States and Complexity

Benjamin Menhorn and Frank Slomka

Ulm University, Germany,
{benjamin.menhorn|frank.slomka}@uni-ulm.de

Abstract. Determining complexity has become a main focus for managing projects. Complexity measures base their statements about a system's complexity on measurement variables (key figures). But fast changes in technology in computer science require complexity measures to be adaptable to changing underlying systems, which should be evaluated. Furthermore, most measurements are based on empirical data which makes projects hard to compare.
This work will discuss whether states could be used as a measurement variable for complexity. Complexity can be considered to be a measure of a system's disorder which is a property of a system's state. Complexity varies with changes made at the amount of possible states of a system. Working with states for measuring complexity has been applied on digital hardware design within the *design entropy concept*. This work will show that states and the design entropy model can also be used to get complexity in other fields under control.

Keywords: states, complexity, design, entropy, measurement

1 Introduction

1.1 Complexity measurement

Before this work discusses aspects of using states to measure complexity this section will point out a general need for complexity measurements. This need for measuring complexity is widely spread over different fields of engineering and other fields. Addressing complexity in an adequate way is essential for a successful project management. It is common to split the project development process into stages. Traditionally, those stages are requirements, analysis, design, implementation and verification stage. The stages and their logical order is illustrated in figure 1. In the following, the importance of measuring complexity for the single stages is pointed out.

Even before a project starts, determining the expected complexity enables a potential contractor to make a solid offer which neither overestimates nor underestimates costs and duration. At the beginning of a project, the most interesting aspects are the needed resources and a time schedule. These factors can only be specified appropriately if it is possible to assess complexity correctly. Furthermore, the calculation of project variables such as costs and duration has to scale

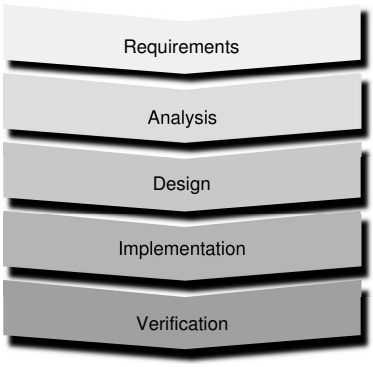

Fig. 1. Stages within the project development (based on [22])

with the measured project size. In other words, it is important to known how to map complexity to project variables. At best, a linear relationship between project variables and calculated complexity is given. For example, realizing one complexity unit takes one hour and implementing ten complexity units takes ten hours in time.

But not only for the planing a project a descent complexity measure is needed. One of the most important tasks within an ongoing project is to measure its progress. This allows for adjusting to changes and difficulties during the development. Measuring progress is commonly done by milestones [13]. Milestones are checkpoints along the way in project work, but it is not always easy to formulate milestones [2]. An often observed phenomenon is the ninety-ninety rule also known as ninety-ten rule by Tom Cargill of Bell Labs which is known to almost every software engineer: "The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time" [4]. This quote illustrates that in most cases the development progress is not a linear progress.

Even after a project is over, measuring complexity plays an important role. The end of a project does not necessarily mean that it has been completed successfully. It may also have been abandoned. Most interesting are lessons which can be learned from finished project for future projects. This includes comparison with other projects, how adequate the project planning was and how the model has adjusted to challenges which have emerged during project execution. One important question arises after each finished project: has the complexity been estimated correctly or does the model need adjustments. Not all deviations in project planning and assessment can be attributed to a faulty assessment of complexity. But it is important that the complexity model is able to adjust to changes during the project appropriately. At the end of a project, the predicted project figures can be compared with the actual figures such as costs, duration, progress and resources. Therefore a method for mapping complexity figures to

the relevant project management figures is necessary. Figure 2 illustrates on the left hand side the chart of different project complexities. These complexity figures are mapped to project figures such as duration, costs, progress and resources. In most cases the relation between these two figures is given by a database which provides the information on how many complexity units for which type of project represent how many time or cost units, for example.

