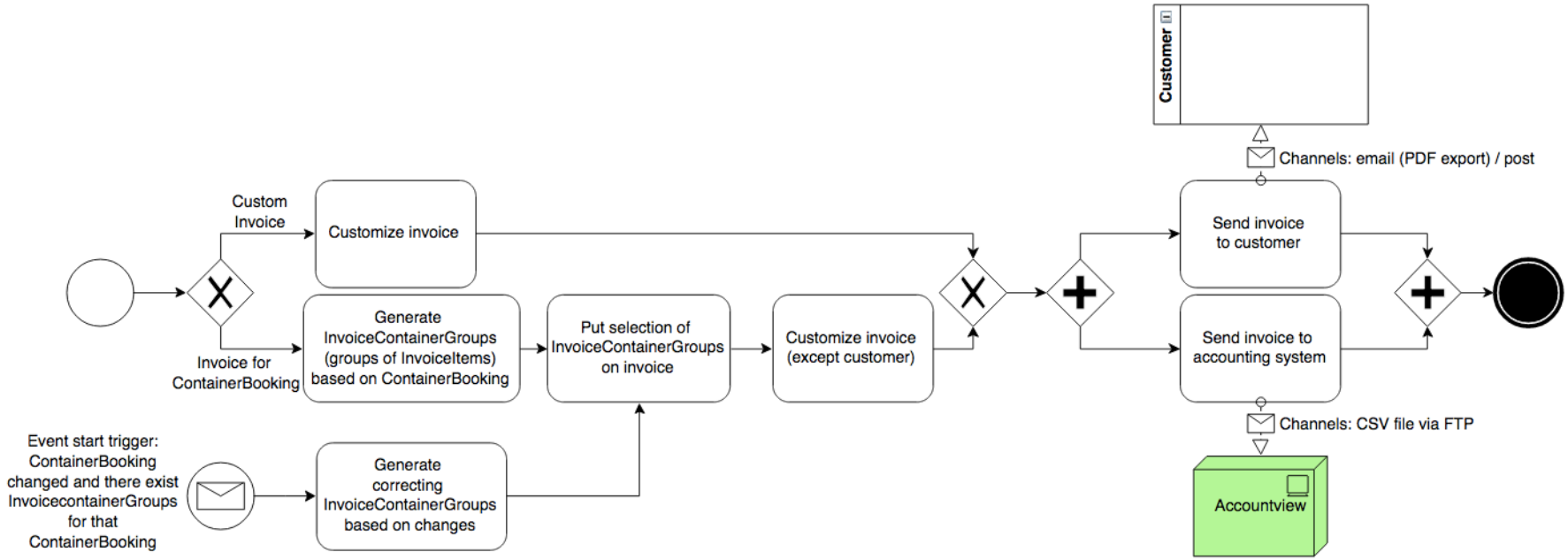

Figure 18 - Invoicing Process

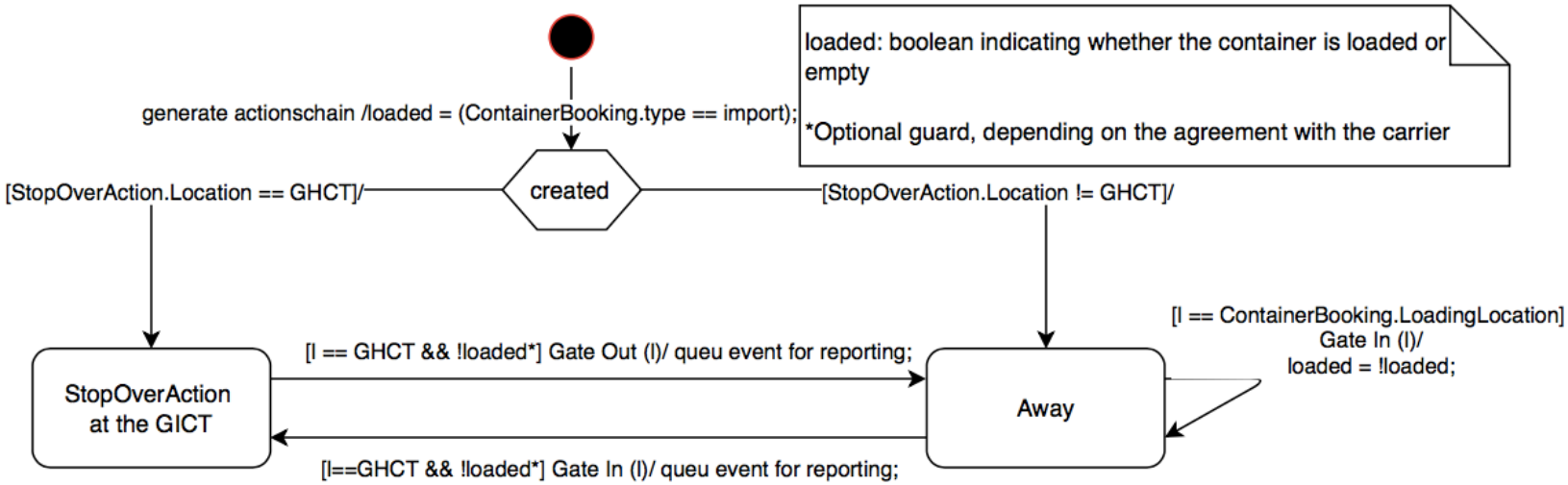

Figure 19 - CODECO reporting state diagram

4.3.11 Reuse subdomain

When the GHCT is a depot holder for a carrier, it basically means that they have an agreement with the carrier that they are allowed to retain an empty container for a certain amount of days (usually 90 days, depending on the agreement) after it finished an import round trip. The GHCT can use this time window to reuse the container for an export round trip, effectively saving a trip to the depot of the carrier. If the time passes without reusing the container, the container has to be returned to the depot after all.

This means that, a container of a ContainerBooking that is near the end of an import round trip, but are not yet returned to the depot, is fit for reuse. By matching these ContainerBookings with export ContainerBookings, both round trips are turned into so called 'single trips' as the trip to the depot for both ContainerBookings are skipped. This possibility is reflected in the state diagram of the Gate Control subdomain in Figure 15.

As you can see, a match depends on the following conditions:

- The export ContainerBooking's ActionCollection's currentStepIndex may not be beyond the first step.
- Both the import and the export booking need to be intededForReuse (sometimes the customer does not want a reused container for their export).
- The carrier of both bookings must match.

If a match is made (i.e. the conditions are met) the ContainerBooking in question starts in the stopOverAction state with its location as the current location of the matched import booking, which saves a trip to the Depot.

At the end of an import, the container of the ContainerBooking may be reused given that it is not yet returned to the depot and it is allowed to be reused. This transition in the state diagram brings the ContainerBooking in the Reused state.

Because the match may be made over a period of 90 days after unloading the container, the customer has usually already been invoiced. This is a typical scenario where the Action chain changes and a correcting InvoiceContainerItemsGroup is created in the Invoicing domain.

In sum, reusing is a matching activity between ContainerBookings: When a match is found between import bookings of which the container is not yet returned to the depot and export bookings that are just created (and thus may start in the reuse state), both ContainerBookings save a trip to the depot.

4.3.12 Replenishment subdomain

In agreement with the Carrier, the GHCT is also allowed to hold a stock of empty containers under the same conditions as reusing them (i.e. holding them for 90 days and match them with an export booking for use). Specifically for this purpose, a DepotBooking can be created, which represents a trip to a Depot to retrieve one empty container.

