# Project Data Incorporating Qualitative Factors for Improved Software Defect Prediction

Norman Fenton, Martin Neil, William Marsh, Peter Hearty and Łukasz Radliński[‡]
*Department of Computer Science*
*Queen Mary, University of London*
*Mile End Road, London*
*and [‡]Institute of Information Technology in Management*
*University of Szczecin, Poland*
*norman, martin, william, hearty, lukrad @dcs.qmul.ac.uk*

Paul Krause
*Department of Computing*
*University of Surrey*
*Guildford, Surrey, UK*
*p.krause@surrey.ac.uk*

## Abstract

*To make accurate predictions of attributes like defects found in complex software projects we need a rich set of process factors. We have developed a causal model that includes such process factors, both quantitative and qualitative. The factors in the model were identified as part of a major collaborative project. A challenge for such a model is getting the data needed to validate it. We present a dataset, elicited from 31 completed software projects in the consumer electronics industry, which we used for validation. The data were gathered using a questionnaire distributed to managers of recent projects. The dataset will be of interest to other researchers evaluating models with similar aims. We make both the dataset and causal model available for research use.*

## 1. Introduction

The proper goal of research in software metrics [7, 11, 12] is to help project managers make decisions under uncertainty. In particular, we wish to be able to estimate the cost of developing software, and to predict the quality likely to be achieved by a given development effort. The MODIST ('Models of Uncertainty and Risk for Distributed Software Development') Project [14], which was part-funded by the EC, was concerned with improved predictions of quality in large distributed software projects. The project partners were Agena, Israel Aircraft Industries, QinetiQ and Philips. As part of this project a group of experienced project managers identified a set of factors influencing cost and quality outcomes, which we formed into a causal model. This model is summarized in Section 2.

The objective of this paper is to describe how we validated this model and to make the data available to other researchers. We needed data that was not available in any publicly accessible form (even though similar factors are used in models supporting software managers, most notably COCOMO-II [2] for software cost estimation). For example, whereas the ISBSG dataset [9] (which is accessible at low cost and contains approximately 3500 projects) helped us to quantify some of the relationships in the model, it does not help in validation because of the absence of the qualitative, causal factors.

To get the necessary data, senior project managers in one organization provided the information for 31 new projects. We had to provide refined and more detailed descriptions and measurement schemes for most of the factors in the model. This process is described in Section 3. The resulting quantitative data is presented in Section 4, with the qualitative data in Section 5. Section 6 describes some issues arising from data collection, while in Section 7 we summarise the model validation results. The results show that the causal model, built using expert judgement and historical data, was able to make accurate predictions for the new projects.

## 2. Background: the causal model

This section is a summary overview of the causal model whose factors were elicited from experienced project managers in the MODIST project [14]. The model, which is a Bayesian Network, is presented in schematic form in Figure 1, where each rectangle represents a subnetwork. A detailed description of both the model itself and the Bayesian Network formalism is not necessary in this paper since our focus is on the data. However, for those interested the necessary details can be found in [4, 5, 6, 8, 10, 15]. The model itself can be downloaded from [13] and viewed and executed using Bayesian net software which can be downloaded for free from [1].

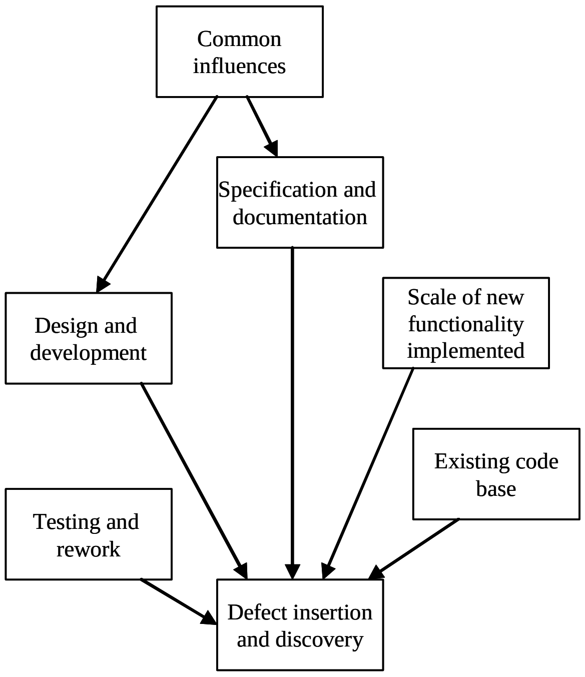
Common influences
Specification and documentation
Design and development
Scale of new functionality implemented
Existing code base
Testing and rework
Defect insertion and discovery

