

Peter Hearty, 2nd year, RADAR.
Supervisor: Norman Fenton.

1. Introduction

Regression based software process models, such as COCOMO [2] are based on large numbers of very different software projects. Their predictions for a “typical” software project are therefore of limited value. They also lack the ability to distinguish between alternative causal relationships.

Causal models based on Bayesian Nets (BNs) overcome these disadvantages [3]. However, even causal models need some form of input. These can be in the form of specification metrics, such as Function Points [1], or code metrics such as lines of code.

Agile development methods [4] have no large scale specification or design phases. Function point counts or equivalent estimates of project size are therefore unavailable. So, with no data to make predictions with, how can we plan or assess the risk of agile projects?

Agile methods rely heavily on iterative development and customer feedback. This allows us to use another feature of Bayesian nets: their ability to learn. We start with generic predictions using the kind of data which informs the COCOMO model, but quickly adapt the model to match local conditions.

2. Extreme Programming

The most popular of all of the agile development methods is Extreme Programming (XP) [5].

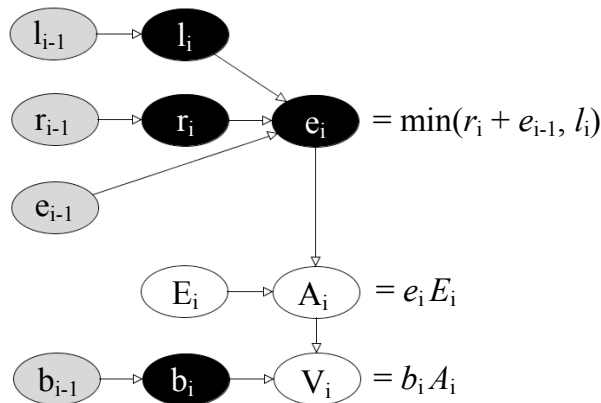


Fig. 1. Project velocity model.

The basic unit of work in Extreme Programming (XP) is the *User Story*, measured in Ideal engineering Days (IEDs). When iteration i finishes, the estimated efforts for the completed user stories are added together to create the Project Velocity (V_i). The customer selects the highest priority uncompleted user stories whose estimates sum to V_i . These user stories are then scheduled for iteration $i + 1$.

A BN model for an XP iteration is shown in Fig. 1. Multiple iterations are linked together to form a full project model. Table 1 summarizes the model variables.

Table 1 Symbol definitions

| Symbol | Meaning |
|---------|--|
| s_i | Productive effort to date. $s_i = s_{i-1} + V_i = \sum V_i, s_i \in [0, \infty)$. |
| E_i | Iteration effort in man-days. $E_i \in [0, \infty)$. |
| U_i^j | Estimated effort of j^{th} user story in iteration i . $U_i^j \in [0, \infty)$. |
| A_i | Actual productive effort in iteration i . $A_i = E_i \times e_i, A_i \in [0, \infty)$. |
| V_i | Project Velocity in iteration i . $V_i = \sum_j U_i^j, V_i \in [0, \infty)$. |
| b_i | Estimation bias. $b_i = V_i / A_i, b_i \in [0, \infty)$. |
| e_i | Process effectiveness in iteration i . $V_i = E_i \times e_i, e_i \in [0, 1]$. |
| l_i | Effectiveness limit. The maximum value that the e_i can take, $l_i \in [0, 1]$. |
| r_i | Process improvement. $e_i = \min(e_{i-1} + r_i, l_i), r_i \in [-1, 1]$. |

To distinguish between a model prediction and a measured value, we use an underscore to denote the measurement. So if V_3 is the predicted value for the velocity at iteration three, then \underline{V}_3 is the measured value.

3. Model Validation

Williams, Shukla and Anton [6] provided a detailed description of an XP project developed at Motorola. The project was developed in a series of eight iterations of between two and three weeks duration.

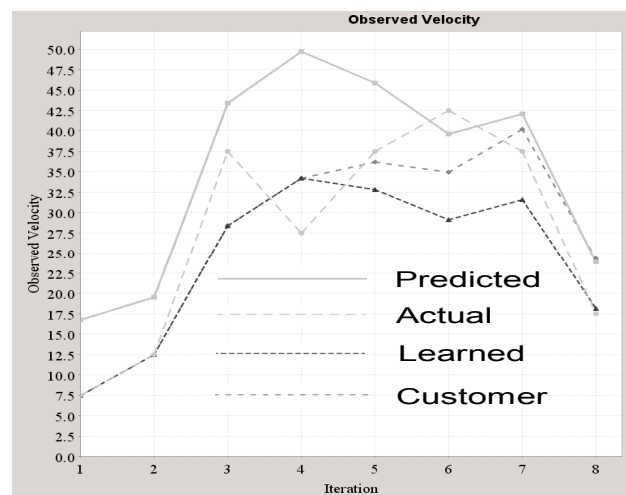


Fig. 2. Predicted vs. actual Motorola V

Initially we simply enter values for E_i into the model (no values for \underline{V}_i entered). The median values from the

V_i distributions are shown in Fig. 2 (the “Predicted” graph). Actual values for V_i are shown in the same figure for comparison (the “Actual” graph).

