ELSEVIER

Expert Systems with Applications 26 (2004) 397–411

Expert Systems with Applications

www.elsevier.com/locate/eswa

An expert system for the best weight distribution on ferryboats

Khaled Shaalan[a,*], Mohammed Rizk[b], Yasser Abdelhamid[b], Reem Bahgat[a]

[a]*Department of Computer Science, Faculty of Computers & Information, Cairo University, 5 Tharwat St, Orman, Giza, Egypt*
[b]*Department of Computer and Information Sciences, Institute of Statistical Studies and Research (ISSR), Cairo University, 5 Tharwat St, Orman, Giza, Egypt*

Received 22 July 2003; revised 22 July 2003; accepted 22 September 2003

Abstract

There are some problems that need expertise in order to get a satisfactory solution. Ferryboat carries goods, fresh water, diesel oil, luggage and storing rooms up to its permissible draft in order to maintain safety according to the international safety regulations. The best weight distribution on ferryboat needs human expertise to handle many variables, such as the amount of the bunker and fresh water that allow us to use more rooms for charging in order to maximize the profit. This sort of problems can be classified under *Configuration Problem*. In this paper, we address the development of a ferryboat expert systems (WDFB) using CommonKADS knowledge engineering methodology. We propose a reusable problem-solving approach, which is an enhancement of the structure-oriented approach, capable of solving the ferryboat configuration problem. The proposed model includes heuristics that make the search of suitable configuration more efficient, taking into consideration the transformation knowledge and the optimality criteria. The results of testing the system on a real-world data from National Navigation Company, Suez, Egypt, were satisfactory.

Keywords: Expert systems; Configuration; Best weight distribution on ferryboat

1. Introduction

Shipping transport is considered to be one of the most important operations of a ferryboat as means of marine shipping, in addition to serving the economy either local or abroad. During the shipping operation, experience can be exchanged among countries. Marine investments have had a clear development and were subjected to some economical and technical circumstances. Commercial competition had arisen among companies regarding the freight and ideal operations of ferryboat.

The ferryboat carries goods, fresh water, bunker, luggage and storing rooms up to its permissible draft in order to maintain safety according to the international safety regulations.

Fig. 1 shows how to deal with these variables to use more rooms for passengers and cargo in order to maximize the profit. Consideration should be taken for determining the amount of bunker needed for sailing, the amount of fresh water for sailing according to the number of passengers (the constant like store rooms for food, equipment, crew weight), luggage dependant on the number of passengers.

In order to meet the international safety regulations and to maximize the profit, we need human expertise for expecting the number of passengers, luggage and fresh water per travel and their effect on the ferryboat draft that allows us to use more rooms for cargo.

In literature, this sort of problems can be classified under *Configuration Problem* (Zdrahal & Motta, 1995; Zdrahal & Motta, 1996). Traditional solutions of the configuration problem are not suitable for solving this type of problems. Consequently, there is a need to introduce a new reusable model that handles the transformation of knowledge and special type of knowledge that is derived from computation.

The main objective of this paper is to investigate how expert systems technology can be applied to ferryboat configuration problem.

We followed the CommonKADS methodology (Breuker & Greef, 1993; Breuker, 1997) for developing a set of reusable modeling components (Breuker et al., 1994a) that are capable of solving such a configuration problem. This model includes a proposed approach that handles the computation in objects that is a part of many types of

* Corresponding author. Tel.: +20-12-2223377; fax: +20-2-761-7628.
E-mail addresses: shaalan@mail.claes.sci.eg (K. Shaalan); mohammedrizk00@hotmail.com (M. Rizk); yasser@mail.claes.sci.eg (Y. Abdelhamid); rbahgat@rite.com (R. Bahgat).

0957-4174/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2003.09.010

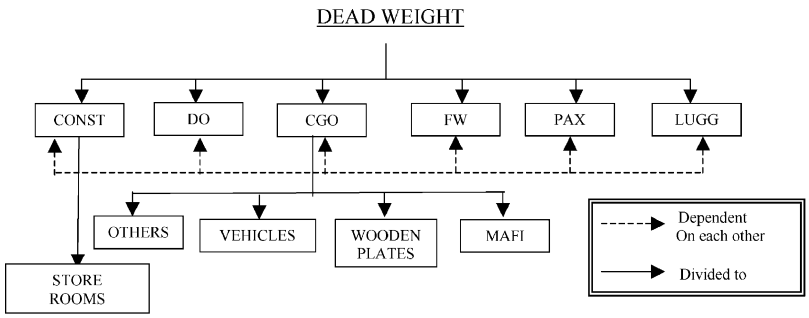


Fig. 1. Illustration of permissible weight for cargo and passenger ferryboat.

configuration problems, and development of heuristics that make the search for suitable configuration more efficient, taking into consideration both the transformation knowledge and the optimality criteria.

The paper is organized as follows. Section 2 is an overview of the configuration problems. Section 3 presents our proposed approach and the suggested problem-solving model, and its applicability to the ferryboat configuration problem. In Section 4, we show how we applied the new model for the best weight distribution on ferryboat (WDFB). Section 5 describes the design of CommonKADS expert systems for ferryboat configuration. Section 6 presents a case study and the calculation of the system. In Section 7, we present some concluding remarks.

2. Configuration problems

2.1. Basic concepts

The term *configuration* has at least three different meanings (Istenes, Tchounikine, & Trichet, 1996). It is used to denote a specific problem type, the process of solving problems of this specific type, and the product, which results from performing this process. For explicitly distinguishing these different meanings, we will use:

- the term *configuration problem* when talking about the problem type,
- the term *configuration process* when talking about the process of solving a configuration problem,
- the term *configuration* when talking about the product.

There are many different definitions of what configuration problems are, and what is rather something else. From these definitions of configuration problem, the following pieces of knowledge evolve as being inputs to solving configuration problems (Istenes et al., 1996):

1. The *components* are the objects of which the final configuration is to be composed. These objects have properties and form altogether a hierarchy of all the available objects.
2. The *compositional structure* gives all the relations between the components and the properties of the components:
   - *Relations among components* describe the connections that can possibly be made among components, that is, to which kinds of components a specific kind of component can be connected. The term 'connection' is used here in an abstract sense, i.e. it is not always meant to refer to a real-world object like a cable. Rather, by word 'connection' is meant that two objects are attached to each other.
   - *Relations among properties of components* describe definition-like dependencies among property values, e.g. 'the cubic capacity of a motor is the product of the cubic capacities of its cylinders'.
3. Constraints come from the environment of the configuration problem. They are either:
   - *Technical constraints*, e.g. 'component A which produces heat as a side-effect may not be located among components B and C which produce coldness',
   - *Legal constraints*, e.g. 'the space in front of a door in a passenger cabin of a plane has to be at least 50 cm wide', or the laws on the equipment of machine tools for human safety,
   - *Ergonomic constraints*, e.g. 'the kitchen block in the passenger cabin of a plane is to be located beneath the sanitary block in order to minimize the space need for the water supply system',
   - *Corporate specific constraints*, e.g. 'the company prefers whenever possible to make use of house products',
   - And any other constraints that have to hold for all configurations.
4. *User requirements* specify the characteristics of the configuration that user at that moment is looking for, thus they form the configuration specification. They restrict the set of all valid configurations to the set of all suitable configurations.