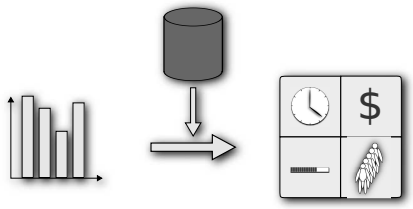

Fig. 2. Comparing different projects and calculating project key figures

Outside of the actual project management, estimating complexity also allows to review projects for other purposes. In safety-critical systems, for example, a complexity ranking of all components can ensure that more complex components are subjected to a detailed verification. Also, when troubleshooting, it helps if the more complex components can be identified. Even in the assessment of (new) technologies or alternatives a proper assessment of complexity brings a decisive advantage.

In summary, it is an important challenge to assess complexity adequately. Only this can ensure that project management models are reliable and projects can be completed successfully in terms of time and resource consumption. Measuring complexity is not only important in all stages of the project development but also to evaluate projects and components for their complexity.

1.2 Complexity estimation

This short section will show the most common approach to address complexity. As it is illustrated in figure 3, the determination of the complexity is basically a two step progress. On the left hand side of the figure, the system under investigation is illustrated. In the first step, the values of the measurement variables are determined. The measurement variables are the key variables, also called key figures of the complexity model. In a second step, the complexity of the system is either calculated or estimated, depending on the model. Therefore the values of the key variables are used. The result is a representation of the system's complexity. This is illustrated by the bar chart on the right side of figure 3.

In order to find an adequate measurement for complexity, the determination is often a cycle as illustrated in figure 4. The system is observed and variable

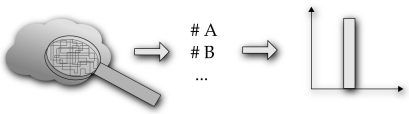

Fig. 3. Determining complexity with key variables

values are measured. Based on the calculation model, the complexity is determined. Mapped on real world variables such as time and costs, the calculated values are compared to the actual duration and costs of the project. If the values don't match, key variables as well as calculation methods need to be adjusted. After enough cycles for a decent number of projects, the calculated figures match the actual figures very well.

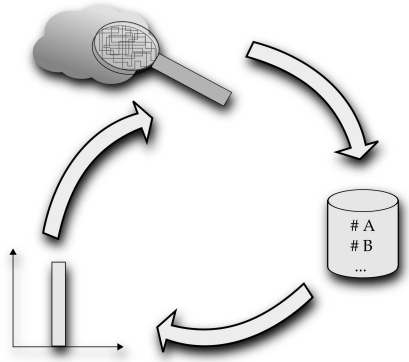

Fig. 4. Adjusting complexity measure

These adjustments make it possible to bring model and reality closer together. The model is always adjusted for one particular field. This approach seems adequate for projects in similar fields and with similar technologies. But it is also necessary to have enough data from previous projects to adjust the model. This makes it hard to adjust the model for fields of engineering with fast changes in technology. For example in software measurement, there is almost an uncountable number of adjustments for COCOMO (COnstructive COst MOdel) [7] and COCOMO 2.0 [6], for instance [18].

The following section will discuss the special demands for a complexity calculation model in computer science.

1.3 Computer Science

In classic engineering disciplines such as architecture or mechanical engineering, engineers can rely one some hundred years of experience. This experience allows a good estimation of complexity in these fields, for most part.

In "modern" fields such as software engineering, digital hardware design or embedded systems, engineers can only rely on a few decades of experience. Designing information technology related systems has become a complex challenge for engineers. Especially in embedded systems, new key demands led to more complex systems in recent years [15]. The success of a system is influenced by its complexity [19]. It has become an important challenge to find an adequate complexity measurement. The determination of complexity makes it possible to give system related statements and allows to compare different systems [11] [12]. Today, most methods for estimating system size use empirical data gained by analyzing previous projects [16]. In order to develop a complexity model, the complexity itself needs to be understood [8].

Figure 5 charts a study from the company Numetronics. About 60% of integrated circuit projects slip at least one quarter and about 16% slip more than one year. In order to provide a solid project management, the whole management progress from the requirement stage to the verification stage has to be reconsidered. Due to almost endless combination possibilities during development progresses an innovative approach is necessary to get complexity under control. The result is a completely new approach in the field of project management and complexity determination: the *design entropy concept*.