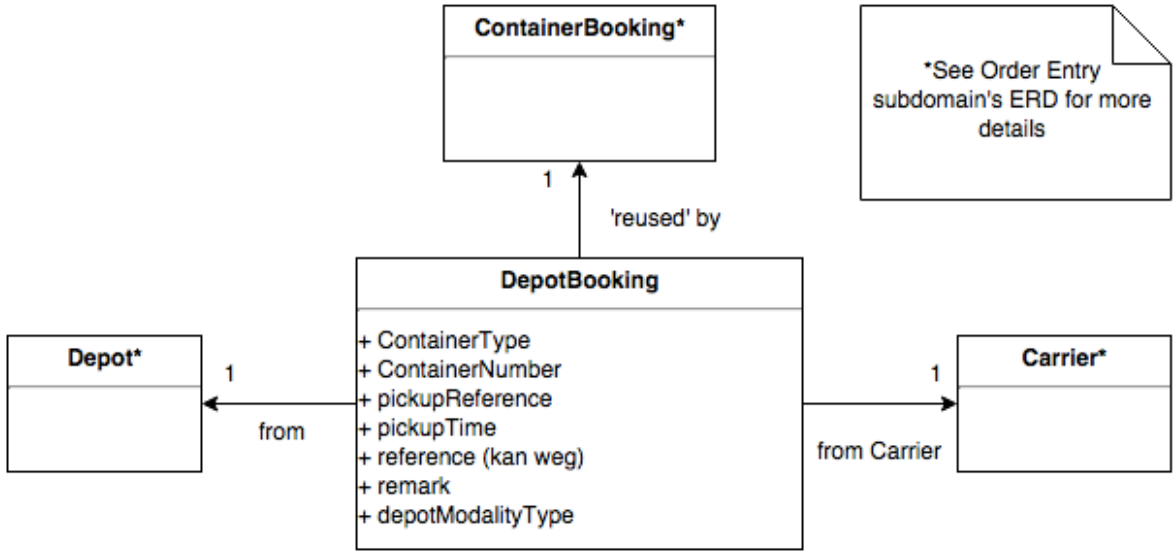

Figure 20 - ERD of the DepotBooking for empty container stock replenishment

The DepotBooking is shown in the ERD for this subdomain in Figure 20. We supplement it with a short dictionary:

DepotBooking. Entity type name. A booking to retrieve one empty container from a Depot of a specific Carrier.

depotModalityType (db: DepotBooking). Attribute. The modality type used to retrieve the empty container from the Depot (i.e. by truck or by barge).

Similar to reusing ContainerBookings, the DepotBooking is 'reused' once it is on the terminal. In that case, the DepotBooking is 'matched' to an export ContainerBooking exactly as described in the Reuse subdomain.

The entry process is a simplified version of the regular order entry depicted in Figure 21. Just like a normal ContainerBooking, after the initial creation of a DepotBooking the chain of Actions (ActionCollection in the ERD of the Transportation domain) is created, which are then to be planned by the normal planning processes.

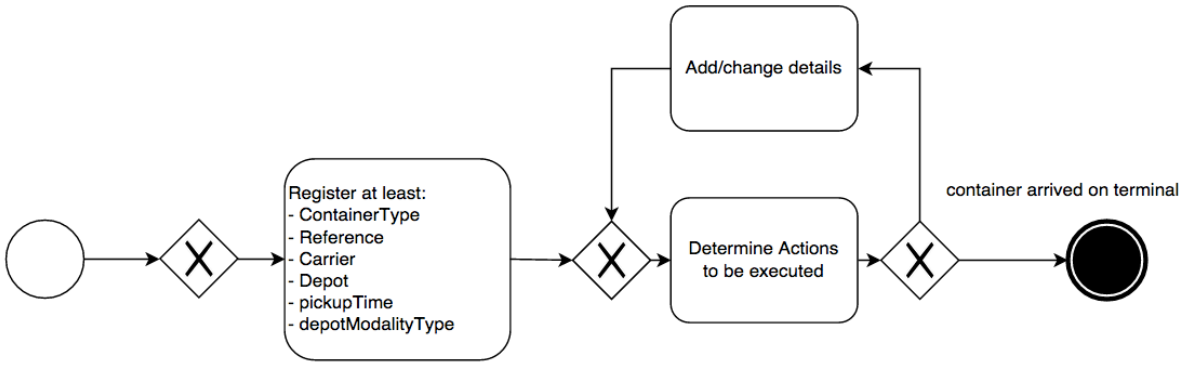

Figure 21 - DepotBooking order entry process

4.3.13 Reporting (Depot) subdomain

Part of the depot holding agreement the GHCT may have with a carrier, is that the GHCT has to report all movements of containers to their carrier. The reporting is based on the Gate-in and Gate-out events at the GHCT. In other words, as soon as the container enters or leaves the GHCT, that event has to be sent to the carrier. (For clarity, this excludes events at other locations.) The events are reported to the carrier through so called 'CODECO' messages (UN, 2001). The frequency of these messages and whether the carrier merely requires messages about empty containers may differ depending on the agreement with the carrier.

To describe this behaviour, we add another state diagram since its focal point are the Gate In and Gate Out events at the GHCT. The diagram is shown in Figure 19 and works in conjunction with the state diagram of the Gate Control subdomain.

After a ContainerBooking has been created, the Action chain may start in either two states in this diagram due to different starting possibilities (i.e. reuse, import, export, see Gate Control state diagram): Either in the StopOverAction state at the GHCT or in another Action state. As you can see, the event can only be queued for reporting to the Carrier when the Gate Out or In event takes place at the GHCT.

An optional condition for queuing the Gate In or Out event is that the container is empty. Whether this condition applies depends on the agreement with the carrier.

4.4 Design principles

In addition to the design principles explained while building the reference model throughout the chapter, we summarise them here such that the reference model can be reproduced.
On a high level:

- Start by building the skeleton based on the main business process in its most elementary form and divide that process in a few crude domains; in this case we take the placing of an order by a customer as starting point and the sending of an invoice after delivery as end point in the process.
- After the skeleton is in place, go through the main process again and identify subdomains, like order entry and invoicing in the customer domain in this case.
- This process of adding levels of details in the building blocks is continued until each block (on whichever level) can be described with enough detail by means of the different perspectives (i.e. process, data flows, entities and relationships, and events).

When giving content to the domains:

- The model describes reality 'according to the system', but not the system itself. The reference model's domains, events and entities **may** correspond with the system's functional domains, events and data structure. In part, it should support the domain properly. However, implementation details are irrelevant for the reference model; all descriptions and models should be in terms of the actual domain.
- What is described in a domain may cross its domain borders; this shows how it is related to other domains and reflects the real world where no organisation can exist with completely autonomous departments.

- What is described should only reflect what is currently supported by the system; no more no less. Otherwise the differences with reality cannot be identified. (The reference model cannot describe what is planned to be supported, since that is contingent on the outcome of its design and consequently the changes it imposes on the organisation.)
- One or more perspectives for each domain are chosen depending on the focal points of that domain; process models for sequenced tasks and data flows, ERDs for entities and relationships and state chart diagrams for events. Each domain has at least a description, but not always a model; the description may refer to models in other domains to prevent redundancy.
- Domains with an * are optionally supported by configuration of the system.
- Domains in red are identified but not yet explored.
- Parts of models in red are part of the current design of the system but not yet (fully) implemented. This is the only exception where future support **may** be modelled as the design (and hence what is supported) are known.