   The difference between constraints and user requirements is that the user requirements may differ each time ask for a new configuration whereas the constraints usually remain the same for quite some time. One could call the user requirements dynamic and parametrizing

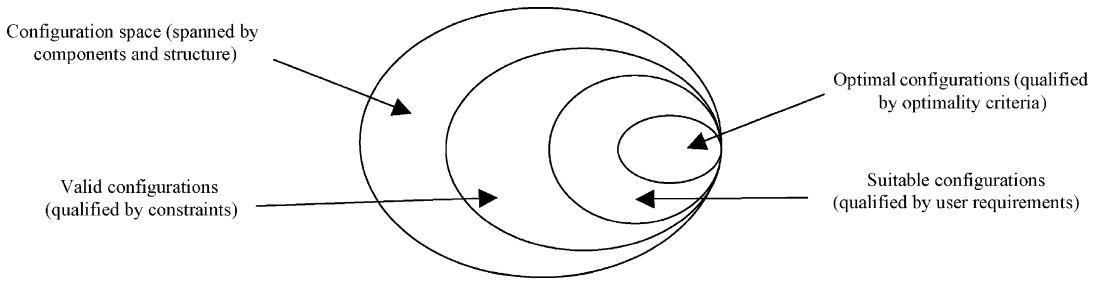


Fig. 2. Configuring understood as restricting the configuration space (Istenes et al., 1996).

the problem-solving process whereas the constraints have a more static nature.

5. *Optimality criteria* have to be differentiated according to their goal:
   - Some optimality criteria measure the distance between a proposed configuration specification (obtained from the user requirements) in order to decide whether the proposal satisfies the specification or not.
   - Other optimality criteria measure the difference between two configuration specifications in order to impose a structure on a library already solved configuration problems and to find the 'nearest' case in a library to the configuration specification at hand.
   - A third kind of optimality criteria is used for differentiating among several configurations which are all valid and satisfy the specification obtained from the user requirements, i.e. which are all suitable.

Thus, we distinguish between optimality criteria for measuring the distance during the search for valid configurations in order to enhance the configuration process and optimality criteria for selecting one of a set of suitable configurations. They restrict the set of all suitable configurations to the set for all optimal configurations.
Transformation knowledge specifies how the user requirements can be transformed or abstracted into a configuration specification. This step is necessary only when the user uses other terms for giving his requirements than those used for describing the components, compositional structure, constraints and optimality criteria.

The output of the configuration process is a (may be empty) set of suitable or, if optimality criteria are given, optimal configurations or one single suitable configuration. The empty set denotes that no valid configuration could be found which satisfies the user requirements. If more than one configuration remains, the configuration is underspecified, i.e. there are degrees of freedom in the user requirements.

Altogether, one can understand the configuration problem as a problem type where a configuration space, spanned by the components and the compositional structure, is restricted stepwise to find the proper configuration(s) (Garcia, de Andrade, Rodrigues, & Moura, 1997): the constraints restrict the space of all possible configurations to the set of all valid configurations, this set in turn is restricted by the user requirements to the set of all suitable configurations, which is finally restricted by optimality criteria to the final set of all optimal configurations, see Fig. 2.

2.2. The configuration function

Before we define the configuration function, we have to introduce two terms (Schreiber et al., 2000):

- The *configuration environment* encompasses all input knowledge that does not come from the user. Thus, the configuration environment contains up to six parts, according to the application, these parts can be classified as mandatory parts and optional parts. The mandatory parts are the components, the compositional structure, and the environment constraints. The optional parts are the transformation knowledge, the optimality criteria, and the heuristics.
- The term *solution space* is used for subsuming the different output possibilities of configuration, namely: a set of suitable configurations, a single suitable configuration, a set of optimal criteria—if optimality criteria are given—or a single optimal configuration. This term denotes the space of configurations to be found out as being suitable with respect to the user requirements and the configuration environment, or as in addition being optimal with respect to some optimality criteria.

The reason for introducing the latter term should be evident. The reason for introducing the former is much more complex. First, in the discussion of the inputs to solve configuration problems, we stated already that some input are optional (Chandrasekaran et al., 1993). If, for example, no case library is used, there is no need for specifying optimality criteria for measuring the distance between library cases and configuration specifications. If the user requirements can directly be used as the configuration specification, specifying transformation knowledge is unnecessary. Which input actually is necessary, depends

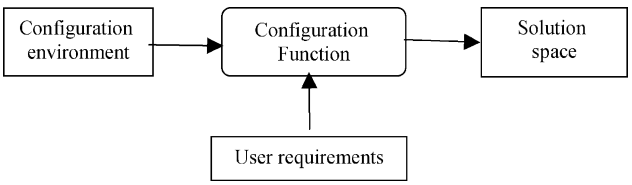


Fig. 3. The configuration function.

to a great extent on the problem-solving process. Thus, there is no such thing as a standard set of input knowledge pieces to the configuration process. Introducing the configuration environment as encompassing all knowledge from the environment of the configuration problem gives the opportunity to define an overall configuration problem type and to state what kinds of inputs are necessary for which approach to solve a configuration problem.

Second, if we leave all the user-independent inputs separated, the consistency of the inputs, i.e. that they are all phrased in the same terms, etc. is not guaranteed. Introducing a configuration environment that contains all inputs and from which they are extracted later on makes explicit that they belong together, are to be phrased using the same terms, and have to be consistent.

Having introduced the two terms, we can now define the configuration function as having the goal to find configurations (solution space) that satisfy some user requirements and are built according to the configuration environment, see Fig. 3.

2.3. Modeling components

There are three different classical approaches for solving the configuration problems. They are: case-based, resources-oriented and structured-oriented approaches.

2.3.1. Case-based approach

In the case-based approach, to solve a configuration problem, the configuration environment must contain a library of previously collected pairs of configuration specifications and suitable configurations. For the actual configuration specification, the most similar specifications in the library are determined. Its corresponding suitable configuration forms the library solution, and thus represents the first proposal for the actual specification.

Afterwards, the distance between the library solution and the actual configuration specification is determined according to some distance function, which must be part of the configuration environment. The distance is then used, together with the description of the components, the constraints, the compositional structure of the configuration environment and the actual configuration specification, to modify the library solution. Thus, we get the final set of suitable configurations.

There are some limitations concerning the applicability of this approach:

- There are many problem types that do not have a library of previously collected pairs of configuration specifications. Thus, this approach is used only for the problem type that has a configuration specification similar to the library specification, because this approach cannot be used for the problem type specification that does not match or similar to the library specification.
- Sometimes we need to have 'Transportation knowledge' component, to transfer the user requirement to the configuration specification.

2.3.2. Resource-oriented approach

When the components can be viewed as requesting and producing resources, suitable configurations can be determined by balancing requested and producing resources. Hence, it is very restricted by the nature of a special type of problems that are concerned with the production lines. Balancing is done in two steps. First, a local balancing takes place, which satisfies needs, afterwards a process of global balancing takes into account non-local needs. For optimizing the resource-oriented configuration process, one can induce:

- A priority per resource on these components that produce this specific resource. This priority usually reflects some cost/profit ratio of the producing components such that the cheapest components can be chosen to produce a requested resource.
- A linearization on the resources. This linearization reflects heuristics knowledge for choosing the best resource from a set of momentarily requested but not yet produced resources. The term 'best' refers to the fact that introducing a component that produces a specific resource imposes restrictions on further components by requesting other resources. Choosing the resource whose production imposes the strongest restrictions to be produced first often makes the search for suitable configurations faster.

Thus, the resource-oriented approach has four variations: either linearize the resources or not, and either define a priority on the resource producing components or not. Accordingly, there are four different function structures for performing the configure step in the resource-oriented approach:

- Only balancing,
- Defining priority + balancing,
- Linearizing + balancing,
- Defining priority + linearizing + balancing.

Suitable configuration in this approach is determined by balancing the requested and produced resources. On the contrary, the case-based approach, where there are suitable configurations in the library that the actual suitable configuration is determined according to some distance function.

This approach is very restricted by the nature of special types of problems that are concerned with production lines.

2.3.3. Structure-oriented approach

In the structure-oriented approach, it is assumed that a compositional structure for the components exists and encompasses all components, i.e. it imposes a compositional and taxonomical hierarchy on the components. The taxonomical hierarchy is mapped onto a graph structure. Finding a suitable configuration then corresponds to instantiating the graph according to the configuration specification, and a suitable configuration to an instantiated path in the graph (Teije & van Harmelen, 1996a,b).

Notice that the case-based and resource-oriented approach are instantiations of the problem-solving method 'propose and modify'. The third one, the structure-oriented approach is different in that it builds the solution in a bottom up manner.

