Project Management through States

Benjamin Menhorn
Department of Embedded Systems/Real-Time Systems
Ulm University
89069 Ulm, Germany
benjamin.menhorn@uni-ulm.de

Frank Slomka
Department of Embedded Systems/Real-Time Systems
Ulm University
89069 Ulm, Germany
frank.slomka@uni-ulm.de

***Abstract*—Looking at computer science, rising complexities, durations and costs for software and hardware development make it necessary to employ a new holistic model to manage those development processes. This paper introduces a new model which is based on abstract states in order to calculate a project's complexity. Based on this complexity it is now possible to estimate key variables such as costs, duration and progress. An abstract definition of states allows users to adjust this model to their project and to their specific requirements.**

I. Introduction

A. Motivation

Project management has become a major issue for modernly organized companies. Most engineering disciplines exert well defined methods and models to manage, control and evaluate projects. But in computer science, the management of hardware and software projects is still very weak. This expresses not only in cost and time overruns but also in a high cancellation rate of such projects. For hardware development it is not enough to simply count transistors on integrated circuits anymore [1].

The most common cost model in software development COCOMO (COnstructive COst MOdel) is based on lines of code [2]. Costs and duration of a project are adjusted by a questionnaire which determines its complexity. An empirical study of Chris Kemerer analyzed the accuracy of cost models on 15 large completed business data processing projects and has shown a mean error of 600% for projects planned with COCOMO [3].

A better model enables project managers to set a tighter framework to a project. A good project management has to cover three basic purpose fields: a project should a) fulfill its requirements (appropriate), b) in the given time (in time) and c) with the given budget (cost effective) [4]. To achieve those goals, mathematical methods will be presented in the next sections which allow the determination of a project's relevant variables by using states.

Furthermore, this model reduces the demand for empirical data. Today, most models base their estimations on empirical data [5]. Gathering this data is expensive and challenging. With fast changes in technology as well as with new abstraction layers, a data comparison from different projects is difficult [6].

B. Contribution

This model is not limited to hardware and software development but has to proof itself primarily in these areas. The demand for a good model in computer science is high in order to get more and more complex projects under control and keep them within certain cost and duration bounds. But the definitions made in the following are based on abstract formulations and can also be used in other development fields.

The primary output of this model is a calculation of a project's complexity. The complexity then allows the determination of costs, duration and progress of a project.

Complexity can often be considered to be a measure of a project's disorder, which is a property of a system's state. It varies with changes made at possible states of a system. Looking at physics, especially at thermodynamics and statistics, the measurement of complexity comes along with entropy [7]. But entropy is not unknown in computer science. C. E. Shannon also used this term to describe information in a transmitted signal or message [8].

With knowledge of the amount of possible states as well as in- and outputs of a project the complexity of this project can be calculated. As in thermodynamics and information technology the term entropy is used to describe complexity. Altogether, states enable one to calculate entropy and with changes on the project and therefore on its states the entropy can be influenced.

II. States

States as a key variable of this model will be now explained in detail. It is important to note that the term ”states” represents the amount of possible states in terms of inputs, outputs or states of a module, component or elementary part. For different projects, states have to be defined by their purpose. Different examples will be explained in the section about working with states.

A. Elementary modules

By adding the complexity of several modules, the complexity of an overlaying module can be determined. This leads to the ability to calculate the complexity of a whole project. Figure 1 illustrates the steps. Module A is an elementary module which can not be subdivided. Module A and the three B modules form a submodule (grey). This submodule and the modules C form together the whole project. The C modules can also have submodules on their own.

The approach to divide a project/module into submodules is similar to other project management methods (e.g. [9]). But it is essential that it has to be mentioned here. The particularty

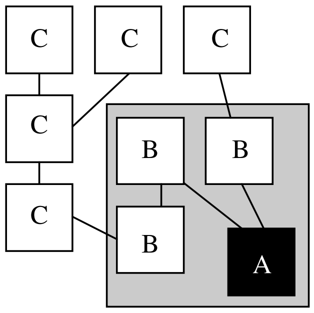


Fig. 1. Calculation a project's complexity

is that also connections and interactions between modules are taken into consideration. This will express in the definitions presented in the next section.

B. Calculating the complexity

The theory of calculating a project's complexity through states has been presented in [10]. The two equations necessary for this paper are given in the following.

