Applying Knowledge Engineering and Ontology Engineering to construct a Knowledge Base for Early Warning and Proactive Control

Yuan Li, Mark R. Kramer, Adrie J.M. Beulens, Jack G.A.J. van der Vorst
1 Information Technology Group, Wageningen University
2 Operations Research and Logistics Group, Wageningen University
Hollandseweg 1, 6706 KN Wageningen, The Netherlands
Yuan.Li@wur.nl

ABSTRACT
Early Warning and Proactive Control systems intend to help managers predict and prevent problems in Food Supply Chain Networks. Since those problems are of various types and have broad scopes, an easy-to-access and easy-to-extend Knowledge Base is needed. This paper presents a structure for such Knowledge Base. The structure is designed with combined techniques from Ontology Engineering and Knowledge Engineering. It contains an ontology and a rule base with inference mechanism. A prototype system, filled with information from case studies, shows that the designed structure supports consulting the Knowledge Base on relevant knowledge for encountered problems, and extending the Knowledge Base with obtained novel knowledge.

Keywords: knowledge based systems, ontology, rule base, food supply chain networks

1. INTRODUCTION
In Food Supply Chain Networks (FSCN), due to complexity in structure, variability in food quality and production, and uncertainty in influential factors, managers are facing various types of problems covering multiple network stages. Moreover, causes for those problems may originate from various sources, such as operations, environment, or product inherent quality. It is impossible for any single manager to possess knowledge of all types of problems in FSCN. Furthermore, new problems and new causes for problems occur frequently. This requires managers to have well-established procedures and facilities for problem investigation. Since problems in FSCN might cause hazards to human health and considerable losses to food industry and shareholders, those procedures and facilities should be efficient and effective.

In our research we aim at developing Early Warning and Proactive Control (EW&PC) systems to help managers in FSCN to predict, prevent, and correct problems. An important additional function of EW&PC systems is to support managers sharing obtained knowledge from data analysis, and using knowledge added by other managers. By comparing his own case with similar cases from FSCN, a manager can quickly locate potential causes and counter-measures for encountered problems. Whenever a manager encounters a problem for which no previous knowledge exists, the EW&PC system helps the manager to investigate the problem and store the obtained knowledge for later reference.

To meet the requirements of knowledge sharing, a Knowledge Base (KB) for problems in FSCN is indispensable [1]. A KB accommodates obtained knowledge from managers and enables managers to retrieve relevant knowledge for their cases efficiently. Due to various types of problems and variability of causes of problems, extensive knowledge will be put into such KB. Therefore, the KB should be designed in an easy-to-access and easy-to-extend way.

We apply results from current research and development in Ontology Engineering and Knowledge Management to build a structure for our KB. We separate knowledge we obtained in case studies into two types, namely manifest knowledge and inferred knowledge. Based on such separation, we build an ontology to model manifest knowledge, and construct a Knowledge-Based System to capture inferred knowledge. Furthermore, a prototype Knowledge Base has been built to link the Knowledge-Based System with ontology. Example results show that this structure is able to help managers. When managers come across problems in FSCN, the system helps them to specify the problem and obtain knowledge on causes and remedies. Managers can also quickly and correctly extend the KB when they obtain novel knowledge.

2. CONTEXT
This section first explains the concept of Early Warning and Proactive Control systems in FSCN, introducing a Knowledge Base as an essential component. Then we review existing research and development in areas that contribute to designing the KB: Knowledge Engineering and Ontology Engineering.

Early Warning and Proactive Control Systems
Early Warning and Proactive Control systems in FSCN are knowledge-based, data- and model-driven decision support systems that are designed for managers to predict and prevent problems associated with food products in FSCN. Managers in FSCN use EW&PC systems to deal with encountered problems by analyzing existing data sets. With quantitative methods managers explore causes for encounter problems, predict upcoming problems based on monitored status of those causal factors, and at the end evaluate different control actions to prevent problems, e.g. discarding products, taking corrective measures, or adapting succeeding processes in order to make amends. The knowledge obtained through data analysis is beneficial for other users that have similar problems. Therefore, EW&PC systems contain a Knowledge Base. Knowledge obtained during data analysis can be easily incorporated into the Knowledge Base. Users can share their knowledge through this KB.