**Figure 1. Schematic view of defect prediction model**

Examples of two of the subnetworks, shown as rectangles in Figure 1, are provided in Figure 2 and Figure 3. Each subnetwork is a part of the Bayesian Network, with nodes representing probabilistic variables and arcs representing causal relationships between variables. It is important to note that the model not only reflects relationships between variables which could be reflected in regression-type models, but also direct cause-effect relationships. For example, a more rigorous testing process leads to an increased probability of finding and fixing a defect and thus to a reduced number of defects left in the software after testing. As an extreme case, the model includes the knowledge that no defects will be found if no testing is done.

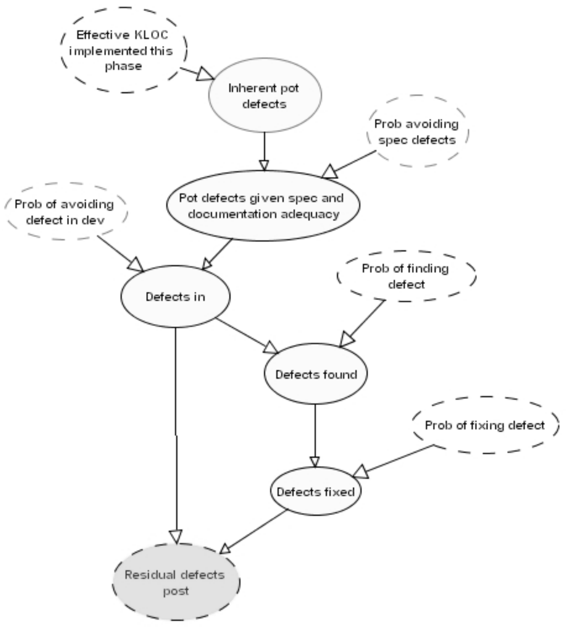
Effective KLOC implemented this phase
Inherent pot defects
Prob avoiding spec defects
Pot defects given spec and documentation adequacy
Prob of avoiding defect in dev
Defects in
Prob of finding defect
Defects found
Prob of fixing defect
Defects fixed
Residual defects post

**Figure 2. Defect insertion and discovery subnet**

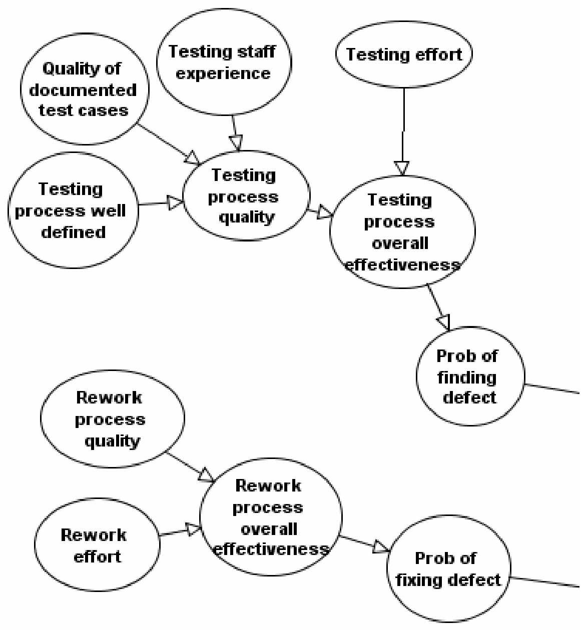

**Figure 3. Testing and rework subnet**

Figure 2 shows the core of the Bayesian Network: the nodes shared between this subnetwork and the others are shown with dashed edges (unshaded). These shared nodes are unobservable quantities, such as the probability of finding a defect present in the software during testing. These values are predicted from qualitative process factors. Figure 3 shows one example of the subnetworks containing these qualitative process factors, with two of the shared nodes shown on the right hand side of the figure.

We developed this causal model (see [8] for further details) based on a combination of the following sources:

- empirical data from the literature;
- empirical data from the project partners;
- subjective judgment of project managers and other experts in the collaborative project, where no relevant data were available.

The causal model was not developed from the data reported in this paper, which was instead gathered after the model had been developed to validate the model.

## 3. Qualitative factors

As the first stage of the development of the causal model outlined in Section 2, partners in the MODIST Project [14] identified qualitative factors that they believed had a significant influence on the outcome of a software project. Once the model had been built the second stage was to gather a dataset to validate the model; to do this effectively a more detailed description of each factor was needed. In Section 3.1 we describe the set of factors, together with the first level of detailed description. Section 3.2 gives an extract of the subsequent questionnaire given to project managers to gather data from completed projects. Section 3.3 discusses some issues arising from this method of measuring the qualitative project factors.