Definition 1 (Behavior entropy)
The behavior complexity of a module or project is given by the number of inputs n, the number of outputs m and the number of possible states z and is called behavior entropy S_B.

$$S_B = m \cdot \ln(z^n) \tag{1}$$

Definition 2 (Structure entropy)
A project/module consisting of k modules/elementary modules with n_i inputs and m_i outputs $i = 1, \ldots, k$ has a structure entropy S_S which can be seen as the complexity of this project/module.

$$S_S = \sum_{i=1}^{k} m_i z^{n_i} \ln(z^{n_i}) \tag{2}$$

It was already mentioned that this model also takes relations between modules, submodules and elementary modules into account. This can be found in the equations. They clearly show that the behavior entropy is different from the structure entropy. The structure entropy can not be determined by just adding the behavior entropy of single submodules.

With these two definitions a project's complexity can be calculated. Therefore the structure entropy of modules is calculated. The structure entropy of all modules is a project's entropy. The behavior entropy is used when other modules refer to other modules in a project. In this case the referred module is used and not created. The usage of a module that is already created is less complex than the creation of it. Therefore the behavior entropy is always less or equal to the structure entropy $S_B \leq S_S$. In the following section it is explained how states in general and their amount in particular can be determined.

C. Working with States

In this section some examples of states and the possible amount of states for a variable are given. As the first example a boolean logic is used. In digital circuits connections can either be set to "low" or "high" expressing a digital "0" or "1". Hence a connection and therefore an input or output of a gate on a chip can have two possible states. In the equations z would then be two ($z = 2$).

If boolean logic is extended to standard logic used in modeling circuits with VHDL (Very-High-Speed Integrated Circuits Description Language) there can be nine different states, $z = 9$ (IEEE Standard 1164_1993) [11]. But the number of states depends on the purpose. The nine states can also be grouped according to their effect on the circuit. For example could states be also grouped by their synthesizeable values "1", "0" and "Z".

It's becoming more clearly when looking at software variables. An integer value, for example, can have (depending on the architecture) 2^8 possible states. While most of the states lead to the same action, it doesn't make sense to set z to 2^8. Allow for the use of a variable, different states can be identified: for example in loops those could be: lower out of bounds, lower loop limit, lower inside loop, higher inside loop, higher loop limit and higher out of bounds. For example with an loop `for(int i = 0; i <10; i++){}` this would be `-1,0,1,9,10,11`.

In further publications standard cases for setting up states will be discussed. Here the idea of states was given in order to understand how it looks like to apply a state based model on practical examples.

III. Project Management

A. From states to the key values

The key values of this model are states. In the previous section it was already shown how states can be used to determine the complexity of a project. But also a project's progress, costs and duration can be determined by using states. Figure 2 shows that states allow to determine the complexity of a project, while complexity then allows to determine a project's progress, costs and duration. Costs and duration are written in italic, because of the human influence. As soon as human beings have to be taken into account an exact calculation of variables is impossible. There are more and less talented workers, faster and slower workers and also unexpected events like illnesses, crashes or machine failures. These influence factors can't be determined in advance.

This model enables a user to determine the ratio between complexity and costs or duration very easily. The complexity of a completed project has to be determined and the total costs can be divided by the entropy. As it can also be done for the duration. The more projects are analyzed the better the average value gets. But basically one previous project is enough to determine how much the realization of one state costs and how long it will take. This leads to the claim of linearity discussed in the next section.

$$\begin{tikzcd}
 & \text{States} \arrow[d] & \\
 & \text{Complexity} \arrow[dl] \arrow[d] \arrow[dr] & \\
\textit{Costs} & \text{Progress} & \textit{Duration}
\end{tikzcd}$$

Fig. 2. Model Purpose

B. Linearity

Using states and calculating the entropy enables a linear calculation of cost, duration and progress. Costs per entropy unit (CPU) and duration per entropy unit (DPU) determined from previous projects can now be multiplied with the entropy in order to the determine the currents projects costs C and duration D. The entropy used is the structure entropy of the project plus the behavior entropy of the project.

$$C = (S_S + S_B) \cdot \text{CPU} \tag{3}$$

$$D = (S_S + S_B) \cdot \text{DPU} \tag{4}$$

The progress P of a project is the ratio between realized structure entropy units and a project's total entropy units.