Figure 1 shows a framework we designed for early warning and proactive control systems in FSCN [2].

![Figure 1: framework for early warning and proactive control systems in FSCN](image-url)
Users of such a system will be guided by Task classifiers to search for applicable causal relations from the Knowledge Base. If no applicable relation is found, users can choose appropriate DM methods to explore causal relation from recorded data. Template approaches will guide users in each step of applying DM methods. The Expert System (ES) for method selection suggests applicable DM methods from DM library. After obtaining causal relations, users can predict potential problems and consider different control measures to reduce or prevent losses due to problems in FSCN. Users can also store obtained knowledge about causal relations and control measures into the Knowledge Base to make the knowledge available for other users with similar problems in FSCN. In order to provide users with relevant knowledge, the Knowledge Base should enable users to browse, navigate through the KB, and compare their cases against the KB. Correctness of stored knowledge should be guaranteed, so the Knowledge Base has to store obtained knowledge correctly and in a comprehensible way, has to ensure that entered knowledge truly represents what users intend, and has to prevent typing mistakes and inconsistencies in naming.

However, two problems have to be dealt with in building a KB. First, the databases being analyzed by different managers are normally collected in different FSCN contexts (e.g. with different processes and different operations involved). Even if they refer to the same object in FSCN, the exact meaning of variables in different databases could be totally different. It is difficult for one manager to correctly understand semantics of databases that come from other managers. Second, the models built by managers through data analysis could have different forms. Those different model forms make it difficult for managers to quickly understand and locate relevant models. The models built by managers through data analysis could have different forms. Those different model forms make it difficult for managers to quickly understand and locate relevant models. This problem is further complicated by the first problem that variables used by one manager in problem solving, they share assumptions and requirements on the problem domain. An ontology captures the underlying conceptualization of those assumptions and requirements.

A variety of activities of Knowledge Management can be supported by ontologies, such as knowledge retrieval, storing, sharing, and dissemination. There are many methodologies, tools, and environments for Ontology Engineering. However, currently there is no definition and standardization of methodologies that drive the development of ontologies. For a comprehensive comparison of those methodologies and tools, please refer to Corcho et al. [14].

3. STRUCTURE DEVELOPMENT

Case studies
The knowledge base we build is based on three case studies in FSCN.

DOA case
This case is about a chicken supply chain. This supply chain has various stages, from hatchery to slaughter house. The monitoring system in this supply chain records data associated with various factors (operational, environmental, etc.) and properties of chickens in this supply chain. When we started the case study, there was a problem in this supply chain that too many chickens arrived dead at the slaughter house. This problem is called Death On Arrival (DOA). Managers involved in the supply chain were not sure about the cause of this problem, particularly because the level of DOA varied considerably between flocks and in time. In total, there are 63 attributes and 1357 records available in the data set. Each record corresponds to one flock of chicken. The whole data set has records from December 2004 to July 2005. After applying DM methods to analyze recorded data, relations were found between genetic factors, transport time, and DOA [15]. When the knowledge base is operational, knowledge obtained can be stored for later reference.

Management farm case
This case comes from a pig supply chain of a Dutch food company. In each stage of the breeding chain, the company recorded information about various attributes of pigs and the way they were managed. Slaughter house data of the pigs are also recorded, such as their meat percentage, muscle thickness, etc. In total 38 attributes were recorded. For research purposes such performance indicators were actually recorded at an individual level. And there are 9659 records available, ranging from October 2001 to October 2005. Through analyzing recorded data, we identified relations between inherent attributes of pigs (e.g. sex, genetics), operational factors, and performance indicators. Those relations contribute ample resources for ontology construction.