Although it was intended that the validation dataset would cover all the data used in the Bayesian Network, this was not achieved. Some questions were not answered by some project managers. A small number of variable were omitted altogether. This arose were the data recording practices of the projects did not match the assumptions of the questionnaires: in particular rework was not distinguished from testing, and the concept of the 'effort' spent on a project phase, relative to what would be expected on average, was not used by project managers. Fortunately, a Bayesian Network handles missing data (see Section 6.3).

### 3.1. Factor descriptions

The factors are conveniently grouped under five topics: specification and documentation process (Table 1), new functionality (Table 2), design and development process (Table 3), testing and rework (Table 4) and finally project management (Table 5). However, this grouping is not part of the dataset; the factors should all be considered to be project attributes.

Each factor is named and described by a question to be answered. The descriptive questions were specifically tailored for the organisation providing the project data.

**Table 1 Specification and documentation process**

| | **Factor Name** | **Descriptive Question** |
|---|---|---|
| S1 | Relevant Experience of Spec & Doc Staff | How would you rate the experience and skill set of your team members for executing this project during the requirements and specifications phase? |
| S2 | Quality of Documentation inspected | How would you rate the quality of the requirements given by the client or other groups? |
| S3 | Regularity of Spec & Doc Reviews | Have all the Requirements, Design Documents and Test Specifications been reviewed in the project? |
| S4 | Standard Procedures Followed | In your opinion, how effective was the review procedure? |
| S5 | Quality of Documentation inspected | What was the review effectiveness in the project for the requirements phase? |
| S6 | Spec Defects Discovered in Review | In your opinion, is the defect density of spec reviews on the high side? |
| S7 | Requirements Stability | How stable were the requirements in your project? |

**Table 2. New functionality**

| | **Factor Name** | **Descriptive Question** |
|---|---|---|
| F1 | Complexity of new functionality | What was the complexity of the new development or new features that happened in your project? |
| F2 | Scale of New functionality implemented | How large was the extent of working on new functionality rather than just enhancing the older functionalities in your project? |

| | Factor Name | Descriptive Question |
|---|---|---|
| F3 | Total no. of Inputs and Outputs | For your product domain, would you rate the total no of outputs/inputs (newly developed / enhanced) as high? |

**Table 3. Design and development process**

| | Factor Name | Descriptive Question |
|---|---|---|
| D1 | Relevant Development Staff Experience | How would you rate the experience and skill set of your team members for executing this project during the design and development phase? |
| D2 | Programmer capability | On an average, how would you assess the Quality of code produced by the team members? |
| D3 | Defined processes followed | What was the review effectiveness in the project for the Design and Development phase? |
| D4 | Development Staff motivation | What is your opinion about the motivation levels of your team members? |

**Table 4. Testing and rework**

| | Factor Name | Descriptive Question |
|---|---|---|
| T1 | Testing Process Well Defined | How effective was the testing process adopted by your project? |
| T2 | Staff Experience – Unit Test | What was the level of software test competence of those performing the unit test? |
| T3 | Staff Experience – Independent Test | How would you rate the experience and skill set of the independent test engineers (Integration, functional or sub-system testing, Alpha, Beta)? |
| T4 | Quality of Documented Test Cases | What was the extent of the defects that were found using formal testing against the intuitive / random testing? |

**Table 5. Project Management**

| | Factor Name | Descriptive Question |
|---|---|---|
| P1 | Dev. Staff Training Quality | What is the coverage of the identified project / process related trainings as well as trainings identified as per the roles, by the team members? |
| P2 | Configuration Management | How effective is the project's document management and configuration management? |
| P3 | Project Planning | Has the project planning been done adequately? |
| P4 | Scale of Distributed Communication | How many sites/groups were involved in the project. |
| P5 | Stakeholder involvement | To what extent were the key project stakeholders involved? |
| P6 | Customer involvement | How good was customer interaction in the project? |
| P7 | Vendor Management | How would you rate the Vendor / Sub-contractor Management (if applicable)? |
| P8 | Internal communication / interaction | How would you the rate the quality of internal interactions / communication within the team? |
| P9 | Process Maturity | What's your opinion about process maturity in the project? |

## 3.2. Questionnaire Design

Qualitative data are expressed on a 5-point ordinal scale. The ordinal values used are: Very High, High, Medium, Low, Very Low. The data values were gathered using a questionnaire, which was completed by the project manager, project quality manger or other senior project staff. Each questionnaire item consists of:

- More detailed questions
- An interpretation of the ordinal scale.