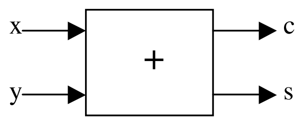
$$P = \frac{S_S(\text{realized})}{S_S(\text{total})} \tag{5}$$

IV. Example

In this section an example project is discussed. The task is to design a half adder. This example is used in order to easily follow the theory of managing the project through states. It is assumed that there are previous completed projects that can be used to find out the costs and duration for the realization of one state. We assume that the realization of an entropy of $\ln(2)$ costs two monetary units (MU) and five time units (TU). As mentioned above, these are the only empirical data needed from previous projects in order to plan the next project.

A. Three kinds of realization

Figure 3 shows the specification of a half adder. It has two inputs labeled x and y and outputs sum s and carry c. There are different kinds of realizing a half adder. Three are shown here. The most simple design would be implementation A shown in figure 4 with an XOR and an AND gate. Implementation B from figure 5 only consists of NAND gates. It has the same amount of gates as implementation C illustrated in figure 6. It will now be shown that the different implementations lead to different complexities, progresses, development durations and costs.



x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 3. Specification of a half adder

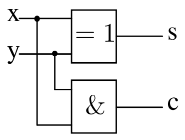


Fig. 4. Half adder with an AND and a XOR Gate (A)

B. Planing the project

First of all the number of possible states needs to be determined. In case under consideration a boolean logic leads to two possible states "1" and "0" or "high" and "low" and therefore z in equation 1 and 2 is two ($z = 2$). For the whole implementation of a half adder two inputs (x and y) as well as two outputs (s and c) can be found. They lead to a behavior complexity of:

$$S_B(\text{HA}) = m_{\text{HA}} \ln\left(2^{n_{\text{HA}}}\right) = 2 \ln\left(2^2\right) = 4 \ln(2) \tag{6}$$

It gets more interesting when taking single gates into account. AND, OR, NAND and XOR gates are each identified by two inputs and one output. Therefore the complexity of those gates is the same and according to formula 1:

$$S_B(\text{Gate}) = m_{\text{Gate}} \ln\left(2^{n_{\text{Gate}}}\right) = 1 \ln\left(2^2\right) = 2 \ln(2) \tag{7}$$

Inverters have only one input and one output. Therefore the behavior entropy becomes:

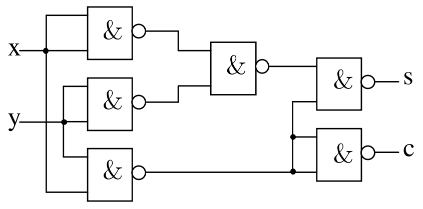


Fig. 5. Half adder with NAND gates (B)

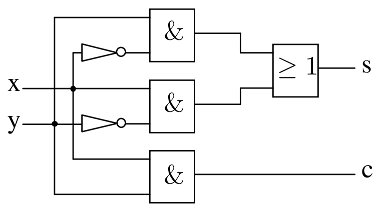


Fig. 6. Half adder with AND, inverter and OR gates (C)

$$S_B(\text{Inv}) = m_{\text{Inv}} \ln(2^{n_{\text{Inv}}}) = 1 \ln(2^1) = 1 \ln(2) \tag{8}$$

It looks logic that all gates have the same complexity. All of them have two inputs and one output, furthermore their logic tables have the same amount of entries. While an inverter has only one input and one output the complexity has to be less then the gates' complexity. Comparing formula 7 with formula 8, the gates' complexity is twice the inverters' complexity.

Now the structure entropy for the three different realizations of a half adder is calculated:

$$S_S(\text{A}) = \underbrace{1 \cdot 2^2 \cdot \ln(2^2)}_{XOR} + \underbrace{1 \cdot 2^2 \cdot \ln(2^2)}_{AND} = 8 \ln(2) \tag{9}$$

$$S_S(\text{B}) = 6 \cdot \underbrace{1 \cdot 2^2 \cdot \ln(2^2)}_{NAND} = 48 \ln(2) \tag{10}$$

$$\begin{aligned} S_S(\text{C}) =& 2 \cdot \underbrace{1 \cdot 2^1 \cdot \ln(2^1)}_{Inv} + 3 \cdot \underbrace{1 \cdot 2^2 \cdot \ln(2^2)}_{AND} \\ &+ \underbrace{1 \cdot 2^2 \cdot \ln(2^2)}_{OR} = 36 \ln(2) \end{aligned} \tag{11}$$

Now costs and durations can easily be calculated by multiplying the structure plus behavior entropy of each realization with the assumed 2 MU respectively 5 TU. It is exemplarily shown for realization A. The results of all three realizations are shown in figure 7.

$$C(A) = (16 \ln(2) + 4 \ln(2)) \cdot 2 \text{ MU} = 40 \text{ MU} \tag{12}$$

$$D(A) = (16 \ln(2) + 4 \ln(2)) \cdot 5 \text{ TU} = 100 \text{ TU} \tag{13}$$

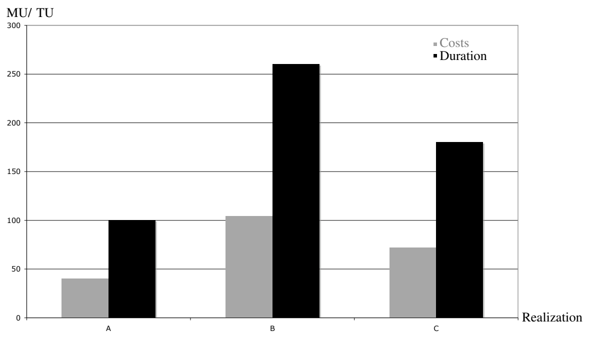


Fig. 7. Project's costs and duration

As equations 3 and 4 already indicate, costs and duration are linear to a projects complexity. Figure 7 illustrates this very well. The project progress is also illustrated by plotting the progress in percent on the y-axis and the relative amount of realized modules on the x-axis. Thereby it is assumed that in project A the inverters are realized at the end of the project. Figure 8 shows the characteristics of the different realizations.

Because inverters have a lower entropy than the other gates the implementation of the two inverters of realization C leads to a lower complexity raise and therefore to a lower progress raise. Comparing realization A and B, the implementation of

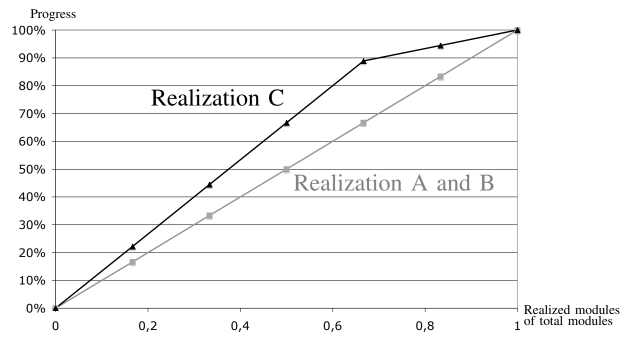


Fig. 8. Project's progress

one gate results in a 50% progress raise for A while in B it is only 16.7%.

V. Conclusion

This paper has shown a new approach to manage the development progress based on states. A project needs to be subdivided into elementary modules, which enables a user to determine the structure and behavior entropy of those modules. A project's costs, duration and progress can now be derived from its complexity (entropy). The use of entropy has the advantage that costs and duration are linear to the complexity. And also the progress of a project is rising linear.

Further publications will test the new model against commonly used models in software and hardware development. Also states should be defined more clearly in standard applications and for commonly used variables and factors in hardware and software development. Remaining are field studies to show the advantage of this model in praxis.

References

[1] V. Ermolayev and W.-E. Matzke, "Towards industrial strength business performance management," in *HoloMAS '07: Proceedings of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 387–400.

[2] B. W. Boehm, *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.

[3] C. F. Kemerer, *An empirical validation of software cost estimation models*. New York, NY, USA: ACM Press, May 1987, vol. 30, no. 5.

[4] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 8th ed. Wiley, February 2003.

[5] N. Hinrichs, P. Leppelt, and E. Barke, "Building up a performance measurement system to determine productivity metrics of semiconductor design projects," in *IEEE International Engineering Management Conference (IEMC), Austin TexasErmolayev2007*, IEEE, Ed. IEEE, 2007.

[6] P. Leppelt, A. Hassine, and E. Barke, "An approach to make semiconductor design projects comparable," in *7th Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS 2006)*. Asian Institute of Technology, 2006, pp. CD–ROM.

[7] K. Stowe, *An Introduction to Thermodynamics and Statistical Mechanics*, 2nd ed. Cambridge University Press, 2007.

[8] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.

[9] T. DeMarco, *Controlling Software Projects: Management, Measurement, and Estimates*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1986.

[10] B. Menhorn and F. Slomka, "Entwurfsentropie: Ein Maß im Schaltungsentwurf," in *7. GI/GMM/ITG-Workshop "Multi-Nature-Systems"*, Ulm University, Department of Embedded Systems/ Real-Time Systems. VDE, January 2009.

[11] K. L. Short, *VHDL for Engineers*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.