Better Software Effort Estimation—A Matter of Skill or Environment?

Jo E. Hannay

Abstract— Estimating the effort of software development is fraught with difficulties, and it is clear that effort should be invested in improving the accuracy and the reliability (consistency) of effort estimates, as well as the assessment of estimate uncertainty. However, it is less clear where to target such improvement efforts. We discuss the degree to which it is feasible to improve the expertise of the person(s) who estimate(s), and the environment in which the estimation is performed. The former hinges on what there is to say about the development of estimation expertise and the task characteristics of effort estimation. The latter hinges on what contextual support may be developed in terms of environment control and tools and methodology. We integrate several theories to make a framework for discussing software effort estimation and planning. On the basis of that discussion, we conclude that present guidelines almost exclusively concern the environment and its influence on broad psychological factors, that task-specific estimation expertise is too weak a signal in the noise of biases, and that strengthening this expertise requires new efforts in understanding the task-specific elements in software effort estimation and planning, as well as environmental measures (tools and methodologies) that support expert behavior and expert learning.

Index Terms— Software Effort Estimation and Planning, Lens Model, Environment, Expertise, Task Complexity, Deliberate Practice

1 Introduction

In an interview with OMNI Magazine in 1994, Herbert Simon gave a, by now, famous parable for describing complex behavior: "When you watch an ant follow a tortuous path across a beach, you might say, 'How complicated!' Well, the ant is just trying to go home, and it's got to climb over little sand dunes and around twigs. Its path is generally pointed toward its goal, and its maneuvers are simple, local responses to its environment. To simulate an ant, you don't have to simulate that wiggly path, just the way it responds to obstacles", see also [136, ch. 3]. In addition to illustrate that one need not implement complex rules in order to achieve complex behavior, the ant's tortuous journey may also be used to illustrate a converse principle: that one might decrease complex behavior by decreasing the complexity of the environment. Make the beach flat and featureless and perhaps the ant will end up on a less tortuous, less complicated path.

The development of a software system is an inherently complex process. Estimating the effort needed to run a large software development project is doubly so and notoriously difficult. The human judgment processes involved in forecasting an estimate are subject to a range of unconscious processes [97], [93], [99] that are subject to complicating environmental factors: They are sensitive not only to what, but also to the nature and format of the information (e.g., requirement documents) that are available when producing the estimate [94], [95], [96], [91].

Visible effects are that effort estimates of software development are inaccurate and generally too low [112], that software professionals tend to exhibit more confidence in their estimates than is warranted [101], and that estimates are unreliable in that the same person may judge the same project differently on separate occasions [60].

With regards to responding to this environment, there seems to be no substantial improvement (learning from history) in estimation accuracy over the past decades [112], and learning from outcome feedback seems difficult [61]. It seems that it is not enough simply to do more estimation of software development effort to become good at it. Evidence suggests that feedback (either passive in the form of historical data or active in the form of immediate feedback) in terms of performance measures alone does not improve on over-optimism, over-confidence, or unreliability.

In attempts to improve effort estimation, one can take one of two extreme stances. At the one extreme, one would study the expertise within a person, exclusively seeking ways to alter that person's mental models so that he would perform better. At the other extreme, one would ignore personal expertise and concentrate instead on altering the environment so as to induce better performance or, perhaps more to the point, hinder adverse performance in a person. Of course, a combination of the two is probably better than either one alone: It is not enough to design straighter roads and cars with more safety features, one also has to improve driving skills. The question is where to invest most effort.

Underlying the ant parable is the idea that behavior exhibited by an agent—be it complex, tortuous, erroneous or simple—is an adaptation or reaction to an environment. It is therefore not meaningful to study the agent's behavior without considering the environment to which the agent is adapting itself [57]. Adaptation is about acquiring relevant and pertinent expertise.[1] In this paper, we explore the boundaries of expertise and its environment with regards to software development effort estimation. Our goal is to comment on how far and where it is possible to go in both developing estimation skill and in tailoring the environment so as to improve estimation performance. There seems to be a limit as to how much estimation skill one can induce, but when it comes to software development, it is certainly not possible to make the beach featureless either.

So in order to improve performance on a task—in our case, the task of estimating effort—one needs to develop expertise and one also has to consider how to form the environment so as to facilitate good task performance.

SIMULA Research Laboratory, Department of Software Engineering, Pb. 134, NO-1325 Lysaker, Norway. johannay@simula.no

[1]Before objecting that ants do not exhibit expertise or learning and only "mindlessly" respond to bumps and dips in the environment, it is worth considering that ants have exhibited tutor-based learning, constructive adaptation to failure, and exhibit speed/accuarcy tradeoffs in decision making [50], [124], [49]. The ant's well-adapted local response to obstacles is precisely its expertise.

The extent to which one should focus on expertise or environment, or both, is the topic of this paper. At the core of this discussion, however, is the concept of *task* and its structure and content. If we were to discuss the task of riding a bicycle, we would not need to analyze the task in any depth. It suffices to go out and practice, unless you want to become a bicycle athlete. When it comes to software effort estimation, matters seem more complicated.

Section 2 presents a framework for discussing software effort estimation that integrates the estimator with the environment. Then, Section 3 and Section 4 relate concepts of expertise and task complexity to this framework. Section 5 discusses how to improve estimation performance, and Section 6 exemplifies the preceding discussion in the context of a large agile development project. Section 7 draws implications and concludes.

2 A Framework for Studying Estimation

In order to reason about complex phenomena it is useful to use or build suitable concepts and to organize them in an appropriate conceptual model of sufficient abstraction and simplicity to facilitate reasoning.[2] For our purposes, we need a conceptual model for the task of estimation that expresses the distinction between the environment and an agent operating as a task-doer in that environment. Stewart and Lusk present such a model for forecasting in general [145], [144] which is theoretically and empirically founded. We will in the following discuss this model in the specific context of software effort estimation.

2.1 Skill Score

We start, as do Stewart and Lusk, by considering the components of a *skill score* due to Murphy [115]. The *observed event* O (e.g., the actual effort of a software development activity) is what the *forecast* Y estimates; prior to the observation being available. The skill score of the forecaster (estimator) relates Y to O (over a series of occasions) and consists of three components:

$$SS = (r_{OY})^2 - [r_{OY} - (s_Y/s_O)]^2 - [(\bar{Y} - \bar{O})/s_O]^2 \qquad (1)$$

Here, r_{OY} is the correlation between the forecast and the observed event, s_Y and s_O are the standard deviations of the the forecast and the observed event, respectively, and $\bar{Y}$ and $\bar{O}$ are the means of the forecast and observed event. The first component $(r_{OY})^2$ may be viewed as a measure of the potential skill of the forecaster in the absence of bias. The second term $[r_{OY} - (s_Y/s_O)]^2$ expresses so-called *conditional bias*; i.e., forecaster bias in uncertainty assessments (e.g., over-confidence) and the third term $[(\bar{Y} - \bar{O})/s_O]^2$ expresses so-called *unconditional bias*; i.e., forecaster bias in point (middle value) forecasts (e.g., under-estimation).

Technically, there are several ways of instantiating Equation 1, and, moreover, the concepts in the equation should be generalized to nonlinear equivalents. However, these technicalities are not the topic of the present discussion. We are here primarily interested in the conceptual decomposition of Equation 1.

We will now elaborate on the three components of Equation 1 with regards to software effort estimation.

[2]This is the essence and purpose of theory building [70], [59].

2.2 Unbiased Skill—Simple Model

Stewart [143] decomposed the first component of Equation 1 further using a Brunswikian [66] lens model; see Fig. 1. The lens model includes cues X $(X_1, \ldots X_n)$ which are used by the forecaster in producing the forecast Y and as input to models for predicting the outcome O. For example, the cues X could be quantitative project characteristics recorded by a software development project, such as function points, lines of code, average developer skill, number of teams, etc. Associated with the lens model are the following equations:

$$O = M_{OX}(X_1, X_2, \ldots, X_n) + e_{OX} \qquad (2)$$

$$Y = M_{YX}(X_1, X_2, \ldots, X_n) + e_{YX} \qquad (3)$$

Here, M_{OX} and M_{YX} are probabilistic models that describe the relations between the cues X and the observed event O and between the cues and the forecast Y, respectively.

One can then calculate reliability measures: The correlation R_{OX} between actual observed O and the corresponding values of M_{OX} expresses the fit of the model; i.e., the predictive strength of M_{OX}, given the cues X. If M_{OX} captures all systematic variance in the relationship between the cues and the event, then R_{OX} is the maximum predictability of the observed event O given cues X. Likewise, the correlation R_{YX} between forecasts Y and the corresponding values of M_{YX} expresses the predictive strength of M_{YX}, given the cues X [145], [144]. The curved lines in Fig. 1 represent correlations between the cues, since some of them are likely to be interdependent. Also, the observed event and the forecast should be correlated, which is represented by the curved line between O and Y. The error terms e_{OX} and e_{YX} represent the residuals of the models; that is, the amount of variability not explained by the models.

The lens model in Fig. 1 incorporates a model for the environment (M_{OX}) and a model for the agent performing the forecast (M_{YX}), and therefore provides a view of the world that suits our focus in this paper. The correlation between the outputs of these two models, denoted G, is then a measure of how well the model of the forecaster and the model of the environment match each other. The overall predictivity of the models; i.e., the correlation $r_{Y,O}$

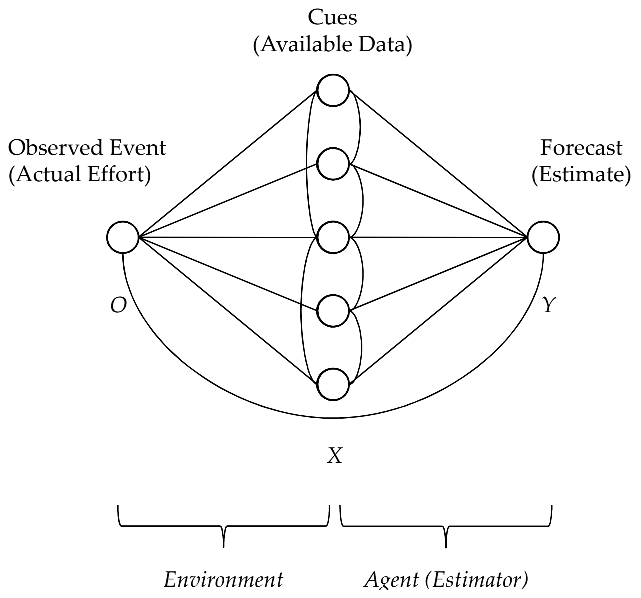

Fig. 1. Lens Model (adapted from [145]).

from Equation 1 between the forecast Y and the outcome O, can be expressed as:

$$r_{Y,O} = R_{OX} G R_{YX} + C\sqrt{1 - R_{OX}^2}\sqrt{1 - R_{YX}^2} \qquad (4)$$

where C is the correlation between the error terms e_{OX} and e_{YX}. The more systematic variance M_{OX} and M_{YX} capture and the better they are correlated, the smaller C will be. Since we are interested in the conceptual properties, we will concentrate on the primary term $R_{OX} G R_{YX}$ of Equation 4.

2.3 Descriptive and Explanatory Models

The nature of the probabilistic models M_{OX} and M_{YX} is worth some consideration. In the original formulation, these models are best thought of as regression models that describe available empirical data: M_{OX} describes data consisting of observed cases of cue values and observed events (actual project effort). Such a model can be generated from historical project data. The model defines a hypercurve in $X_1 \times \cdots \times X_n \times O$-space that best fits the data under the chosen model type; e.g., a hyperplane under linear assumptions. Similarly, M_{YX} describes data consisting of observed cases of cue values and forecasts (effort estimates) and defines a hypercurve in $X_1 \times \cdots \times X_n \times Y$-space.

Models that describe empirical data are useful for demonstrating phenomena. A first step beyond mere data description is to move from exploratory model fitting to confirmatory model checking in which the regression models are validated on several data sets. However, regression models are extensional, or "black box", in that they do not describe underlying mechanisms. They therefore do not provide the means to manipulate phenomena or to improve on events. For that, one needs to move toward intentional "white box" explanatory models that describe cause-effect relationships underlying phenomena [127], [59], [70]. Improvement is then achieved by manipulating appropriate causes. We will therefore extend the discussion in [145], [144] to include not only models that describe empirical data (quantitatively), but also explanatory models that represent state-of-knowledge (perhaps qualitatively).

Consider again the example cues X_i above for a software project: function points, lines of code, average developer skill, etc. If the observation to be forecast is project effort, then an explanatory model M_{OX} should reflect the state-of-knowledge of how these cues influence project effort, while an explanatory model M_{YX} should reflect the state-of-knowledge of how these cues influence the forecast of a human estimator. That such explanatory models also quantitatively capture all systematic variance is an ideal (an optimal model) to strive for, but is far from achieved in most disciplines. Only disciplines with extremely strong theoretical models (such as in some fields of physics) come close to such optimal models. In any case, explanatory models are by necessity simplified abstractions—here of the environment and forecaster—that allow us to comprehend and reason about those features of the environment and forecaster that are important to us. What is important is that the theoretical models give us means to assess and improve the practice of estimation and planning.