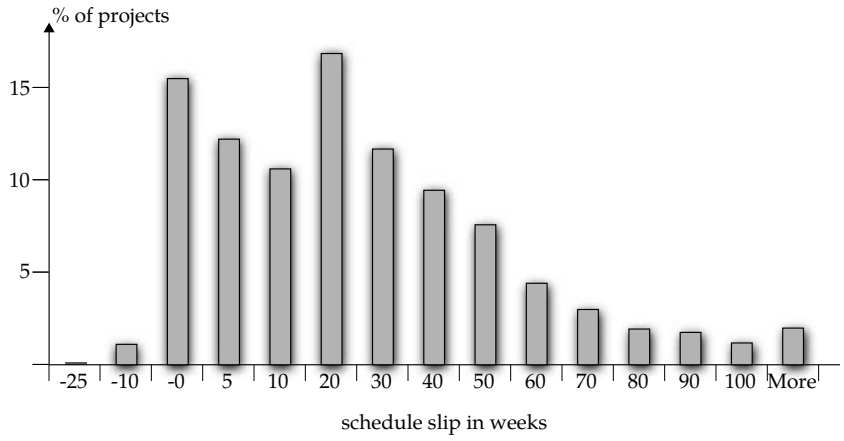

Fig. 5. Schedule Slip (based on [24])

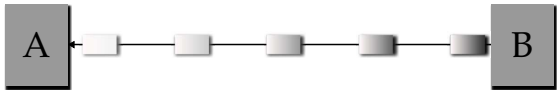

Fig. 6. Transmission of information between components

1.4 Design Entropy concept

The formulas for the design entropy concept can be derived from Shannon's information entropy. Signals between components can be seen as a transmission of information. Connections are the channel and information are symbols transmitted from a pool of available symbols. Figure 6 shows two components A and B. Connections allow to interchange information. In digital hardware design connections are normally implemented by wires. In software design, information interchange can happen though assignments, calls or statements, for instance. But it is still transmission of information between components. For instance an assignment between two variables: `a := b`: The information *(=value)* from component *(=variable)* `b` is transmitted *(=assigned)* to component *(=variable)* `a`.

In order to give mathematical statements about transmitted information, Claude Elwood Shannon developed a model which became famous as Shannon's information theory[23]. The form of his theorem (see (1)) is recognized as that of entropy as defined in certain formulations of statistical mechanics (e.g. [25]), where p_α is the probability of a system being in cell α of its phase space. H is from Boltzmann's famous H theorem and the constant K merely amounts to a choice of a unit of measure [23]. According to C. E. Shannon, J. W. Tukey suggested to call the information content $I = 1$ bit for devices with two stable positions.

$$H = -K \sum_{i=1}^{N} p_\alpha \log p_\alpha \tag{1}$$

(1) can be rewritten as (2) as shown in [20] and [21]. Where c is a component with $n(c)$ inputs (component sources) and $m(c)$ outputs (component drains). Each in- and output can presume $z(c)$ possible states. $H_B(c) \in \mathbb{R}_0^+$ is then the behavior entropy of component c.

$$H_B(c) = \log \left(\prod_{i=1}^{n(c)} z(n_i(c)) \cdot \prod_{j=1}^{m(c)} z(m_j(c)) \right) \tag{2}$$

According to the design entropy concept, complexity is calculated over inputs, outputs and states as illustrated in figure 7. Key variable is the amount of states a system can be in. The values for the variables can be determined according to rules. With these values, the complexity can be calculated. It is not necessary

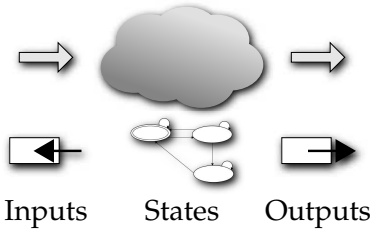

Fig. 7. Variables for calculating complexity

anymore to rely on empirical data. The complexity determination becomes a calculation instead of an estimation.

The following section will deal with states from different scientific fields in order to understand why states make it possible to describe complexity in an adequate way.