5 Process model

As described in the previous chapter, the product model is the derivative of the reference model. Such derivative is a description of the world according to the system and differences identified. This combination gives us a description of the problem context as far as we know it.

Therefore, the process of creating derivatives equals the process of finding differences. In this chapter, the current development process aimed for at Cofano is described. Thereafter, the process is augmented with the use of the reference model.

5.1 As is

The process as is currently aimed[4] for is described in a dataflow diagram (Figure 22). Overall the process contains practices obtained from Scrum (Rubin, 2012). The customer tells about its domain and gives feedback about a previous iteration. Using this input, the analyst (within the organisation often referred to as 'lead developer') forms a mental model of the problem context and what needs to be supported. With this insight, the analyst creates issues to resolve negative feedback or such that the product properly supports the domain as the customer has explained it.

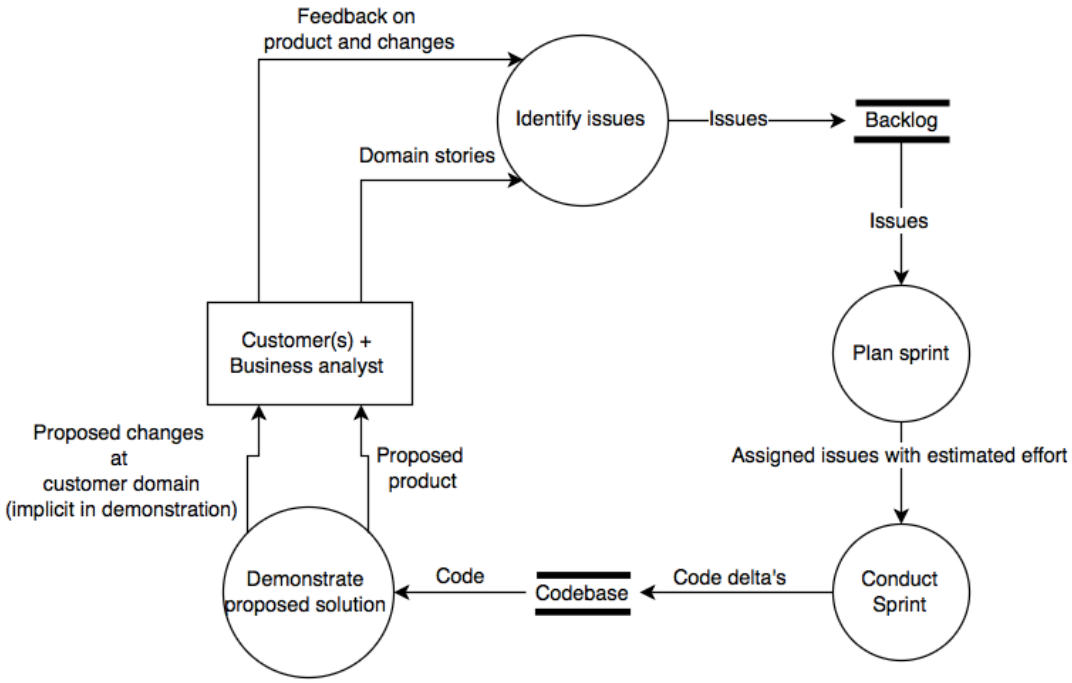

Figure 22 - Current development process at Cofano

The team works in sprints, of which the time span is variable but often 2 weeks. A sprint planning is made by selecting a subset of the backlog's issues that are to be resolved during the sprint, assigning them to programmers and estimating their time to resolve. It is not unusual for the development team in Sliedrecht to have stand-up meetings.

[4] Officially, this is the process that is aimed for. However, often this is not the case. In example, often there is no sprint planning or the planning is (partly) ignored and issues to be solved are picked up ad hoc from the backlog.

Throughout the sprint, changes to the code are committed to the codebase ("code delta's" In the diagram). Once in a while, usually after a sprint, the product is released for feedback from the customer. It is either demonstrated or made available to the customer to experiment with the product. Note that currently, changes to the domain implied by the system are implicitly proposed to the customer.

5.2 To be

In the 'to be' situation, the mental model of the problem contexts are made explicit by use of the reference model. In addition, the reference model is kept up to date as part of the development cycle. The overall process is described in the dataflow diagram shown in Figure 23.

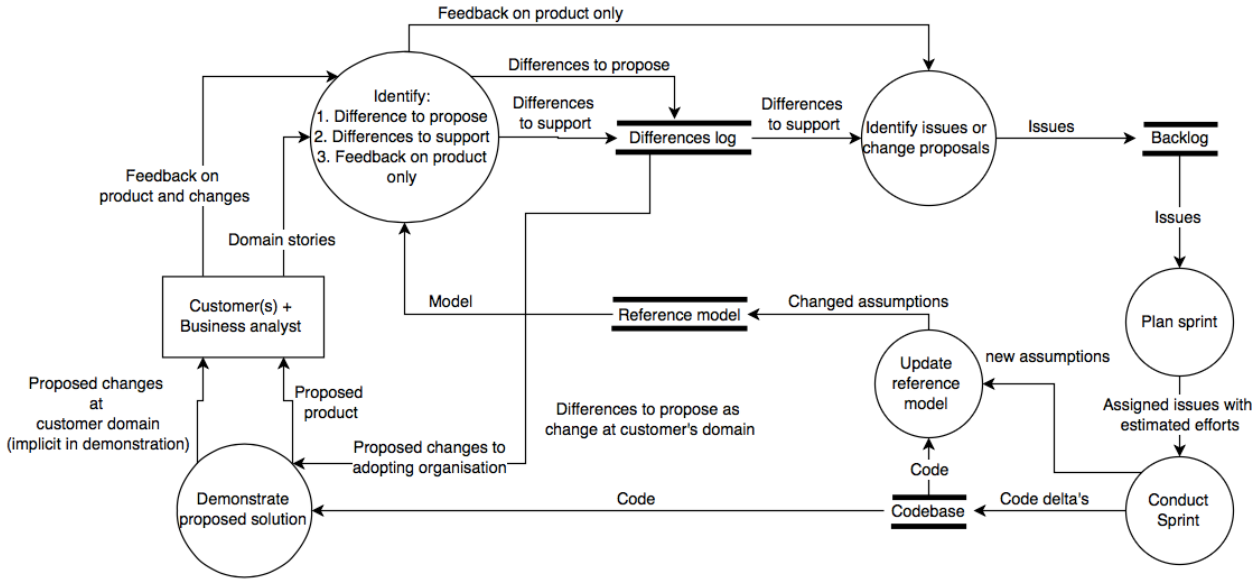

Figure 23 - Development cycle with use of the reference model

The process starts with the customer's input. The analyst again forms a mental model of the problem context. The analyst compares this mental model to the reference model and identifies the differences. Besides input about the problem context, feedback exclusively about the product itself, which does not imply a difference in assumptions, follows the traditional path. It is directly translated into issues.

5.2.1 Options for identified differences

For each difference, there are a few options: first the difference could be of such degree that a change to the system must be made without question. This could be the case if a certain subdomain is missing entirely, for example. Another option is to propose a change to the problem context; perhaps the process assumed by the system is different but not unworkable for the customer or even better than the current practice. Lastly, a change to both the customer's business and the system may be combined.