Our discussion in this paper concerns the possibility of increasing G; the correspondence between the environment model and the forecaster model. What this means precisely depends on the nature of these models. If the models are data-fitted models (i.e., extensional, black box and quantitative), it is natural to regard the environment model as the standard to which one measures the performance of the forecaster model. The input-output relationship of cues to event is all one has, and the relationship has presumably been modeled to the best of ones knowledge. For example, the data-derived regression model M_{OX} might describe the best fit (within the type of model) to the given cues X in relationship to O. Steps should then be taken to manipulate the forecaster to the best of one's knowledge and then check if new data generates a model M_{YX} of the forecaster that better corresponds to M_{OX}.

However, if the models are intentional, it is not clear that the environment model should be the standard. In some physical sciences, where one has near optimal models of the environment, an obvious way to increase G is to increase the forecaster's knowledge of, and practice with respect to the environment in general and M_{OX} in particular. But in disciplines such as economics, management, and the behavioral sciences (upon which much of the theoretical basis used in software engineering is built [70]) theoretical models are much less accurate and many are qualitative rather than quantitative. In such circumstances, epistemological issues come into play more prominently: There is no longer a single consensus model. Rather, different viewpoints serve different purposes, and the question becomes one of how to choose models so as best to comprehend and manipulate a complex world for the purpose at hand. In such a setting, optimizing G might mean adjusting M_{OX} and M_{YX} to each other, rather than simply using M_{OX} as a gold standard for M_{YX}. For example, comprehensive environment models may be too complex for a human forecaster to relate to, since humans are not good at processing masses of information analytically [57], [72], [73], [54]. In this sense, it is a misnomer to call the first component of Equation 1 "skill" since this only refers to the qualities of the forecaster and not to qualities of the environment model. We shall say more on this in Section 3.

2.4 Unbiased Skill—Extended Model

Several authors have extended the model in Fig. 1; see e.g., [42]. Stewart and Lusk [145], [144] extended the model by taking into account *true descriptors* T and *subjective cues* U; see Fig. 2 (correlations omitted). True descriptors are facts about the environment, free from measurement error, that the cues X are intended to capture. For example, true function points may be only approximated in a project database based on historical data, and the true skill level of developers may be imperfectly measured conditional on the assessment instrument administered. The lens equation for predicting O from T is

$$O = M_{OT}(T_1, T_2, \ldots, T_n) + e_{OT} \qquad (5)$$

where M_{OT} is a probabilistic model describing the relation between the true descriptors and the observed event. The correlation R_{OT} between O and M_{OT} expresses the predictability of the environment given the true descriptors. The ratio $V_{TX} = R_{OX}/R_{OT}$ is then, in the terminology of [145], the *fidelity in the information system*; values less than 1 being the effect of measurement error. Efforts to improve data quality in business intelligence are precisely efforts to improve the fidelity in the information system. On the other hand, the choice of what data to gather pertains to

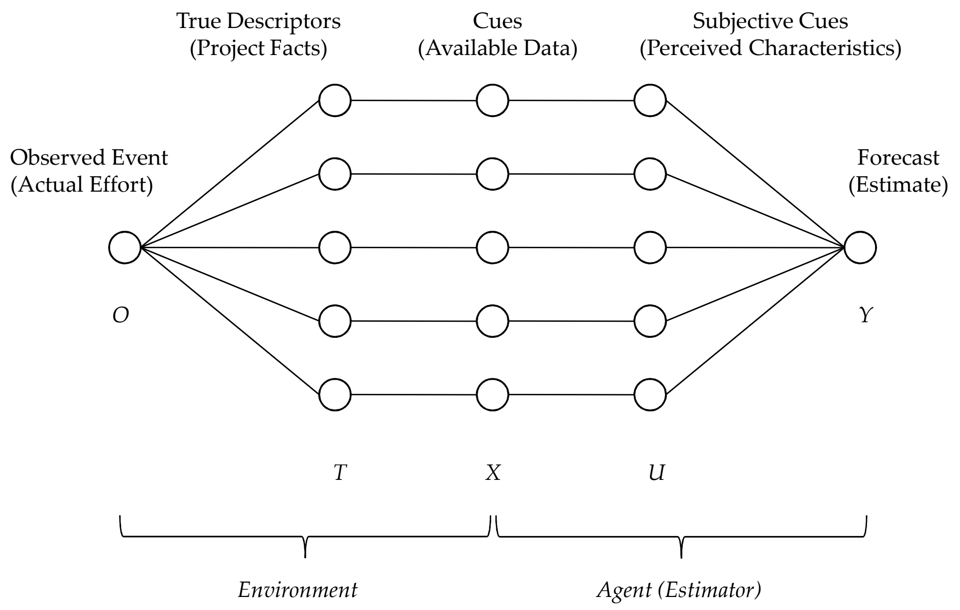

Fig. 2. Extended Lens model (adapted from [145]).

which true descriptors one thinks are appropriate to gather information (cues) about.

For extensional models (e.g., regression models) that describe data, M_{OT} and M_{OX} will likely be distinct models, unless X and T are very similar. For explanatory models, M_{OT} and M_{OX} will likely be the same model, since measurement error is not part of a conceptual model.[3]

Subjective cues U are the forecaster's interpretation or redefinition of the cues X. The cues might be interpreted in various ways according to the forecaster's mental model. Cues might also be replaced by vicarious cues, when so-called accessibility effects are present [103]. Accessibility effects occur when a problem is approached by (subconsciously) substituting the problem with a perceived easier problem (as when heuristics such as the availability, representability, and affect heuristics step into action) or when the cues for the original problem are given a reference point (anchoring). The lens equation for predicting Y from U is

$$Y = M_{YU}(U_1, U_2, \ldots, U_n) + e_{YU} \tag{6}$$

where M_{YU} is a probabilistic model describing the relation between the subjective cues and the forecast. The correlation R_{YU} between Y and M_{YU} expresses the reliability of the forecast given the subjective cues; in other words, the reliability of the information processing of the forecaster. The ratio $V_{UX} = R_{YX}/R_{YU}$ represents, in the terminology of [145], *the reliability of information acquisition*. It is assumed that R_{YX} is less than R_{YU}, since the subjective cues are what are actually incorporated into the forecast.

Again, in the setting of extensional models (for example regression models) that describe data, M_{YU} and M_{YX} will likely be distinct models, unless X and U are very similar. For models that set out to represent knowledge, M_{YU} and M_{YX} will likely be the same model.

Equation (4) can now be decomposed into further components (using $R_{OX} = R_{OT}V_{TX}$ and $R_{YX} = R_{YU}V_{UX}$):

$$r_{YO} = R_{OT}V_{TX}GV_{UX}R_{YU} \tag{7}$$

[3]This is only true for probabilistic models. In contrast, in so-called fixed-effects models, measurement error is confounded with conceptual variance.

2.5 Areas for Improving Performance

Equation 1 can now be written as

$$SS = R_{OT}V_{TX}GV_{UX}R_{YU} - [r_{OY} - (s_Y/s_O)]^2 - [(\bar{Y} - \bar{O})/s_O]^2 \tag{8}$$

Each component of Equation (8) represents an area for improving forecasting performance. We list the components together with their interpretations:

1) R_{OT}: Environmental predictability
2) V_{TX}: Fidelity in the information system
3) G: Match between environment and forecaster
4) V_{UX}: Reliability of information acquisition
5) R_{YU}: Reliability of information processing
6) $[r_{OY} - (s_Y/s_O)]^2$: conditional bias
7) $[(\bar{Y} - \bar{O})/s_O]^2$: unconditional bias

Areas 1 and 2 pertain to the environment, Areas 4 and 5 pertain to the forecaster, and Area 3 pertains to the match between environment and forecaster. Areas 6 and 7 originally pertain to the forecaster, but in software engineering, they pertain to both forecaster and environment because the forecaster is part of the environment and because methods to avoid forecaster bias are measures taken in the environment. In the following, we will discuss these areas with regards to software effort estimation. We will treat the environment first, then the forecaster, and then the match between the two. Areas 1, 3, and 5 concern the environment model and the forecaster model directly, and for these areas, we shall discuss both descriptive and explanatory models (recall Section 2.3), prior to discussing ways to improving estimation performance. For the other areas, we will discuss briefly how they relate to the models and then discuss ways to improve estimation performance.

2.5.1 Environmental predictability

Area 1 concerns the environment model M_{OT}.

Descriptive: If a descriptive model M_{OT} captures all relationships (all systematic variance) between true descriptors T and observed event O, R_{OT} will provide a quantitative upper bound of the environmental predictability, given the true descriptors [145]. The effects of improving forecasting performance across all areas of the skill score model are limited by this upper bound. Achieving a good descriptive model for software engineering field phenomena is generally hard, and building a good explanatory model is even harder. In fact, the difficulty of constructing such models are one of the reasons why human forecasting is necessary. Note that R_{OT} is relevant even for non-optimal models; in this case, it reflects the environmental predictability given the state of knowledge embedded in M_{OT}.

Explanatory: With regards to building explanatory models, the issue in judgment tasks such as software effort estimation is to identify better descriptors T and to understand how they interact to produce the event O. A lot of research has gone into identifying salient cost drivers of software development projects. Formal estimation models such as the COCOMO family, SLIM, or ANGEL are, in fact, attempts at making quantitative models M_{OT}. However, the performance of formal estimation models is debatable [89], and as of yet, no model comes close to being an optimal model in the sense that it describes and explains environmental predictability fully, and at the same time provides means to improve environmental predictability.

Improving environmental predictability: It may seem obvious that more effort should go into constructing better quantitative models M_{OT} for software effort estimation. However, this too is under debate; e.g., [92]. Arguments against are that decades of effort has produced sub-optimal quantitative models that perform no better than human forecasters, and that software development projects are so diverse that it is only possible to produce good estimates if one knows when to use project-specific cues; so-called "broken-leg" cues (highly irregular, but available, information) [111]. This somewhat contradicts arguments in other judgment disciplines where it is found that formal models are more reliable and do outperform human forecasters, and where broken-leg cues are viewed as disturbances that offset reliability [43], [7].

Nevertheless, increasing understanding about which true descriptors that should be included in models will theoretically increase environmental predictability R_{OT}. It is therefore relevant to continue work on understanding process drivers. Also, some studies have shown that a decrease in environmental predictability (when this is determinable) is associated with a decrease in judgmental consistency [26], [27], [32], [33]; see [145], [144] for an overview. For software effort estimation, the vast variability in project variables and an unclear understanding of the effects of this variability (unclear M_{OT}) and of what variables are relevant (unclear T) have been proposed as one of several reasons for why software effort estimation is hard in general [89], [90]. The bottom line is that "judges respond to unpredictable tasks by behaving less predictably themselves" [144, p. 88].

2.5.2 Fidelity in the Information System

Area 2 concerns the discrepancy between true descriptors T and available cues X.

Relationship to models: True descriptors are hard to access in general and sometimes impossible to access in software projects in particular. Therefore, inputs to models of the environment are often approximations X to the true descriptors. For example, inputs to formal estimation models such as those mentioned above are usually available cues X, prone to measurement error, or subjective approximations, or themselves estimates by judgment-based methods. The latter point has led Jørgensen to suggest that formal estimation is judgment-based estimation in disguise [89], [92], since the models rely on inputs (cues) from the user that are non-trivial to supply.

There is therefore usually a discrepancy between the true descriptors T and the available cues X. The fidelity in the information system is a measure of this discrepancy; i.e., the data quality brought both to the environment model and to the forecaster (model). It is reasonable to expect that if the available cues do not accurately reflect the true descriptors, then both output from environment models and forecasts will deteriorate. As mentioned, the environment models for software estimation are vulnerable to data quality. For forecasting, though, certain studies, e.g., [8] indicate that measurement error in cues may be less of a problem than anticipated. The noise from such error could be of less consequence than other noise in the system; see [145], [144] for further pointers. Certainly, in software engineering, an imminent problem is to gather the appropriate business intelligence data in the first place. Also, the structure of the models; i.e. the estimation methods used, may be a larger problem than the quality of the input cues.

Improving fidelity in the information system: For software effort estimation, an aspect of fidelity in the information system has been uncovered in studies on the effects of irrelevant information in requirements documents [95], [94]. For example, in attempts to provide a comprehensive account of the system, a customer might include information that goes beyond the requirements for the first product release. Such information may concern accounts of the complexity of possible user patterns and about possible future extensions of the system. The information may also be presented in a verbose manner; perhaps illuminating the requirements from various point of view. These and other factors have been shown to inflate estimates. While *what* information that is irrelevant is a question of appropriate T and environmental predictability, the act of removing such information pertains to data quality. Environmental measures to reduce such effects are therefore steps to increase the fidelity in the information system. Further, the format of, and the manner in which information is presented, has also been shown to have effects on estimates [65], [96]. Time estimates of how much time to complete a given task are, on average, lower than time estimates of how much work can be done in a given time slot. Deciding the appropriate format in which to present information is a data quality issue and pertains to the fidelity in the information system. All these effects sort under biases (components 6 and 7 in Equation 8). Therefore, steps taken to increase the fidelity in the information system may lessen the effect of bias-inducing factors. The following are some of the current pieces of advice for software effort estimation pertaining to increasing the fidelity in the information system.