MAF case
This is a second pig supply chain case. Contrary to the previous case, this one concentrates on feeding regimes. Most data relate to the fattening farm and aggregated slaughter data. In each of the growth stages, the farms recorded information about feed and genetics. There are 31 attributes and 19390 records available, covering the whole year of 2006. Through analyzing those recorded data, we identified relations between feeding regimes, genetics and meat quality. These relations supplement relations found in the second case.

Knowledge Engineering
Knowledge Engineering aims at modelling various aspects of domain experts’ problem solving expertise, and hence producing Knowledge-Based Systems (KBS) to help non-experts in dealing with problems [3]. It intends to model not only the domain knowledge of experts, but also the inference procedures applied by those experts in problem solving. Knowledge Engineering has some overlap with Knowledge Management, which comprises of “a set of practices used by organizations to identify, select, organize, disseminate, and transfer important information and expertise that are part of the organization’s memory and that typically reside within the organization in an unstructured manner” [4]. Current research and development in Knowledge Engineering yields various methodologies to build knowledge-based systems, of which we use CommonKADS [3]. Knowledge Engineering techniques have been applied in various areas, such as hospital management [5], weight distribution on ferryboat [6], and classifier design [7].

Ontology Engineering
Ontology Engineering deals with organizing domain knowledge by formal descriptions of the concepts with their properties and relations. The aim is to improve knowledge sharing by structuring the concepts according to human understanding [8-10]. The resulting structures are called ontologies. As defined by Neches et al. [11], “An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.” With ontologies we can reuse knowledge in a knowledge base. The ontology in a knowledge base plays the role of its backbone [12]. When people use a knowledge base for problem solving, they share assumptions and requirements on the problem domain. An ontology captures the underlying conceptualization of those assumptions and requirements.

A variety of activities of Knowledge Management can be supported by ontologies, such as knowledge retrieval, storing, sharing, and dissemination. There are many methodologies, tools, and environments for Ontology Engineering. However, currently there is no definition and standardization of methodologies that drive the development of ontologies [13]. For a comprehensive comparison of those methodologies and tools, please refer to Corcho et al. [14].
Manifest knowledge and inferred knowledge

When organizing the Knowledge Base for EW&PC, we find that knowledge consists of two distinguishable kinds of knowledge. On the one hand there is domain knowledge on various stages in FSCN, involved entities, and their attributes and relations. On the other hand, there is knowledge on the influence between those stages, entities, and attributes. Such knowledge is normally obtained from either data analysis or investigation.

Therefore, we categorize the knowledge in the knowledge base into two types: manifest knowledge and inferred knowledge. Manifest knowledge represents the objects and their properties (e.g. about operations, staff, production means, environmental indicators, and performance indicators) and those relations between them that people can perceive without need for data analysis or inference. For example, there is a relation between ‘Chicken’ and ‘Genetics’ indicating the breed of chickens. Different people share the same perception upon such relations. This kind of knowledge is quite stable in the sense that it is not subject to influence from outside factors.

Unlike manifest knowledge, to identify inferred knowledge, people have to analyze data or employ domain experts. For example, the knowledge that ‘long transport time causes more DOA to chickens of breed Cobb’ is derived from data analysis. Inferred knowledge is a result of and contributes to the process of problem solving. Due to the fact that different data analysis methods may yield different outcomes, people may have varied perceptions of inferred knowledge. Such knowledge may need adaptation when it occurs in a different situation.

Ontology Engineering is suitable for modeling manifest knowledge. The predefined terms in an ontology (e.g. class, instance, properties) provide a way to capture features of objects and static relations between them. The class hierarchy in an ontology makes it possible to model the structure associated with semantics of databases in FSCN. With an ontology, users can share understanding of entities in FSCN during consulting or extending the Knowledge Base. They can correctly navigate to the relevant knowledge in the Knowledge Base. But ontologies are not well suited for representing inferred knowledge. It is difficult for different people to reach consensus on inferred knowledge. Moreover, inferred knowledge is more likely to change than manifest knowledge.