The analyst makes the differences and the fitting option explicit by documenting it in the differences log. An example of the differences log is shown in Table 1. In case the difference needs to be supported, the system must change. The analyst describes what assumptions has to be different about the problem context. If a difference may be resolved by proposing it to change at the customer's domain, the 'change at customer' column is checked. If a combination is in order, the 'to be' situation of the problem

context is described as a difference with the reference model and the 'change at customer' column is checked.

From the identified differences that need to be supported documented in the differences log, the issues are created to develop that support. Issues created to support a particular difference are documented at that difference in the differences log in the 'issues' column.

5.2.2 The sprint

The sprints are conducted as usual, with the slight addition once in a while (preferably after each sprint), changes to the system should be reflected in the reference model. As the system changes, *the world according to the system* changes as well. Therefore, the reference model describing that world must change with it. Otherwise differences are no longer identified correctly. Once the change in the system is reflected in the model, the change can be indicated as 'modelled' in the differences log.

5.2.3 Feedback

Once the sprint is completed, the difference to propose as changes together with the latest version of the software product form the proposed solution. The input of the customer on this proposed solution forms input for the next development cycle.

Depending on that input, a change can be indicated as being fixed, meaning the customer has agreed with the proposed solution. On the other hand, if the proposed solution is rejected, the analyst can describe another difference in the differences log and indicate it is the next iteration of a previously indicated difference. The result is a history of identified differences and decisions made to mitigate those differences (hence the name 'differences log'). As the reference model may not be fully up to date with the latest changes, the analyst can see the latest status of it in Table 1.

In practice, the customer's input will often be in the form of a story about the problem context (i.e. telling us how things are done at their company called 'Domain stories' in the diagram) or in the form of feedback on the proposed solution. In the latter case, this could also be feedback directly on the application itself. Small preferences on the user interface aside, the analyst should realize when such feedback implies a difference in the assumptions in the reference model.

As an example, the customer may want multiple loading locations when entering a container booking. The analyst can indicate this difference in the reference model through the ERD and the process diagram. The ERD should have the possibility to describe multiple loading locations in relation to a container booking. If this is not the case, the analyst describes this difference in the table and chooses one of the options as described.

A counter example is when feedback on the application itself that does not imply a difference between the domain and the reference model. This type of feedback is more common in practice. The customer may not like the graphical interface of the system, wishes certain information to be displayed (like a map of the current location of a ship on the planning screen), or wishes to be able to use the tab button to jump to the next input field of a form.

Customer	Description of identified difference	Iteration of	Issues	Change at customer	Fixed	Modelled
Cust. A, Cust. B	It is possible to assign multiple loading locations to a container booking (independent of booking type). Currently only one is assumed.		1234, 1235, 1236, 1237		Yes	Yes
Cust. B	The related ContainerBooking's TerminalBooking may either be import or export: import going from the deep sea terminal to the GHCT's local customer and export vice versa.		1456, 1457			
Cust. B	The ContainerBooking's TerminalBooking may also be unknown until further notice	2	1456, 1457		Yes	Yes

Table 1 - Differences log

5.2.4 A fit for each customer

As each difference identified is tracked per customer in the differences log (second column in Table 1), filtering the table on a specific customer results in a summary of how the system and that customer's organisation fit together. The list may contain changes made to the organisation as well as the variations added to the system. On the other hand, variations added to the system can be tracked to its original customer(s).

6 Validation

Aside from the analytically reasoned validity of the method chunk developed, based on the theoretical foundation and its reasoned design principles, input for external validity is required. Throughout its development, the method chunk iterates between researcher and practitioners from the organisation as described in the Action Design Research (ADR) approach (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011). However, the iterations have not yet approached the point of adoption or rejection by the organisation. Therefore, an alternative is required to obtain external input for the validation of the method chunk. The goal is to explore the plausibility that the method chunk will be effective in use for future customers of Cofano to:

- Identify commonality and variability
- Cope with multiple customers
- Improve coordination within the developing organisation
- Quick and precise identification of requirements (or 'issues')

6.1 Approach

The options identified to that end are:

- Applying the method in a test case within a workshop for Cofano.
- Using the method in the field at a future customer of Cofano.
- Asking experts for their opinion.

Applying the method in a test case will require the attendees of the workshop to be competent modellers and abstract thinkers. This will require time and effort to develop these competencies within the organisation and is outside of this project's scope.

Using the method in the field is seen as commercially risky, as the method is unproven and may give the future customer a negative impression of Cofano's abilities.

Therefore, the last option is chosen for this project. The method chunk is explained to experts and asked for their opinion. The experts asked for their opinion with their qualifications, and from which perspectives they provide insight are summarized in Table 2. In short, the experts asked for their opinion had the following qualifications:

- Experienced in management of, and overseeing large IT implementation projects for core operations in the logistics industry (preferably intermodal container transport).
- Preferably experienced with generic models (ability to understand the level of abstraction)

Interview	Qualifications	Stakeholder perspectives
Marco Huijsman	- Consultant (often project leader) in the logistics industry for process and IT application optimisation and implementation (for both adopting and supplying organisations) (18 years) - Co-owner and product owner of Cofano Software solutions (5 years)	End user, owner and the adopter perspective
Michel Mensink	- Implementing SAP ERP finance modules (9 years) with use of reference models.	End user and owner
Marten van der Velde	- Project management of IT implementation projects for Portbase (Previously Port infolink, provider of nearly nationwide shared services for the logistics industry in the Netherlands) (4 years)	End user, owner and the adopter perspective
Richard Klaassen	Key user SAP implementation at GE Bayer Silicones (1 year) Terminal Manager Markiezaat Container Terminal (7 years)	Adopter perspective

Table 2 - Experts

The experts were given a presentation on the method chunk and the context it is placed in. The presentation included an explanation of the core concept of the reference model, a tour of the reference model, an explanation of the process model with a real life example to illustrate the method chunk's use in practice, and its most important design principles. The experts are asked to identify at least three challenges or pitfalls, and three advantages of the method chunk.

6.2 Advantages

The advantages identified by experts are listed in Table 3.

In general, all experts that provide insight from the end user or owner stakeholder's perspective are convinced that the method chunk will:

- Identify commonality and variability between customer's problem contexts
- Cope with multiple customers
- Improve coordination within the developing organisation
- Quick and precise identification of requirements (or 'issues')

However, there are challenges and pitfalls that may impede each of these advantages. These are discussed in the next section.

A few of the advantages identified are worth elaborating. First and foremost, as more terminals adopt the TOS, the practices that the system supports will become a collection of 'best of breed' in the industry. The method chunk makes this knowledge explicit as added value to potential future adopters of the system.