- Reduce irrelevant information [88], [60]
- Use standardized formats
- Reduce or control factors that trigger availability biases
- Identify the "swamping forces" represented by the most important cues [57]
- Use group estimates [88]
- Use tools that offer cognitive decision support [15]

Since true descriptors are hard to access, environmental predictability R_{OT}, and therefore V_{TX}, is in general not possible to compute; although it is possible to estimate its value using probabilistic methods. Nevertheless, these reliability measures provide a conceptual framework that gives an opportunity to structure our knowledge of estimation.

2.5.3 Reliability of Information Processing

Area 5 concerns the model of the forecaster M_{YU}. Note that reliability here concerns consistency of estimates, not necessarily that estimates are accurate.

Descriptive: To develop a quantitative descriptive model, one might start exploratory by modeling empirical data on how subjective cues relate to human estimates, and then confirmatory by validating the model on other data sets. Then, R_{YU} is a measure of the distance between actual estimates and the model's predicted estimates. If the model captures all relationships (all systematic variance) between subjective cues U and forecast Y, then R_{YU} is a quantitative measure of the reliability of the forecast, given the subjective cues [145]. It is non-trivial to develop such a model to perfection. Nevertheless, preliminary studies have shown that software estimation reliability is low: Estimators are inconsistent in their estimates both between and within subjects [60]. A recent study also suggests that prioritizing user stories for release planning suffers from a similar lack of consistency and even that the technique used for setting priorities affects the priorities between user stories [15].

Explanatory: Explanatory models of how human forecasters produce software effort estimates are mostly qualitative. For example, models of analogical thinking in general [77], [52] and with respect to estimation [140], [23], [100] in particular, set out to model some of the (cognitive) steps involved, and other models describe top-down versus bottom-up strategies of estimation [98]. We also know that estimators are influenced by environmental factors such as irrelevant information, information format, and availability biases. However, the process of producing the numerical estimate is not well-understood. This unknown quantification process has been coined "the magic step" [87]. This magic step is not atomic, since it can be decomposed into sub-steps; e.g., the similarity/dissimilarity testing set forth in [146], [116], but these sub-steps are also qualitative, producing several "magic sub-steps".

Improving reliability of information processing: A good explanatory model of the forecaster may help to improve the reliability of human software effort estimation. Such a model should embody what it is to be an expert software effort estimator and might be a conceptual model of the mental models of expert estimators [118] (Section 3.1). This would be an explication of tacit knowledge into a common task strategy (Section 4.3) and is theorized to improve performance as long as practice undergoes reflection in the practitioner [3], [4], [80] (Section 5.3). Such an improvement should then manifest itself in improved readings from the quantitative descriptive model in terms of better reliability.

Stewart and Lusk state that unreliability in processing subjective information is pervasive in human judgment, and that certain studies show that this unreliability increases as the predictability of the environment decreases. Further, unreliability may increase as the amount of information available to the forecaster increases [145], since humans are not optimized to deal analytically with large quantities of information [57]. An explanatory model of the forecaster should therefore reflect this.Tool support should be developed on the basis of such a model and might aid the estimator in simplifying and reducing information, rather than prompting the estimator to gather as much information as possible.

2.5.4 Reliability of Information Acquisition

Area 4 concerns the extent to which the forecaster can reliably interpret the objective cues X [145].

Relationship to models: In software engineering, the meaning of cues is often not clear in the first place, and reliability is bound to be low for this reason, since forecasters are likely to interpret cues differently from one occasion to the next. Stewart and Lusk [145] cite studies that indicate that unreliability in judging cues makes learning more difficult and has an effect similar to that of unpredictability in the environment. Building consistent mental models is thence hard for the practitioner, and deriving stable conceptual models of these mental models is therefore also hard.

Improving reliability of information acquisition: Software effort estimation is now often done in group sessions, and work studies indicate that elaborate processes come into play in which a common understanding of cues is achieved gradually through "boundary concepts" [23]. Cue interpretation therefore also relies on input from other forecasters; however, it is likely that reliability of cue interpretation increases through group work.

Stewart and Lusk [145] argue that unreliability of information acquisition limits forecasting skill more than lack of fidelity in the information system and that the former is also less costly to correct. However, it is not clear whether the unreliability observed in software estimators is due to unreliability of information acquisition or lack of fidelity in the information system.

2.5.5 Match between Environment and Forecaster

Area 3 concerns the correspondence G between the environment model M_{OX} and the forecaster model M_{YX}. In the words of [145], it is an estimate of the potential skill that the forecaster's current strategy could achieve if the environment were perfectly predictable (given the cues) and the forecasts were unbiased and perfectly reliable. (Recall that bias and unpredictability are factored out in the other components of Equation 8.)

Descriptive: In a purely descriptive setting, and if the environment and forecaster models adequately describe the phenomena, the match between the models reflects how well the forecaster predicts relative to the environment. A good match means that reliability and accuracy levels correspond between the two models.

Explanatory: In the explanatory mode, one can have an explicit goal to match the environment and forecaster models better. This might involve finding cues that more easily allow for matching the models, and by transferring knowledge from one to the other. For example, models of human analogical reasoning in estimation may provide a guide to environment models [108] which use project data to calculate nearest analogies. Conversely, environment models may give valuable guidance on cost drivers, etc. to models of human analogical reasoning. Notice how this implies deliberately altering the model of the forecaster. This is one of the essentials in learning and building skill. In software engineering, it is less relevant to view an environment model as the standard, since too little is known about the environment (Section 2.3).

Improving the match between environment and forecaster: In the lens model, the environment model and the forecaster model share the same cues. This assures the so-called Brunswick symmetry [151], [152], in which the comparison of the forecaster and his performance in the environment is conducted on the same level and area of generality. According to Wittman, studies routinely fail to observe this symmetry and may therefore ignore true relationships. In an estimation setting, asymmetry will occur analytically if the forecaster model is run on a certain type of project data and is evaluated with respect to an environment model which uses a different type of project data. Asymmetry will occur in practice if an estimator is given certain cues and is then evaluated on project outcomes based on different cues. Moreover, if an estimator is simply evaluated on project outcome, this says something about the adequacy of the cues given to the estimator, rather than the estimator's performance, unless one has an explanatory environment model that links the cues to the outcome. Since the latter is for the most part missing, one is constantly running the risk of asymmetry and, therefore, missing true relationships between estimator and environment. This may contribute to erroneous conclusions about the failure of software effort estimation (both formal and human).

An instance of this problem is the following: One knows that various ways of manipulating the environment affect estimates through largely unconscious mental processes. However, we have very little normative knowledge about whether these manipulations induce estimates that are better or worse. Consider again the case of irrelevant information in requirements documents. One knows that adding irrelevant information or simply inflating format will increase estimates. However, one does not know the extent to which the resulting estimates become less accurate. In fact, since estimates in general tend to be too low [112], it would actually seem sensible to take steps to inflate estimates. However, to introduce such a practice would introduce an asymmetry in Wittmann's sense: the set of cues used by the forecaster are different (inflated) from those that would be used in an explanatory environment model, since the purpose of the latter model is to understand the effect of true descriptors on actual project outcomes.

Indeed, current recommendations urge us to avoid irrelevant information and, in general, to take environmental measures so as to minimize the effects of unconscious processes on the estimator [85]. Although not explicitly stated, the underlying motive for such recommendations is based on a desire for rationality: One does not wish such unconscious processes to determine what experts do. The problem, however, is that we do not know how to manipulate experts' performance in other, rational ways. We do not know what the expertise of estimation is. We have the unfortunate situation where we discourage certain factors in the environment without providing a viable alternative; in effect, we're discarding the only forecaster model we have, at least in terms of it being a normative model.

Nevertheless, asymmetry in cues between environment model and forecaster model is bad for learning, because the interchange of explanatory elements between models can no longer take place. We therefore need to build a rational forecaster model. This requires that the forecaster is empowered with expertise so that the signal from this expertise stands out from the noise of unconscious processes induced by the environment.

Bias—conditional and unconditional

Biases are collectively called *calibration* in [145], and are an add-on to the forecaster model; recall that the latter is a model of an unbiased forecaster. The observation that estimators seem to be mis-calibrated with regards to estimates—they tend to under-estimate—and uncertainty—they tend to be unduly confident about their under-estimates—motivated a line of research that has demonstrated that several known effects from the heuristics and biases literature (of which Kahneman and Tversky are two of the foremost contributors) are also present in software effort estimation. In fact, this research constitutes the main body of what has been done toward understanding the forecaster in software effort estimation [64]. Our present knowledge is therefore almost exclusively about forecaster bias rather than about forecaster skill or expertise.

Judgmental biases are hard to unlearn [105]. Many of them persist even when forecasters (estimators) are made aware of them [60] and when measures are taken to neutralize them [95]. Moreover, in the complex setting of actual projects, multiple biases are expected to augment and counteract each other and to give rise to emergent biases [136]. At present, the disturbing effects of interacting biases overshadow forecaster skill. We do not know what a true (unbiased) M_{UY} is. We have, in fact, only measured biases. We therefore contend:

> *To improve on software effort estimation one should start to make efforts to strengthen the signal of expertise (skill) to stand out from the noise of biases.*

In fact, forecaster bias, as researched in software effort estimation, pertains to the environment, rather than to the forecaster. In physical sciences, the outcome O is not affected by the forecast Y. It will rain regardless of what the meteorologist says. In software engineering, effort estimates and project dynamics mutually affect each other to produce the project outcome. Research on biases has uncovered influences from the project environment that induce the biases, and current advice urges one to alter the environment (e.g., requirements documents) so as to avoid inducing biases [88].

3 Expertise

In the previous section, we argued that estimator performance was swamped by biases and that a stronger signal of estimation skill is necessary to break the dominance of biases. Formal models of the environment do not at present produce adequate estimates in general, and are in any case reliant on nontrivial human input. On the other hand, estimator skill must rely on knowledge embedded in models of the environment. All this adds up to building expertise in the forecaster, and we need to elaborate what we mean by that. To do this we will first introduce the concept of *mental model*.

3.1 Mental Models

A mental model is a person's, or a group of persons', implicit understanding or conception of some given topic of interest [53], [82], [84], [83]. In contrast to theoretical knowledge, or conceptual models, mental models represent tacit knowledge and are not explicit. A first step in influencing how people elaborate about a given topic, is to elicit and make explicit their mental models concerning the topic. Norman [118] has a useful description of mental model elicitation, which we depict in Fig. 3. The focus of interest is a system S. For us, this is the complex techno-social-psychological workings of a software development project

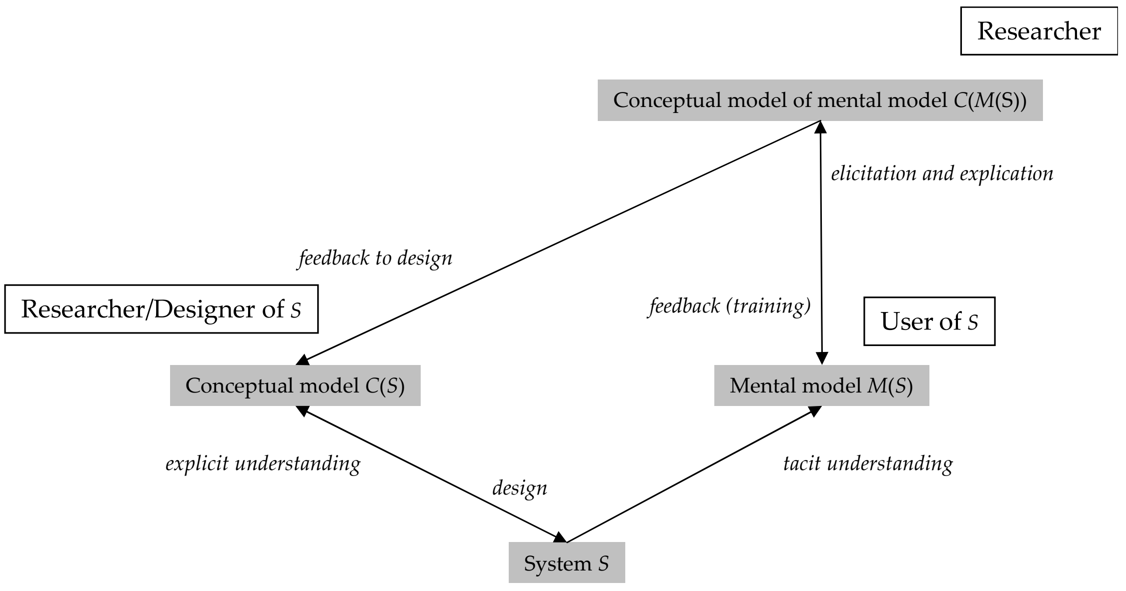

Fig. 3. Mental Models Elicitation in Research, Training and Design.

or more specifically, the estimation process, but foci for mental model exploration could also be a physical system such as an advanced machine [53], or it may be a socio-economic societal system or some other system [48], [104], [107], [113]. A mental model $M(S)$ of S then represents a the tacit knowledge and understanding of someone using or participating in S. A conceptual model $C(S)$ represents the explicit knowledge and understanding of researchers trying to understand S, or of designers of S if it is designed. In our context, $C(S)$ is the environment model M_{OT}.

The elicitation of a mental model $M(S)$ results in a conceptual model $C(M(S))$ of the mental model. The conceptual model $C(M(S))$ goes under several names; e.g., "cognitive map" [45] and "conceptual map" [9], [119], [106]. In our context, $C(M(S))$ is a forecaster model M_{YX}.