For example, for factor S1 'Relevant Experience of Spec & Doc Staff', the additional questions are:

1. Did the Requirements team have adequate experience in analysing and generating requirements?
2. Did the Requirements team have adequate domain expertise?

and the ordinal scale points are:

**Very High:** Software engineers with greater than three year's experience in requirements management, and with extensive domain knowledge.

**High**: Software engineers with greater than three year's experience in requirements management, but with limited domain knowledge.

**Medium:** Software engineers having between one and three year's experience in requirements management.

**Low:** Software engineers having between one and three year's experience, but with no experience in requirements management.

**Very Low:** Software engineers with less than one year's experience, and with no previous domain experience.

In some cases the questionnaire used a set of criteria and a score. An example is the factor S4 'Standard (Review) Procedures Followed'. The detailed questions, giving the criteria, are:

1. In case of changes after baselining, have the major changes been re-reviewed?
2. Are there any re-review triggers/criteria defined?
3. Have some domain specific standards been adhered to (like design rules, re-engineering guidelines, architectural guidelines, etc)?
4. Was the requirements document checked for review worthiness or pre-review checklist filled before the review?
5. Have the reviews been planned upfront?
6. Have the reviewers been assigned upfront?
7. Were the reviews role-based?
8. Were the reviewers identified appropriate and experienced enough for reviewing?
9. Was there adequate preparation time available for the reviewers?
10. Were there overview sessions for all complex work products?

The scale point is then derived as follows:

**Very High**: All the 10 sub questions answered yes
**High**: 7-9 of the sub questions answered yes
**Medium**: 5-6 of the sub questions answered yes
**Low**: 4 of the sub questions answered yes
**Very Low**: less than 4 of the sub questions answered yes.

### 3.3. Measurement Issues

The factors used in the model were originally identified by a group of project manager from different partners in the MODIST project. Although from different organisations, it was possible for the project managers to agree on the importance of factors such as 'Requirements Stability'. A further issue is whether it is possible to measure such values consistently between organisations.

As shown by the example in Section 3.2, we designed the questionnaire to used objective criteria, such as the number of years of experience, whether possible. However, we do not claim external validity of these measurements, since this is not needed for our approach, as we explain below.

One way experts were used in building the model was to estimate the 'strength' of the effect of each qualitative factor in the causal model. This information is represented in the conditional probability table for each node in the Bayesian Network. As a result of this process, the model is applicable within the organisation where the experts have gained their experience. Since the model itself is not universal, there is no need for the measurements to be so. It does not follow that the scope of the validation (see Section 7) becomes trivial, even tautological, as a result. Instead, the validation shows that a model constructed using expert judgement and historical data, within one organisation, can be used within the same organisation to predict accurately the outcome of new projects. On the other hand, the validation described here does not consider issues such as the external validity of the causal structure of our model (see Section 6.4).

## 4. Quantitative data

The projects developed software for consumer electronics products. Each project developed or enhanced some functionality provided by a product. The developed software was not stand-alone but was integrated with other software subsystems in the product.

A waterfall lifecycle was followed. The software engineering part of the lifecycle covered a specification review, design, a design review and development up to unit testing. The software was then passed to independent test in several phases, from software integration testing to overall system (i.e. product) testing.

**Table 6. Size, effort and defects**

| Project | Hours | KLoC | Language | Defects |
|---|---|---|---|---|
| 1 | 7109 | 6.0 | C | 148 |
| 2 | 1308 | 0.9 | C | 31 |
| 3 | 18170 | 53.9 | C | 209 |
| 4 | 7006 | - | C | 228 |
| 5 | 9434 | 14.0 | C | 373 |
| 6 | 9441 | 14.0 | C | 167 |
| 7 | 13888 | 21.0 | C | 204 |
| 8 | 8822 | 5.8 | C | 53 |