The limitation of this approach is that it is not suitable for the type of problems that the user requirement needs to be transformed to configuration specification.

To sum up, each of these approaches has its strengths, limitations and geared towards a specific type of problems.

3. The proposed model

Our study indicates that the case-based approach and the resource-oriented approach are not suitable for the ferryboat configuration problem for the following reasons:

- The configuration environment does not contain a library of previously collected pairs of configuration specifications or suitable configurations, thus the case-based approach does not apply for this type of problems.
- The components of this type of problems cannot be viewed as requesting and producing resources. Thus, the resource-oriented approach is not suitable for this type of problems.

We found that the structured-oriented approach is a closely related model to configure the ferryboat problem. There is a compositional structure (relation between components and relation between properties of the component) exist and encompasses all components. However, there are needs for additional support to handle the limitations of this approach. This has led us to propose a new reusable model for ferryboat

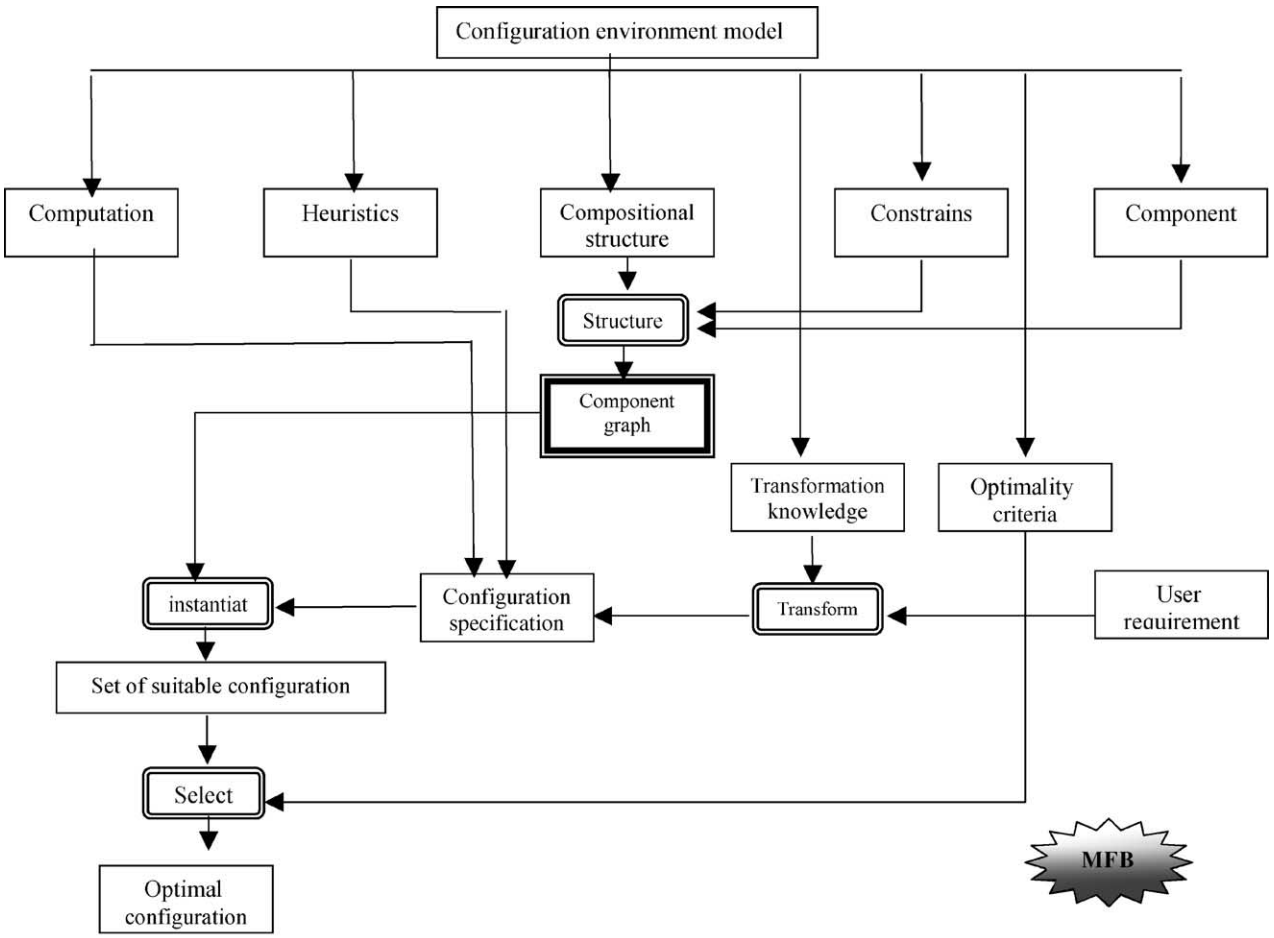


Fig. 4. The proposed architecture of reusable problem-solving model MFB.

configuration problem (MFB) that overcomes these limitations. This is shown in Fig. 4.

From this Figure we notice that the reusable MFB model specify the component, constrains, compositional structure, heuristics, transformation knowledge and computation in object. The following describes the entities of the new model:

- The compositional structure exists and encompasses all components.
- Constraints come from the environment of the configuration problem. The new model supports the technical, legal, ergonomic and corporate specific constrains.
- Heuristics make the search of suitable configuration more efficient.
- Specify transformation knowledge component to the user requirements that are needed to be transformed to the configuration specification.
- The configuration specification is collected from transformation knowledge (if the user requirements need to be transformed to configuration specification), heuristics and computation in object.
- The mandatory components are, namely: components, compositional structure, and constrain, structure the component graph.
- The suitable configuration is an instantiation of the component graph and configuration specification.
- If an optimality criterion exists, there is an optimal configuration from the set of suitable configurations.
- Both the instantiated graph and the configuration specification determine the set of suitable configuration.

Major benefits of the model

The following summarizes the major benefits of the MFB model:

1. The model addresses the type of problems where the user requirements need to be transformed into a configuration specification.
2. The use of heuristics makes the search of suitable configuration more efficient.
3. The model supports optimality criteria to select the optimal configuration.
4. The compositional structure encompasses all components.
5. Computation in object takes the knowledge that is derived from a computation or collected from a database and adds it to the configuration specification.
6. This model can be reused for similar type of problems, hence the reusability is realized. For example, this problem can be applied for the best weight distribution on airplanes.

4. Applying the MFB model for the best weight distribution on ferryboat

In this section, we will demonstrate the capability of our model in solving the best weight distribution problem using real-data taken from the National Navigation Company, Suez, Egypt. In particular, we cover 30 voyages of 'Wady el neel' Ferryboat.

Our problem is classified as MFB approach by traversing a decision tree trying to answer the following questions as follows:

- Is the application domain hierarchically structured? Yes
- Is there any computation in object? Yes
- Had the user requirements to be transformed first to configuration specification? Yes
- Is there a heuristic knowledge? Yes

In the following subsections, we specify the components of configuration environment model that contains components, constraints, compositional structure, heuristics and computation in objects.