Based on the understanding established in $C(M(S))$, and on our understanding $C(S)$ of the software estimation process, we may provide feedback to users in the form of training. In addition, $C(M(S))$ may be used to provide feedback to designers of S. Designers of S, where S is the software estimation process, are researchers advising software developers how to estimate better and designers of formal estimation methods. Thus, in the cognitive tradition, mental model elicitation, the conceptualization of mental models, and ultimately altering mental models are steps in building expertise. Mental models research has produced much of what we know about expertise today.

3.2 Concept of Expertise

Expertise is a classic concept of social and behavioral science and has undergone several stages of elaboration with respect to definitions and operationalizations. At present, *expertise* is given several aspects [46]:

1) in terms of *individual differences in mental capacities*,
2) in terms of *extended experience*,
3) in terms of *superior knowledge representation and organization*; more specifically [78], [141]:
   - Expert knowledge
   - Expert deductive reasoning
   - Expert working memory
4) in terms of *reliably superior performance on representative tasks* (so-called *expert performance*)

Expertise is understood as task specific. Being an expert on one task does not necessarily relate to being an expert on a different task, and expertise often does not transfer across domains [36]. Differences in general mental capacities (1) is, for very many tasks, not specific enough, and as a definition of expertise, not adequate [2], [46]. In particular, this is the case for most software engineering tasks which demand extensive domain knowledge.

Extended experience (2) reflects the observation that people tend to get better at tasks through experience doing the task. For software effort estimation, such a definition of expertise is, ostensibly, problematic since research shows inadequate effects of experience on estimation performance.

Superior knowledge representation and organization (3) concerns cognitive structure and enables one to understand how expertise is formed and represented in the brain. It also enables one to consider what other cognitive factors, such as intelligence, may influence the development of expertise. It is a fundamental aspect of expertise and also difficult to operationalize. Nevertheless, notable general results about cognitive differences between novices and experts have been achieved through mental models research. Models based on this aspect of expertise play the role of explanatory models M_{UY} of the forecaster (Section 2).

Reliably superior behavior on representative tasks (4) is in many respects the most useful definition. It focuses on determining and enhancing expertise by concentrating on performance on small work samples. This is extremely time saving. However, a valid construct here would demand that one (i) is able to define the real-world task, (ii) is able to design representative test tasks, (iii) is able to determine relative difficulty of test tasks, and (iv) is able to assess the relative skill of performers on these test tasks. All this entails a level of control over the concept that allows for both the determination of the level of expertise and for developing means to improve expertise. For example, recent work on assessing programming skill has produced a test instrument that goes a long way toward meeting the above criteria [17], [18], [19]. For effort estimation, hardly any of the above criteria are met to any useful extent [67]. Notice how reliability is inherent in this aspect of expertise: One needs to exhibit reliably superior performance. Hence the

TABLE 1
Domains in which Good and Poor Expert Performance Have been Observed (based on [133])

Good performance	Poor performance
Weather forecasters	Clinical psychologists
Livestock judges	Psychiatrists
Astronomers	Astrologers
Test pilots	Student admissions
Soil judges	Court judges
Chess masters	Behavioral researchers
Physicists	Counselors
Mathematicians	Personnel selectors
Accountants	Parole officers
Grain inspectors	Polygraph (lie detector) judges
Photo interpreters	Intelligence analysts
Insurance analysts	Stock brokers
Nurses	Nurses
Physicians	Physicians
Auditors	Auditors
	Software effort estimators

focus on reliability in the framework of Section 2 captures this essential aspect of expert performance. The literature on *skill* usually defines skill in terms of this aspect.

There are interdependencies among the aspects of expertise. For example, extended experience often leads to superior knowledge representation and organization which manifests itself in reliably superior performance on representative tasks. Expertise comes and goes with practice. If you stop practicing, your expertise will likely drop below your optimal level [78]. To gain expertise on certain types of task, it is necessary to engage in deliberate practice; i.e., reflective practice on tasks and subtasks in conjunction with plentiful feedback on task performance [46].

3.3 Superior Performance

Superior performance on real tasks within the domain of expertise is the desired effect of expertise in the field. For the task of software effort estimation, superior performance is characterized by *more reliable and accurate estimates and more realistic uncertainty assessments in software development projects*. Expertise is one cause of superior performance. Other postulated causes are intelligence, personality, and as discussed in Section 2, unconscious biases. A large body of research concludes that expertise is the main determinant of superior performance while personality and intelligence have less predictive power [12], [13], [14], [128], [129]. However, software effort estimation is in an awkward position, since (i) there is a dearth of superior performance that distinguishes would-be experts from novices, and (ii) one does not know what it is that gives superior performance. In short, it is unclear what expertise in software effort estimation is. That biases have such a large effect in the absence of expertise is therefore unfortunate.

Software effort estimation shares its lack of superior performance with other domains and tasks. Shanteau [133] lists a series of professions and activities for which expert performance is good and for which expert performance is bad. Here "expert performance" alludes to performance by people with extended experience or who have been assigned the expert role by consensual agreement among peers. One may add software estimators to the "poor" side of this list (Table 1).

Superior performance does not define expertise. But whatever definition of expertise we adhere to, the definition should be such that more of it leads to better performance, that is, we are interested in the criterion validity of expertise with respect to superior performance (Fig. 4). We are interested in finding ways to increase performance, and therefore we need a useful definition, in two ways: We must be able to use it to predict performance, but it cannot be the same as what we wish to predict, because this would compromise the content validity of the expertise construct[4]. Note that the aspect of expertise, reliably superior performance on representative tasks, is not the same as superior performance on real-world tasks, since the former is a measurable test on smaller tasks that only represent real-world tasks.

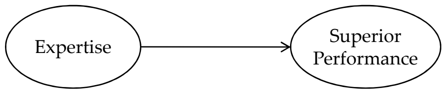

Fig. 4. Criterion Validity.

3.4 Operationalization

Concepts need to be operationalized into observable variables to have practical impact. In many disciplines, expertise is often operationalized by amount of experience in various ways [139], [132], even when expertise is not conceptually defined in terms of extended experience. For example, it is postulated that superior cognitive structure develops over time as a result of task-relevant experience, see [62] for a review of operationalizations of IT-expertise in the management literature.

For software effort estimation, this operationalization of expertise has not proved appropriate, for the same reasons that make the conceptual definition of expertise in terms of extended experience difficult. Other domains experience similar problems, and when lacking objective measures for expertise, Shanteau [133] suggests to let each domain define their experts, whether that be by peer consensus, official recognition (job titles), or other consensual acclamation.

However, operationalizations must reflect what they are intended to measure. Extended experience on a task should be measured by amount of experience on that task and not by a more general measure such as years of professional experience; unless one is out to measure and predict general professional performance. In other words, observable variables should respect the Brunswik symmetry.

Empirical work on software engineering expertise is not abundant, but there are interesting cases of operationalization. For example, among the 103 articles reporting software engineering experiments surveyed in [137] only three explicitly investigated the expertise construct related to the relevant task in some way. Expertise was operationalized in terms of various indicators: by type of mental model, e.g., [28], by degree of semantic knowledge, e.g., [123], by students versus professionals, e.g., [79], and by IT consultant fee category [6], [5], [68], etc. None of these operationalizations observed the Brunswik symmetry with regards to the purposes of the studies. Note that the resulting criterion validity therefore also falls below standard.

[4]A *construct* is a concept with a means to determine or measure variation in that concept (an operationalization). Any operationalization inevitably restricts the degree to which a construct represents the intended concept, and content validity is the extent to which a concept is expressed in a construct. If one merely defines a predicting construct in terms of the criterion construct, then one is in effect not defining a separate construct and one is failing in the outset to obtain a means for predicting anything.

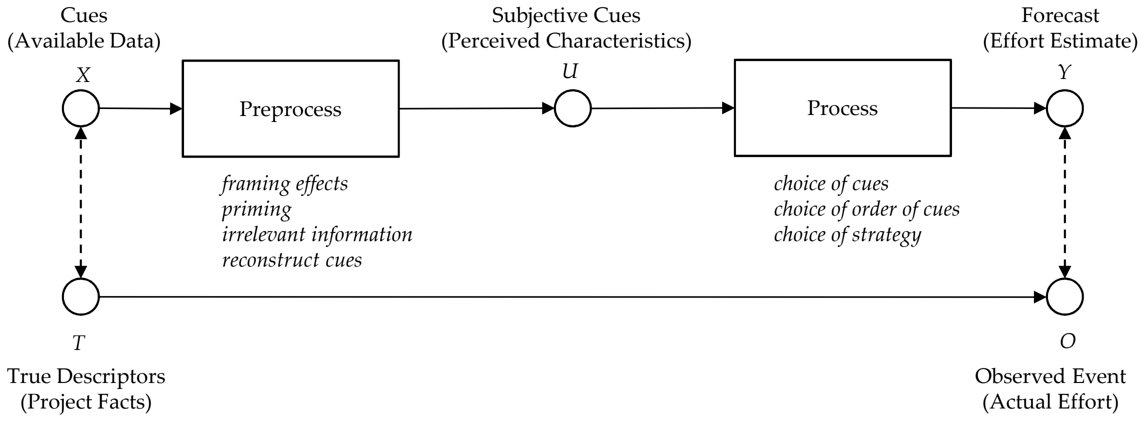

Fig. 5. Model of Estimation Task.

4 Tasks

Expertise relates to specific tasks. A further development is a more detailed analysis of task characteristics and how such characteristics and expertise relate to each other.

4.1 Task Structure

A task may be defined simplistically in terms of three elements: a set of *cues* (*input*), a set of *acts* (*processes*), and a set of *products* (*output*) [22], [153]. We can relate this to the model in Section 2. We invert Figure 2 to give the task-focused diagram in Figure 5. The model takes into account cues (input), a process, and output (the forecast) in the manner of traditional task definitions. In addition, the model takes into account subjective cues, which are the forecaster's interpretation or redefinition of the cues. Thus, the simple task model can be extended with a preprocess in which the (re)interpretation of cues takes place.

The preprocessing stage includes the weighting and (re)interpretation of objective cues in requirements documents and project characteristics. As mentioned in Section 2, objective cues are subject to a range of unconscious processes that produce bias and unreliability.

When taking a closer look at the process part of judgment tasks, unconscious processes play a major role here as well. Judgments can be seen as one of two kinds: *comparative*, where an observer identifies some relation between two sets of cues present, or *absolute*, where a single set of cues is assessed. In performing an absolute judgment, the immediate cues are related to information held in short-term memory, to information about former comparisons, or to some previously experienced measurement scale [146], [20]. Thus, even in absolute judgments a comparison takes place; namely between immediate cues and information stored in memory of related cues. The practice of comparing user-stories in planning poker and various other comparison techniques [16] make this comparison process explicit.

If comparison is at the heart of judgment processes, then we can consider Mussweiler's *selective accessibility process* model [116]; see Figure 6. This model suggests the cognitive steps that are involved in making comparisons. Comparisons are made between a target and a standard. In our context, the target might be a property of a current development project and the standard could be a related property in a past project. The main idea is that comparisons are done either by assessing similarities between the target and the standard or by assessing dissimilarities between the target and the standard. The choice between these two paths depends on an initial overall assessment of target-standard similarity. Once the choice is made, the decision-maker searches selectively for accessible information that supports this initial choice; that is, he searches for information that supports either the initial perceived similarity or the initial perceived dissimilarity. The outcome is a comparative judgment that is, respectively, an assimilation or a contrast.

Experiments demonstrate that subjects may be manipulated either way [116], [146]. This suggests that the selective accessibility process is largely beyond subjects' control; hence relegating also the main process part of judgment tasks to the unconscious processes together with those of the preprocess part.

However, there is, in fact, an important distinction. Whereas preprocess biases tend toward being task independent, process biases tend toward being task specific. First, the overall initial judgment that prescribes similarity or dissimilarity testing relies on domain-specific knowledge. Secondly, the ensuing hypothesis-testing process that is involved in the selective accessibility mechanism focuses on knowledge that relates specifically to the target of the comparison. As a consequence, specific target knowledge rather than general semantic knowledge is activated [116]. Further, the magnitude of comparison effects depends on the amount of available target knowledge; see [116], [34] for details and references. Thus, increasing the availability of user-story-related knowledge before making a comparisons in a planning poker session should lead to more pronounced comparison effects.

According to [116], this focusing characteristic of the selective accessibility mechanism should reduce the complexity of comparative evaluation, and since comparisons are involved in every judgment, the selective accessibility mechanism should become proceduralized [11]. Therefore:

> *This task-specificity is the signal of expertise that must be strengthened to overcome the noise of biases.*

In terms of the Brunswik symmetry, the broad semantic concepts behind the preprocess biases do not match the narrower task-specific concepts pertaining to estimation expertise and skill in particular. It makes little sense to evaluate a person's estimation skill based on how affected (s)he is on general biases.