In stead of using an ontology, we model inferred knowledge with facilities provided by Knowledge Engineering. In systems built with Knowledge Engineering techniques, one of the essential components is a rule base. Rules in the rule base capture expert knowledge in problem solving. Since normally that kind of knowledge is obtained indirectly (e.g. from data analysis or expert interviews), rules provide a more mature mechanism to model inferred knowledge. Modification of rules will not harm the consistency of a built ontology. Furthermore, rules may contribute to extra functionality, such as inference for problem investigation.

As a result, we first employ Ontology Engineering techniques to build an ontology that describes manifest knowledge in the Knowledge Base. The ontology captures such aspects as the classification of various stages in FSCN and relations between different processing stages. Then we use Knowledge Engineering techniques to model inferred knowledge. Figure 2 gives an example of this idea. ‘Chicken’ and ‘Transport’ are two objects in the ontology for FSCN. From data analysis and expert interviewing, we found a rule ‘transport time is longer then 2.4 hours and chickens have genetics Cobb is a cause for DOA rate larger than 0.5%’. This rule is based on the objects in the ontology. It describes the inferred relations between the two objects ‘Chicken’ and ‘Transport’.

Modeling manifest knowledge and inferred knowledge with Ontology Engineering and Knowledge Engineering techniques separately has a number of advantages. Firstly, knowledge is easy to access. The rule base makes it possible to use inference mechanisms to search all relevant rules for causes that explain some observed symptom. Such causal knowledge contributes to what is called ‘diagnose’ type of applications in CommonKADS. The existence of different types of rules enables a reasoning process containing multiple steps for problem solving, thereby providing users with more extensive knowledge. The ontology, which is the foundation for the rule base, prevents users to misunderstand the rules in rule base, and enables them to navigate through the class hierarchy for exploring relevant knowledge.

Secondly, the separation of manifest knowledge and inferred knowledge facilitates Knowledge Base designing. The clear distinction between knowledge of different types helps designers in Knowledge Modeling. Also, designers can take advantage of developments in both Ontology Engineering and Knowledge Engineering to ease the design process. In addition, such a separation makes it possible for designers to build a rule base using existing ontologies.

Thirdly, such a structure makes a Knowledge Base easy to extend and maintain. When users have generated different rules, they can incorporate them into a rule base instead of changing the ontology. The ontology helps users to find correct terms that constitute target rules. Therefore, added rules can truly reflect what users intended to add. Changing rules in the rule base does not affect other rules, whereas changes to the ontology might influence any existing rules. Designers can even put restrictions on the rule base in order to ensure consistency of new rules incorporated.

![Figure 2: illustration of relations between rule base and ontology](image)

To build ontologies for manifest knowledge, we mainly follow the process provided by METHONTOLOGY, one of the most mature approaches for ontology construction [14]. METHONTOLOGY prescribes a set of activities for building ontologies, from objectives specification, through knowledge acquisition, conceptualization, implementation, to evaluation.

To model inferred knowledge, we use the methodology CommonKADS [3] to build a Knowledge-Based System. CommonKADS provides methods to analyze knowledge-intensive tasks and processes, and supports the development of Knowledge-Based Systems that support selected parts of the business process. Contrary to simple ‘if-then-else’ rules, CommonKADS enables designers to define different types of rules in order to capture different kinds of knowledge in a problem domain. Furthermore, CommonKADS provides a suite
of models to capture different aspects (inference steps, tasks, etc.) of expert problem solving process. The inference structure and task models in CommonKADS provide a way to utilize constructed rule bases, thereby they facility the creation of systems to help users solve problems.

Ontology

In this section we present an ontology build from our case studies. We developed this ontology with OWL (Web Ontology Language) in Protégé 3.3 beta [16]. Besides sufficient expressive power, OWL has well-defined syntax and formal semantics and hence provides efficient reasoning support [17]. Protégé comes with an IDE for ontology construction (with tabs, widgets, menus, etc.) developed in Java. It is open source so its Java APIs can be easily used in other applications. Furthermore, Protégé is backed by a large community of active users and developers, and the feedback from this community is valuable for its further development.

