Case-Based Reasoning in a System Architecture for Intelligent Fish Farming

Axel TIDEMANN [a], Finn Olav BJØRNSON [a], Agnar AAMODT [b]
[a] *SINTEF Fisheries and Aquaculture AS*
Trondheim, Norway
[b] *Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway*

Abstract. Fish farmers manage assets of considerable value on a daily basis. Many aspects of the daily operation are automated in some way, such as the feeding system. Sensory equipment steadily becomes cheaper and more ubiquitous, yielding data that can be used by automated systems and for post-processing (i.e. data mining) to discover hidden trends in the data. However, a lot of information is only known informally by the fish farmers themselves, through years of experience. Companies that can store this information and reuse it will have an advantage; even more so if high-level human expertise can be linked to low-level sensor data. This paper presents early developments of a system that stores this informal knowledge using case based-reasoning, combined with corresponding sensor data.

Keywords. Case based-reasoning, Decision support system, Industrial application, Machine learning

1. Introduction

Norway is among the biggest exporters of farmed fish in the world. In 2009, the total export value of Norwegian farmed seafood was NOK 26 billion[1] (roughly €3.3 billion/US$ 4.4 billion). An average site contains biomass worth around 100 MNOK (12.7 million €/17.2 million US$). The fish farmer has practical knowledge of how to operate and run the site, which is not stored formally. When managing assets of such value, there are clear advantages to storing and re-using these experiences. There are three main areas where the advantages of such a system could manifest: increasing revenue, improving fish welfare and reducing environmental impact.

The research presented in this paper aims to implement such a system. To store and retrieve similar experiences, case-based reasoning (CBR) is employed [1]. CBR is both a problem solving and learning technique in which the knowledge of the system grows over time as new experiences (or, more formally, *cases*) are added to the system. The CBR mechanism will be able to utilize the rich sensor information acquired from various sensor platforms on the site. In the architecture presented machine learning techniques are applied to the sensor datasets, and if particular patterns are discovered, they can be linked to human experiences. The approach intends to bridge the gap between high-level

[1] www.fisheries.no/aquaculture/facts_statistics/Export-trends-for-2009/

human expertise and low-level sensor data. The system can then be used as a decision support system (DSS).

This research is part of an ongoing project called Simulation and Optimization Framework (SimFrame), which in turn is part of the CREATE (Centre for Research-based Innovation in Aquaculture Technology)[2] programme, hosted by SINTEF Fisheries and Aquaculture[3]. CREATE involves several industry partners and research institutions that influences the direction of the research conducted in SimFrame. The main partners are AKVA Group[4], Egersund Net AS[5], Erling Haug AS[6], NOFIMA Marin[7], the Norwegian University of Science and Technology[8], Institute of Marine Research[9] and SINTEF ICT[10]. The top three Norwegian salmon farming companies Marine Harvest[11], Lerøy Seafood Group[12] and SalMar[13] are new partners from 2011. The goal is to integrate several independent tools and models that are used in modern fish farming, so that the process can be both simulated and optimized. One direct result, and the focus for this paper, is the development of a decision support system for operational planning on a fish farm.

The paper continues with a background section, before detailing the general architecture for intelligent fish farming. Subsequently, implementations and preliminary results will be presented and discussed, before pointers to future work are addressed.

2. Background

Concerning expert systems, Metaxiotis et al. [10] surveyed expert systems and their role in production planning and scheduling. They conclude that expert systems are generally perceived to be very useful in production planning and scheduling. The benefits reported from the use of expert systems include more accurate decisions, time gains, improved quality and more efficient use of resources. They also believe that the usefulness of expert systems can be improved if they are integrated with operations research techniques like simulation. Liao [7] reviewed expert systems and their applications from 1995-2004. He concludes that expert systems methodologies are tending to develop towards expertise orientation and that expert system application development is a problem-oriented domain. He further suggests that different social science methodologies, such as psychology, cognitive science, and human behaviour could implement expert systems as another kind of methodology.