In summary, one must distinguish between the cognitive processes that are amenable to deliberate practice and therefore relevant for building expertise, and those that are not. Klein states that "... for the most part, the recommended techniques for overcoming biases are not firmly grounded

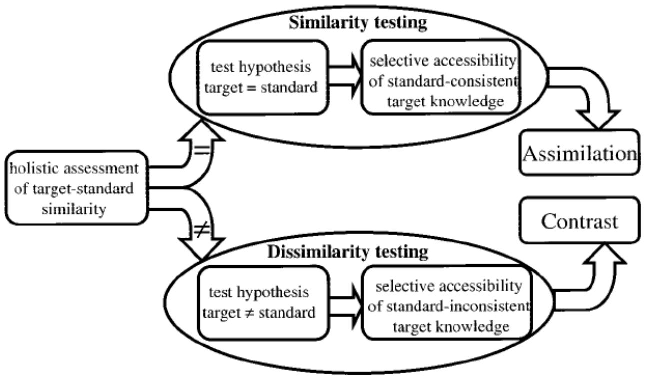
Similarity testing
test hypothesis target = standard
selective accessibility of standard-consistent target knowledge
Assimilation
holistic assessment of target-standard similarity
=
≠
Contrast
Dissimilarity testing
test hypothesis target ≠ standard
selective accessibility of standard-inconsistent target knowledge

Fig. 6. Selective Accessibility Process [116].

in evidence that the strategies can be learned and transferred to naturalistic settings" [105]. Moreover, there are arguments and evidence that several of the biases studied in the laboratory do not degrade performance in natural settings and that the biases may even disappear in real world settings [57], [105].[5] Along this line of thought, it is not clear how the extensive research on unconscious general biases in software engineering should be applied to improve estimation in practice. Indeed, general traits such as handedness [102] and GMA may influence the propensity to be affected by general biases, but such traits are impossible to manipulate.

Instead one should design environmental facilitation so that the signal of expertise becomes more prominent. The comparison mechanism above is affected by numerous conditions. For example, comparisons that involve non-extreme standards tend to trigger similarity testing, as does comparisons that involve ambiguous targets. Comparing user stories in prioritizing backlogs or in planning poker sessions involves comparing a user story to other user stories or to analog user stories from earlier experiences. In addition, user stories are purposely kept low in specificity. One would therefore expect a rather strong propensity for similarity testing. This could lead to prioritizing with too little variance with ensuing difficulties in backlog maintenance. To stimulate clearer decisions, one can use methods that stimulate dissimilarity testing; i.e., that stimulate the task-specific expertise that relates to discernibility. A useful method is the repertory grid elicitation technique [51], [10], [126], [69], [15].

4.2 Task Complexity

Based on the input(-preprocess-)process-output model, one can further analyze tasks according to complexity. There are several task complexity dimensions in the literature, and efforts have been made to systematize them [29]. It is reasonable to consider task complexity in relation to the human performing the task, and researchers have discussed task complexity from the point of view of *primarily a psychological experience*, *an interaction between task and person characteristics*, and *a function of objective task characteristics*; all of which are useful characterizations, see [29]. Nevertheless, in discussing expertise, it is particularly useful to isolate a concept of objective task complexity. This does not preclude the fact that a task of a specified complexity may be experienced differently by performers. One can reserve the notion of *difficulty* for the interaction between task and person: a task of given complexity may be less difficult for a trained person than for a novice, or it may be more difficult due to stable abilities, such as intelligence or height [29].

4.3 Objective Characteristics

There are several ways of expressing objective task complexity, but the most prominent elaborations can be forged into the input-process-output format. Common to all is a focus on task characteristics that demand cognitive load in the performer. Wood [153] argues in terms of *component complexity* (number of subtasks and cues and their dependencies), *coordinative complexity* (measure of precedence, timing, and other dependency relations between subtasks), and *dynamic complexity* (measure of changes in the two previous complexity components during task execution). This schema distinguishes tightly between variations on a task. "Landing a plane every 90 s requires much higher levels of coordination between acts and more frequent monitoring of cues than landing a plane every 60 min. In this sense, 'landing a plane safely at O'Hare airport, Chicago' is a different task than 'landing a plane safely at Champaign-Urbana airport' " [153]. One might also think that this tight discernibility implies that experts actually solve different tasks than novices. For example, chess masters use pattern recognition or better chunking [35] when contemplating a chess position, rather than analyzing the particular position of every piece as a novice is likely to do [58]. On the face of it, it may therefore seem that the master uses fewer cues and therefore performs a different task than the novice. However, the holistic approach of the master is dependent on all cues (the pieces) being there, and in this sense, the task and its objective complexity are the same as for the novice. However, other tasks have less defined structures and are not equally amenable to Wood's definition.

Other accounts of task complexity are therefore looser, and degrees of complexity are given by the distribution of emphasis on each part of that structure. These accounts are better suited to characterize judgment tasks. For example, Campbell [29] summarizes the literature on objective task complexity along the lines of Wood, but tasks may hold several resolution paths through the input-process-output structure; i.e., one has (1) the presence of multiple potential paths to arrive at the desired end-state, (2) the presence of multiple desired end-states, (3) the presence of conflicting interdependence among paths and end-states, and (4) the presence of uncertain probabilistic links among paths and end-states. In particular, judgment tasks such as software effort estimation are characterized by an emphasis on (3) and (4). Thus accurate judgment or prediction requires the task-doer to (a) determine which pieces of information to pay attention to, (b) weight these pieces appropriately, and (c) combine the weighted information to arrive at an overall judgment" [29]. Here, (a), (b), and (c) correspond to *input*, *process*, and *output*, respectively.

Abdolmohammadi and Wright [1] sort tasks according to *well-structuredness*. They use a model due to Simon [134] which divides a judgment task into three phases:

1) *Intelligence*—gather information to understand the problem, risks involved, and key factors to consider
2) *Design*—identify the alternative courses of action

[5]For example, the so-called Conjunction Fallacy, where most people commit a logical blunder when deciding from a description of "Linda" what is most likely: her being a bank-teller, or a bank-teller and a feminist [103], has been argued to be based on a disregard for natural language semantics [57].

TABLE 2
Task Complexity and the Decision Process ([1])

Task Complexity	Intelligence	Design	Choice
Unstructured	Unique, undefined, few/no guidelines available	Infinite/undefined alternatives	Judgment & insights needed
Semi-structured	Repetitive, semi-defined, guidelines available	Limited; specified alternatives	Judgment needed
Structured	Routine, well-defined problem & cues	Limited/well-specified alternatives	Little judgment needed

3) *Choice*—arrive at decision on which alternative to take.

These three phases correspond to *input*, *process*, and *output*, respectively. Task complexity is then defined in terms of the presence of structure on the phases (Table 2).

A related classification uses a notion of *consistency* which relates to the reliability discussion in Section 2. A task is said to be *consistent* if, over time, the best performers develop similar strategies to solving the task. An *inconsistent* task, on the other hand, is a task for which substantially different strategies for solving the task emerge [31], [30] Judgment tasks, such as software effort estimation, sort under so-called *ill-defined* tasks [81]. They transcend merely inconsistent tasks in that successful strategies seem difficult even to define [135], [125], [150]. Ill-structured tasks have no well-defined *a priori* solutions and it is the task doer who adds structure to the task.

In such conditions, a general result is that experts fare no better than novices, and even simple regression models often outperform experts. In software effort estimation, mathematical models are not better than experts, but this says perhaps more about the models than the experts.

In summary, software effort estimation as a task is traditionally seen as ill-defined, unstructured, has conflicting interdependence among paths and end-states, and has uncertain probabilistic links among paths and end-states. Along with attempts to increase expertise on such a task, it is relevant to ask if it is possible to evolve the task so that it has less of these undesirable characteristics.

4.4 Performing a Task

Several authors make a distinction between task structure and the process of solving the task. For example, Stuart and Prawitt [147] report that it is better with an unstructured rather than structured auditing routine on a complex task (high component and coordinate complexity, in Wood's terminology [153]). Their findings suggest that a structured routine is too rigid to gain experience with a task that is complex, i.e., a task that is less routine, more cognitively demanding, and less amenable to standardization.

Wood's task structure is defined in terms of subtasks and their precedences and which cues to consider for each subtask. Thus, there is considerable overlap in what task structure is and how to solve the task. Instead of distinguishing between task structure and the way to solve the task, one may view both various ways of solving a task and variations of task structure as different resolution paths to solving the task. Thus, if one finds that a certain routine solves a task better, then this routine can be seen as simply a more efficient resolution path through the input(-preprocess-)process-output configuration. Whether one views variants of a performing a task as defining a new resolution path or actually defining a new task depends on the granularity one chooses for defining subtasks and cues.

A related point is the statement that complex tasks do not necessarily demand complex solutions [54]. This again means that a task can be solved in several ways, and that although it is possible to focus on a task's complex structure, it is also possible to focus on an alternative simple structure. From this perspective, this entails that the notion of task complexity is volatile. Several studies suggest that a simpler resolution path gives better performance in certain circumstances. There may be several reasons for this, but in some cases it may be that following the complex structure would give superior performance but that the structure is too complex to be executable in sufficient detail in practice. The optimal strategy is therefore too complicated, and satisficing [136] by following the simpler structure might give better performance in practice. Even in cases where it is practically possible to follow the optimal strategy, it may be more cost-effective to follow a simpler satisficing strategy rather than a time-consuming optimizing strategy.

Complex behavior is not necessarily the result of a complex rule or complex mental strategies. The ant's complex route across a sandy beach as a result of a simple strategy interacting with a complex environment is prototypical to research on adaptive behavior, e.g., [57]. In an uncertain environment, good intuitions must ignore information. Gigerenzer and the Adaptive Behavior and Cognition Group [57] argue that simplicity is an adaptation to an uncertain and complex environment. Several simple heuristics outperform more analytical strategies on ill-defined tasks in judgment under uncertainty [71], [55], [56].

A slightly different account arises from the observation that experts follow abstract deeper-lying principles in solving tasks, while novices use surface information available in the concrete task instance at hand [37], [38], [47]. This has inspired a Chomskyan view [39] of task resolution paths as pertaining to *deep* (or *abstract*) structure versus surface structure. Based on the notions of surface structure and deep structure, Hærem and Rau link cues (input) and products (output) to a task's surface structure, and the act (process) part of a task to its deep structure. The *critical complexity* of a task is the complexity of the resolution path that minimizes the amount of information processing [63]. A task's objective task complexity can then be retro-defined in terms of its critical complexity as follows: A *surface structure task* is a task in which the critical complexity resides in its surface structure, and a *deep structure task* is a task in which the critical complexity resides in its deep structure. A mixed structure task is a task whose critical complexity resides in both its surface structure and its deep structure [63]. For example, software effort estimation can be done in a bottom-up manner or in top-down manner. According to [63], bottom-up estimation is a deep-structure resolution path and involves breaking down and analyzing the project and the system under development into sub-activities, estimating each sub-activity, and accumulating the estimates into a total project estimate. Top-down estimation is then closer to a surface-structure resolution path and is based on properties of the project as a whole and comparisons with similar projects, and then dividing this estimate into portions for parts of the project [86]. However, it is not clear that deep structure here coincides with what is meant by utilizing abstract properties in [37], because using the inner workings of a software system (rather than the input/output relation) does not necessarily involve abstract knowledge to a larger degree. For some tasks, it may be meaningful to claim that critical complexity resides statically in deep or surface structure, but for other tasks, this seems less viable.

5 Improving Software Effort Estimation

The lens model of Section 2 gave us a way to conceptualize the forecaster and the environment with an emphasis on symmetry between cues for the forecaster and environment (Brunswik symmetry). In Section 3 it also gave us a link to expertise via the forecaster model and the notion of mental models. We noted that software effort estimation as a whole is seen an ill-defined task, with little structure. However, the task model of Section 4 made it possible to indicate which stages of an estimation task that are affected by broad psychological factors and for which stages it is feasible to build task-specific expertise. We noted that most of the recommendations for improving software effort estimation concern the preprocess structure of the task that pertain to the broad psychological factors; e.g, avoiding biases in the the transformation of cues to vicarious cues. We also noted that task-specific expertise was to be found in the ensuing comparison sub-processes. We further suggested that methodological structure (e.g., pairwise comparisons, repertory grid elicitation, planning poker) can be given to the task to support that expertise.

We have thus given some conceptual and practical structure to the task of software effort estimation. We think that this can be a basis for further substantial improvement of performance on this and other planning tasks in software engineering. We will now outline how we can use our deliberations so far to proceed further. In summary, we propose three avenues:

To improve the signal of expertise in software effort estimation, we should, simultaneously

- *construct forecaster models,*
- *design tools and methodologies (environmental measures) that support expert behavior, and*
- *design tools and methodologies (environmental measures) that support expert learning.*

5.1 Constructing the Forecaster Model

What then, is the expertise that comes into play in comparisons? We identified the task-specific knowledge accessed in the selective accessibility process and we mentioned how one might encourage dissimilarity testing. But how is this accessed knowledge used to generate an estimate? Answering this question clearly would unravel the magic quantification step of estimation. Moreover, this would explain the step in terms of expertise, rather than in terms of broad non-task specific biases. Some studies have been conducted to investigate how estimators use knowledge. For example, in [149] both analogical and analytical-style reasoning was identified. However, it has been challenging to elicit the mental models of individuals, and not much progress has been made. However, group estimation has now become common. Recently, promising advances [23], [25], [24] have been initiated that take advantage of the explicit communication that group estimation necessitates. This studies use a socio-cultural perspective and discourse analysis to classify the knowledge elements that are verbally exchanged in group sessions along with their roles in generating an estimate. Rather than relegating the quantification step to "magic", this conceptualization has means to describe how reasoning to reach an estimate occurs on two levels—qualitatively; e.g., in terms of discussions around requirements specification and quantitatively in terms of relative or absolute estimates. Moreover, these levels are intertwined rather than in succession [24]. Thus, one can begin to map out how a forecaster model might look like, even though there are differences in individual and group processes. Work-life studies based on theoretical communicative concepts are a major step forward in this difficult terrain. We recommend that more such studies be initiated.