4.1. Specifying components

The component found in our problem is as follows:

DW dead weight
Voy. No. voyage number
Pax. number of passengers
Luggage the passenger luggage on trailers 20 or 40 ft
FW fresh water in the tank
Vessel way the way of vessel if (Suez-Geddah or Suez-Safaga-Geddah… and so on
DO bunker in the tank
Cargo goods to be reserved
Banned goods list in banned goods in ports
Reserved goods list of reserved goods beforehand
Constant such as (store rooms, crew, etc. and so on)

4.2. Specifying compositional structure

Concerning our problem, the relations among components in the best WDFB are as follows:

Vessel way ⇔ bunker
Season ⇔ passenger
Passenger ⇔ luggage
Passenger ⇔ fresh water
Date ⇔ vessel way
Dead weight ⇔ all other components

The following are the relations among properties of ferryboat configuration problem components

Passenger.number ⇔ luggage.weight
Passenger.Number ⇔ fresh water.weight
Season.date ⇔ passenger.number

4.3. Specifying constraints

The constraints in the best weight distribution of ferryboat are classified under legal constraints as follows:

DWT (Dead weight): is the total lifting capacity of a vessel (i.e. its possibility to carry cargo, DO, stores, fresh water) = 1069 tons
Maximum permissible height of garage = 4 m
The maximum number of charge = 400 ton
Number of 20 ft trailers = 18 unit
Number of 40 ft trailers = 15 unit
Maximum tonnage of fresh water = 450 ton
Maximum tonnage of bunker = 200 ton

4.4. Specifying computation in object

We conducted an analytical study of 'Wady el Neel' departure and arrival voyages from January 1st, 2000 to October 9th, 2001. 'Wady el Neel' is a passenger ferryboat that is owned by the National Navigation Company and serves Suez-Geddah voyages passing by Safaga port. The sailing duration (SD) is 42 h for a single voyage. Besides, carrying passengers, 'Wady el Neel' carries also general goods as well. The data collected during this study considers the following items:

1. Voyage number.
2. Sailing duration.
3. Number of passengers for each trip (travel and arrival).
4. Total number of passengers.
5. Consumption of fresh water for each trip
6. Total consumption of fresh water.
7. Average consumption of fresh water for passenger.
8. Average consumption of bunker of ferryboat per trip.
9. Number of passengers luggage trailers (arrival voyages).
10. Average number of passengers per trailer (arrival voyages).

In addition, we rely on a recent study conducted by domain experts. This study considers the following items:

1. Cargo weight in accordance to fresh water.
2. Cargo reservation according to number of passenger.
3. Fresh water consumption (42 h).
4. Bunker consumption.
5. Average number of luggage trailer for arrival passenger.

From the collected data and the results of domain expert study, we can conclude the following results concerning SD, fresh water consumption (FW cons.), bunker consumption (DO cons.) and number of traveling passengers:

1. Average SD:
(a) Suez-Geddah: 1245/30 = 41.5 ≈ 42 h
(b) Safaga-Geddah: 173.5/6 = 28.9 ≈ 29 h
(c) Suez-Safaga: 45.5/3 = 15.16 ≈ 15 h
2. Average FW consumption:
(a) Suez-Geddah: 3885/30 = 129 ≈ 130 ton
(b) Safaga-Geddah: 621/6 = 103.5 ≈ 104 ton
(c) Suez-Safaga: 333/3 = 111 ton
(d) Geddah-Safaga: 1667/18 = 92.6 ≈ 93 ton
(e) Safaga-Suez: 881/14 = 62.9 ≈ 63 ton
(f) Geddah-Suez: 2263/19 = 119.1 ≈ 119 ton
3. Average DO consumption:
(a) Suez-Geddah: 1137/30 = 37.9 ≈ 38 ton
(b) Safaga-Geddah: 156.4/6 = 26.1 ≈ 26 ton
(c) Suez-Safaga: 46.3/3 = 13.4 ≈ 13 ton
(d) Geddah-Safaga: 496.1/18 = 27.5 ≈ 28 ton
(e) Safaga-Suez: 186.8/14 = 13.3 ≈ 13 ton
(f) Geddah-Suez: 708/19 = 39.5 ≈ 40 ton
4. Number of passenger study.

Also conducted detailed study on the passengers historical data in order to estimate the number of passengers for the next seasons.

As shown in Fig. 5, the peak comes in the seasons of Ragab, Shaban and Ramadan; when the number of passengers increases, a lot due to El-Omra.

(i) The next peak comes on Rabee 1st when people like to make El-Omra during the occasion of Prophet Mohamed's Birth ε.
(ii) Another smaller maximum comes in Heja month, which is the season for Hej. It is noticeably less than other maxima because Hej is much more expensive than Omra, so little people can manage Hej.
(iii) Very least number of passengers travel in the seasons of: moharam, safar, gamad 1st, and gamad 2nd because visas to Saudi Arabia are blocked in those months.

The set of suitable configuration is derived from the result of computation in objects component, heuristics, configuration specification and the instantiation graph.

4.5. Specifying heuristics

As shown in Section 2, heuristics component is used for making to search for a suitable configuration more efficient. Heuristics recommended by the domain expert study are as follows:

If number of passenger = 100 then amount fresh water consumption = 110
If number of passenger = 200 then amount fresh water consumption = 120

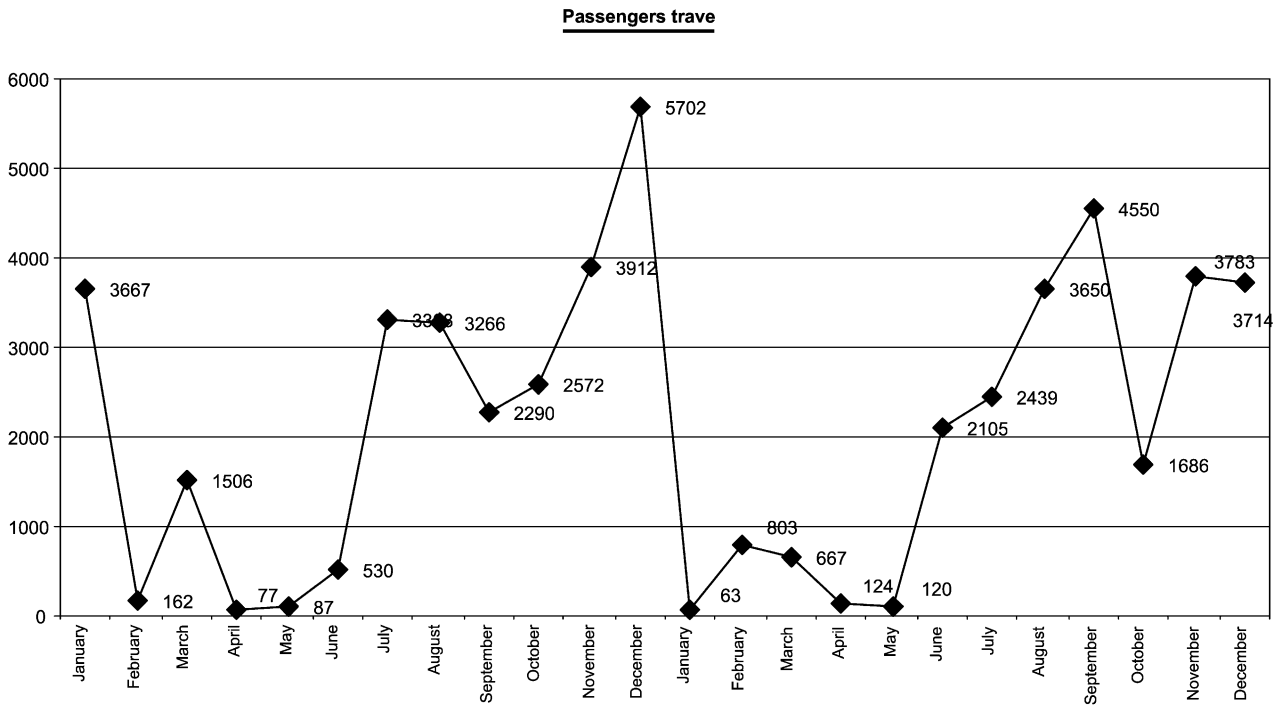


Fig. 5. Seasons of Hej and Omra during 2000–2001.