By	Advantage
Mensink, van der Velde, Klaassen, Huijsman	As terminals adopt the system, more business practices and solutions become available; the reference model becomes a collection of best practices in the industry. The knowledge is added value that comes with the system that is sold and is made explicit by the reference model.
van der Velde, Klaassen, Huijsman	Business process optimisation (developing and selecting best practices) becomes part of the software implementation project as opposed to blindly supporting the process in place.
Mensink, Huijsman	Opens possibilities for even more distributed development teams: perhaps even outsourcing/offshoring
Huijsman	Helps the integration process with third party applications in communicating the process implied by API's.
Mensink, Klaassen, Huijsman	Dictionaries in the reference model helps to standardise jargon.
Mensink	The method chunk will save resources (programming and configuration activities), estimated at 15-20% from experience.
Mensink, Klaassen	Gives a professional impression at the customer of the developing organisation's capabilities to provide suitable solutions.
Mensink, van der Velde, Huijsman	Identifies commonality and variability between customers and helps finding a fit at multiple customers. (Helps coping with multiple customers.)
Mensink, van der Velde, Huijsman	Improves coordination through central documentation of customers' problem contexts resulting in a shared mental model in addition to newcomer to learn more quickly about the problem contexts.
Mensink, van der Velde, Klaassen, Huijsman	Quick and precise identification of requirements (or 'issues'). Will help to stabilise requirements earlier in the project (problems are more quickly understood because the assumptions are made explicit).
Klaassen	Problems resolved or improvements made for one adopter become available for the entire customer base.
Klaassen	Reference model leaves room for unexplored areas to be added by simply adding blocks.

Table 3 - Advantages identified in expert interviews

In addition, as the decision to add support for certain practices to the system and developing new practices are part of the implementation projects, the developing organisation does not merely supply software. Rather, business process optimisation becomes part of the implementation project, with knowledgeable insights provided by Cofano from prior adopters.

It is even thought that, due to the improved coordination, the method chunk may even open up possibilities for outsourcing; sharing the mental model with an off shore or near shore developing team might improve coordination.

From the customer's perspective, Klaassen noted that problems solved or improvements made for one adopter become available for the entire customer base. He also said that the reference model leaves room to add new unexplored parts of the problem contexts due to its structure and build-up.

6.3 Challenges and pitfalls

As said, there are challenges and pitfalls that may impede the main advantages of the method chunk. All identified challenges and pitfalls are listed in Table 4. The challenges and pitfalls that require elaboration are discussed in more detail.

6.3.1 Sharing knowledge with the competitor

As the knowledge of practices and best practices developed throughout its implementation projects are gathered in the TOS and the reference model (as described as an advantage in the previous section), adopters may fear sharing this knowledge and losing their competitive advantage. A counter argument suggested by Mensink is that their competitive advantage comes from their location and the quality of their operational execution. Huijsman noted that the adopters' customer base comes from their geographic location and rarely regard each other as competitors.

6.3.2 Required capabilities and skill in use

An important pitfall from Mensink's experience, is that the reference model may give the impression that any person can analyse the customer's problem context for differences. Insight in the customer's business, knowledge of the reference model's content, modelling skills and a certain capacity for abstract thinking are impertinent for the task. Performing the task by an under qualified person will neither result in a correct identification of commonality and variability, nor in the correct requirements. Ultimately, it may result in a poor fit with the organisation where its processes are poorly supported.

Conversations between people about the reference model's contents also require certain capabilities. These may be the analyst and the customer, or co-workers discussing an identified difference within Cofano. In discussions about the reference model's contents and identified differences, the capability to talk in high levels of abstraction and a deep understanding of the business processes is required. Indeed, a source of difficulty in enterprise software implementation projects often involves the customer's poor understanding of their own business processes and limited ability to simplify and model them (Dalal, Kamath, Kolarik, & Sivaraman, 2004).

Another challenge noted by van der Velde, is that programmers tend to think strongly in terms of technical solutions. This tendency results in difficulties in separating the domain described in the reference model, from the technical solutions to support it. The separation becomes even more difficult as certain parts of the domain exist only in the system, but nevertheless is part of the domain.

By	Challenge or pitfall
Mensink, Huijsman	A consultant knowledgeable of the reference model's content, the actual business processes it describes and the application is still required
Mensink, Klaassen, Huijsman	Conversation partner needs to be able to abstract at the level of the reference model.
Mensink, Klaassen, Huijsman	Changes to the organisation require careful change management
Mensink, van der Velde, Klaassen, Huijsman	Differences identified and changed in the system may not have added value for future or past customers, or even result in unwanted changes at existing customers.
Mensink, Klaassen, Huijsman	Changes made to the reference model, with implications for the existing customer base, should be agreed upon through a focus group and/or a group of representatives of the customer base.
Klaassen, Huijsman	Too many variations added to win over adopters may results in all these variations described in one complex model and embodied by one system, resulting in high cost and complexity.
Mensink, Huijsman	A variation to the system should be made with consideration for added value for the customer base and its cost.
Mensink, Huijsman	Changes to the codebase may break existing links with other systems at existing customers.
Mensink, Klaassen, Huijsman	The reference model will always lag behind the actual state of the system's assumptions in practice.
Mensink	Work saved in the project is estimated from experience to be merely 15-20% (the programming and configuration activities).
van der Velde, Klaassen, Huijsman	The knowledge adopted in the TOS and consequently the reference model is shared by all adopters, which may result in fear of losing competitive advantage over other container terminals. Also mentioned in the review of Pajk et al. (2011).
van der Velde, Huijsman	In general, programmers tend to think in terms of technical solutions straight away and may find it difficult to separate the domain, which is described in the reference model, from the technical implementation.
Huijsman	The method should be accepted by the developing organisation for it to work. One fear is that the system implied by the method may impose too much structure on the informal organisation.

Table 4 - Challenges or pitfalls identified in expert interviews

6.3.3 Organisational and software changes

Part of the method is to propose changes to an organisation; perhaps even more than at normal software implementation projects as the adaptations of the system to the problem context are kept to a minimum. Therefore, for a successful implementation, the organisational change should be managed with care.

On the other hand, when variations are added or changes are made to the codebase of the TOS, these may be unwanted by prior adopters of the system or even break existing processes and communication links. Mensink and Klaassen suggests a careful communication plan towards the existing user base upon changes and a focus group of 'reference customers' to reflect the impact of the changes made.

In addition, Klaassen noted that too much variation added to tailor to each and every customer's wishes may result in a too complex system and reference model and high cost. Mensink made a related remark that the decision to add a variation to the system should be made with consideration for added value and its cost. Huijsman nuanced the notion that as an exception, integration with existing systems of an adopter cannot be prevented and results in variations that may only be used by one adopter.

Another organisational change is that at Cofano adopting the method. For it to work, the method should be accepted by the organisation. Huijsman noted that too much structure imposed by the method might impede its acceptance, as the organisation has an informal way of working.

7 Generalisation

The applicability of the method chunk is limited to use by Cofano, as the process model is tailored to their development process. Furthermore, the reference model is developed with HCTs in mind. Its design principles and its building blocks are all based on our experience with HCTs so far, in conjunction with the TOS' design. Therefore, the method chunk developed in this project is limited to instantiate for development projects conducted by Cofano, where the TOS' codebase is to be implemented at organisations that qualify as HCTs.

Organisations that qualify as HCTs, are those which:

- Organise (at least a part of) the transport of intermodal containers or bulk goods between the shipper and deep sea terminals.
- Of this transport the organisation itself, at least, receives, stores and loads (at least part of) the intermodal containers or bulk goods for limited periods in between transports at their location.

8 Discussion

Despite the insight the experts give from their experience, the method chunk is built and validated almost purely analytically. Therefore, its effects from use in practice are yet to be seen. In addition, there are a few other critical remarks to be made.