The graphs in Fig. 2 show the change in predicted values when the first two V_i values have been entered. The “Predicted” graph becomes the “Learned” graph. The predictions for V_i in iterations 3 and 4 improve as a result. However, the predicted V_i for iterations 5, 6 and 7 are significantly worse.

At the start of the 5th iteration, the Motorola team had constant access to an onsite customer. The “Onsite Customer” indicator node was therefore added (Fig. 3) and set to “Very High” for these iterations. The result is the “Customer” curve.

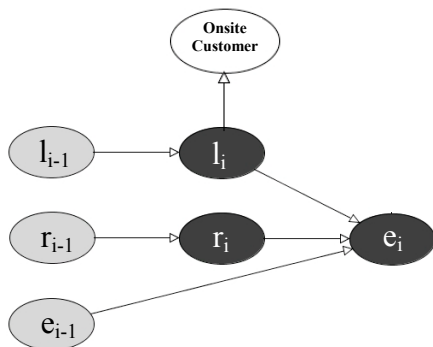


Fig. 3 The "Onsite Customer" indicator node

Fig. 4 shows a modified version of the velocity fragment of the model. This includes an additional link node, s_i , which acts as the cumulative sum of V to date.

Plots of s_i for the initial prediction, the learned prediction and the actual scenarios are shown in Fig. 5. If the total estimate to complete the entire project is, say, 200 IEDs, then we can immediately read off from the graph how long it will take to complete the project.

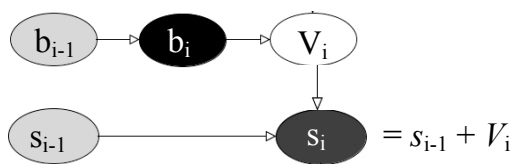


Fig. 4. Project Velocity summed to date.

The initial predictions of the model are too optimistic. However, once the model has learned from the V_1 and V_2 observations, and account has been taken of the onsite customer, the predictions are virtually indistinguishable from the actual outcome.

The model can also quantify the uncertainty involved in completing 200 IEDs within 8 iterations. Fig. 6 shows the cumulative distribution functions for the s_i node in iteration 8.

The vertical line allows us to read off the probability of completing up to 200 IEDs by the end of the 8th iteration. For the “Initial” scenario, there is only a 10% chance of completing less than 200 IEDs, i.e. there is a 90% chance of completing 200 IEDs or more.

Once the model has learned from V_1 and V_2 , the probability is revised down to a 65% probability. The actual number of IEDs delivered was 224.

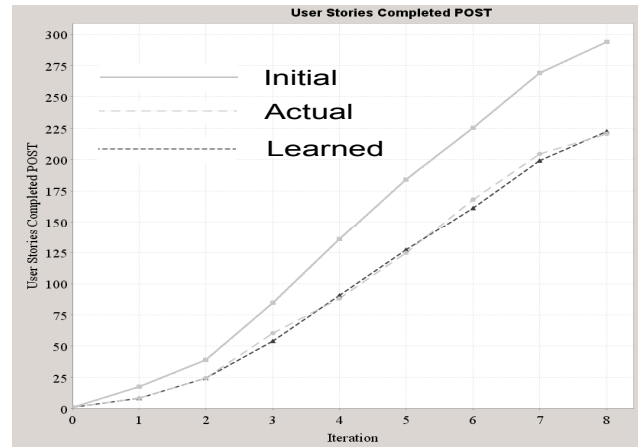


Fig. 5 Sum V_i to date

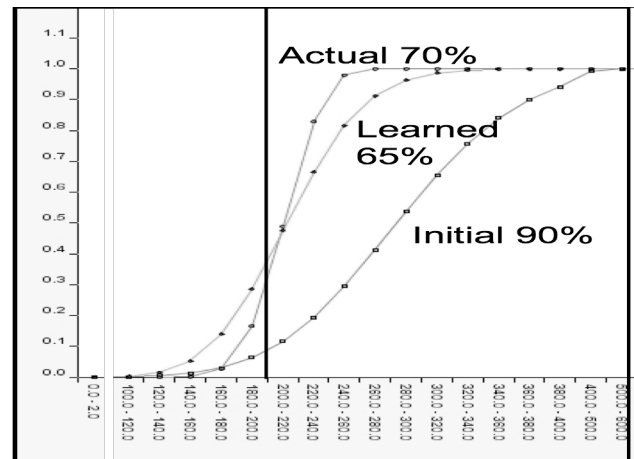


Fig. 6 Iteration 8 cumulative distributions, Sum V to date

4. References

- [1] Albrecht A.J., “Measuring Application Development Productivity,” Proc. Joint SHARE/GUIDE/IBM Application Development Symp.,pp. 83-92, 1979
- [2] Boehm B, Software Engineering Economics, Prentice-Hall, 1991.
- [3] Fenton NE, Neil M, “A Critique of Software Defect Prediction Models,” IEEE Transactions on Software Engineering, 25(4):675-689, September 1999.
- [4] Agile Manifesto, <http://www.agilemanifesto.org/>
- [5] Beck K, Extreme Programming Explained, Embrace Change, Addison-Wesley Professional; 1st edition (2000)
- [6] Williams L, Shukla A, Antón AI, An Initial Exploration of the Relationship Between Pair Programming and Brooks’ Law, Proceedings of the Agile Development Conference (ADC’04)