At the same time, normative guidelines are important. Exactly what such guidelines should be, depends on which estimation method one uses, but according to the lens model, this ultimately comes down to relevant domain knowledge and knowledge about process drivers. Moreover, according to the theory of expertise the knowledge embedded in the guidelines should be structured appropriately. However, since we do not have adequate environment and forecaster models, giving extensive normative guidelines for the comparative processes in software effort estimation seems difficult. So even if we have added some structure to the task, the comparative steps remain ill-defined, until we have developed better forecaster and environment models.

For tasks that are difficult to define, Klein [105] suggests two alternatives to normative guidelines. The first is to *think* like experts, and when even thinking like experts is hard to convey, one can instead teach people to *learn* like experts.

5.2 Thinking Like an Expert

It is common to build support tools and methodologies with a disregard for mental models. In the parlance of Section 3.1, one builds tools and methodologies on the basis of a conceptual model $C(S)$ (i.e., an environment model M_{OX} in our setting) rather than on the basis of a mental model $M(S)$. When the environment model is rudimentary, the resulting tool is necessarily haphazardly developed. A tool may also miss its mark if the correspondence G between the environment model M_{OX} and the forecaster model M_{YX} is insufficient, but also when this correspondence is high, a tool or methodology may be inadequate if it does not support the mental model(s) underlying the forecaster model (the latter is $C(M(S))$ in mental model terms). This happens a lot, since tools are often designed from abstract conceptual models that are tuned to aiding clear conceptual reasoning about the environment. By nature, such tools often reflect optimal ideal conditions and therefore fall short in the messy actualities in the workplace.

An approach when good models are lacking is not to design tools in terms of specific estimation strategies, but rather to focus on supporting expert behavior in its essence. A key point is the observation that the main effort of superior performers goes into assessing the nature of the situation rather than on comparing alternative courses of action; in particular, Klein [105] summarizes the essence of expert behavior as:

- Recognizing patterns (situation awareness).
- Making fine perceptual discriminations
- Recognizing typicality and detecting anomalies
- Mentally simulating future states (to evaluate courses of action) and past states (to generate explanations)
- Improvising (satisficing instead of optimizing)
- Adapting to events (i.e., meta-cognition)

This is in line with other research: Instead of explicitly simulating the complex environment grain by grain, one should adapt by following simple heuristics [54], [74]. Cleverness is to know when the environment allows the use of simple rules [75]. Many studies on successful experts conclude that assessing when to brake the rules of "best practice" is valuable. For example, in a study of entrepreneurs [44], successful entrepreneurs were found to follow means-driven rather than text-book goal-driven strategies.

To further explicate this, we can relate to the expertise aspect of *superior knowledge representation and organization* of expertise (Section 3). It has three interacting cognitive components: expert knowledge, expert deductive reasoning, and expert working memory [78].

Expert deductive reasoning is the main component. It is knowledge based, deductive, and inferential. From the repository of expert knowledge, which has been organized and structured in a superior way, experts should quickly, by using expert short-term working memory, comprehend and categorize the situation at hand before deducing alternative task solutions. It is useful to contrast this to the cognitive components of general intelligence. Whereas the central component of general intelligence, fluid intelligence (Gf), is inductive reasoning from first principles (e.g., the rules of chess), expert reasoning is deductive and inferential (e.g., from the battery of experienced and learned chess positions). Whereas short-term apprehension and retrieval from short-term working memory SAR is retrieval from short-term working memory, which holds seven chunks of information, plus or minus two, experts seem to be able to use an expanded expert working memory in their domain of expertise that holds much more than SAR's short-term working memory [78]. Novices must rely on Gf alone and will inductively generate a large number of alternatives based on first principles that may not bear sufficient relevance to the problem at hand. The inaccessibility of expert knowledge to novices is why general intelligence influences performance on entry-level jobs and on training, but does not influence expert-level performance to the same degree [128], [142]. This underpins our remarks in Section 4.4: A lack of knowledge forces a task performer to use first principles to induce an answer analytically, thereby possibly missing the mark which is deductively evident to someone with knowledge. For example, studies suggest that better software effort estimators use analogical thinking based on experience, and that reverting to analytical thinking produces less accurate estimates [149], [114], [121].

Expert deductive reasoning relies especially on the item *recognizing patterns* in the above list. This is central in analogical reasoning which probably occurs in most expert estimation strategies. Research suggests that estimators succeed in finding appropriate analogies only when there is a close surface similarity with the present project, but that the validity of this approach breaks down as soon as obvious similarities fail to be apparent. Thus, high levels of expertise is achievable, but only locally on very similar projects. There is little transferability of obtained expertise to other dissimilar projects. In this sense, expertise in software effort estimation is currently transferable through its *external validity*; i.e., in terms of variations on a project's specific variables [131]. In order to transfer expertise across substantially diverse projects, one must master recognizing patterns in terms of *construct validity*.

Transferring expertise by its construct validity means to recognize variables in a particular project as instances of abstract concepts (that, furthermore, are measurable). This enables one to recognize deep similarities (rather than surface similarities) between projects. Thus, projects that may seem dissimilar on the most visible characteristics, may actually share characteristics on a deeper level. If the effect of these deeper level variables on, say, productivity are well understood, then one may extend the scope of analogical reasoning. For example, rather than looking at superficial likenesses in the type of system, the programming language, the number of function points, etc. [110], one could look at similarities in organizational structure [117]. However, in general, the appropriate deep similarities to look for are not well known, and the prospect of fully understanding the underlying mechanisms of software projects is at present unrealistic. Instead, it is possible to concentrate on a smaller number of most important variables. This still remains a challenge since constructs and construct validity are not well developed in software engineering.

Studies on process suggest that experts and novices in unstructured and ill-defined tasks use similar processes as experts and novices in other types of problem solving [133]: Experts use top-down strategies, use more domain knowledge, etc., than do novices. The quandary is that this difference in process does not seem to bring about the expected differences in performance. Neither the task of software effort estimation nor the expertise required to be good at this task are within our scientific grasp yet; in other words, we are struggling with construct validity on these two concepts.

5.3 Learning Like an Expert

If thinking like an expert is not viable, Klein suggests to take aim to *learn* like an expert; i.e., we can make tools and methodologies that support learning to become an expert, in addition to, or instead of, support to be an expert. The following list presents strategies identified by a review of work on gaining expertise [105]:

- Engaging in deliberate practice, so that each opportunity for practice has a goal and evaluation criteria
- Using attentional control exercises to practice flexibility in scanning situations
- Sampling alternative task strategies
- Compiling an extensive experience bank
- Obtaining feedback that is accurate and diagnostic and reasonably timely
- Enriching experiences (i.e. reviewing prior experiences to derive new insights and lessons from mistakes)
- Building mental models
- Obtaining coaching

Several of these points are implemented to some extent in project data repositories and in agile practices such as burn-down charts and sprint reviews and retrospectives. Attempts to sample mental models and feed them back to practitioners has been done; e.g., [126], [69], [15]. However, tools or methodology that serve continuous mental model building [80], [4] are not implemented for software effort estimation to our knowledge. This, and support for deliberate practice are goals for a research project we are initiating, whose aim is to develop training and assessment modules for essential software project planning skills. Each planning skill concept should thus be a full-fledged construct with associated means for measuring variation in that concept (assessment) and means to manipulate the concept (training). The training scheme will be longitudinal with simulations and targeted feedback, thus offering intensive extended experience (Section 3) through deliberate practice.

We think the approach of withdrawing the researcher's position to that of giving support to practitioners' own theorizing and expertise building is crucial to achieving progress in software effort estimation and planning. This transfers the ownership and responsibility of expertise back to the practitioner where it belongs and is therefore likely to raise the necessary enthusiasm for success. It is important that this avenue proceeds in parallel to the two others outlined in Sections 5.1 and 5.2.

6 Example: Large Agile Project

We will now exemplify the most central concepts of this paper by relating to an actual project. A Norwegian public service company that administers pensions and loans has to acquire a new case handling system due to political reforms and new legislation. A large software development project was therefore initiated and organized using Scrum concepts [130] in a large-scale "agile fractal" manner with iterations on several levels [138]. The project is now in progress, and an excess of 300 epics (very high-level user stories) [40] define the functional scope. Each epic is a short description and acts as a placeholder for development tasks requiring in the magnitude of one person-year in effort. Each epic has been given a priority according to value for customer and an effort estimate produced by Wideband Delphi [21] in a top-down manner [86] involving stakeholders with business, functional, and technical expertise.

In addition to the scrum teams, there is one cross-cutting architectural team, one cross-cutting test team, and one development environment team supporting the Scrum teams. The choice for agile practices was taken on grounds that requirements were guaranteed to change during the project's lifetime due to political reforms that were not fully decided upon yet, and whose indirect effects through other governmental bodies were not yet clear.

The project selects, designs, builds, and deploys a subset of the epics every four months, giving three releases per year. Three different subcontractors supply ten on-site Scrum teams consisting of eight developers on average. The teams should deliver production-ready code in three-week iterations between these release points. A six-week release planning period precedes each release period, wherein epics are elaborated upon and split into user stories, possible interdependencies between development tasks are clarified, and priorities are reevaluated and detailed. Toward the end of this release planning, the detailed backlog for the release is split among the three developer companies who then proceed to negotiate a contract on the deliverables for the release with project management. The three subcontractors use their preferred estimation method to estimate effort for each item in their portion of the release backlog: Two use planning poker [41], while the third uses their own proprietary tool. The subcontractors establish an hours per story-point ratio (120 man-hours per story point, say). This ratio is then used in negotiations with project management to determine a target cost for the release with a 50-50 shared responsibility agreement for over-/underruns. The parties use the target cost-based [120] *PS2000* contract [122].

In this project, prediction happens at various occasions at different levels for different purposes and involving different stakeholders. We will discuss two occasions: project inception and release monitoring.

6.1 Project Inception

In the inception phase, a central estimation task was to deliver an estimate Y of the actual project effort O so that negotiations with governmental funding bodies could commence. As a part of this task, the master backlog consisting of epics with rough estimates was constructed. The available cues X were the functional and non-functional requirements of the system, as well as other project characteristics such as organization and technical environment.

Environment model: The forecasters did not use or construct an environment model M_{OX} at this stage. A descriptive environment model might have taken the form of a systematized database with historical project data and actual efforts of "similar" systems. Efforts are underway by the governmental IT regulatory body to collect historical data from large public service development projects, but at present, the results are not perceived as useful. An explanatory environment model, on the other hand, would have to provide usable insight into project drivers based on the available cues for this project. At present, there seems to be no comprehensive models or theories that are easy to apply in inception phase settings such as this one.

Environmental predictability and fidelity in the information system: True cues T in this phase are the actual requirements of the customer and true project characteristics pertaining to organization and technical environment, etc. Whether the actual requirements of the customer are captured by the requirements documents is a profound question that is addressed in research on requirements elicitation, capture and representation. In line with the project's choice of Scrum, the project uses the user story format. Although the good intentions of this format are plausible, it is an open question whether this format actually does entail a closer match of available cues X to true cues T in general. In the very early stages of the project, requirements were listed in a so-called migration plan which was reportedly very hard to understand for the customer, particularly after the lead architect left the project. Thus, the project did experience a substantial improvement in requirements handling when the master backlog was constructed. An environment model M_{OT} would describe or explain the relationship between what the customer *really* wants and project outcome. In some sense, efforts in agile methodology to involve the customer are intended to aid the customer in gradually becoming more aware of her requirements (true descriptors T) and hence also in updating the requirements documents (available cues X). The fidelity in the information system $V_{TX} = R_{OX}/R_{OT}$ would express predictability of the project using the requirement documents and available project characteristics versus somehow accessing the true requirements of the customer and using true project characteristics. In this project, V_{TX} is not computable, but still provides relevant conceptual concerns to the participants. Indeed, the extent to which user stories contribute to core functionality and added value remains a prime concern with project participants [15]. Related to this, there is concern that the project lacks a shared vision for the end-product [69]. Finally, environmental predictability R_{OT} of an environment model can be approximated by R_{OX}, which could be assessed over all available historical data in a database over projects.

Forecaster model: A forecaster model M_{YX} that is both descriptive and in part explanatory can be derived from the explicit estimation methodology the project used in this phase. First, the build effort of each epic was estimated using Wideband Delphi/planning poker using story points in the Fibonacci sequence. Based on the build estimate, estimates for analysis, design, system testing, debugging, Scrum ceremonies, and functional disambiguation are calculated as percentages of the build estimate. The sum of these estimates together with the build estimate constitutes the construction estimate. Then, estimates for business value analysis, epic deconstruction and elaboration, and verification are calculated as percentages of the construction estimate. Finally, estimates for other activities such as administration, technical environment, etc. are calculated as percentages of that total. All percentages are predefined. This then gives the overall project base estimate used as input to establish governmental funding.