| Project | Hours | KLoC | Language | Defects |
|---|---|---|---|---|
| 9 | 2192 | 2.5 | VC++,MFC | 17 |
| 10 | 4410 | 4.8 | C | 29 |
| 11 | 14196 | 4.4 | C | 71 |
| 12 | 13388 | 19.0 | C | 90 |
| 13 | 25450 | 49.1 | C | 129 |
| 14 | 33472 | 58.3 | C | 672 |
| 15 | 34893 | 154.0 | C | 1768 |
| 16 | 7121 | 26.7 | C | 109 |
| 17 | 13680 | 33.0 | C | 688 |
| 18 | 32366 | 155.2 | C | 1906 |
| 19 | 12388 | 87.0 | C | 476 |
| 20 | 52660 | 50.0 | C | 928 |
| 21 | 18748 | 22.0 | C | 196 |
| 22 | 28206 | 44.0 | C | 184 |
| 23 | 53995 | 61.0 | C | 680 |
| 24 | 24895 | 99.0 | C | 1597 |
| 25 | 6906 | 23.0 | C | 546 |
| 26 | 1642 | - | C | 261 |
| 27 | 14602 | 52.0 | C | 412 |
| 28 | 8581 | 36.0 | C | 881 |
| 29 | 3764 | 11.0 | C | 91 |
| 30 | 1976 | 1.0 | C | 5 |
| 31 | 15691 | 33.0 | C | 653 |

Most of the software development was at one site, but the overall development was distributed over different locations in a global organisation. Both the software specification and the independent testing were typically at a different location to the software development.

The data values are shown in Table 6:

- Software size: the size, in KLoC of the developed code and the development language (Figure 4 shows the distribution of code size in the dataset). Note that for two projects, this data was not available: the Bayesian Network can still be used and it will assume the projects to be ‘average’ but of uncertain size.
- Effort: development effort measured in person hours for the software development, from specification review to unit test

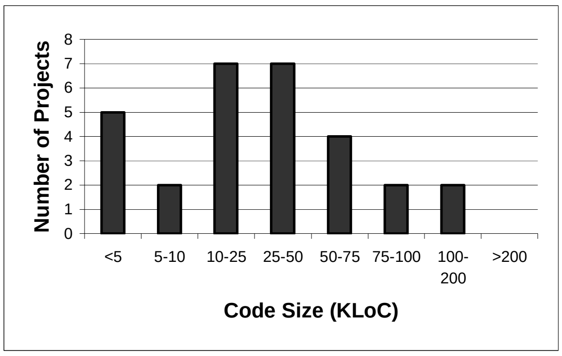

**Figure 4. Code size distribution**

- mDefects: functional defects discovered during all the independent testing phases, following the software development.

In some projects existing software was reused as part of the development. The impact of this on the dataset is considered in Section 6.

This new dataset could, of course, be used to build traditional statistical/regression based models, as indicated, for example, by Figure 5. This could be the basis for a simple regression model relating KLoC to defects; indeed the correlation coefficient here is quite high (0.78). However, this does not correspond to the way that we used this data, which was to validate a model created before the data was gathered. Therefore, we do not pursue this comparison.

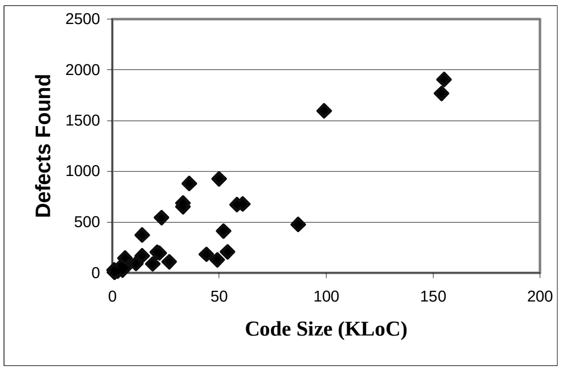

**Figure 5. Code size versus defects**

## 5. Qualitative data

The data values are shown in tables 7 to 10. Missing data values are marked with ‘-’ (see Section 6.3). The letters VL, L, M, H, VH correspond to the ordinal scale described in Section 3.2 (very low to very high).