If number of passenger = 300 then amount fresh water consumption = 130
If number of passenger = 400 then amount fresh water consumption = 140
If number of passenger = 500 then amount fresh water consumption = 150
If number of passenger = 600 then amount fresh water consumption = 160
If number of passenger = 700 then amount fresh water consumption = 170
If number of passenger = 800 then amount fresh water consumption = 180
If number of passenger = 900 then amount fresh water consumption = 190
If number of passenger = 1000 then amount fresh water consumption = 200
If number of passenger = 50 (arrival Safaga) for normal trips then luggage = 1 × 40 ft
If number of passenger = 100 (arrival Suez) for normal trips then luggage = 1 × 40 ft
If number of passenger = 100 (arrival Safaga)for omra trips then luggage = 1 × 40 ft
If number of passenger = 200 (arrival Suez) for omra trips then luggage = 1 × 40 ft
If number of passenger = 75 (arrival Safaga) for Hej trips then luggage = 1 × 40 ft
If number of passenger = 150 (arrival Suez) for Hej trips then luggage = 1 × 40 ft

4.6. Specifying transportation knowledge

Transportation specifies how the user requirements can be transformed or abstracted into a configuration specification. This step is necessary because the user requirements of our problem need to be transformed into configuration specification. Concerning our problem, the configuration specification is as follows:

DO in tanks ⇐ 263 tons.
The expected time for discharging trailers in Geddah = 4 days.
FW in tanks ⇐ 490 tons, (adding 200 ton of FW for safety).
Weight of PAX (Suez-Jeddah) + luggage for Suez-Geddah trip = 0.1 ton
Weight of PAX (Jeddah-Suez) = 0.075 ton.

4.7. Specifying optimality criteria

The optimality criterion leads to optimal solution. Full cargo and passengers is an optimality criteria in ferrysboat configuration problem, thus the optimality criteria may be as follows:

- Number of passengers = 1200
- Weight of passenger + cargo = 400 ton

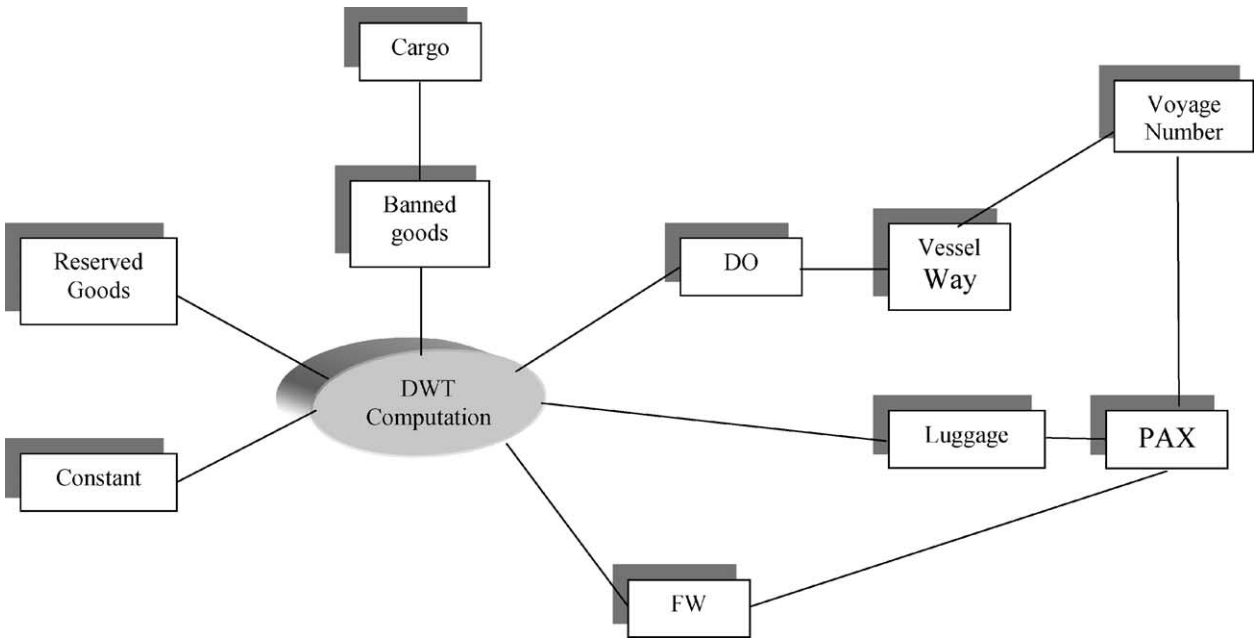


Fig. 6. Taxonomical structure.

4.8. Taxonomical structure

The taxonomical structure of the ferryboat problem, including all components and compositional structures, is shown in Fig. 6.

Fig. 6 shows all components of the ferryboat configuration problem and the relations among components. As shown there are 10 components, voyage number, number of passengers, luggage weight, amount of fresh water, vessel way, amount of bunker, cargo weight banned goods, reserved goods, and cargo. This figure shows the relations among components, for example, there is a relation between the vessel way and the amount of bunker, another relation is between number of passenger and fresh water consumption.

5. The design of CommonKADS expert system for ferryboat configuration expert system

This section introduces our design of a ferryboat expert system using CommonKADS methodology (Abdelhamid, El-Azhari, Hassan, & Rafea, 1999; Gennari, Grosso, & Musen, 1998; Breuker et al., 1994b). First, we describe the design of the domain layer, then we proceed to the inference layer, and finally the task layer.

5.1. Domain layer

The aim of knowledge engineer in this layer is to collect the domain specific knowledge about the problem and all the relations between the domain parts (Jansweijer, 1996; Sure et al., 2003).

The static knowledge of the domain is described in terms of concepts, properties, relations between concepts, and relations between property expressions.

Domain is represented in the form of classes (Brown, 1997) for each class we defined different properties. Each property has characteristics that can be specified through the use of facets. We used seven types of facets to describe properties, namely: Boolean, Date, Integer, Multi-value, Nominal, Real, and string. There are 10 concepts in our domain, namely: dead weight, trip, constant, bunker, cargo, trailers, fresh water, luggage, request reserved cargo, and passengers. The properties required to describe these concepts are acquired. Relations between these concepts that describe the dependencies of concepts in our domain are also acquired. In addition, we described for each relation between expressions, the functions that compute the value of property for a given concept.

Notice that the naming convention in CommonKADS for a property name takes the form: *Concept name property.*

Table 1
An extraction of concepts in the domain layer of ferryboat expert system

Primitive	Name	Description
Concept	Dead weight	Total lifting capacity of a vessel, i.e. its possibility to carry cargo, bunker, stores, fresh water
Property	Deadweight.weight	Dead weight has the property weight
Concept	Voy	The ferryboat trip
Property	Voy.path	The path of the ferryboat
Property	Voy.number	The number of voyage
Property	Voy.season	The season of voyage
Concept	Constant	The constant on the ferryboat
Property	Constant.weight	The weight of the constant

Table 1 shows an extraction of the domain schema of ferryboat expert systems. This table includes three concepts, namely: DW, Voy and constants.

5.1.1. Concepts in ferryboat expert system

The following example shows the concept Voy as described in the design document:

voy

Properties	Facets	
Voytype	Type	String
	PV	Hej trip
		Normal trip
		Omra trip
	SV	Default
Voydate	Type	Date
	Default	1/1/2003
	PV	
	SV	User
Voyno	Type	String
	Default	0
	PV	
	SV	User
Voypath	Type	String
	Default	Suez-geddah-suez
	PV	Suez-geddah
		Geddah-suez
		Geddah-safaga
		Suez-geddah-safaga-suez
		Suez-safaga-geddah-safaga-suez
		Suez-safaga-geddah-suez
		Suez-geddah-suez
	SV	Default

5.1.2. Relation between concepts

Fig. 7 shows the relations between concepts in ferryboat expert system.

5.1.3. Relation between expressions

See Tables 2 and 3

5.2. Inference layer

The inference defines the relations between problem parts called, meta classes (Ford & Topp, 1996; Decker et al., 2000) which may include more than one domain concept. During run time instances of meta classes created dynamically to represent the state of execution. Each relation is a step in the inference, which performs an action in the knowledge structure, which called knowledge source. Fig. 8 shows the inference structure of the ferryboat expert systems.