Reliability of information acquisition and reliability of information processing: Subjective cues U are the estimator's reinterpretation of the information in the requirements documents and of the other project characteristics. The epics are high-level, and their further elaboration is intentionally left for later. At inception phase, the epics leave a lot of room for subjective interpretation and one would expect that inter- and intra-estimator reliability would be decreased by this fact. Although not done in the project, intra-estimator reliability, an aspect of R_{YX}, could be assessed by recording the span in epic points given for each epic in the planning poker sessions. A study on the project's user stories indicated that judges are inconsistent when assessing the same user story on different occasions and that the method used to rank user stories affects value assessments [16]. Ongoing research aims to find methods that aid forecasters in being consistent (reliable). The reliability of information acquisition $V_{UX} = R_{YX}/R_{YU}$ would express the extent to which the estimator can reliably interpret the subjective cues compared to the objective cues. To assess this, one would first have to gain insight into the extent to which estimators reinterpret epics and user stories and the variability in such interpretations. In any event, efforts were made to control unreliability: group estimation involves stakeholders holding different expertise, and (extreme) estimates are to be justified to the group; both measures recognized to increase reliability of estimates [144].

Match between environment and forecaster: Note that the distinction between what might be a model of the environment and what might be a model of the forecaster is not clear cut here. The forecaster is a part of the environment; more precisely, the method used by the forecaster, and therefore the forecaster model, forms how the project will evolve; in other words, it forms the environment model. This is in contrast to physical systems, such as weather, where the forecaster has no (direct) influence on the environment. One could also argue that the percentages schema used by project management constitutes an environment model. We choose not to, for two reasons. First, the schema needs human forecasts as input and second, the underlying presumption in the lens model is that the environment model describes or explains project outcome from cues in terms of scientific knowledge. We regard the percentages schema closer to a description of a mental model, and therefore belonging to the domain of the forecaster model.

Since the forecaster model is integrated in the would-be environment model, facilitating the environment for the forecaster is all the more pertinent and viable. It is important that this is done in ways that support building and utilizing expertise (Sections 5.2 and 5.3). Matching the forecaster and environment models is not possible in this case, since there is no environment model as such.

Bias: Bias at epic level is unknown. At sprint level, there seems to be marginal conditional bias (over-/underestimation). However, this is at subcontractor level, and there is a lack of traceability up to epic level. Unconditional bias (overconfidence) is not recorded in the project.

Expertise: We assume that the essence in judgment is comparison and that within comparison one can locate the task-specific factors that pertain to expertise in, first, the holistic initial (dis-)similarity assessment, and secondly, in the hypothesis-testing selective accessibility process (Section 4.1). In planning poker, this is explicit in that a user story or epic (target) is to be compared to a reference user story or epic (standard). Standards are conventionally chosen to be a small simple user story/epic, in part so that other user stories/epics may be compared in multiples of the target.[6] Thus, standards are deliberately chosen to aid similarity testing. One could argue that the use of a small standard makes it extreme w.r.t. very large user stories/epics and that dissimilarity testing is therefore triggered. However, very large user stories/epics are usually identified and broken down into smaller ones, and it is likely that the standard is not perceived as extreme so that the propensity for similarity testing persists. This means that as each epic is is considered, there is a propensity to assess it according to how similar it is to the reference epic, by selectively exciting target-specific knowledge to test the similarity hypothesis. Ambiguous perceptions of the target would strengthen this propensity. Under the assumption that it is beneficial to augment the elements of expertise so as to strengthen the deliberate signal against the noise of unconscious biases, one should here ensure that the forecaster has as much similarity-relevant expertise as possible on the target; i.e., the epic under consideration.

However, the actual work flow in planning poker is usually more complex. Although a standard is explicitly provided, analogical reasoning from past experience is pervasive in planning poker sessions in this project [24], and this introduces at least one additional standard that probably overshadows the provided reference epic/user story in terms of semantic content, although the original standard still serves as the definition of "unit". This new standard would virtually by definition trigger similarity testing. In group settings with different stakeholders, knowledge is integrated, rather than put forth competitively [25]. For example, both business flow and technical issues are elaborated when breaking down epics into user stories prior to casting one's bid in planning poker in the project [24]. The

[6]Recent findings suggest that one should choose a medium-sized user story as a reference, due to bias introduced by thinking in terms of multiples.

process of elaboration involves multiple comparison and reconceptualization and involves a substantially more complex series of steps in contrast to the search-find-compare-adjust sequence of formal models of analogical reasoning [23]. Moreover, group processes are in play, which could mean that social comparisons concerning competence, role, power, etc. are in play as well. To enhance the elements of expertise in such a situation is all the more important, and a thorough understanding of work flows and construction of the forecaster model are instrumental (Section 5.1).

Expert knowledge. The forecaster's mental model must include an efficiently organized knowledge base that is applicable to the deployment of large agile projects. Extended experience on a variety of such projects could give such a knowledge base, but as mentioned before, expertise is likely to be gained only on very similar projects. To speed up the learning process, a deliberate practice training regime might have exposed stakeholders to simulations of large agile projects where essential parameters are varied. It would further have been important to support the abstraction of salient factors of events to give an efficient organization of knowledge. In the inception phase, the diversity of stakeholders' backgrounds and interests are represented by various mental models. In the project, a major part of actual project deployment is fixed in the percentages (due to the mental model of the project manager) mentioned earlier, and variety is then factored out in the Wideband Delphi/planning poker sessions. The percentages are the results of experience from very similar past projects.

Expert deductive reasoning. The forecaster's mental model must include ways to deduce efficiently, from expert knowledge, how the requirements expressed in the epics will affect project deployment. In the project, stakeholders held varying levels of expert knowledge and very likely applied varying degrees of deductive reasoning, but this is unconfirmed, since studies on the project were only conducted after the inception phase. It may well have been the case that the reasoning process was "educated guessing" rather than expert deductive reasoning, and that the planning poker sessions adjusted variability in these guesses to relatively reasonable estimates. A deliberate practice training regime prior to project start might have enabled expert deductive reasoning to a larger extent.

Expert (short-term) working memory. This element allows the forecaster to quickly categorize the situations expressed in epics; possibly using rapid intuitive thinking [73] rather than time-consuming analytical processes. This requires a solid base of expert knowledge. Again, it is uncertain to what extent this element was utilized at inception phase. To support this cognitive element, is to support the building of effective chunking by repeated exposure to the task (extended experience) or small representative tasks (training).

6.2 Release Monitoring

During a release, it is important to monitor progress according to plan. Although not defined as an estimation phase, monitoring progress involves predicting future effort (outcome O) based on current progress. Project management uses elements from earned value management (EVM) [148], [109] using project data stored and monitored in JIRA (available cues X) to obtain an estimate Y of future effort. Based on the estimates for the epics, *planned value* $PV_t^{t'}$ is defined (mis-nominally, perhaps) as the budget for planned work between time t and time t' (e.g., the period included in a release plan). The *budget at completion* $BAC(t_C)$ is the total planned value up to some time of completion t_C (e.g., of production or of the entire project). *Earned value* $EV_t^{t'}$ is the amount of planned value realized between time t and time t'. In the project, 10% of planned value for an epic is regarded as realized when the epic enters elaboration, 30% when the target value contract has been signed, 85% when the epic is accepted by the customer at the designated check point, and 100% when the epic is put into production. *Actual cost* $AC_t^{t'}$ is the actual expenditure between time t and time t'. Then the *cost performance index* $CPI_t^{t'} = EV_t^{t'}/AC_t^{t'}$ gives the realized value to cost ratio between time t and time t' and is a measure of productivity. Another measure of productivity is the *schedule performance index* $SPI_t^{t'} = EV_t^{t'}/PV_t^{t'}$ which gives the realized value to planned value ratio. From these productivity metrics, future outcomes can be forecast; e.g., by computing the *estimate at completion* $EAC_t^{t'}(t_C) = BAC(t_C)/CPI_t^{t'}$ at some completion time t_C based on observations for time t to t'.

Environment model: The $EAC_t^{t'}(t_c)$ metric and others constitutes a model M_{OX} of the environment based on available cues X (project data). The model is descriptive (quantitative), not explanatory.

Environmental predictability and fidelity in the information system: The project data X are approximations of true project descriptors T that are subject to measurement error; e.g., by imprecise registering of AC or by errors in recording data in JIRA. Project data are in some instances also missing due to a lack of discipline in entering data. A study of two releases in the project showed that the three subcontractors had valid estimate/actual effort data for, respectively, 1066 (97.4%) of 1095 sprint tasks (23 sprints), 2118 (92.7%) of 2286 sprint tasks (32 sprints), and 1891 (95.9%) of 1972 sprint tasks (21 sprints) [76]. These are issues of the fidelity in the information system V_{TX}. Environmental predictability R_{OT} (which must be approximated by R_{OX}) can be calculated by correlating the actual cost AC with that predicted by, e.g., EAC over several periods of time (releases). A quicker way to gain insight into environmental predictability, is to sample data of completed sprint tasks and compute a distribution of actuals at sprint level or release level by using Monte Carlo simulation. This can be used for successively predicting uncertainty (environmental predictability) for the next sprint or release [98], [76]. For this project, the mean actual/estimate ratio over the sprints above where, for the three subcontractors, 1.05, 0.97, and 0.94, respectively. The distributions generated from the data are somewhat more informative; see [76] for details.

Environmental predictability could be improved by collecting project data more consistently. For example, there is only occasional data that links estimates and actual effort of sprint tasks to estimates and actual efforts on the epic level. Earned value is calculated based on how far the user stories belonging to an epic have proceeded in construction; see the 10-35-85-100% ranking above. However, the subcontractors do their own estimation on user stories, and actual cost (under-/overruns) are dealt with according to the deliverables contract which is subject to mercantile discretion [69]. While earned value according to plan is traceable, detailed estimates and actual effort is not traceable from the sprints up to epic. This compromises transparency and analyzability of exact locations in the production where problems or successes occur, which in turn compromises project learning and building of experience and therefore expertise. Nevertheless, it is important to note that, in this agile setting, the environment model is revised at regular intervals according to new project data; the intention being to improve environmental predictability for the future of the project according to the project's past.

Forecaster model: The forecaster model M_{YX} for release monitoring can only be sketched. It is clear that the environment model M_{OX} is integrated somehow in the forecaster model, since project management uses information obtained from the model to make decisions. Human judgment is needed in addition. For example, low CPI values may appear due to unforeseen events that trigger work not accounted for in planned value. In the project, nine types of uncertainty have been predefined and given designated epics complete with estimates. When an unforeseen event occurs, it is classified to one of the nine epics and a user story is allocated to the event. The user story is then included in the backlog, and thus the event is included in the earned value calculations. This simulates unforeseen events as being planned for. However, this may necessitate that an existing user story has to be taken out, and since there are non-trivial dependencies between user stories, consequences need to be elaborated, and the backlog may have to be re-estimated. The set of cues for the forecaster model is therefore larger than those for the environment model. Since the latter is used by the former, this should be regarded as a deliberate breach of the Brunswik symmetry due to the present status of the models.

Reliability of information acquisition and reliability of information processing: When monitoring progress, the forecaster interprets the project data, the outcome of the earned values management calculations, and other relevant data to understand anomalies in the earned value calculations or factors that might affect future productivity. The opportunity for reinterpreting cues X to subjective cues U are many and depend on the forecaster's mental model of the project. In this project, this was not investigated specifically, but indications as to the variability of subjective perceptions were that various project members had diverging views on how on track the project was. Reliability in information processing concerns how consistent the forecaster is in assessing progress. This was not assessed in the project.

Match between environment and forecaster: The environment model generates input cues to the forecaster model, and it is important that the forecaster model includes additional cues. To paraphrase a manger (whom we interviewed for a different occasion): *it's fine to have a dashboard with instruments showing speed and progress, but you have to look at the road too*. It is extremely important to acknowledge that a current environment model is suboptimal (but still useful) and that one needs to integrate the model in a larger mental model (practitioner) and forecaster model (researchers). In this setting, the goal can still be to match the environment model and the forecaster model, but neither model is the gold standard. We postulate that conscious and explicit reference to these models and to the asymmetry in cues will benefit projects such as this. Conscious efforts can then be made to expand the two models mutually, with an ultimate goal that they observe the Brunswik symmetry.

Bias: Although we have viewed the release-monitoring activity as an estimation activity, the purpose is to guide when to adjust resources in real time. For this reason, biases have not been assessed at this level in the project.

Expertise: Here when unforeseen events happen, project management must recognize that such an event is occurring and classify it to one of the nine predefined uncertainty categories, define a user story for the event and estimate the effort/cost of the event. The target is the event and the standards are the categories and their contents. Targets cannot be considered extreme here, so there should be a tendency for similarity testing rather than dissimilarity testing. This means that as each category is considered, there is a propensity to assess an unforeseen event according to how similar it is to a category, by selectively exciting target-specific knowledge to test the similarity hypothesis. Ambiguous perceptions of the target would strengthen this propensity. Under the assumption that it is beneficial to augment the elements of expertise so as to strengthen the conscious signal against the noise of biases, one should here ensure that the forecaster has as much similarity-relevant expertise as possible on the target; i.e., the unforeseen event. This would strengthen the adherence of the target to the most appropriate category of multiple categories that may in the outset fit the event.