**Table 7. Specification and documentation process data**

| Project | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|---|---|---|---|---|---|---|---|
| 1 | H | M | VH | H | M | H | L |
| 2 | H | H | VH | H | M | H | H |
| 3 | H | H | VH | H | H | VH | H |
| 4 | L | L | M | L | L | L | L |
| 5 | H | M | H | M | H | - | M |
| 6 | VH | M | VH | M | H | - | H |
| 7 | L | M | VH | H | H | L | M |
| 8 | M | M | H | M | H | L | H |
| 9 | H | VH | VH | H | VH | M | VH |
| 10 | H | H | H | M | H | M | H |
| 11 | H | M | H | M | H | H | H |
| 12 | H | M | H | M | M | M | L |
| 13 | VH | M | M | L | M | H | L |
| 14 | H | H | H | H | H | H | H |
| 15 | H | H | H | H | H | VH | VL |
| 16 | H | H | H | H | H | H | M |
| 17 | VH | H | M | L | H | H | M |
| 18 | M | H | H | H | H | VH | VL |
| 19 | H | M | H | H | H | H | M |
| 20 | L | L | M | VL | L | M | VL |
| 21 | H | H | H | M | L | M | M |
| 22 | L | L | M | M | M | M | L |
| 23 | M | H | VH | H | L | M | M |
| 24 | M | M | M | H | M | H | L |
| 25 | M | H | - | H | M | M | M |
| 26 | M | M | H | M | H | H | H |
| 27 | H | M | VH | M | M | VH | M |
| 28 | H | L | VH | M | M | M | L |
| 29 | H | M | VH | H | M | M | VH |
| 30 | H | H | VH | H | H | M | VH |
| 31 | - | H | H | M | M | H | M |

**Table 8. Data for new functionality, design and development process**

| Project | F1 | F2 | F3 | D1 | D2 | D3 | D4 |
|---|---|---|---|---|---|---|---|
| 1 | M | L | M | L | H | H | H |
| 2 | L | VL | M | L | H | H | H |
| 3 | H | H | VH | H | VH | H | VH |
| 4 | M | L | M | L | M | L | M |
| 5 | H | H | VH | L | M | H | H |
| 6 | M | M | VH | M | H | M | M |
| 7 | L | VL | M | M | VH | H | H |
| 8 | M | L | M | H | H | M | M |
| 9 | L | L | M | H | VH | VH | H |
| 10 | M | L | M | H | H | H | H |
| 11 | H | H | H | H | H | H | H |
| 12 | H | H | H | VH | M | M | H |
| 13 | H | H | H | H | H | H | H |
| 14 | VH | H | H | H | H | H | H |
| 15 | H | H | M | H | H | H | H |
| 16 | L | VL | M | H | H | H | H |
| 17 | L | VL | M | M | M | H | H |
| 18 | VH | VH | H | M | H | H | H |
| 19 | H | H | H | H | H | H | H |
| 20 | VH | H | VH | VL | VL | L | H |
| 21 | L | M | VH | H | H | H | H |
| 22 | M | M | VH | H | M | L | H |
| 23 | H | VH | VH | L | H | H | H |
| 24 | M | M | H | M | H | H | M |
| 25 | H | VL | H | M | H | M | H |
| 26 | M | H | M | L | M | M | M |
| 27 | H | VH | VH | M | L | M | H |
| 28 | VH | VH | VH | M | L | H | H |
| 29 | M | M | H | VH | VH | H | H |
| 30 | L | L | M | H | H | H | H |
| 31 | M | M | H | H | H | H | H |

**Table 9. Testing and rework data**

| Project | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 1 | M | H | L | H |
| 2 | H | H | L | H |
| 3 | H | H | H | H |
| 4 | VL | VL | VL | L |
| 5 | M | M | L | M |
| 6 | H | - | M | M |
| 7 | H | M | M | H |
| 8 | H | M | M | M |
| 9 | H | VH | VH | H |
| 10 | H | M | M | M |
| 11 | H | H | M | M |
| 12 | H | H | M | M |
| 13 | M | M | L | M |
| 14 | H | H | H | H |
| 15 | M | H | M | M |
| 16 | M | H | M | M |
| 17 | M | L | L | H |
| 18 | H | H | M | M |
| 19 | H | M | M | H |
| 20 | VL | VL | VH | H |

| Project | T1 | T2 | T3 | T4 |
|---|---|---|---|---|
| 21 | H | H | H | H |
| 22 | H | M | M | H |
| 23 | H | H | H | H |
| 24 | H | M | M | M |
| 25 | VL | M | H | L |
| 26 | M | L | H | M |
| 27 | M | M | M | M |
| 28 | M | M | M | M |
| 29 | H | VH | VH | H |
| 30 | H | H | H | H |
| 31 | M | H | M | M |