5.2.1. Meta classes

Cargo to reserve	Voy.date
	Cargo.description
	Cargo.banned goods
Weight account	Reserved cargo
	Constant.weight
	DWT.weight
Cargo reserved	Trailer.type
	Trailer.number

5.2.2. 5.2.2 Knowledge source

Name	Specify_a
Function	Voy number, path, and number of passenger are specified depending on voy date
Input	Voy date
Output	Voy path
	Voy number
	Number of passenger
Method	Match voy date with tourism agent table
Name	Specify_b

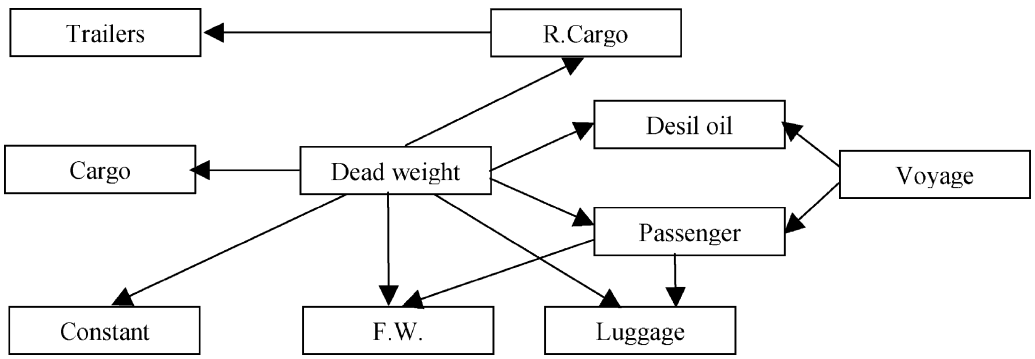


Fig. 7. The representation of relation between concepts of ferryboat expert system.

Function	The weight of fresh water and bunker and passenger is specified depending on voy path and number of passenger
Input	Voy number and path and number of passenger
Output	The weight of passenger, fresh water and bunker
Method	Match the number of passenger to the fresh water consumption

5.3. Task layer

The task layer describes recursive decomposition of a top-level task in subtasks, specifies what it means to achieve these tasks, and describes when these various subtasks are to be executed in order to achieve a parent task (Mitchell & Sundstrom, 1997; Stuckenschmidt & Van Harmelen, 2001).

In ferryboat expert system, the task knowledge consists of three parent tasks, The first one is DW calculation, second is cargo reservation, and the third is the voy (see Fig. 9).

Ferryboat weight distribution expert system is implemented using Knowledge Sharing and Reuse (KSR) tool (Abdelhamid, Hassan, & Rafea, 1997). KSR tool is an Expert system building shell, which has been developed using visual C++ to facilitate building the KADS-based expert systems.

6. Case study

In this section, we present the results of running and testing the weight distribution expert systems on a sample of real-world data gathered from National Navigation Company (NNC) for reserving goods in year 2003. The schedule of Wadi el neel ferryboat voyage number, date and path were acquired from tourism agent; taking into consideration the estimated number of travel passengers and its FW consumption, the bunker consumption, goods description and number of trailers. An advantage of WDFB over the existing manual system is that it can reserve goods in the coming trips. Section 6.1 gives a brief description of our system. In Section 6.2, we evaluate our system.

Table 2
Examples of relation between expressions in the ferryboat expert system

Name	Description
Passenger: Pax.number	The number of passenger determines the amount of fresh water and luggage weight.
Determine Fresh water: FW.weight Luggage: luggage.weight	
Trip: trip.path	The trip path determine the weight of bunker
Determine Bunker: bunker.weight	
Passenger: PAX.weight	The weight of passenger, bunker and fresh water determines the available weight of charging cargo.
Bunker: bunker.weight Fresh water: FW.weight Determine Cargo: cargo.weight	

Table 3
An example of a rule cluster that computes bunker amount (Cluster id: C1)

Rule id	LHS	RHS
R1	voy.voypath = "suez-geddah-suez"	bunker.weight = 78
R2	voy.voypath = "suez-safaga-geddah-suez"	bunker.weight = 87
R3	voy.voypath = "suez-geddah-safaga-suez"	bunker.weight = 87
R4	voy.voypath = "suez-safaga-geddah-safaga-suez"	bunker.weight = 95

LHS refers to the rule premise and RHS refers to the rule conclusion.

6.1. An Expert system for the best weight distribution on ferryboat: a brief description

WDFB is an expert system for handling the good reservation on ferryboats. The system reserves goods for one coming year; which is a major drawback in the existing system.

The input will be a selection of Gregorian reservation date. Then, the system will show the corresponding trip number and the corresponding Hijri date. According to this Hijri date, the number and weight of passengers and the FW amount will be expected. According to the trip number, both the trip path and the bunker amount will be determined. The total weight of the passengers, fresh water, bunker, and the constants are subtracted from

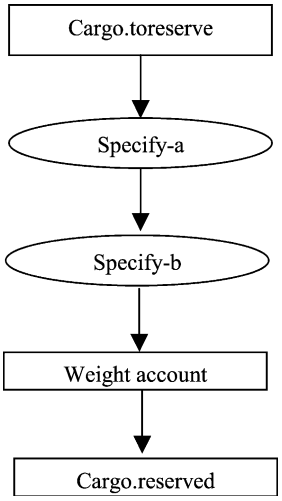
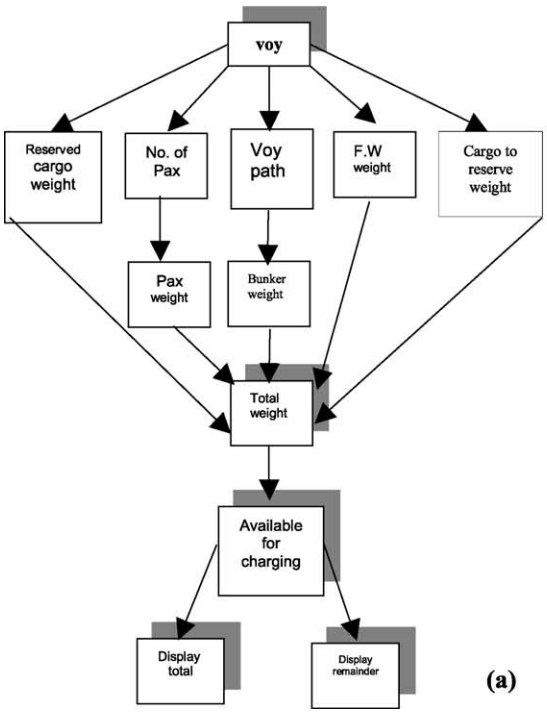


Fig. 8. Inference structure of ferryboat expert system according to CommonKADS. Rectangle represents meta classes; ovals represent inference steps. Arrows are used to indicate input–output dependencies.



(a)

Task: dead weight calculation

task definition: goal: get the available spaces for charging

input: cargo weight, voy date, number of passenger, weight of fresh water

output: Remainder spaces for charging.

task-body: type: composite

Subtask: get weight of passenger, get weight of bunker

Control-structure:

Repeat

Trip no.= the value takes place by the user

Get-weight (trip date hijri, trip way, Pax weight, F.W. weight, bunker weight, reserved cargo (weight, 20/40 feet))

cargo to reserve = the value takes place by the user (Type, weight, 40/20 feet)

Action = none % the action takes place by the user it may be exit, add, save, or none.

Total-weight (const +bunker weight + passenger weight + F.W. weight)

available for charging (dead weight – get_weight)

IF action = save THEN

Begin

Get_weight = get weight + cargo to reserve

Available for charging = DW – get weight

Display (Remainder), DISPLAY (No. of 40 feet trailers), DISPLAY (No. of 20 feet trailers)

End

(b)

Fig. 9. (a) Task structure. (b) Dead weight main task.

the dead weight to give the maximum permissible cargo weight. Fig. 10 shows the trip numbers on which the goods will be reserved and shows the corresponding Gregorian date and the trip path.

Fig. 11 shows the selection of the type of cargo to be reserved. Notice that the filled check box refers to banned goods.