Figure 3 shows the major class hierarchy of the ontology we build for the Knowledge Base in EW&PC systems (‘class’, ‘instance’, ‘property’ are terms for types of components in an ontology). Relations between classes can be either subclass relations or property relations. The subclass relation, denoted by a solid arrow, means that the class is a specialization of the other class. The property relation, denoted by a dashed arrow, indicates that a property of one class takes instances of another class as its value set. The name of the property is shown by a term beside the arrow.

![Figure 3: part of the class hierarchy in the ontology built for the knowledge base](image)

The ontology in Figure 3 has the following major classes:

- **Organization_unit**: Organization units represent various kinds of human organizations in FSCN, such as farmer, feed producer, slaughter house, and production manager.
- **Production_process**: This is the class for various processes in food production. The processes in chicken production include: hatchery, breeding, transport, and slaughter. Pig production has its own types of processes.
- **Production_means**: This class represents facilities used in food production along the FSCN, such as feed, heating equipment, machines, and medicine.
- **Production_object**: Production objects represent food products being produced. Since we have cases in a chicken supply chain and pig supply chains, we introduced objects_in_chicken_production and objects_in_pig_production as two major subclasses. In the chicken supply chain, production objects include chicken and egg.
- **Genetics**: This class represents genetics of animals in FSCN. It has two major subclasses, chicken_genetics and pig_genetics.

Each class in the ontology contains some properties. For example, from the case study we can define following properties for class chicken: has_body_weight, has_genetics, has_transport_time (from farm to slaughter house), and DOA_percent. The property has_genetics is shown by a dashed arrow in Figure 3, to illustrate that its value set consists of the instances of class chicken_genetics.

Knowledge-Based System

In order to model inferred knowledge in problem solving, we apply CommonKADS to build the Knowledge-Based System. This system is used to infer required knowledge from user input and internal rules. Figure 4 shows the inference structure of this system. This particular inference structure is a variation on the ‘diagnose’ template from CommonKADS.

![Figure 4: inference structure for Knowledge-Based System for EW&PC systems](image)

Square boxes represent objects in the Knowledge-Based System, such as problem, cause, suggestion. Ovals represent functions within the system. Each function has inputs and outputs, e.g. the function cover takes the specified problem and causal rules as inputs, and outputs a set of known hypotheses about the causes of the problem. The rounded box represents a function that needs information from outside of the system. The function obtain poses questions to the user and collects his answers into the system. Items within parallel lines represent types of rules in the rule base. In this system, there are two types of rules, causal rules and remedy rules.

- **Causal rules** capture knowledge about causal factors of problems in FSCN. For example, the causal rule “chicken has genetics=Cobb & transport.transport_time>2.4hrs \[17\]” means that if chickens with...
• Remedy rules contain possible solutions for the identified causes. For example, the remedy rule "transport.transport_time>2.4hrs HAS_REMEDY transport.departure_time<=6am" suggests to transport before rush hours.

To start the inference procedure, the user should specify the problem encountered. Then the function cover looks into causal rules for hypotheses on possible causes. For each potential cause in the generated hypotheses, the system asks the user to input its status in the case at hand. With hypotheses and obtained findings, the system can verify the validity of hypothesized causes. With verified causes and remedy rules, the system appoints all available suggestions on how to solve this problem.

4. APPLICATION

To illustrate how the presented structure helps achieving the requirements specified in section 0, we have built prototypes user interfaces to link the ontology and the Knowledge-Based System. These prototypes have been built using Java and Protégé APIs.

Accessing the Knowledge Base

When managers encounter problems in FSCN, they can specify the problem by choosing corresponding class, property, and value from the ontology. The sequence of dialog windows in Figure 5 shows the interfaces for choosing class, property, and value.

First, users have to select the class denoting the domain of the problem. As shown in the first dialog window, the class chicken is selected. Then the relevant property should be selected: in the second window, property DOA_percent of class chicken is selected from the four available properties. After that, its value set is shown in the third window. In this example the value set is enumerated as "\(<0.5\%" or "\(>0.5\%". If "\(>0.5\%" is selected for the value of chicken.DOA_percent, the problem has been fully specified as chicken.DOA_percent>0.5%. The specified problem will be shown in a text field of the user interface, as illustrated at the left hand side of Figure 6.