**Table 10. Project management data**

| Project | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VH | H | H | L | H | M | - | VH | H |
| 2 | VH | H | H | L | H | M | - | VH | H |
| 3 | H | VH | H | - | VH | VH | - | VH | VH |
| 4 | L | M | VL | L | M | M | M | H | M |
| 5 | H | H | H | M | M | H | L | VH | M |
| 6 | H | H | H | M | M | VH | L | VH | H |
| 7 | H | H | VH | VL | VH | VH | - | H | VH |
| 8 | M | H | H | VL | H | H | - | H | H |
| 9 | VH | VH | VH | L | VH | VH | - | VH | VH |
| 10 | H | H | H | VL | H | H | - | M | H |
| 11 | H | H | H | VL | H | H | - | M | M |
| 12 | H | H | H | L | H | H | - | M | H |
| 13 | M | H | H | VL | H | M | H | M | M |
| 14 | H | H | H | - | H | H | - | H | H |
| 15 | VH | M | H | M | VH | VH | - | VH | H |
| 16 | VH | M | H | M | VH | VH | - | VH | H |
| 17 | M | M | M | M | M | H | - | H | M |
| 18 | VH | M | H | H | VH | VH | - | VH | H |
| 19 | M | H | H | L | H | H | - | H | H |
| 20 | H | M | L | H | H | M | - | H | H |
| 21 | H | H | H | H | H | H | - | H | H |
| 22 | H | H | M | H | H | H | - | H | H |
| 23 | H | H | H | H | H | M | - | H | H |
| 24 | H | H | M | L | M | H | - | VH | H |
| 25 | M | M | M | M | M | M | M | H | H |
| 26 | L | M | M | L | H | H | L | H | M |
| 27 | H | M | L | L | M | H | H | H | M |
| 28 | H | M | L | L | M | M | - | H | M |
| 29 | M | H | H | L | VH | VH | - | H | H |
| 30 | M | H | H | L | VH | VH | - | H | H |
| 31 | H | H | H | H | VH | VH | - | VH | VH |

## 6. Issues arising from the data collection

The complexity of software projects make gathering data challenging. The most important challenges we faced and lessons we learned during this work are described below. Some of these challenges are not fully resolved by the data included in this dataset; how these issues were addressed in our models is described elsewhere [8].

### 6.1. Software size: intrinsic complexity

Because of the need to have a size based measure based only on the amount of functionality to be implemented, we had hoped to use function points as the key metric for this purpose, as recommended from the MODIST work. Unfortunately, function points were not used by the software development organisation providing this data. It is well known [7] that KLoC measures program length but the length of the program is only one aspect of the size of the development task – which we term the 'intrinsic complexity'. The factors F1-F3 were included in the data gathered to give a better estimate of the intrinsic complexity than code size alone. Unfortunately, intrinsic complexity is not an observable quantity, so finding sufficient factors to estimate the size of the development task remains a challenge.

### 6.2. Code reuse

It is very common for software development to be carried out as part of a product line development, naturally giving rise to software reuse. This complicates the measurement of software size – the lines of developed software differs from the length of the developed program – and also impacts the prediction of defects, since the quality of the reused software is variable.

### 6.3. Missing Data Values

Given the complexity of a dataset that attempts to cover relevant software cost and quality drivers, it is inevitable that some data values will be missing. It is essential that software prediction methods are able to cope with missing data values.

The Bayesian net model used in this study is one such method that handles missing data, since the model includes prior probability distributions for all the project data.

### 6.4. Generality of the data

An objective of the partners in the MODIST project was to identify only factors (and the means of measuring them) that were generally relevant to complex software projects. Achieving this objective would enable different organizations to make use of the causal model. We recognize that the more detailed descriptions and questionnaires refer to process-specific information. The objective of generality would still be partly achieved if other organizations (using different processes) used the same factors, but adapted the questionnaire as a means of measuring them.

## 7. Model validation

We validated the causal model using the presented project dataset. We did this by entering, for each project, data excluding the defect data and ran the Bayesian net model. This produces a (predicted) probability distribution for number of defects found in independent testing. Using the median values of these distributions enables us to calculate the accuracy of the predictions. As presented in Figure 6, we achieved a very high accuracy with an $R^2$ value of 0.9311.

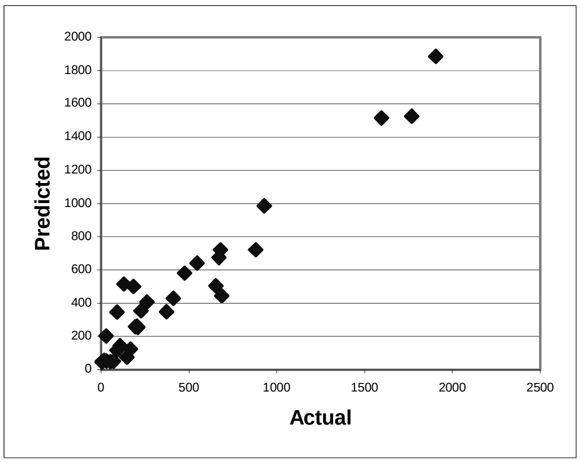

**Figure 6. Predicted and actual values**

The outstanding result of the model validation gives great confidence in the value of the causal model, but of course further validation using additional datasets would provide even greater confidence in the integrity and robustness of the model. Moreover, the validation we have described does not cover all the ways that the Bayesian net model can be used, since it is possible to enter data at any of the variables and obtain the probability distributions at any of the unobserved variables.