8.1 How versus what

First off, as the reference model describes the customer's problem context, it describes **what** is supported by the TOS, not **how** it is supported. Some may consider the distinction as a limitation. Although very important, whether a process is supported with a fax machine and a homing pigeon or a state of the art web application in the cloud is not of concern for the reference model. However, one is not better than the other. Rather, understanding what is to be supported is the step before developing how to support it. How to support a process is up to the creativity and innovation of the developing organisation, but not before it is understood what is to be supported. The process of understanding the customer's problem context is documented and made explicit by use of the reference model.

8.2 Shared knowledge benefits realisation only when used

As the method chunk's potential advantages in part come from its shared knowledge facets, these advantages will only come into effect when it is actively used by the developing organisation. Often, the task of understanding the problem context is mainly placed at the person in charge of the customer dialogue. If only that person were to use the reference model as a tool for its business analysis and updates it after each sprint, these advantages will not be realised if programmers do not contribute to the development of the shared mental model of the customers problem context and only focus at its outcome (i.e. the issues created).

8.3 Maintenance

In addition, the reference model's effectiveness is highly dependent on its maintenance; an out-dated reference model can no longer identify differences with the actual system's assumptions about the customer's problem context. In a pragmatic organisation, the reference model can easily be considered unnecessary 'red tape'. Activities other than documenting differences and updating the reference model could very well receive higher priority. In example, as small changes are made ad hoc, these may skip the reference model's maintenance activities all together. It is therefore pertinent to keep these activities as lightweight as possible.

In theory, the reference model will require less maintenance over time as more HCTs have adopted the TOS: As more organisations have adopted the codebase, the more generic the reference model's domain it describes becomes. There will be less feedback from customers that imply a change in the reference model and more on how their domain is supported.

8.4 Reference model's precision

There is a trade off between the reference model's precision and the effort required in the reference model's maintenance activities; on the one hand one might model a highly detailed state diagram, but it soon becomes near impossible to comprehend (i.e. Figure

15 is considered difficult to comprehend). On the other hand, too little detail may impede the ability to identify relevant differences.

In addition, the reference model may state an assumption about a limitation, which in theory is technically possible by the system. For example, shuttles will never be meant to drive a container to a deep sea terminal (why use an hinterland terminal in the first place if the deep sea terminal is that close by the shipper?), but technically, if a deep sea terminal is indicated to be a 'shuttle location' in the system, the user is capable of planning a container transport to the deep sea terminal on a shuttle.

9 Conclusion

At the beginning of this research, we asked the following question:

What is a suitable method chunk for analysing the problem contexts of HCTs in order to identify their needs and priorities for software solutions?

We've seen that a methodology consists out of at least a product model and a process model. We asked the following subquestion in order to develop the product model:

- *Which pieces of information about the problem contexts can be identified that are relevant for analysis?*

As the same codebase should be able to support more than one customer's problem context, the identification of variability and commonality is key. Because commonalities do not require action by the developing organization, we argue that the relevant information describes the differences between problem contexts and not yet explored parts of the problem contexts that may require support from the system.

In order to effectively and efficiently document and store the differences, the question is:

- *What is a suitable method for documenting the relevant pieces of information?*

Reference models laid the theoretical foundation for the product model. The generic model developed in this project describes currently supported domains by the TOS, such that variability can easily be identified when comparing problem contexts. Even stronger, because the same codebase is used to support different problem contexts, the same reference model can be used to compare against the actual problem context of adopters. This assures efficiency be reusing the model and only documenting differences identified.

How well the model describes the problem context assumed by the system and supports variability in detail is yet to be seen in practice. The adoption of the method chunk is a long term change project at Cofano. We suspect that, as the model matures in use by Cofano, it will be able to describe all the details it requires about the problem context, as it is up to the modellers to add these details for future reference. On a higher level, the model can add or remove parts of the problem context it describes by adding or removing building blocks. This allows for expansion of the model as well as the addition of levels of detail.

In effect, the derived models describe the to be situation together with the customer; looking for processes that the customer cannot or will not adopt or that are non-existent in the system. The outcome is therefore a design of how the system and the organisation will fit together, by mending on both sides (sometimes the organisation changes, sometimes the system changes.)

In order to adopt the reference model in the overall development process at Cofano, the question asked is:

- *What is a suitable process for applying the product oriented fragment in the overall development process of Cofano?*

For this project, the current process model of Cofano is identified and extended with use of the reference model, detailing what activities, and in what order, are needed to document differences identified and the assumptions made that have become part of the system. In addition, practices for documenting differences identified in the differences log are discussed in detail.

The process model makes use of the agile approach already present at Cofano by using the feedback cycle with the customer as input for the identification of differences and the end of a sprint as input to update the reference model.

Returning to the main research question: The process model in combination with the product model form a potential basis for a method chunk at Cofano that may be instantiated for future implementation project at HCTs. From discussion with experts, the method chunk yields the potential to effectively identify commonality and variability (and consequently the identification of 'issues) for multiple customers and improve coordination, preparing Cofano for its future growth and its expanding customer base. In addition, it adds value for the customer by making the acquired embodied knowledge in the system explicit and by providing the potential to optimise the business processes through the implementation process. Overall, the method chunk improves the alignment of the system with the HCTs. Its added value makes the adoption of TOS a unique proposition for Cofano's customers.

On the other hand, important pitfalls and challenges may impede these advantages. As best practices are to be adopted and new best practices developed are shared through the adoption of the TOS, potential adopters may fear the loss of their competitive advantage, or may find the organisational change troublesome. Furthermore, the use of the reference model requires capabilities in abstract thinking and modelling in use, both within the developing organisation as by the customer. Also should changes to the reference model and codebase be made with careful consideration for the existing customer base.

All in all, the most quantifiable advantage is estimated to be between 15-20% of the implementation project's resources in terms of programming and configuration activities and is most likely to become more as more variations are added and the customer base expands. Not including the other qualitative benefits in the equation, this implies that at least 15% of an implementation projects resources may be spent to maintain the reference model and cope with its challenges and pitfalls before a break even point is reached and further investment in the method no longer yields any advantages.

9.1 Recommendations for adoption

It is apparent that the method chunk will require some polishing for use in practice. In addition, the concept of the method chunk is to be adopted by an organisation with a limited modelling capabilities and capacity for abstract theoretical thinking while this is

mentioned as stringent requirements for its use by experts. With this in mind, a few recommendations for its adoption by Cofano are in place.

Overall, we recommend a continuation in similar fashion as the ADR method described by Sein et al. (2011), to promote learning by the organisation as well as continued development of the method chunk. As the majority of the organisation is not trained in modelling languages and unfamiliar with taking a comprehension of the customer's problem first approach in problem solving, we recommend a few workshops in which:

- In the first, the task is given to identify differences between the reference model and the current state of the TOS.
- In the second, a real life case (observed at a future potential customer) is given with the task to use the reference model to identify differences.
- In the second, or a third: When identifying differences, the task is given to identify customer feedback that merely applies to a particular solution and does not imply a difference with the reference model (in order to learn the difference by the attendees of the workshop)
- In the third or fourth: The task is given to create and link issues to identified differences.

These workshops will bring the reference model up to date and show what parts of the method chunk (reference model design and practices) work in practice and which parts require some additional thought: It is recommended to take the learning points from each workshop to improve the reference model and practices before conducting the next.

As the organisation improves the method chunk, so does the organisation itself learn to model and understand the customer's problem context by doing, and learns how to start at a deeper understanding of the customer's problems before attempting to solve them.

An additional commercial advantage of this internal approach is that it prevents the customer from being confronted by an unproven method and an organisation that is untrained in its use.