The Best Weight Distribution of ferryboat expert system (WDFB) must ensure that the banned goods are not selected to accept this cargo.

Fig. 12 shows the input of the weight of cargo to be reserved and the type of trailers to be carried on. The total weight of the reserved cargo is subtracted from the maximum permissible cargo weight of the chosen trip

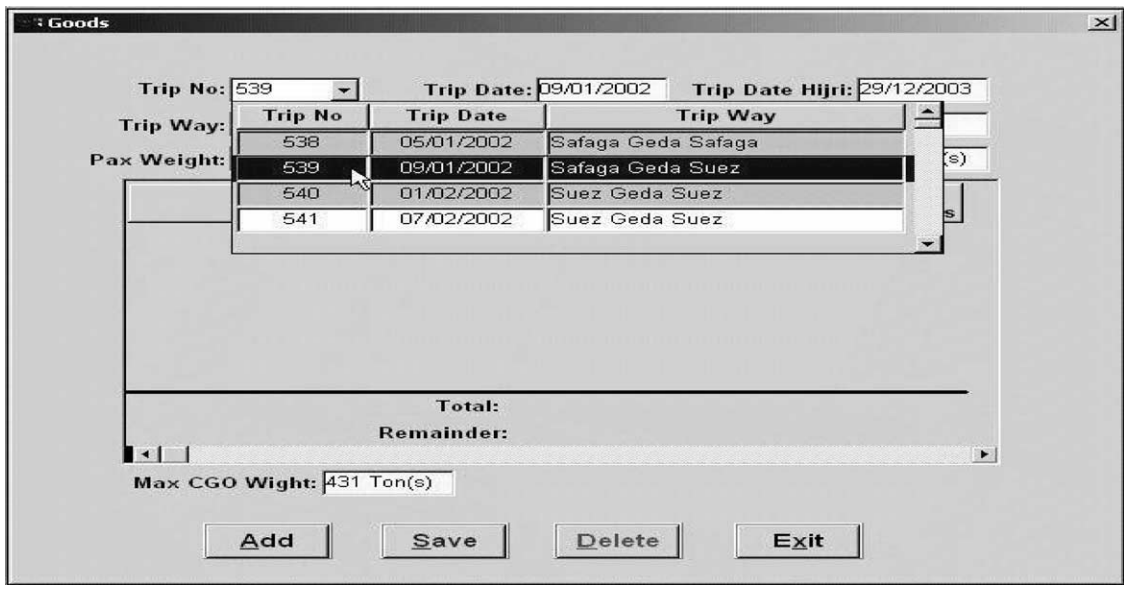


Fig. 10. Trip selection.

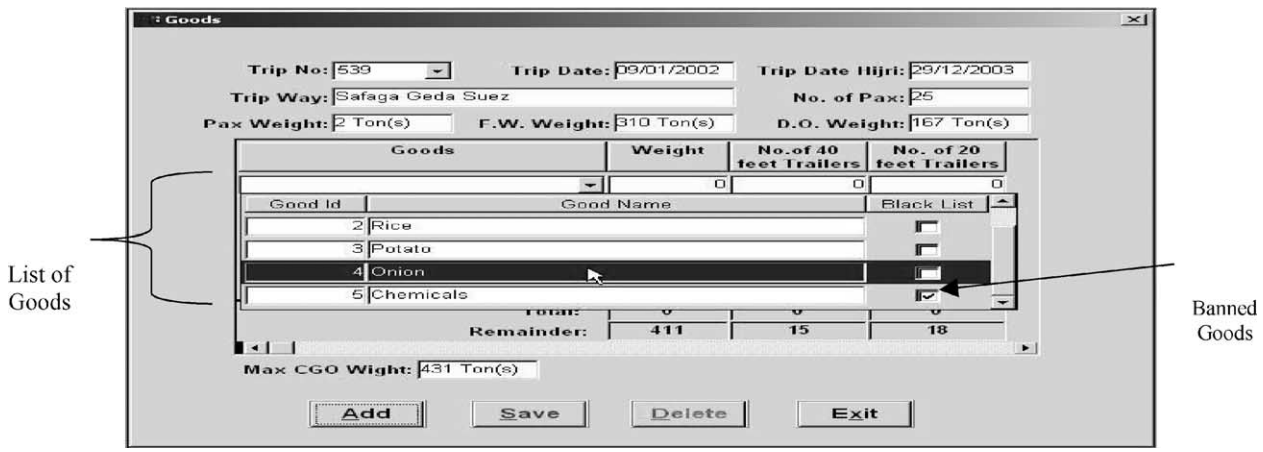


Fig. 11. Type of cargo.

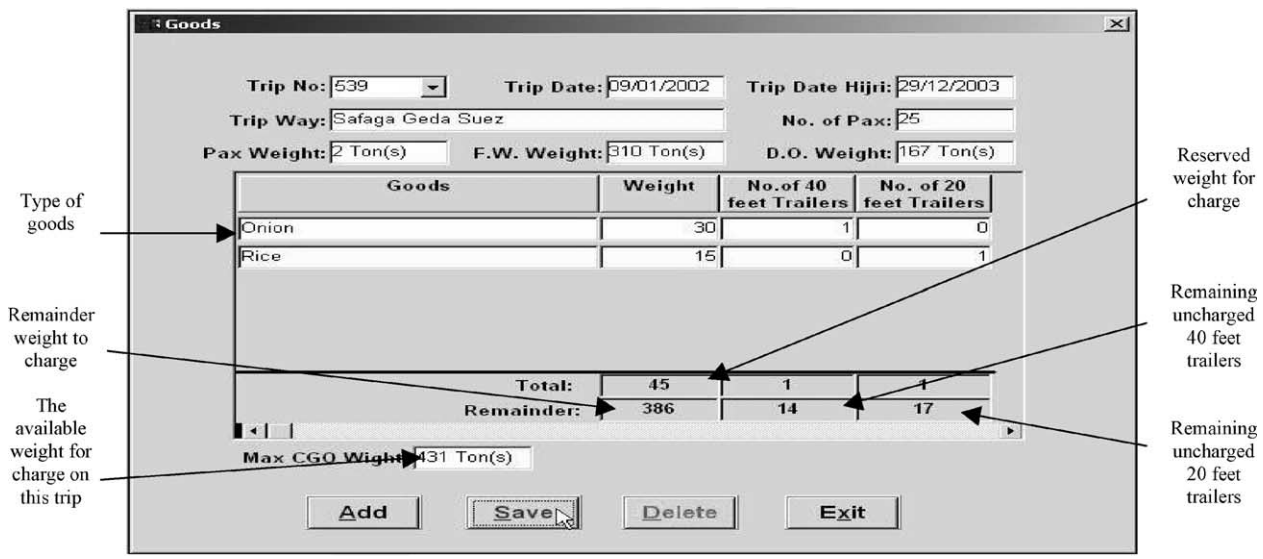


Fig. 12. Cargo reservation.

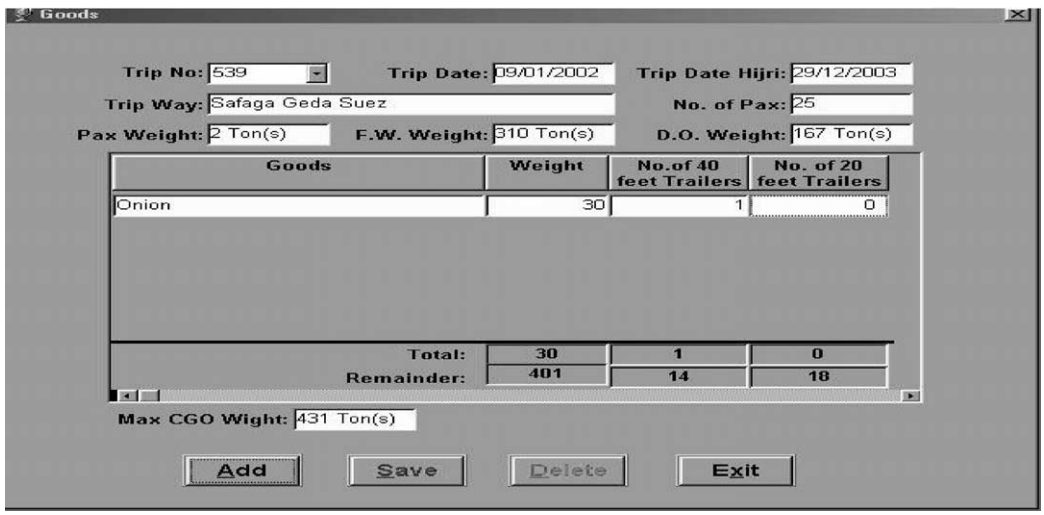


Fig. 13. Results of onion reservation.

and added to the list of reserved goods on this trip, and will be available to the coming reservation.