## 8. Conclusion

We have presented a comprehensive dataset for 31 software development projects. This dataset incorporates the set of quantitative and qualitative factors that were previously built into a causal model of the software process. The factors (which had been identified by a consortium of software project experts) include values for code size, effort and defects, together with qualitative data values judged by project managers (or other project staff) using a questionnaire. We have used these data to evaluate the causal model and the results are extremely promising. Specifically the model predicts, with remarkable accuracy, the number of software defects that will be found in independent testing.

Although it is beyond the scope of the paper to discuss the causal model in detail, it should be noted that good predictions of the defects can be achieved by entering relatively few of the project factors (size plus 2 or 3 others). Hence this kind of model can be used for effective decision-support and trade-off analysis during early development phases.

By presenting the raw data in this paper, we hope to enable other researchers to evaluate similar models and decision-support techniques for software managers (the dataset can of course also be used for evaluating more traditional types of software prediction models). We also hope that similar datasets will become more widely available in future.

To ensure full visibility and repeatability, we also provide an electronic version of the causal model for researchers [13]. The model can be viewed and executed by downloading the free trial version of the Bayesian network software [1].

## 9. Acknowledgments

This report is based in part on work undertaken on the following funded research projects: MODIST (EC Framework 5 Project IST-2000-28749), SCULLY (EPSRC Project GR/N00258), SIMP (EPSRC Systems Integration Initiative Programme Project GR/N39234), and eXdecide (EPSRC project EP/C005406) [3]. We also acknowledge the contributions of individuals from Agena, Philips, Israel Aircraft Industries, QinetiQ and BAE Systems.

## 10. References

[1] AgenaRisk Bayesian Network Software Tool, www.agenarisk.com, 2007.

[2] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, “Cost models for future software life cycle process: COCOMO 2.0”, *Annals of Software Engineering*, 1995.

[3] eXdecide, “Quantified Risk Assessment and Decision Support for Agile Software Projects”, *EPSRC project EP/C005406/1*, www.dcs.qmul.ac.uk/~norman/radarweb/core_pages/projects.html .

[4] N.E. Fenton, P. Krause, and M. Neil, “Probabilistic Modelling for Software Quality Control”, *Journal of Applied Non-Classical Logics* 12(2), 2002, pp. 173-188.

[5] N.E. Fenton, P. Krause, and M. Neil, “Software Measurement: Uncertainty and Causal Modelling”, *IEEE Software* 10(4), 2002, pp. 116-122.

[6] N.E. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M Tailor, “Making Resource Decisions for Software Projects”, *Proceedings of 26th International Conference on Software Engineering (ICSE 2004), (Edinburgh, United Kingdom, May 2004)* IEEE Computer Society 2004, ISBN 0-7695-2163-0, pp. 397-406.

[7] N.E. Fenton, and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach (2nd Edition)*, PWS, ISBN: 0534-95429-1, 1998.

[8] N.E. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra, “Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets”, *Information and Software Technology*, Vol. 49 Issue 1, January 2007.

[9] ISBSG International Software Benchmarking Standards Group, www.isbsg.org.

[10] F.V. Jensen, *An Introduction to Bayesian Networks*, UCL Press, 1996.

[11] C. Jones, *Programmer Productivity*, McGraw Hill, 1986.

[12] C. Jones, “Software sizing”, *IEE Review* 45(4), 1999, pp.165-167.

[13] MODIST BN Model, 2007, http://www.dcs.qmul.ac.uk/~norman/Models/BN_Model_PROMISE.html.

[14] MODIST, “Models of Uncertainty and Risk for Distributed Software Development”, *EC Information Society Technologies Project IST-2000-28749*, www.modist.org.

[15] M. Neil, P. Krause, and N.E. Fenton, “Software Quality Prediction Using Bayesian Networks”, in *Software Engineering with Computational Intelligence*, (Ed T.M. Khoshgoftaar), Kluwer, ISBN 1-4020-7427-1, Chapter 6, 2003.