Once the organisation feels confident to do so, the method chunk can be instantiated for its first sprint in practice. It is recommended to continue the learning on both ends to keep improving both the method chunk and the organisation's ability in using it.

It should be kept in mind that, as the method and the organisation's capabilities require further development, it requires investment in terms of time and effort at first before its benefits materialise. In addition, the TOS' variations and customer base are currently small but expected to expand in the future. Both imply that the benefits of its adoption come with a delay.

10 Scientific contribution

Thomas, Horiuchi & Tanaka (2006) describe that in the past years development of reference models predominant in the scientific research has distanced itself from their use in practice. Scientific contributions in the number of methods and practices for reference modelling are numerous, often in high abstraction, but few recommendations for the case-specific selection of these techniques have been made. Consequently, a practitioner has no overview of what to use. This project adds to the void between scientific research and practice by pragmatically applying the reference model concept with design principles deciding on specific modelling techniques to use.

In practice, reference models are still rarely used in small and medium sized companies (Pajk et al., 2011), implying that this project is one of those rare cases. For most systems a reference model does not yet exist and creating one for a mature system is time and cost consuming. As the TOS is still relatively small in terms of variations, this project created a reference model in an early stage of the system's development. It shows that for a software system this scale, a reference model can be created at relatively low cost (this entire project took approximately 840 hours). However, if the developing organisation is not yet familiar with the concept, the cost become significantly more to adopt it in its implementation process.

In addition, this research introduces a new concept in the use of reference models. Its purpose in this project is to identify variability and commonality between multiple organisations, as opposed to one organisation. Traditionally, customisations of a particular ERP implementation are not added to the codebase as variation and made available to the entire customer base. This feedback loop back to the system's codebase and consequently the reference model is not present in Nes' (2007) overview of techniques of reference model use nor is it present in Pajk et al.'s (2011) description of comparing an ERP's capabilities with the organisation. Confirmed by Mensink's experience in SAP implementation projects and van Dongen, Jansen-Vullers, Verbeek, & van der Aalst's (2007) description of SAP's application of the reference model, traditional customisations are made to the particular instance implemented at a customer's organisation and not added to the codebase available to all, resulting in *not* reusing the identified differences with the reference model as in this project.

11 Future research

Future research can be conducted in a couple of directions. One of which is in the same direction as this project, focusing on further validation and development. It is recommended to conduct more iterations to further develop and validate the method chunk until definitive adoption or rejection by the organisation (Sein et al., 2011), similar to the recommendations for adoption described in paragraph 9.1. Through such iterations, the method chunk can be:

- Extended with practices to mitigate the challenges and pitfalls identified during its validation so far.
- Improved in the quality of the reference model's ability to help identify important differences in assumptions by adding details of missing assumptions
- Extended with practices for keeping track of configurations, as these are barely handled by the current version of the method chunk, as the configuration possibilities of the TOS are barely present.

Another direction of future research is recommended in a widening the scope of the implementation projects, adding to its generalizability. Perhaps the same design principles can be employed to develop a similar method chunk for a software system that is to support other actors in the same industry, i.e. expediters, barge operators, trucking companies, etc. On the other hand, the method chunk may be suitable for other developing organisations as well. The research would have to look into fitting it into a difference development processes as well and the impact that has on the effects in the new context.

Another topic for future research would be to explore the use of the method chunk in combination with a near shore or off shore development team, a potential advantage thought of by Mensink (2015), one of the experts. The research would explore the coordination improving capabilities in such setting.

Bibliography

Ågerfalk, P. J., Brinkkemper, Sj., Gonzalez-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., & Ralyté, J. (2007). Situational Method Engineering: Fundamentals and Experiences. In J. Ralyté, S. Brinkkemper, & B. Henderson-Sellers (Eds.), *Situational Method Engineering: Fundamentals and Experiences* (Vol. 244, pp. 359 – 368). Boston, MA: Springer US. doi:10.1007/978-0-387-73947-2

Blank, S. G. (2006). *The four steps to the epiphany*. *Cafepress. com* (Third Edit.). Lulu.com.

Dalal, N. P., Kamath, M., Kolarik, W. J., & Sivaraman, E. (2004). Toward an integrated framework for modeling enterprise processes. *Communications of the ACM*.

Espinosa, J., Slaughter, S., Kraut, R., & Herbsleb, J. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, *24*(1), 135–169. doi:10.2753/MIS0742-1222240104

Harmsen, A. F. (1997). *Situational Method Engineering*. Ph.D. Thesis, University of Twente.

Mensink, M. (2015). (Interview, January 20, 2015).

Nes, P. (2007). *Reference Models*. Msc. Thesis, University of Twente.

Osterwalder, A., & Pigneur, Y. (2009). *Business Model Generation*. Amsterdam.

Pajk, D., Indihar-Štemberger, M., & Kovačič, A. (2011). Enterprise Resource Planning (ERP) Systems: Use of Reference Models. In J. Grabis & M. Kirikova (Eds.), *Perspectives in Business …* (Vol. 90, pp. 178–189). Berlin: Springer Berlin Heidelberg. doi:10.1007/978-3-642-24511-4

Porter, M. E. (2008). *Competitive advantage: Creating and sustaining superior performance*. Simon and Schuster.

Rolland, C., Prakash, N., & Benjamen, A. (1999). A Multi-Model View of Process Modelling. *Requirements Engineering*, *4*(4), 169–187. doi:10.1007/s007660050018

Rothengatter, D. C. F. (2012). *Engineering situational methods for professional service organizations - AN ACTION DESIGN RESEARCH APPROACH*. Ph.D. Thesis, University of Twente.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Upper Saddle River, NJ, USA: Addison-Wesley.

Schoonhoven, C. B. (1981). Problems with Contingency Theory: Testing Assumptions Hidden within the Language of Contingency "Theory." *Administrative Science Quarterly*, *26*(3), 349. doi:10.2307/2392512

Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *MIS Quarterly*, *35*(1), 37–56. Retrieved from http://bada.hb.se/handle/2320/9888

Soffer, P., Golany, B., & Dori, D. (2003). ERP modeling: a comprehensive approach. *Information Systems*, *28*(6), 673–690. doi:10.1016/S0306-4379(02)00078-9

Svensson, C., & Hvolby, H.-H. (2012). Establishing a Business Process Reference Model for Universities. *Procedia Technology*. doi:10.1016/j.protcy.2012.09.070

Thomas, O., Horiuchi, M., & Tanaka, M. (2006). Towards a reference model management system for business engineering. In *Proceedings of the 2006 ACM symposium on Applied computing* (pp. 1524–1531).

UN. (2001). UN/EDIFACT Message CODECO Release: 01B. *United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport*. Retrieved from http://www.unece.org/trade/untdid/d01b/trmd/codeco_c.htm

Van Dongen, B. F., Jansen-Vullers, M. H., Verbeek, H. M. W., & van der Aalst, W. M. P. (2007). Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, *58*, 578–601. doi:10.1016/j.compind.2007.01.001

Weill, P., & Olson, M. H. (1989). An assessment of the contingency theory of management information systems. *Journal of Management Information Systems*, 59–85.

Weske, M. (2007). *Business Process Management*. Mairdumont Gmbh & Co. Kg.

Wieringa, R. J. (2003). *Design Methods for Reactive Systems: Yourdon, Statemate, and the UML*. Morgan Kaufmann Publishers.