6.2. The evaluation of WDFB expert system

The evaluation is done by comparing WDFB results to those obtained from the same cargo reservation data on trip (539) of the existing manual system, such a comparison has been considered as means of evaluation of the WDFB system.

In this example, we assume that a client wants to reserve 30 ton onion on trailer with 40 ft, and the tourism agent reserved 25 passengers on this trip. The existing manual system goes into the following steps to reserve this trailer.

1. Weight of passenger $= 25 \times 0.75 = 2$ ton
2. FW consumption $= 400$ ton
3. Bunker on travel $= 200$ ton
4. The available weight of cargo to use $= 1069 - (400 + 200 + 159) = 310$ ton

The remainder weight available for charging $= 310 - 30 = 280$ ton.

The output of running these data using WDFB system is shown in Fig. 13.

The results of running WDFB expert system on these data for reserving the 30 ton cargo leave 401 ton available for charging. This shows that more rooms can be used for additional cargo compared to 280 ton that remains from the manual system.

There are two reasons for this difference in the cargo reservation.

First, the existing manual system fixing Fresh water amount for each trip, while WDFB takes fresh water depending on passengers consumption, this save more rooms for charging goods.

Second, the existing manual system fixing bunker amount for each trip separating from the path of the trip while WDFB takes bunker depending on the trip path, this also saves more rooms for charging goods.

7. Conclusions

This work has been concentrating on issues in the design and implementation of an expert system for the best WDFB. This type of problem is classified under the configuration problem.

We conducted a comparison between three different classical approaches for solving different types of configuration problems. We figured out that the structure-oriented approach is a closely related approach to configure the ferryboat problem. Consequently, we proposed a new approach, called ferryboat expertise approach (MFB) based on the structure-oriented approach that overcomes its limitation. The limitation of the structure-oriented approach is that it is not suitable for the type of problems that the user requirements need to be transformed to configuration specification. We followed the CommonKADS Knowledge engineering methodology for developing our best WDFB expert system.

KSR is successfully used to implement ferryboat expert system.

We compare between the Best weight distribution of ferryboat expert system (WDFB) and the existing manual system by a case study to evaluate the two systems.

The results of running the case study demonstrated the capabilities of WDFB in leaving more rooms available for charging goods and transporting passengers as compared to the existing manual system. This has an impact of maximizing the profit.

References

Abdelhamid, Y., El-Azhari, S., Hassan, H., & Rafea, A. (1999). *An expert system for cattle and buffalo health management. Seventh international conference on AI applications*, Cairo, Egypt: Egyptian Computer Society (EGS).

Abdelhamid, Y., Hassan, H., & Rafea, A. (1997). An approach to automatic KBS construction from reusable domain-specific components. *Proceedings of the ninth international conference on software engineering and knowledge engineering (SEKE'97)*, Madrid.

Breuker, J. A. (1997). Problem in indexing problem-solving methods. In R. Benjamins (Ed.), *Proceedings of the workshop on problem solving methods*, Nagoya.

Breuker, J. A., & Greef, P. (1993). Modeling system user cooperation. In A. T. Schreiber, B. J. Wielinga, & J. A. Breuker (Eds.), *KADS. A principled approach to knowledge engineering* (pp. 47–70). London: Academic Press.

Breuker, J., & Van de Velde, W. (Eds.), (1994a). *CommonKADS library for expertise modeling. Reusable problem solving components*. Amsterdam: IOS-press.

Breuker, J., & Van de Velde, W. (1994b). *Expertise model document. Part II. The CommonKADS library, KADS-II report KADS-II/TM2/VUB/TR/054/3.0,AI-Lab*. Brussels, Belgium: Vrije Universiteit Brussel.

Brown, D. (1997). *An introduction to object-oriented analysis*. New York: Wiley.

Chandrasekaran, B.a.T.R.J. (1993). Generic Tasks and Task Structures: History, Critique and New Directions, in Second Generation Expert Systems, J. David, J. Krivine, & R. Simmons (Eds.). Springer-Verlag. p. 232–272.

Decker, S., Fensel, D., Van Harmelen, F., Horrocks, I., Melnik, S., Klein, M., & Broekstra, J. (2000). *Knowledge representation on the web*.

Ford, W., & Topp, W. (1996). *Data structures with C++*. Englewood Cliffs, NJ: Prentice-Hall.

Garcia, A. C. B., de Andrade, J. C., Rodrigues, R. F., & Moura, R. L. A. (1997). *ADDVAC: Applying active design documents to the capture, retrieval and use of rationale during offshore platform VAC design. Proceedings of the 14th national conference on artificial intelligence and ninth innovative applications of artificial intelligence conference, AAAI-97/IAAI-97 Providence, RI, USA, July 27–31, 1997*, AAAI Press, pp. 986–991.

Gennari, J. H., Grosso, W., & Musen, M. (1998). *A method-description language: An initial ontology with examples. Proceedings of the 11th Banff knowledge acquisition for knowledge-based systems workshop, Banff, Canada*.

Istenes, Z., Tchounikine, P., & Trichet, F. (1996). *Using zola to model dynamic selection of tasks and methods as a knowledge level reflective*

activity. *In European knowledge acquisition workshop, Nottingham. Position papers*, pp. pages 42–53.

Jansweijer, W (1996). *Recommendations to EuroKnowledge*. KACTUS deliverable KACTUS-DOLF.1-UvA-V0.2, University of Amsterdam

Mitchell, C. M., & Sundstrom, G. A. (1997). Human interaction with complex systems: design issues and research approachs. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, *27*, 256–273.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., & Wielinga, B. (2000). *Knowledge Engineering and Management, the CommonKADS Methodology*. MIT Press.

Stuckenschmidt, H., & Van Harmelen, F. (2001). *Knowledge-based validation, aggregation and visualization of meta-data validation: Analyzing a web-based information system*.

Sure, Y., Akkermans, A., Broekstra, J., Davies, J., Ding, Y., Duke, A., Engels, R., Fensel, D., Horrocks, I., Iosif, V., Kampman, A., Kiryakov, A., Klein, M., Ognyanov, D., Reimer, U., Simov, K., Studer, R., & Van, J. (2003). *On-to-knowledge: Semantic web enabled knowledge management*.

Teije, A., & van Harmelen, F. (1996a). *Computing approximate diagnoses by using approximate entailment*. *In Proceedings of the fifth international conference on principles of knowledge representation and reasoning (KR'96), Boston, Massachusetts*.

Teije, A., & van Harmelen, F. (1996b). Using reflection techniques for flexible problem solving. *Future Generation Computer Systems*, special issue Reflection and Meta-level AI Architectures, in press.

Zdrahal, Z., & Motta, E. (1995). An in-depth analysis of propose and revise problem solving methods. In B.R. Gaines, & M.A. Musen (Eds.), *Proceedings of the ninth Banff knowledge acquisition workshop* (pp. 38.1–38.20).

Zdrahal, Z., & Motta, E. (1996). *Improving competence by integrating case-based reasoning and heuristic search*. *Proceedings of the 10th Banff knowledge acquisition for knowledge-based systems workshop*.