2 States

2.1 Defining states

Despite other meanings of the term "state", the Encyclopædia Britannica describes states as "any of various conditions characterized by definite quantities ... in which an atomic system may exist" [1]. In physics the term state is mostly related to quantum mechanics and thermodynamics. Within the theory of energy conservation an atom can only exist in one certain characteristic state associated with a discrete energy, called quantum state [14]. In thermodynamics, a system's properties can be characterized by state variables [26].

But the term state is used in computer science and electrical engineering. In the context of automata theory, an automaton contains a set of states. At each time step and in conjunction with reading a symbol, the automaton transits to the next state. The next state is determined by a transition conjunction. In finite state machines, states have the purpose of remembering the relevant portion of the system's history [17]. In information theory, each state corresponds to one or more of the possible patterns that the recent input sequence has passed through [5].

A decent definition for our purpose can be found in [9]: A state is a situation in which a system or system's component may be at a certain point of time. It refers to the interior of a system and ignores external influences such as input and output. The set of states is the abstraction of a real system.

2.2 States and complexity

Complexity can be considered to be a measure of a system's disorder which is a property of a system's state. Complexity varies with changes made at the

amount of possible states of a system. States are abstract variables which depend on the analyzed property of a project and do not directly depend on technologies, architectures and abstraction layers. These factors have an influence on the amount of possible states.

Adding the connections between components as a second variable allows to consider the increase in complexity with a rising number of components connected. For instance, a simple switch has two states: on and off. A second switch has also two states. When two switches are combined into one component, the components has four possible states which is the sum of number of states from both switches. But it is not linear with the number of switches. A third switch increases the number of states for this component to eight instead of six when adding up the single amount of states. As the complexity doesn't increase linearly, the amount of states does neither. But the complexity is not only depending on the amount of states a component can be in. The second influence is the amount of connections. Increasing the number of connections gives more possibilities to connect components. This increase is also not linear and neither is the increase in complexity.

Considering the number of states also allows to determine project progress in an adequate way. Having again the example with the three switches. The project is completed when all eight states are implemented. But the first two switches only realize four states. Therefore not two thirds of the project are accomplished but only fifty percent. This approach could solve the planing challenge for the ninety-ninety rule.

At the beginning of a project there is a high uncertainty. This uncertainty could also be described by states. Knowledge about the project reduces the number of possible states. The more of an upcoming project is specified and defined the less possible states are available. For example, if the switches have to be choosen. The tighter the criteria for the switches are, the less possibilities there are to choose a switch. If the contractor specifies which switch to use, there is only one possibility and therefore only one state.

The number of states has always to be defined for the property under investigation. For example in digital hardware design, complexity refers to the amount of states of a component and its states. If the layout is analyzed potential orientations and positions of elements are possible states.

In software engineering, source code could be analyzed. With the theory of abstract interpretation [3] [10], programs are analyzed for their behavior (semantics). Values for variables are examined in intervals (interval arithmetics). Each interval leading to the same result could be understood as one state. This would then also allow to measure complexity for software systems. Using for example SystemC to describe an embedded system enables one to analyze hardware/software systems as well.

The same model can be used for different application fields. Therefore, it is possible to compare different fields of application. Different technologies and

abstractions layers have an influence on the complexity over the states. Abstraction layers reduce the degrees of freedom and therefore the amount of possible states. Different technologies allow more or less degrees of freedom and influence the amount of possible states.

States are an abstract variable. Properties of systems can be measured with states. When analyzing the complexity of a system its properties are analyzed. This can be expressed by states. Using states as a key variable allows the calculation of complexity.

3 Conclusion

Modern fields of engineering are subjected to fast changes and developments in technology. Complexity measure models can hardly adjust to these fast changes. But a decent complexity measure is fundamental for the success of a project. Abstract measurement variables for complexity would allow to apply one model on different fields of application. Different projects could be directly compared. Furthermore, an abstract model would be independent of technologies and abstractions layers.

In this paper, it was discussed whether states as an abstract variable for measuring complexity would have the possibility to address complexity in an adequate way. Complexity can be considered to be a measure of a system's disorder which is a property of a system's state. Complexity varies with changes made at the amount of possible states of a system. The design entropy concept uses states to measure project size. States are abstract variables which depend on the analyzed property of a project. Because of the abstraction, the concept can be applied on different engineering fields, and states can be used as measurement variables.