In CBR a computer model is built up of a set of concrete past situations, called cases, stored in a knowledge base referred to as a case base. A case in its basic form has two parts, a problem description part and a problem solution part. The problem description

[2]www.sintef.no/Projectweb/CREATE
[3]www.sintef.no/Home/Fisheries-and-Aquaculture
[4]www.akvagroup.com
[5]www.egersund-traal.no
[6]www.haug.no
[7]www.nofima.no
[8]www.ntnu.no
[9]www.imr.no
[10]www.sintef.no/Home/Information-and-Communication-Technology-ICT
[11]www.marineharvest.com
[12]www.leroy.no
[13]www.salmar.no

part constitutes the set of input features to the reasoning process, while the problem solution part is the output from the system's reasoning process and hence the system's suggested solution to the problem. A third part is often added: outcome, i.e. the result after having applied the solution to the problem. Reasoning methods of similarity assessment, pattern recognition, and analogical mapping, rather than theory-driven methods, operate over this knowledge base of cases. A CBR process contains four steps: 1) A new problem is solved, or a state is interpreted, by finding a past case in the case base with a problem description part that matches the current state to a sufficient degree. 2) The problem solution part of the matching case is then reused, either by reapplying the past solution as it is, or modifying it to better fit the current situation or problem. The latter two steps enables the system to learn from its problem solving attempt: 3) The solution proposed by the system is evaluated in some way, e.g. by being applied to the problem, assessed by an expert, or run in a simulation model. 4) The solution, possibly revised after evaluation, together with description part of the new problem and the possible outcome, becomes input to the final step, in which the system learns from the problem solving session by storing a new case or making other updates in the case base. These four steps are often named Retain, Reuse, Revise, and Retain, respectively [1].

CBR, which is central to our implementation, is considered a lazy learner. As opposed to eager learners, which construct generalizations of observed data at learning time, lazy learners postpone the generalization step until problem solving time. The generalization step in CBR is implicit in the partial matching method that enables an input case to retrieve the best matching past case. The advantage is that this generalization is made on the basis of as much information about the problem as possible, i.e. the information available when a problem situation has been presented. Eager methods, on the other hand, do not have that information, and have to make assumptions about the real problem situation which may or may not be correct. When making a classification or prediction, lazy learners can be computationally expensive. They require efficient storage techniques that may be suited to implementation on parallel hardware. They offer little explanation or insight into the structure of the data. Lazy learners, however, naturally support incremental learning. They are able to model complex decision spaces that may not be as easily describable by other learning algorithms [4].

There are several practical applications of decision support systems that make use of CBR in the literature. Liu et al. [8] describe a system for knowledge support of problem solving in a production process, based on knowledge discovery and case-based reasoning. Raphael et al. [11] describe a system for computing the cost of construction projects, using a case-based reasoning strategy. Shimin et al. [13] seek to combine case-based reasoning and rule-based reasoning for a system for emergency decision making. Arshadi et al. [3] use data mining for case-based reasoning in a biological domain. They conclude that CBR systems perform remarkably well on complex and poorly formalized domains.

There are not that many examples of DSS in the literature from the field of aquaculture, but some do exist. Schulstad [12] describes work done on a DSS for hatchery production management for Atlantic salmon in Norway. Bolte et al. [5] developed a decision support tools for aquaculture to assess economic and ecologic impacts of alternative decisions on aquaculture production. Their main approach was a system based on simulation models and enterprise budgeting. Li et al. [6] describes a web-based expert system for diagnosing fish disease in aquaculture facilities in China. One of their main experiences is that a good expert system require tight cooperation and collaboration among users, human experts, knowledge engineers and system developers.

Combining CBR with low-level sensor data has also been used in the offshore oil industry. Drilling oil wells is a complex and costly process, and downtime is to be avoided. By combining sensor data with historical cases, the drilling process can be monitored to avoid problems in the future, i.e. the system *predicts* possible states that can occur [2,14].

3. The SimFrame architecture

The research presented in this paper is part of the SimFrame project. This section describes the SimFrame *architecture*, which is a generic platform that can hold various methods and applications. The decision support system is therefore an *instantiation* of the architecture.

Figure 1 shows the SimFrame architecture. The top level module is a site (i.e. a physical fish farm on a specified location), since all operations and predictions on a regional level are based on combinations of input from sites in the region. The next level (indicated by the boxes that read *Application N*) represents the different applications within SimFrame. Currently, there are two applications on this level. One is regional decision support, the other is a registration application. The registration application is used by operational managers to record important incidents during the life cycle of the salmon at the fish farm. Both these applications will be described in the next section.

The third level (seen as *Low-level model N*) represents methods and other programs accessible by the applications on the second level. Examples of functionality on this level includes growth models, waste models, simulation of current, regression analysis and CBR. An emphasis is put on the inclusion of already existing software, by providing wrappers to the application level. A goal is to make it easy to add an existing model to the SimFrame architecture without having to re-implement the model. There is a clear divide between the application (2nd) level and the model (3rd) level; the models simply compute data, whereas the applications can make qualitative assertions based on the results from the models.

The modularity also serves another purpose: it should be easy for external research partners to make use of the different models and applications in the architecture. The SimFrame archicture is intended to be an open platform that facilitates co-operation among different researchers.

4. Demonstrator system

The research is done in close cooperation with industry and research partners, where workshops have been organized, displaying partial implementations of the SimFrame architecture. The advantage of such an approach is to avoid waiting for a full implementation (which may be faulty or lacking functionality in itself), and instead get faster feedback on the system development path. Data has been acquired from a salmon farming research site called “Tristeinen” off the coast of Fosen, Sør-Trøndelag, Norway. SINTEF Fisheries and Aquaculture has two research licenses for salmon production, and has established the company Aquaculture Engineering (ACE)[14] together with NTNU and local

[14] www.aceaquaculture.com

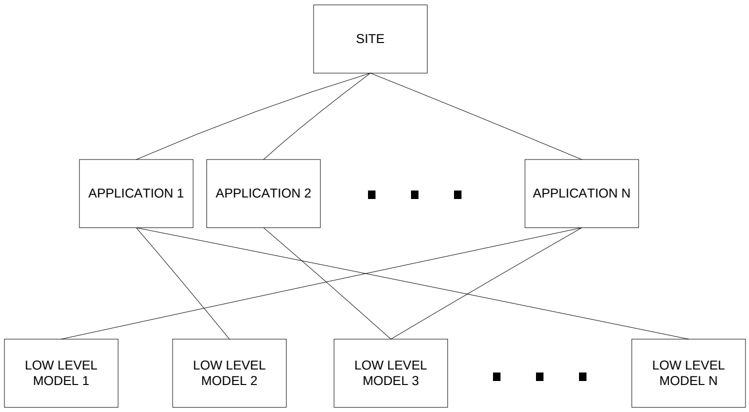

Figure 1. The SimFrame architecture. On the top level is the *Site* object, which is a logical starting point, since all applications and simulations are performed on a site. The second level (i.e. *Application N*) is called from the *Site* object. The applications make use of the models available on the third level (i.e. *Low level model N*).

authorities and companies for managing such large-scale research facilities. The site is operated by SalMar, one of the world's largest producers of salmon. Tristeinen is being run as a proper full-scale salmon production site in itself, with the added advantage of facilitating data acquisition for research studies.

Data from Tristeinen is imported from two separate systems: 1) AquaFarmer[15], a system that collects data of the daily operation on a fish farm (such as feed, lice count, disease), and 2) SeaWatch[16], a system for recording environmental data (such as wave height and direction, current, temperature, salinity) . The data is imported from XML and text files into the SimFrame database. Other data sources (e.g. sensor data) are available and will be added at a later stage.

The SimFrame architecture also aims to be a platform for development of other services by external developers. The architecture is implemented in Django[17], a Python-based framework for web applications. Django was chosen for several reasons: 1) A web interface is well suited as a presentation and deployment approach for the data and methods that the SimFrame architecture aims to provide to its users. It therefore made sense to select a web application framework. 2) Django's strict modularity was appealing to such a modular architecture. 3) Django has a focus on rapid development. 4) Django provides a database abstraction layer, that makes it possible to directly manipulate data stored in the database with Python code (i.e. not requiring to write SQL statements). This also eases the integration of data from multiple databases/data sources. 5) An open-source codebase, should the need arise to modify core elements of the framework to suit our own needs. 6) Many other web application frameworks could be selected for the same reasons, so why was Django chosen? A key factor was the developers' preference for Python as a programming language, with its easy syntax, extensive library, and support of multiple programming paradigms (i.e. object oriented as well as functional programming).

[15] www.ocea.no/ocea/Mercatussoftware/AquaFarmer/tabid/181/language/en-GB/Default.aspx
[16] www.oceanor.no/systems/seawatch/buoys-and-sensor/Seawatch
[17] www.djangoproject.com

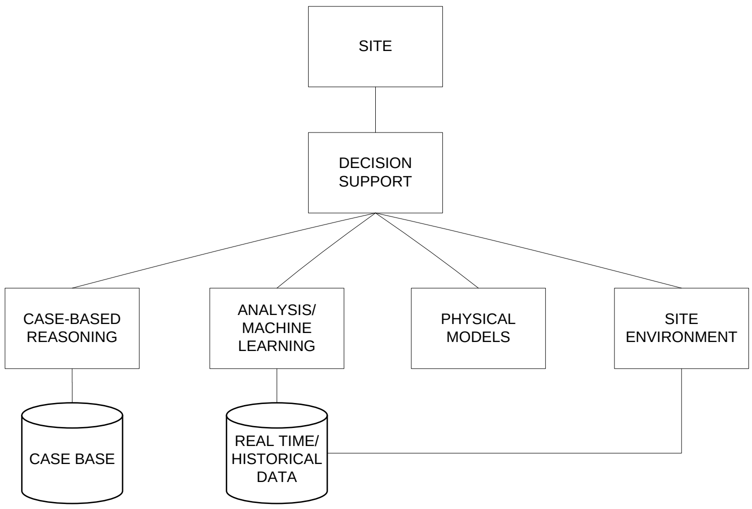

Figure 2. The decision support system. This is an instantiation of the architecture shown in figure 1, note how the box "Decision support system" correspond to the "Application" in figure 1. The decision support system operates on every site. The DSS makes use of various models in order to calculate the current state of the site, the trends of various sensor streams, and offer advice using CBR. For instance, growth and emission models are represented by "Physical models".

As mentioned, two applications have been implemented in the SimFrame architecture so far, one being a decision support system, and the other a registration application for operational events related to mortality. These will now be explained.

4.1. Decision support system

The first application to be developed in the SimFrame architecture is a decision support system. Its intended users are regional managers surveying 10-20 sites. The manager needs to get quick feedback on the state of the different sites. Each site can be in one of the following three statuses: normal, inquiry suggested and problem. The status is determined based on an analysis of the environmental data in the database for each site. If there are deviations from a specified norm, the status of the site will change, depending how severe the deviation was. The input parameters considered for deviation are temperature, oxygen level, feed consumption and lice counts. Depending on how much the parameters deviate from a manually specified norm, the system suggests actions to take, based on a very simple CBR mechanism. An overall goal of the SimFrame project is to be able to *predict* states or events given a set of parameters. To achieve this, the system must analyze trends in the sensor data to be able to predict future trend developments. In order to predict future trends, Levenberg-Marquardt curve fitting has been used [9], with good results. This DSS is an *instantiation* of the general SimFrame architecture, see figure 2.

In the decision support system, two main areas of CBR have so far been implemented. The first is the search for similar situations regarding the state and trend of the site, for instance if the oxygen level is high and have been rising for some time, this is an indication of the presence of algae. The second is based on reports regarding mortality of the fish. When fish dies on site, a written report about the event is standard procedure within SalMar's organization. Among the most important records in such a report are how many fish died, at which point the dead fish were counted, which vaccines were used, and the cause of death. All these parameters are then used in the CBR application. The user of the system can look for similar cases by specifying the current values of the given parameters.

The user interface has been designed so that when operating in a normal state, the program will not give any feedback (i.e. not showing a color when the site was operating normally). The state of the site would only be shown if there was a problem, or a possibility of a problem in the future. In other words, the program would draw attention to *problems*, not when a site was operating normally. If a situation should be investigated, a yellow symbol would appear. A significant problem would show up as a red symbol. An early implementation of the user interface can be seen in figure 3.

When implementing a CBR-system, deciding on a method for assessing the similarity between two cases is paramount. In our implementation this boils down to defining the *similarity metric* (a mathematical formulation of how important a parameter is when finding a similar case). The similarity metric for the system was developed based on the parameters found in the current case base. The current case base was limited to 16 cases. In total there were 20 parameters where 5 were weighted as more important in the global similarities. The parameters given most weight in decreasing order of importance were: diagnosis by veterinary, observations of the dead fish by the farmer, number of dead fish, vaccines used, and fishgroup. For the similarity metric of the individual parameters, primarily two kinds of similarity modes were used: advanced number similarity, and symmetric table similarity. Even though only 16 cases were present in the case base, it does demonstrate how such a system can aid in finding similar cases when mortality occurs. As previously noted, a superior goal within the SimFrame project is to be able to *predict* future cases. Such a CBR system could be queried on possible future outcomes by an automated system (e.g. an online CBR query done on a daily basis), making it possible to predict possible scenarios by searching for similar cases in the CBR base. This mechanism is similarly employed in the oil industry [14]. The CBR mechanism was implemented in Protégé[18] using the myCBR[19] plugin.

4.2. Registration of fish mortality

As mentioned in the previous section, an important source of cases for the CBR system were written reports detailing mortality of the fish. However, the amount of cases was severely limited, only 16 cases. Cases needed to be manually added to the case base, based on written reports. In order to facilitate the registration process, it was suggested by industry partners that the cases could be registered electronically. A new instantiation of the SimFrame architecture (i.e. another *application*) became a web application where the most important operations related to fish mortality could be registered. There were four type of operations: 1) delousing[20], 2) delivery to food processing plant[21], 3)

[18] protege.stanford.edu

[19] mycbr-project.net

[20] Delousing consists of adding a chemical to the environment of the fish to reduce lice. Delousing is strictly regulated in Norway. From the 1st of January 2011, it is illegal to delouse without using tarpaulin to protect the surrounding environment while using chemicals.

[21] This is the final process in the life cycle of the fish. It is pumped from the cages onto a live fish carrier and delivered to a plant that processes and packages the fish.

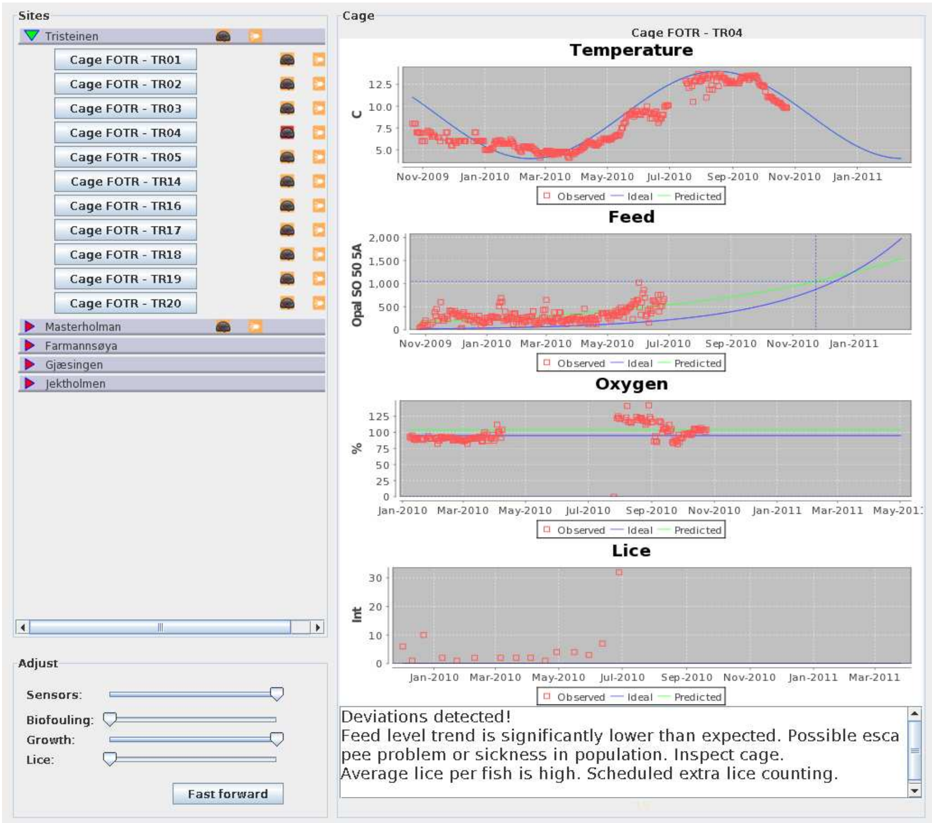

Figure 3. The regional decision support system. To the left, sites are listed. By clicking on the corresponding cage for each site, it is possible to see the recorded data for that particular cage. The CBR system suggests countermeasures on the bottom right. The program for analyzing and displaying the recorded data along with the predictions were written in Java (this was made before the decision to develop the architecture in Django was taken).

deployment[22] and 4) sorting of the fish[23]. Related to these operations are information regarding the environment, chemicals used, methods for moving the fish, which ship was used to transport the fish, how many people were involved, and time and date for the various subprocesses during the operation, to name some of the most important parameters. This system has been under active development during winter 2010/2011, and is being deployed early 2011 on some of SalMar's sites in the middle of Norway.

[22]Deployment is the operation responsible for most deaths among the fish. When the smolt is transported and administered to the different cages on the site, it is crucial that it has gone through the smoltification process. Smoltification refers to the physical changes the smolt goes through, so it can change habitat from fresh water to salt water. If the smoltification has not been completed, the consequences are fatal for the fish.

[23]An important process during the growth phase is sorting of the fish. The fish is sorted based on weight into different cages, for optimum feeding. If the weight of the fish differs greatly within one cage, the smaller fish tend to get less food because the food gets eaten by the bigger fish. By grouping similarly sized fish in on cage, the food is more justly dispersed among the individuals.

5. Discussion

So far, the main result of the SimFrame project has been the experiences gained by implementing the demonstrators in close cooperation with the involved industry and research partners. By making incremental demonstrators and getting early feedback, it is ensured that the developmental path and the projected end result will be both interesting and useful to the industry partners. The technologies and methods used have been chosen on a pragmatic basis, by looking at similar approaches. The fusion of low-level sensor data and high-level human experiences is similar to the approach described by Shokouhi et al. [14], which is being employed in the oil industry. The use of open-source software will also lead to a reduced cost if implemented in a product by one of the industry partners. However, *what separates this architecture from similar system architectures?* Although the architecture is designed specifically for the fish farming industry, it is open for both industry and researchers. The emphasis on using open source tools to implement the architecture further facilitates the reuse of research results, as well as adapting the software to suit specific needs. The design also specifically targets the reuse of already existing software, by making it available as low-level models. By making wrappers, external models can easily be integrated without the need for re-implementation. This is a crucial aspect. Researchers and industry develop new methods and tools over time, and an easy way to integrate them will provide a useful platform for further research and development. The flexible architecture can be seen as lacking in rigidity, however this design approach was taken deliberately to anticipate future changes in applications, operational regulations and new models. The farming industry is dynamic, with new technology and government legislation changing the way business is done on a regular basis. Another novelty is the composition of several data sources into one interface. Integration of knowledge and models from specialized fields is essential for improving fish farming planning and operations. The iterative approach used in SimFrame ensures that experiences with integrating data and models can be utilized quickly in other projects within the domain. The SimFrame architecture provides an environment for innovation by facilitating integration of various systems and models, making it easy to develop novel applications.

6. Future work

This paper describes the early steps towards the desired functionality described in section 3. The deployment of the registration application described in section 4.2 will lead to rapid growth of the case base. When the case base gets bigger, the focus will be on developing and refining the CBR application. Currently, the CBR mechanism is very simple, due to the small amount of cases in the case base. It is certain that as the case base grows, further investigation is needed into defining the similarity metric underlying the search for similar cases. We are now moving from myCBR to jColibri[24] as the CBR development platform, which enables more expressive case representations and increased method flexibility.

The next iteration will consist of integrating more computational models into the architecture, shown in figure 2. Furthermore, data collection will be expanded to include

[24] gaia.fdi.ucm.es/projects/jcolibri/

other SalMar sites, making it possible to exploit machine learning algorithms to search for interesting cases and trends. In the introduction, it was envisioned how low-level sensor data and high-level human expertise would be fused using CBR and other machine learning techniques. The implementation of this ability is still in an early stage, and will also be the focus for the next development phase in the SimFrame project.

At the core of the decision support system is CBR, a way of storing and reusing knowledge that relies on an case base. CBR does not contain a generic model. Integrating CBR with probabilistic models (such as Bayesian networks) can further enhance the CBR mechanism. The current domain is a data-rich environment that opens up this possible opportunity for future research.

Acknowledgements

The authors would like to thank Brad Schofield, Hans Vanhauwaert Bjelland, Gunnar Senneset, Thomas Bekken, Arnfinn Aunsmo and Morten Malm for contributing to the development of SimFrame, and comments and criticism that helped improve the paper.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

[2] Agnar Aamodt. Cbr for advice giving in a data-intensive environment. In *Proceeding of the 2008 conference on Tenth Scandinavian Conference on Artificial Intelligence: SCAI 2008*, pages 201–205, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[3] N. Arshadi and I. Jurisica. Data mining for case-based reasoning in high-dimensional biological domains. *Transactions on Knowledge and Data Engineering*, 17(8):1127–1137, 2005.

[4] J. Han, M. Kamber, and J. Pei. *Data Mining, Second Edition Concepts and Techniques*. Elsevier, 2005.

[5] Bolte John, Nath Shree, and Ernst Doug. Development of decision support tools for aquaculture: the pond experience. *Aquaculture Engineering*, 23:103–119, 2000.

[6] D. Li, Z. Fu, and Y. Duan. Fish-expert: a web-based expert system for fish disease diagnosis. *Expert Systems with Application*, 23:311–320, 2002.

[7] S. H. Liao. Expert system methodologies and applications - a decade review from 1995 to 2004. *Expert Systems with Application*, 28:93–103, 2005.

[8] D.-R. Liu and Ke C.-K. Knowledge support for problem-solving in a production process: A hybrid of knowledge discovery and case-based reasoning. *Expert Systems with Application*, 33:147–161, 2007.

[9] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[10] K. S. Metaxiotis, D. Askounis, and J. Psarras. Expert systems in production planning and scheduling: A state-of-the-art survey. *Journal of Intelligent Manufacturing*, 13:253–260, 2002.

[11] B. Raphael, B. Domer, S. Saitta, and I.F.C. Smith. Incremental development of cbr strategies for computing project cost probabilities. *Advanced Engineering Informatics*, 21:311–321, 2007.

[12] G. Schulstad. Design of a computerized decision support system for hatchery production management. *Aquaculture Engineering*, 16:7–25, 1997.

[13] D. Shimin, H. Shen, and H. Liu. Research on case-based reasoning combined with rule-based reasoning for emergency. In *IEEE Interntational Conference on Service Operations and Logisitcs, and Informatics*, 2007.

[14] S. Shokouhi, A. Aamodt, P. Skalle, and F. Sørmo. Determining root causes of drilling problems by combining cases and general knowledge. *Case-Based Reasoning Research and Development*, pages 509–523, 2009.