Wieringa, R. J. (2012). The Design Cycle. Design Science Methodology. Lecture conducted from University of Twente, Enschede.

Appendix A

Container Booking Information

Container:	ASDF1231237
Carrier:	MAERSK
Reference:	1234
Invoice Reference:	1234
Bill of Lading:	
Pin:	123

Product	Weight	UNDG	Quantity
Friet	0	-	0

Truck Transportation

Truck:	-
Pick-up Location:	MCTstraat 21, 1234aa Bergen op zoom, Nederland
Pick-up reference:	-
Pick-up Window:	2014-12-01 00:00
Departure Time:	-
Transit Time (minutes):	1,800
Deliver Location:	A, door 1, asdf 12, 1234aa Rotterdam, Nederland
Deliver reference:	1324
Deliver Window:	-

Truck:	-
Pick-up Location:	A, door 1, asdf 12, 1234aa Rotterdam, Nederland
Pick-up reference:	1324
Pick-up Window:	-
Departure Time:	-
Transit Time (minutes):	-

- 1 -

Figure 24 - Example of a printout of a Truck transport order given to the trucker on hardcopy (page 1 of 2)

Deliver Location:	MCTstraat 21, 1234aa Bergen op zoom, Nederland
Deliver reference:	-
Deliver Window:	-

Figure 25 - Example of a printout of a Truck transport order given to the trucker on hardcopy (page 2 of 2)

Appendix B

Africo
Africostraat 1a
1234aa Rotterdm
Nedreland

Africostraat 1a
1234aa Rotterdm
Nedreland

-
-

Rotterdm - Nedreland

Bergen op Zoom - Nederland

Vervoerder is niet aansprakelijk voor kwaliteit of kwantiteit van goederen in deze container alsmede schade door verkeerde belading of manco's aan gebruikte container.

Containernummer: ASDF1234560
Laad/Los referentie: -
Uithaal Referentie:
22G1 (20' Dry van)
Zegel:

Net: -

Aankomst:

Vertrek:

Bergen op Zoom, 27-11-2014

Rotterdm,

Figure 26 – Example of a CMR printout (normally printed on the pre-printed template shown Figure 27)

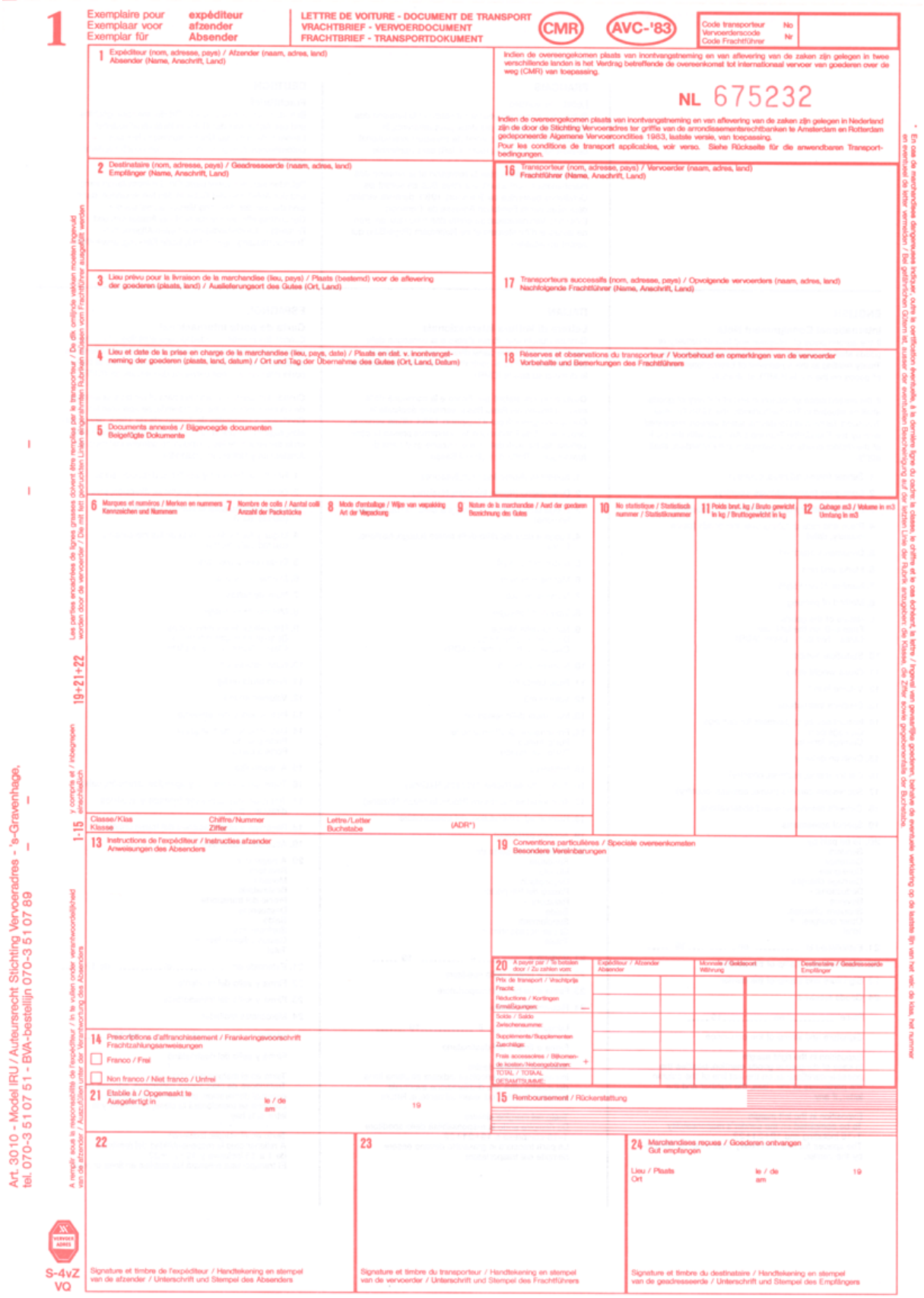

Figure 27 - CMR template (pre-printed on paper)

Appendix C

De volgende containers op terminal per 08-12-2014 12:45 bevatten gevaargoed

Container	Type	IN datum	Debiteur	Netto (kg)	Beschrijving	UN No.	Klasse	Cijfer	Etiketten
CRXU4785327	20G1	02-12-2014	MCT	1,000	HARS, OPLOSSING, brandbaar	1866001	3	F1	3
HLBU1270319	45G1	02-12-2014	MCT	1,000	TRAANVERWEKKENDE MUNITIE, met verspreidings-, uitstoot- of voortdrijvende lading	0018000	1	1.2G	1+6.1+8
HLBU1270319	45G1	02-12-2014	MCT	2,000	HARS, OPLOSSING, brandbaar (dampspanning bij 50 Â°C hoger dan 110 kPa) 640C	1866002	3	F1	3
HLXU4083242	20G1	02-12-2014	MCT	1,000	TRAANVERWEKKENDE MUNITIE, met verspreidings-, uitstoot- of voortdrijvende lading	0018000	1	1.2G	1+6.1+8
HLXU4083242	20G1	02-12-2014	MCT	2,000	HARS, OPLOSSING, brandbaar (dampspanning bij 50 Â°C hoger dan 110 kPa) 640C	1866002	3	F1	3

- 1 -

Figure 28 - Example printout of the environment list (a.k.a. 'milieulijst' in Dutch)