![Figure 5: Interfaces for selecting class, property, and value](image)

The system searches the rule base for rules that apply to the specified problem. All constituent clauses of those rules are presented to the user, who has to answer whether they are satisfied in his case. The presented clauses correspond to hypothesized causes for the specified problem. Based on the answers that the user gives, the system determines causes for the given problem and suggests solutions. As shown in the right pane of the user interface, the system reports that the cause for the given problem is chicken.has_genetics=Cobb & chicken.transport_process.transport_time>2.4hrs. It also makes suggestions for possible solutions, in this example to start transportation before rush hours.

Extending the Knowledge Base

The structure designed in this paper eases extending the Knowledge Base with obtained knowledge. Mostly, extending the Knowledge Base means adding rules to the rule base. Each rule in the rule base consists of items that directly correspond to objects (classes, properties, instances) in the ontology. Therefore, a user can simply select relevant objects from the ontology to construct a rule. Furthermore, the ontology facilitates that users extending the rule base share an understanding about objects and terminology in FSCN. Therefore, extending the Knowledge Base can be achieved by a few mouse clicks, eliminating the chance for typing mistakes. As shown in Figure 7, the user interface for extending KB enables users to add, edit, or delete rules from the rule base. By choosing corresponding classes, properties, and values, users can generate front part and rear part of a rule. The selection interfaces are the same as those shown in Figure 5. Rule types (either "CAUSE" or "HAS_REMEDY" type) can also be selected from a list. After the user confirms the constructed rule, it will be added to the rule base as shown in the right pane of the interface.
Sometimes, it is necessary to add items to the ontology as well. Extending the Knowledge Base in this sense can be realized by using the Protégé IDE. Because such additions are less frequent than changes to the rule base, we did not design a special purpose user interface for this part in our prototype.

5. CONCLUSION AND DISCUSSION

The aim of our research is contributing to solving problems in FSCN by designing EW&PC systems. One of the essential components of such systems is a Knowledge Base that enables knowledge reuse and knowledge sharing among managers in FSCN. In order to facilitate designing a Knowledge Base, we propose a structure for such Knowledge Base. The structure separates manifest knowledge from inferred knowledge, and represents them with ontology and rule base respectively. As a result, designers and users can take advantage of current developments and advances in the areas of Ontology Engineering and Knowledge Engineering. To test whether this structure supports accessing and extending the Knowledge Base, we have built a prototype based on this structure. We filled the ontology and rule base with knowledge obtained from case studies in FSCN, and experimented with changing the KB contents. Results show that it is possible to obtain relevant knowledge from this KB for encountered problems in FSCN, and to extend this KB with obtained knowledge.

This structure is a first step towards a comprehensive Knowledge Base for EW&PC systems in FSCN. The prototype supports extending the Knowledge Base, but changing existing information in the Knowledge Base is not yet well-supported. Changing existing rules is relatively straightforward. However, when the user wants to change items in the ontology, consistency and correctness of the rules may become jeopardized. Further research is needed on how modification in the ontology influences rules in the rule base. The target is to devise automatic routines for changing the rule base when the ontology is changed.

Within the current design, there are still interesting issues to research. For example, how to maintain the consistency of the rule base when new rules are added? Duplication of rules is easy to detect and solve. For conflicting rules there could be several solutions, however. Should the system retain the conflict and leave it to the user to choose, or should some mechanism prevent conflicting rules to be added at all?

In the larger context of EW&PC systems for FSCN, the Knowledge Base presented here is one of the components of a general framework for such systems. As Figure 1 shows, a complete EW&PC system would contain other components as well. However, without the Knowledge Base it would be impossible to share findings between managers in FSCN by means of EW&PC systems. For every new problem encountered in FSCN, managers would have to supply information about objects and rules again.

Reference: