Abalone Harvest Prediction using AI methods

Technical Paper CS04-15-00

Department of Computer Science

University of Cape Town

Rashin Maharaj rmaharaj@cs.uct.ac.za
Shaan Bheekun sbheekun@cs.uct.ac.za
Dr Anet Potgieter anet@cs.uct.ac.za
Justin Kelleher jkr@cs.uct.ac.za

ABSTRACT

"Foreknowledge of the future makes it possible to manipulate both enemies and supporters." --Raymond Aron in "The Opium of the Intellectuals"

The above quote describes perfectly the motivation for developing a prediction tool. Being able to minimise uncertainty to any possible degree will give any business that engages in prediction or forecasting, a competitive advantage that is becoming necessary for economic prosperity.

The main task of this project was to use Artificial Intelligence Methods to support Abalone Harvest Prediction. The first step was to choose a suitable graphical probabilistic network so that the abalone growth, given the factors that affect it, can be successfully modelled.

An implementation structure for the chosen model was designed and then implemented. Once a suitable model was designed, the two core components of the system were implemented. A learning engine to learn the parameter values for the chosen model, and an inference engine to perform probabilistic inference on the learnt parameters.

A graphical user interface that is user-friendly and easy to understand by the people at the farm was then developed and implemented. This graphical user interface hides the complexities of artificial intelligences techniques of which can intimidate the novice user.

1. INTRODUCTION

In today's business world, where the dynamics of market demand and supply are constantly changing, it has become increasingly important and difficult to establish and maintain competitive advantage.

Uncertainty has been the most universal trait of market behaviour. This has made businesses realize that the innovative handling of uncertainty can lead to adding of value.

The high costs of developing and maintaining systems today have led to the realization that standardization is a must in all domains. Standardization of processes can lead to the decreases of development costs financially and in terms of time and expertise.

The high costs of developing systems and solutions today have led to the realization that standardisation is a must in any domain. Development costs can be decreased significantly in financial terms as well as in terms of time and expertise.

A popular problem in the business world is that of prediction, applied to both supply and demand. In the case of this project, the problem to be addressed is one of predicting supply. To solve this problem, artificial intelligence techniques were evaluated and the most relevant technique was employed to provide a solution.

This project was undertaken in conjunction with the I&J abalone culture division in Gansbaai, South Africa.

The solution consists of two parts; the first being a supply prediction tool that predicts the growth rate of abalone using environmental data. The second part of the solution is a browser-based software development process platform. The platform is to be used by the target organization to understand the work done on this project in order to allow the system's extension.

In order for the system to be implemented, there were a number of design decisions that first had to take place. The first decision involved choosing a model with which to represent the situation at the farm. Two technologies to construct models were evaluated based on their appropriateness to the problem domain. These were Bayesian Networks and Dynamic Bayesian Networks.

The second decision involved deciding on an inference algorithm to implement the given model, and the third decision was to decide on a learning algorithm to employ in the model.

Besides the design of the system, the design of the process platform had to be considered as well, so agile modelling, knowledge engineering, user-centred design and business process modelling had to be covered as well.

2. BACKGROUND

2.1 Bayesian Networks

The statement *"The heart of Bayesian Techniques lies in the celebrated inversion formula"* in Pearl [1] refers to the formula:

$P(H \mid e) = P(e \mid H)\, P(H) / P(e)$.

This is where H is a belief of a hypothesis, given evidence e and $P(e \mid H)$ is the probability of the hypothesis given e. Another name for the inversion formula is Bayes rule; this formula provides the basis for all Bayesian techniques.

A Bayesian network is a directed graphical model, where nodes represent random variables and arcs represent conditional dependence assumptions where the lack of an arc can be said to describe conditional independence assumptions as stated in Murphy [2].

An example of a Bayesian Network can be seen in figure one, adapted from Murphy [2]. It represents the situation where the grass can be caused to be wet by a sprinkler or the rain. It also shows that the rain can be cause by the fact that the weather is cloudy.

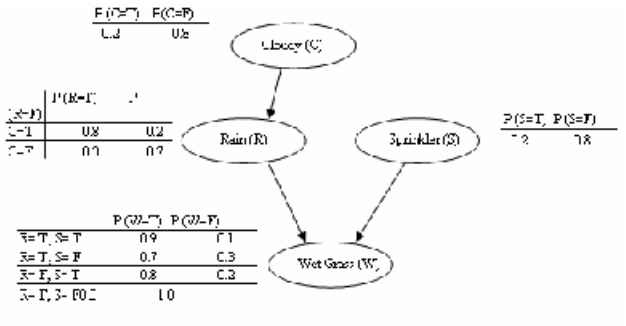

Figure 1. Example Bayesian Network

2.1.1 Inference in Bayesian Networks:

Murphy [2] explains that the most common task to be solved by Bayesian networks is probabilistic inference. Charniak [3] goes on to state that the main constriction in using Bayesian networks is that inference can be an NP-hard problem. The problem arises in multiply-connected networks, if there is more than one path between any two nodes; computation of the probabilities of the network becomes more complex and takes much longer.

Figure 2: Example of a singly-connected network

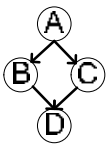

Figure 3: Example of a multiply-connected network

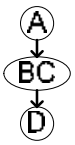

Figure 4: A clustered multiply-connected network

The ideal form of inference is known as "Exact Inference". Exact inference cannot always be used. Whether or not it can be used for a network depends heavily on the structure of the network itself. It can be used in singly-connected networks, also called polytrees. A polytree is a network where there is a maximum of one path between any two nodes; a simple example can be seen in figure two.

Exact inference cannot be used in multiply-connected networks, such as in figure three. A detailed mathematical description of the problem is complex and not necessary in the context of this paper.

The problem can best be explained in the context of figure three. If evidence is provided for node D, and the conditional probabilities at node C are known, the state at C should then be inferred. The problem is that C is not just directly influenced by the evidence at D, but also by A which is influenced by D via B.

There are cases where exact inference can be used despite the network being multiply-connected. A process called clustering can be used, in which pairs of nodes of a multiply-connected network are combined to form single nodes until a polytree is formed. This process is described in detail in Charniak [3].

An example of clustering can be seen by the transition of the network from figure three to figure four. Nodes B and C are combined into one so the network can become a polytree. The parameters contained in node BC are the results of the cross product of the values of B and C. The inference would then be applied to the polytree and the values for B and C can be found individually from the BC node. The new values for B and C can then be integrated into the original multiply-connected tree.

In cases where clustering cannot be used to accommodate exact inference, a technique known as approximate inference is used. There are many algorithms for approximate inference, but there are common features in all of them.

They all randomly collect values for a certain number of nodes, and then use the values of those nodes to assign values to other nodes in the network. A popular technique described in Charniak [3] is that of "Logic sampling". The values of root nodes are guessed based on their conditional probabilities. The algorithm is then propagated down the network by taking the guessed parent node's value as evidence and then assigning a value to the child node.

A notable algorithm is the Message-Passing algorithm by Judea Pearl Kim[], which can be applied as a mechanism for both exact, as well as approximate inference. Message-passing will be described in detail later on in this paper.

2.1.2 Parameter Learning

The aim of parameter learning is to derive the probabilities for the CPT held at each node. Deriving these probabilities is considered to be the most straightforward learning situation when the structure is defined in its totality and the dataset containing the parameters is complete as described by Krause [4].

The probabilities are derived from the dataset by using Maximum likelihood estimate algorithms. These algorithms, in short, ensure that the probabilities for the CPTs are representative of the parameter values in the dataset. An in-depth discussion of Maximum Likelihood Estimate algorithms can be found in [2].

When values for certain parameters are missing, i.e. the dataset is incomplete, and the structure of the network is defined, learning of parameters is not a straightforward process. Expectation Maximization or gradient ascent methods have to be employed in order to derive the probabilities for the CPT.

Both these methods use an inference algorithm as subroutine. The inference algorithm estimates values for the parameters with missing data. Then an iterative process is used during which the estimated values are refined until the probabilities in the CPTs converge.

Once the probabilities in the CPTs converge, they represent the learnt probabilities. A detailed explanation of the Expectation Maximization method is given in Nilsson [5].

2.1.3 Structure Learning

The goal of structure learning is to mine the interdependencies between the nodes of a Bayesian network from the data containing the parameters. When graphically portrayed, structure learning attempts to find the arcs between the nodes and its directionality.

A naïve way of learning the structure is to enumerate all the possible network structures that can be constructed from a set of pre-defined nodes. Each of these structures is then evaluated using a scoring metric. A scoring metric is a set of measures that can be use to evaluate a Bayesian network.

There are a number of scoring metrics and one of them is the description length scoring metric. A good discussion of it can be found in Nilsson [5]. In the case of the description length scoring metric, the network structure among all the possible network structures that minimises the metric is the structure that best describes the data and is hence the learnt structure.

The evaluation of all the possible network structures using a scoring metric is generally not feasible as stated by Deviren [6]. This is due to the fact that the number of possible network increases exponentially with an increasing number of nodes. For example, as stated in Krause [4], the number of possible networks is approximately $4.2x10^8$ for 10 nodes.

There are many different methods that can be applied to solve the problem of evaluating all the possible network structures. One of the most discussed methods in literature is described in Krause [4]. It is the hill-descending or greedy search method in which changes to network i.e. the merit of adding, deleting or reversing an arc is evaluated using a scoring metric instead of evaluating the whole network. The greedy search method relies on the property that the scoring metric is decomposable.

The scoring metric is decomposable in the sense that the scoring metric for whole Bayesian network is the sum of the local scoring metric at each node. Hence changes to a node only require the computation of the local scoring metric at that node as opposed to computation of the scoring metric of the whole Bayesian network.

2.2 Dynamic Bayesian Networks

A dynamic Bayesian network, when described in brief can be thought of as a probabilistic graphical network such as a Bayesian network that encodes the notion of time within it so as to accommodate sequential data.

In more technical terms, dynamic Bayesian networks are directed graphical models of stochastic processes that in their simplest form generalise hidden Markov models as described by Pearl [1]. Stochastic processes are processes that involve some kind of randomness or unpredicted effects often associated with probabilistic or statistical treatments. Dynamic Bayesian networks generalise hidden Markov models hereafter referred as HMMs in the sense that they provide an easier way to specify the conditional independencies involved in HMM. The concepts involved in HMM and dynamic Bayesian networks are however similar.

In order to understand the concepts involved in dynamic Bayesian networks it is best to give an example of an HMM. The example described below is adapted from Dugad [7] and Barber [8].

Let there be 3 urns, U_1, U_2, U_3 such that each of them can contain different proportions of Blue and Red marbles. These proportions of Red and Blue marbles in each urn can be represented as probabilities as shown in the matrix below.

	U_1	U_2	U_3
Blue	0.3	0.8	0.5
Red	0.7	0.2	0.5

Table 1: Probability matrix of marble colour and urn

Marbles are drawn from the urn in a sequence, e.g. $U_2\, U_1,\, U_3$. Any of the three urns can be chosen as the first urn from which a marble is drawn as long as the sequence is respected. For example if U_1 is chosen as the first urn, then the next urn chosen will be U_3, then U_2 and so on. There is a probability associated with each urn which represents its chances of being the first urn from which a marble is drawn i.e. the urn chosen at time t =1. This is shown in the table below.

U_1	U_2	U_3
0.2	0.5	0.3

Table 2: A clustered multiply-connected network

Sometimes mistakes happen and the sequence of choosing the next urn is not respected. The probability of moving from one urn to another is given in the matrix below.

		From U_i		
		U_1	U_2	U_3
To Uj	U_1	0.1	0.8	0.1
	U_2	0.1	0.1	0.8
	U_3	0.8	0.1	0.1

Table 3: Transition matrix for moving from U_i to U_j.

When the marbles are drawn from the urn in the sequence described above ($U_2\, U_1,\, U_3$), only the colors of the marbles are shown to an observer. The urn from which a marble comes from is not shown to the observer. The urns can be thought as being hidden (representing hidden states) and the colors represent observed variables from the hidden states. From the above information a hidden Markov model can be constructed. It consists of the following elements that are referred to as the parameters of the model:

1. The probabilities of being in a particular state at time= 1 which is shown by table two.
2. The probabilities of moving from one state to another that is shown by Table three.
3. The probabilities of each observed variable in each state which is shown by Table one.

The model can be graphically portrayed with hidden states represented as h_i and observed variable as o_i and as shown below.

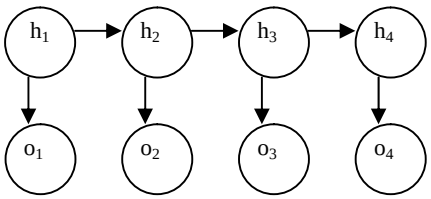

Figure 5: A Hidden Markov Model.

There are three problems for HMMs to solve, as described in Dugad [7]. Some HMM notation is required to understand them.

N = number of states in the model.

M = number of distinct observation symbols.

T = number of symbols observed.

i_t denotes the state at time t.

$V = \{v_1,\ldots,v_M\}$ denotes the discrete set of possible observation symbols.

$\pi = \{\pi_i\}$, $\pi_i = P(i_1 = i)$, the probability of being in state i at time $t=1$.

$A = \{a_{ij}\}$ where $a_{ij} = P(i_{t+1} = j \mid i_t = i)$, the probability of being in state j at time $t+1$ given that we were in state i at time t.

$B = \{b_j(k)\}, b_j(k) = P(v_k \text{ at } t \mid i_t = j)$, the probability of observing the symbol v_k given that we are in state j.

O_t will denote the observation symbol observed at instant t.

$\lambda = (A, B, \pi)$ will be used as a compact notation to denote an HMM.

Problem 1: Given a model $\lambda = (A, B, \pi)$ how do you compute $P(O \mid \lambda)$, the probability of occurrence of the observation sequence $O = O_1, O_2 \ldots O_T$.

Problem 2: Given a model $\lambda = (A, B, \pi)$ how do we choose a state sequence

$I = i_1, i_2 \ldots i_T$ so that $P(O, I \mid \lambda)$, the joint probability of the observation sequence

$O = O_1, O_2, \ldots, O_T$ and the state sequence I given the model λ is maximized.

Problem 3: How do we adjust the HMM model parameters $\lambda = (A, B, \pi)$ so that $P(O \mid \lambda)$ is maximized?

2.1.1 Inference in DBNs

Inference deals generally with the solution to Problem one. There are different algorithms that can be used to perform inference in a DBN. Two of the most popular algorithms are the Forward-Backward algorithm, described fully in Dugad [7] and the Junction tree algorithm, described fully in Barber [8].

2.1.2 Learning in dynamic Bayesian network

As in Bayesian networks, there are two types of learning that can occur, structural learning and parameter learning. The algorithms for learning Dynamic Bayesian networks are extensions of algorithms for learning Bayesian networks as described by Murphy [9]. A discussion of these extensions is out of the scope of this paper, as it will require a full mathematical description of the learning algorithms. However a detailed description of the algorithms can be found in Murphy [9].

2.2 Agile Modelling

Adapted from Ambler [10], agile modelling, hereby referred to as AM, is a 'chaordic' practice-based methodology that describes how to perform modelling and documentation in software engineering in an agile manner. The term 'chaordic' refers to the fact that it combines the chaos of simple modelling with the order that is found in the production of artefacts that occurs in typical software engineering techniques. The word agile in AM, as adapted from Beck [11] refers to the principle of welcoming requirements changes throughout the development process.

2.3 Knowledge Engineering

Context in any decision-making process represents the secondary properties of a cognitive or motivational state of an individual that modify the effect of a stimulus or activity.

In any domain, there are usually domain experts who, whether they have the 'know that' knowledge or not, are known to be experts because of their 'know how'. Tacit knowledge is that knowledge that is implicitly held by an individual. There is tacit knowledge that can be made explicit. Explicit knowledge is accessible to everyone and shared. By observing and interacting with the expert in context, the explicitable tacit knowledge can be translated to explicit knowledge. This is extremely useful for knowledge engineers in a domain-specific environment. The dynamics of this process are illustrated in figure six, adapted from Brezillon [12].

Context can be divided into two parts; the first is called Contextual knowledge which represents the fixed primary characteristics of a situation. The second is called External knowledge, and it represents the part of the context that does not influence the decision-making at a particular time. When the domain expert performs a task that includes contextual knowledge, the act is known as a procedural context. If the contextual knowledge is their activity within context, the procedural context is a single task. By observing and documenting this process, tacit knowledge can be made explicit.

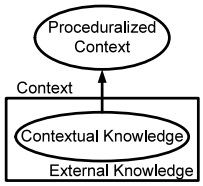

Figure 6: Contextual knowledge and procedural context

3. APPROACH

3.1 Model

Due to the time related nature of the causal relationships at the farm, Dynamic Bayesian Networks (DBNs) seemed the obvious choice. There was a complication however, as seen in figure seven. The complication is that the hidden and observable variable are observable at the same time. The variable 'ev' stands for an environmental variable and the variable 'g' stands for the growth

rate of an abalone. What is meant by hidden to the same degree is that they are both made observable at the same time i.e. once a month. When 'ev' is observed, 'g' will be inferred, but if 'g' and 'ev' are made observable at the same time, then why infer 'g' when it can just be observed? It is for this reason that it was decided that DBNs would not be an ideal graphical model for the processes at the farm.

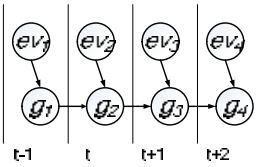

Figure 7: Proposed DBN for farm (Left)

Figure 8: BN for one size-range (Right)

When considering modelling the farms processes using a BN, it was clear that the fact that the process was time-related would be a complication. This turned out to not be a big design challenge, as the number of time-steps is fixed. This is because each time-step represents one month. The network has the additional 't' node because the month is the direct cause of the value of 'ev', due to the seasonal nature of the variance of 'ev'.

What this means is that each size range of animals can be represented by a Bayesian network. This is because different-sized animals grow at different rates. For any particular month, the 't' node will be given evidence, in the form of the current month, as it should be known. This is the model that was implemented as the backbone of AhPT.

3.2 Structural Implementation

It can be observed from the model in figure nine, that there are BNs for each size class, but more importantly, that for a particular month, they will all have the same evidence provided for 't', as t represents the current month. Linked to this is the fact that the relationship between 't' and 'ev' is the same irrespective of the size of the abalone. This might not be intuitive to the reader, but these issues could not be clarified specifically because of confidentiality.

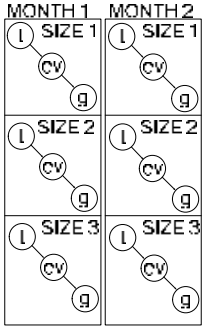

Figure 9: Structure of BN for three size classes over two months

The first implication of the model is that there is only need for one CPT for the relationship between 't' and 'ev', because the relationship is the same irrespective of the size of the animal. The other implication is that there is a need for a CPT for the 'ev-g' relationship for each time-step.

These implications remove the need for the storage of an explicit structure containing all of the nodes. The 'structure' chosen to implement in the end was to persistently store the farm's population containing all of the various sizes. A global CPT for the 't-ev' relationship is persistently stored, as well as CPTs for each size class. The CPTs and population data are stored in XML format

3.3 Inference

Exact inference, as described in 2.1.1 was chosen, as opposed to approximate inference. Approximate inference can yield similar results as exact inference when applied to extensive data. Due to the sparseness of data at the farm, exact inference was chosen as a safer option. This was possible, as the networks in figure nine meet the criteria to be considered polytrees/singly-connected.

The algorithm chosen was Judea Pearl's message passing algorithm, taken from Diez [13] and Kim [14]. A detailed explanation of the algorithm from Diez [13] is below:

A BN can be represented as directed acyclic graph, where each node represents a random variable with a probability distribution:

$P(x_1, \ldots, x_n) = \Pi_i P(x_i \mid pa(x_i))$, (where $\Pi_{i=1}^{3} i = 1*2*3$) where x_i represents a possible value of variable X and $pa(x_i)$ is an instantiation of the parents of X_i in the graph. The basic problem of inference is computing $P(x \mid e)$ of variable X, given evidence e.

$e \equiv \{X_i = x_i, X_j = x_j, \ldots, X_n = x_n\}$.

The message passing algorithm:

The goal of the algorithm is to find $P(x/e)$ as described above, i.e. the value of the probability that $X = x$ given that we have evidence e. In a polytree such as the one depicted in figure fifteen, any node X divides the evidence into the evidence causing X, e_X^+ ; and evidence connected to it's effects, e_X^-. Similarly a link XY divides evidence into evidence above the link, e_{XY}^+ ; and evidence below the link, e_{XY}^-. This division of evidence is called d-separation and it justifies the definition of the following messages used in the Message Passing algorithm.

$\pi(x) \equiv P(x, e_X^+)$

$\lambda(x) \equiv P(e_X^- \mid x)$

$\pi_X(u_i) \equiv P(u_i, e_{UX}^+)$ (Note that UX is actually U_iX in this equation)

$\lambda_y(x) \equiv P(e_{XY}^- \mid x)$ (Note that Y in λ_y is actually Y_i, and XY is actually XY_j)

The derivations of these messages are described in detail in Kim [19], but are not necessary for this report. D-separation results in two subsidiary properties of polytrees:

- For a node X, two children of X, Y_i and Y_j are independent of each other given the value of X.
  i.e.: $P(y_i \mid x) = P(y_i \mid x, y_j)$.
- A parent U_i and a child Y_j of a node X are independent given the value of X.
  i.e.: $P(u_i \mid x) = P(u_i \mid x, y_j)$.

To avoid confusion, it must be stated that although the first property above implies an independency between adjacent child nodes, this does not apply to parent nodes of X.

i.e.: $P(u_i \mid x) \neq P(u_i \mid x, u_j)$.

Given the independency properties described above, the recursive expressions for computing the messages are:

$P(x \mid e) = \alpha\pi(x)\lambda(x)$

$\pi(x) = ?_{u1,\ldots un} P(x \mid u_1, \ldots, u_n) \Pi_{i=1}^{n} \pi_X(u_i)$

$\lambda(x) = \Pi_{j=1}^{m} \lambda_Y(x)$ (Note that Y is actually Y_j)

$\pi_Y(x) = \pi(x) \Pi_{k \neq j} \lambda_Y(x)$ (Note that Y in is actually Y_j and Y in λ_Y is actually Y_k)

$\lambda_Y(x) = ?_{y} [\lambda(y_j) ?_{v1,\ldots vp} P(y_j \mid x, v_1, \ldots, v_p) \Pi_{k=1}^{p} \pi_Y(v_k)]$

Where:

α is a normalization constant to be computed after finding $\pi(x)$ and $\lambda(x)$.

$V_1, \ldots, V_p$ are the causes of Y_i other than X.

The inference engine is an implementation of the message passing algorithm discussed above. The only modification is that the lambdas ($\lambda(x)$) were ignored because of the nature of the causal relationships of our model.

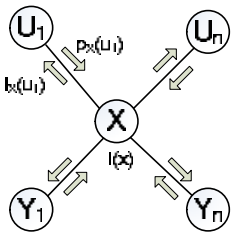

Figure 10: Evidence propagation using Message Passing

3.4 Learning

A learning algorithm was required in order to learn the parameters of the data for the system. In order to achieve this, a Maximum-likelihood estimate-learning algorithm was implemented for the learning engine to achieve its goals.

Following is a description of learning algorithm implemented, which is best explained in the context of an example adapted from Russell [15].

Suppose a bag of lime and cherry candy is bought from a manufacturer whose lime-cherry proportion is unknown and the type of candy cannot be determined from its wrapping. The Maximum-Likelihood algorithm can be used to estimate the proportion of cherry in the bag. Let the proportion of cherry, i.e. the parameter to be learnt, be denoted by θ. Suppose N candies are unwrapped of which c are cherries and $l = N - c$ are limes. Under the assumption that these observations are independent and identically distributed, the likelihood of this particular dataset is given by:

$$P(\mathbf{d} \mid h_\theta) = \prod_{j=1}^{N} P(d_j \mid h_\theta) = \theta^c \cdot (1-\theta)^l$$

Where h_θ is a non-observed hypothesis about the data and **d** represents the dataset

The log-likelihood L of the dataset is then calculated from the above equation

$L(\mathbf{d} \mid h_\theta) = \log P(\mathbf{d} \mid h_\theta) = \log P(d_j \mid h_\theta) = c \log \theta + l \log (1-\theta)$

A differentiation of the above equation is then perform and equated to zero to obtain θ

$$\frac{dL(\mathbf{d} \mid h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0$$

$$\Rightarrow \quad \theta = \frac{c}{c+l} = \frac{c}{N}$$

In order to calculate the parameter θ the values c and N first had to be obtained from the data.

3.5 Production Simulation

In order for a prediction to be made, both the inference engine and production simulations have to be used. Essentially, the inference engine is to provide one of the inputs for the production simulation. The production simulation gets the other input by reading in the current population from an XML file of the current stock. Product objects containing size class objects are then generated using the XML as input.

The production simulation then updates the sizes of the animals using the growth rates generated by the inference engine. The other production rules include exporting of animals, injection of new stock to the population, and moving of animals between size and product classes. Once all of the production simulation is done, the new population is written to a new XML file. This process can be clearly seen in figure sixteen.

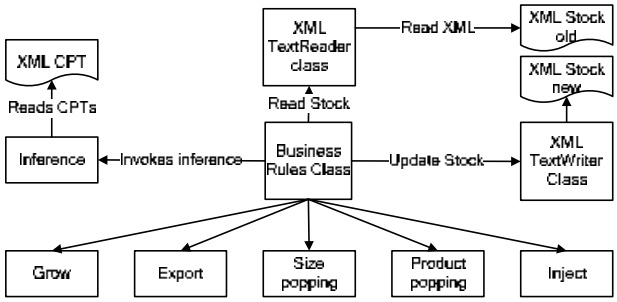

Figure 11: Overview of Production Simulation

3.6 Process Platform

The AhPT Process Platform was implemented as a browser-based process platform, using HTML. The reason for it being browser-based is that it can be used on the web for easy access and that it is completely platform unspecific. It is iterative in nature, comprising of four phases, "Requirements Management", "Software Specification", "Implementation" and "Testing".

4. RESULTS

The results of a heuristic analysis test on our data were positive; Mr Loubser (A domain expert) chose the graph of our results as the second-most likely of a collection of datasets. Further more there was only one notable difference between the graph chosen as most likely and the results of our system. The difference is a 'kink' in our graph, which we found was caused by a lack of data and sampling error in the data gathering at the farm.

An ad hoc prediction accuracy test at the farm where we predicted September's environmental variable was conducted and the result was extremely pleasing. Our prediction of 'ev' was extremely close to the real value. The actual value can not be stated in this report for confidentiality reasons.

The bench-marking done for the inference algorithm against a package called 'JavaBayes' Cozman [17] yielded positive results. The inference returned the same results except that JavaBayes is more accurate as AhPT rounds off the 17th decimal place, due to the nature of floating point manipulation in C#.

The learning engine was tested by comparing parameters learnt from the same dataset using a product called BayesiaLab [18] and those learnt using AhPT. Parameters learnt using BayesiaLab are expressed as a percentage while the parameters learnt using AhPT are expressed as a proportion but they represent the same information. When learnt parameters using BayesiaLab and using AhPT are expressed in the same manner, it can be seen that the difference between them is negligible. This shows that learning that occurs in AhPT is acceptable when compared to the one in BayesiaLab and was successfully implemented.

5. CONCLUSIONS

Given the initial problem description we have delivered:

- The Abalone harvest Prediction Tool:
  AhPT is a prediction tool that uses artificial intelligence techniques to perform predictions. The tool learns trends from historical data and uses probabilistic inference to perform harvest predictions.
- The AhPT Process Platform:
  The AhPT Process Platform is a browser-based software development process platform that was designed to adhere to the practices and principles of Agile Modelling. The AhPT Process Platform makes us of both user centred design and requirements traceability. It has been evaluated heuristically to have fulfilled its main design goal.
- The Inference Engine:
  An implementation of Judea Pearl's Message passing algorithm benchmarked against JavaBayes and has been found to result in adequate inference.
- The Learning Engine:
  An implementation of the Maximum likelihood estimate algorithm benchmarked against BayesiaLab and has been found to have a negligible difference in results with BayesiaLab.
- Production Simulation:
  This consists of the simulation of the production events at the farm, which covers the growth, exporting and movement between products and sizes of the abalone.

Given the comments above, we conclude this project to be a success

6. REFERENCES

[1] PEARL, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference.* San Mateo, California: Morgan Kaufmann

[2] MURPHY, K. 1998. *A brief introduction to Graphical Models and Bayesian Networks.*[Online]. Available: http://www.ai.mit.edu/~murphyk/Bayes/bintro.html

[3] CHARNIAK, E. 1991. *Bayesian Networks without* Tears. Menlo Park, California: American Association for Artificial Intelligence

[4] KRAUSE,J P. 1999 . *Learning probabilistic network*. New York, USA: Cambridge University Press. [Online]. Available : http://citeseer.ist.psu.edu/cache/papers/cs/430/http:zSzzSzwww.aai.orgzSzbayesUS_krause.pdf/krause98learning.pdf

[5] NILSSON, N.J. 1998. *Artificial Intelligence: A New Synthesis.* San Francisco, California: Morgan Kaufmann

[6] DEVIREN, M et al. 2001. *Structural Learning of Dynamic Bayesian Network in Speech Recognition.* [Online]. http://citeseer.ist.psu.edu/deviren01structural.html

[7] DUGAD, R et al. 1996. *A tutorial on Hidden Markov Models*. Indian Institute of Technology, Bombay, Signal Processing and Artificial Neural Networks Laboratory, Technical Report SPANN-96.1

[8] BARBER, D. 2003. *Probabilistic Modelling and Reasoning Dynamic Bayesian Networks: Discrete Hidden Variables* [Online]. Available: http://anc.ed.ac.uk/~dbarber/pmr/pmr.html

[9] MURPHY, P. 2002 .*Dynamic Bayesian Network: Representation , Inference and Learning.*[Online]. www.ai.mit.edu/~murphyk/Thesis/thesis.pdf

[10] AMBLER, S. 2001. *An introduction to Agile Modelling.* [Online]. Available: http://www.agilemodeling.com/essays/agileModeling.htm

[11] BECK, K et al. 2001. *The Manifesto for Agile Software Development.* [Online]. Available: http://www.agilemanifesto.org/

[12] BREZILLON, P & POMEROL, J. 2001. *Is Context a Kind of Collective Tacit Knowledge?* European CSCW 2001 Workshop on Managing Tacit Knowledge, Bonn, Germany, Jacovi and A. Ribak (Eds.), pp 23-29.

[13] DIEZ, F. 1996. *Local Conditioning in Bayesian Networks*. Dpto Inteligencia Artificial. UNED, Sena del Rey, Madrid, Spain: [UNKOWN]

[14] KIM, J & PEARL, J. 1983. *A Computational model for Causal and Diagnostic Reasoning in Inference Systems.*

Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Los Altos, California, W. Kaufmann.

[15] RUSSELL, S. & NORVIG, P. 1995. *Artificial Intelligence: A modern approach*. Upper Saddle River, New Jersey: Upper Saddle River: Prentice Hall.

[16] BECK, K et al. 2001. *The Manifesto for Agile Software Development.* [Online]. Available: http://www.agilemanifesto.org/

[17] COZMAN, F et al. 1998. *JavaBayes - version 0.346.* [Online]. Available: http://www-2.cs.cmu.edu/~javabayes/Home/

[18] Bayesia. 1998.*Bayesia Your decision partner.* [Online]. Available: http://www.bayesia.com