References

1. Encyclopaedia Britannica 2003: ultimate reference suite. Encyclopaedia Britannica Inc. (2003)
2. Andersen, E., Grude, K., Haug, T.: Goal Directed Project Management: Effective Techniques and Strategies. Kogan Page Series, Kogan Page (2009)
3. Bagnara, R., Hill, P.M., Pescetti, A., Zaffanella, E.: On the design of generic static analyzers for imperative languages. Quaderno 485, Dipartimento di Matematica, Università di Parma, Italy (2008)
4. Bentley, J.: Programmimg pearls. Commun. ACM 28, 896–901 (September 1985)
5. Blahut, R.: Principles and practice of information theory. Addison-Wesley series in electrical and computer engineering, Addison-Wesley (1987)
6. Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., Selby, R.: Cost models for future software life cycle processes: Cocomo 2.0. Annals of Software Engineering 1(1), 57–94 (December 1995)

7. Boehm, B.W.: Software Engineering Economics. Prentice Hall PTR, Upper Saddle River, NJ, USA (1981)
8. Calvano, C.N., John, P.: Systems engineering in an age of complexity: Regular paper. Syst. Eng. 7, 25–34 (March 2004)
9. Claus, V., Schwill, A.: Duden Informatik A - Z. BI-Wissenschaftsverlag, 4 edn. (2006)
10. Cousot, P.: Abstract interpretation based formal methods and future challenges, invited paper. In: Wilhelm, R. (ed.) Informatics - 10 Years Back, 10 Years Ahead, Lecture Notes in Computer Science, vol. 2000, pp. 138–156. Springer-Verlag (2001)
11. DeMarco, T.: Controlling Software Projects: Management, Measurement, and Estimates. Prentice Hall PTR, Upper Saddle River, NJ, USA (1986)
12. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, Revised. Course Technology (February 1998)
13. Frame, J.: The new project management: tools for an age of rapid change, corporate reengineering, and other business realities. The Jossey-Bass business & management series, Jossey-Bass Publishers (1994)
14. Halliday, D., Resnick, R., Walker, J.: Fundamentals of physics. No. v. 2 in Fundamentals of Physics, Wiley (1993)
15. Henzinger, T., Sifakis, J.: The embedded systems design challenge. In: Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science. Springer (August 2006)
16. Hinrichs, N., Leppelt, P., Barke, E.: Building up a performance measurement system to determine productivity metrics of semiconductor design projects. In: IEEE (ed.) IEEE International Engineering Management Conference (IEMC), Austin TexasErmolayev2007. pp. CD–ROM Proceedings. IEEE (2007)
17. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts (1979)
18. Lum, K., Baker, D., Hihn, J.: The effects of data mining techniques on software cost estimation. In: Engineering Management Conference, 2008. IEMC Europe 2008. IEEE International. pp. 1 –5 (june 2008)
19. McCabe, T.J., Butler, C.W.: Design complexity measurement and testing. Commun. ACM 32, 1415–1425 (December 1989)
20. Menhorn, B., Slomka, F.: Entwurfsentropie: Ein Maß im Schaltungsentwurf. In: 7th GI/GMM/ITG-Workshop, Multi-Nature-Systems (2009)
21. Menhorn, B., Slomka, F.: Design entropy concept. In: International Conference on Hardware/Software Codesign and System Synthesis 2011 (2011), accepted
22. Royce, W.W.: Managing the development of large software systems: concepts and techniques. In: Proc. IEEE WESTCON. IEEE Press (August 1970), reprinted in Proc. Int'l Conf. Software Engineering (ICSE) 1989, ACM Press, pp. 328-338
23. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
24. Silbey, A., Johnston, S.: Delivering During the Downturn: Engineering Management Strategies at Top IC Companies. Numetronics (2009)
25. Tolman, R.C.: The principles of statistical mechanics. Oxford Univ. Pr., London (1938)
26. Vliet, C.M.V.: Equilibrium and Non-Equilibrium Statistical Mechanics. World Scientific Publishing Co. Pte. Ltd., Singapore (2008)