Expert knowledge. The forecaster's mental model must include an efficiently organized knowledge base that is applicable to unforeseen events. Extended experience on unforeseen events in the project, or rather across projects, would give such a knowledge base. To speed up the learning process, a training regime might expose students to a battery of unforeseen events that are found (by research and experience) to be most often occurring in projects. It is important to support abstracting salient features of events to give an efficient organization of knowledge.

Expert deductive reasoning. The forecaster's mental model must include ways to deduce efficiently how the event is similar to one of the nine uncertainty categories in the project. The same extended experience and training above is applicable. In addition, environmental measures to aid the forecaster would be to give comprehensive definitions of the categories perhaps using checklists, or to provide the quick estimation heuristic [71]. Note that the categories themselves support abstraction and principled thinking, which are both consistent with associating expertise with the abstract or deep structure of a task. Optimally, the nine categories in this project should relate to cardinal uncertainties taught in training.

Expert (short-term) working memory. Massive experience in using the nine categories to classify events would train expert memory. This would enable rapid and clear classifications as to what type of unforeseen event the project is experiencing. This would replace e.g., checklists for analytical classification. It is unclear which processes project members use when classifying unforeseen events in the project.

7 Conclusions and Implications

We draw the following conclusions and implications from our deliberations in the previous sections:

> *Conclusion*: The current nature of insight into estimation processes is most useful for taking ameliorative action in the environment.

Because: Studies into what affects estimates have almost exclusively focused on bias effects between groups, irrespective of levels of expertise. In other words, experimental groups have been subjected to environmental manipulation (anchors, amount/format/nature of information) and then systematic differences in estimates have been observed as consequences of such manipulation. It is therefore possible to recommend various measures to be taken in the environment (avoid ill-founded anchors, avoid irrelevant information, etc.) to facilitate better estimates. However,

> *Conclusion*: There are common-sense but non-validated assumptions underlying these insights.

For example, one assumes implicitly that irrelevant information inflates estimates wrongfully and that anchors wrongfully affect the "true" estimate, but there is no validation as to whether biased estimates are not, in fact, closer to actual effort than are unbiased estimates. Therefore,

> *Implication*: Advice on taking environmental measures may not give better estimates

Because: Such advice may reduce the effect of a bias that actually gives better estimates, and therefore,

> *Implication*: Estimates may be good or bad estimates for the "wrong" reasons.

This is frustrating and clearly unacceptable as a basis for reflective practice aiming to build expert knowledge on which to make rational decisions using expert deductive reasoning. If research were to reveal that inserting 33% irrelevant information does produce better estimates, then acting on that information is not acting on the root cause of the problem, and would be a solution more akin to evidence-based magic. It is important that practice reflects our ideas of reducing complexity in the environment, and adding irrelevant information would to most people mean adding complexity, which would not make the estimation journey any less tortuous. Rather than making the beach smoother, we may inadvertently be introducing even more complexity for the software engineering ant.

Further, it is not acceptable for generating explanatory models of the environment and estimator that environmental measures have spurious effects. We know that putting up highway road dividers is life saving and that driving on the wrong side of such a road divider is wrong. But we would not know if correcting for judgment biases is cost saving and we would not know which side of such a bias is the right side.

To actually reduce complexity in the environment and make the beach smoother:

> *Implication*: We need to develop normative theories for rational effects of environmental measures.

In other words, we should understand which environmental measures that do, in fact, facilitate better estimation. However, there are a host of bias-inducing factors in the environment which in naturalistic settings may interact and even induce emergent or canceling effects. Working guidelines would have to consider all of these factors at once, which would amount to rearranging the sand on the beach to a painstaking detail.

> *Conclusion*: Controlling the environment so it does not induce biases is impossible.

To produce better estimation, we can make the beach smoother as recently discussed. But we can also alter the way software engineering ants respond to obstacles. However, training people to be unaffected by biases is not feasible due to the robustness of biases and the broad stable psychological factors that mediate them. On the other hand, it is possible to identify the task-specific elements of judgments tasks that can be altered by training regimes such as deliberate practice with feedback. This is building expertise; more specifically, it is strengthening the mental models of estimators so that their deliberate actions become stronger. To alter the way software engineering ants respond to the environment;

> *Conclusion*: We need to strengthen the deliberate signal of task-specific expertise to overcome the noise of unconscious broad psychological biases.

This is also hard, but at least there are feasible steps forward that are very promising. We propose to follow a three-way endeavor that aims to unearth the task-specific elements in play in the comparison processes that underlie judgments such as estimation, to design environmental facilitation to support expert thinking and behavior, and to design environmental facilitation to support expert learning.

> *Conclusion*: We need to conduct facilitating measures in the environment after all, but in order to support task-specific expertise, rather than to cancel broad psychological bias-inducing factors.

Acknowledgements

The author is grateful to Magne Jørgensen, Gunnar Bergersen, and Thorvald Hærem for many enjoyable discussions regarding topics in this article.

References

[1] M. Abdolmohammadi and A. Wright, "An examination of the effects of experience and task complexity on audit judgments," *The Accounting Review*, vol. 62, no. 1, pp. 1–13, 1987.

[2] P.L. Ackerman and M.E. Beier, "Methods for studying the structure of expertise: Psychometric approaches," in *The Cambridge Handbook of Expertise and Expert Performance*, K.A. Ericsson, N. Charness, P.J. Feltovich, and R.R. Hoffman, Eds. Cambridge Univ. Press, 2006, ch. 9, pp. 147–166.

[3] C. Argyris, *Knowledge for Action*. Jossey-Bass Publishers, 1993.

[4] C. Argyris and D.A. Schön, *Organizational Learning II. Theory, Method, and Practice*. Addison-Wesley Publishing Company, Inc., 1996.

[5] E. Arisholm, H. Gallis, T. Dybå, and D.I.K. Sjøberg, "Evaluating pair programming with respect to system complexity and programmer expertise," *IEEE Trans. Software Eng.*, vol. 33, pp. 65–86, Feb. 2007.

[6] E. Arisholm and D.I.K. Sjøberg, "Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software," *IEEE Trans. Software Eng.*, vol. 30, pp. 521–534, Aug. 2004.

[7] J.S. Armstrong, "Forecasting of software development work effort: Introduction," *Int'l J. Forecasting*, vol. 23, p. 447, 2007.

[8] J.S. Armstrong, R.J. Brodie, and S.H. McIntyre, "Forecasting methods for marketing: Review of empirical research," *Int'l J. Forecasting*, vol. 3, pp. 355–376, 1987.

[9] D.P. Ausubel, J.D. Novak, and H. Hanesian, *Educational Psychology: A Cognitive View*, 2nd ed. Holt, Rinehart & Winston, 1978.

[10] N. Baddoo and T. Hall, "Practitioners roles in software process improvement: An analysis using grid technique," *Software Process Improvement and Practice*, vol. 7, pp. 17–31, 2002.

[11] J.A. Bargh, "The automaticity of everyday life," in *Advances in Social Cognition*, R.S. Wyer, Ed. Erlbaum, 1997, vol. 10, pp. 1–61.

[12] M.B. Barrick, M.K. Mount, and T.A. Judge, "Personality and performance at the beginning of the new millennium: What do we know and where do we go next?" *Int'l J. Selection and Assessment*, vol. 9, no. 1/2, pp. 9–30, 2001.

[13] M.R. Barrick, G.L. Stewart, M.J. Neubert, and M.K. Mount, "Relating member ability and personality to work-team processes and team effectiveness," *J. Applied Psychology*, vol. 83, no. 3, pp. 377–391, 1998.

[14] S.T. Bell, "Deep-level composition variables as predictors of team performance: A meta-analysis," *J. Applied Psychology*, vol. 92, no. 3, pp. 595–615, 2007.

[15] H.C. Benestad and J.E. Hannay, "A comparison of model-based and judgment-based release planning in incremental software projects," in *Proc. 33rd Int'l Conf. Software Engineering (ICSE 2011)*. ACM, 2011, pp. 766–775.

[16] H.C. Benestad and J.E. Hannay, "Does the prioritization technique affect stakeholders selection of essential software product features?" *To be submitted to ESEM*, 2012.

[17] G.R. Bergersen, "Assessing programming skill, to be submitted in 2012," Ph.D. dissertation, Simula Research Laboratory/University of Oslo, 2012.

[18] G.R. Bergersen and J.E. Gustafsson, "Programming skill, knowledge and working memory among software developers from an investment theory perspective," *J. Individual Differences: To appear*, 2010.

[19] G.R. Bergersen and J.E. Hannay, "Detecting learning and fatigue effects by inspection of person-item residuals," in *Probabilistic Models for Measurement in Education, Psychology, Social Science and Health*, 2010.

[20] A.L. Blumenthal, *The Process of Cognition*. Prentice Hall, 1977.

[21] B.W. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

[22] S. Bonner, "A model of the effects of audit task complexity," *Accounting Organization and Society*, vol. 19, pp. 213–234, 1994.

[23] K. Børte, "Challenges faced by software professionals in analogy-based top-down estimation," *European J. Information Systems; submitted*, 2010.

[24] K. Børte, S. Ludvigsen, and A. Mørch, "The role of concepts in professional work: Unpacking the "magic step" in software effort estimation," *Information & Software Technology*, 2012.

[25] K. Børte and M. Nerland, "Software effort estimation as collective accomplishment: An analysis of estimation practice in a multi-specialist team," *Scandinavian J. Information Systems; to appear*, 2010.

[26] B. Brehmer, "Note on the relation between clinical judgment and the formal characteristics of clinical tasks," *Psychological Bulletin*, vol. 83, pp. 778–782, 1976.

[27] B. Brehmer, "Response consistency in probabilistic inference tasks," *Organizational Behavior and Human Performance*, vol. 22, pp. 103–115, 1978.

[28] J.M. Burkhardt, F. Détienne, and S. Wiedenbeck, "Object-oriented program comprehension: Effect of expertise, task and phase," *Empirical Software Engineering*, vol. 7, no. 2, pp. 115–156, June 2002.

[29] D.J. Campbell, "Task complexity: A review and analysis," *Academy of Management Review*, vol. 13, no. 1, pp. 40–52, 1988.

[30] J.P. Campbell, "Modeling the performance prediction problem in industrial and organizational psychology," in *Handbook of Industrial and Organizational Psychology*, 2nd ed., M.D. Dunnette and L.M. Hough, Eds. Consulting Psychologists Press, Inc., 1990, vol. 1, pp. 687–732.

[31] J.P. Campbell, R.A. McCloy, S.H. Oppler, and C.E. Sager, "A theory of performance," in *Personnel Selection in Organizations*, N. Scmitt and W.C. Borman, Eds. Josey-Bass, 1993, pp. 35–70.

[32] C. Carmerer, "General conditions for the success of bootstrapping models," *Organizational Behavior and Human Performance*, vol. 27, pp. 411–422, 1981.

[33] C. Carmerer, "Why are judgments less consistent in less predictable task situations?" *Organizational Behavior and Human Performance*, vol. 63, pp. 247–263, 1995.

[34] G.B. Chapman and E.J. Johnson, "Anchoring, activation, and the construction of values," *Organizational Behavior and Human Decision Processes*, vol. 79, pp. 1–39, 1999.

[35] W.G. Chase and H.A. Simon, "The mind's eye in chess," in *Visual Information Processing*, W.G. Chase, Ed. Academic Press, 1973.

[36] M.T.H. Chi, "Two approaches to the study of experts' characteristics," in *The Cambridge Handbook of Expertise and Expert Performance*, K.A. Ericsson, N. Charness, P.J. Feltovich, and R.R. Hoffman, Eds. Cambridge Univ. Press, 2006, ch. 2, pp. 21–30.

[37] M.T.H. Chi, P.J. Feltovich, and R. Glaser, "Categorization and representation of physics problems by experts and novices," *Cognitive Science*, vol. 5, pp. 121–152, 1981.

[38] M.T.H. Chi, R. Glaser, and E. Rees, "Expertise in problem solving," in *Advances in the Psychology of Human Intelligence*, R. Sternberg, Ed. Lawrence Erlbaum Associates, Inc., 1982, pp. 17–76.

[39] N. Chomsky, *Syntactic Structures*, 2nd ed. Mouton, 2002.

[40] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[41] M. Cohn and R. Martin, *Agile Estimating and Planning*. Prentice Hall, 2005.

[42] R.W. Cooksey, "The methodology of social judgement theory," *Thinking and Reasoning*, vol. 2, no. 2/3, pp. 141–173, 1996.

[43] J. Dana, "Is task complexity an exception to the superiority of mechanized judgement, or a barrier to it?" *Int'l J. Forecasting*, vol. 23, pp. 463–464, 2007.

[44] N. Dew, S. Read, S.D. Saravasthy, and R. Wiltbank, "Effectual versus predictive logics in entrepreneurial decision-making: Differences between experts and novices," *J. Business Venturing (in press)*, 2008. [Online]. Available: doi:10.1016/j.jbusvent.2008.02.002

[45] J.K. Doyle and D.N. Ford, "Mental models concepts for system dynamics research," *System Dynamics Review: J. System Dynamics Society*, vol. 14, no. 1, pp. 3–30, 1998.

[46] K.A. Ericsson, "An introduction to Cambridge Handbook of Expertise and Expert Performance: Its development, organization, and content," in *The Cambridge Handbook of Expertise and Expert Performance*, K.A. Ericsson, N. Charness, P.J. Feltovich, and R.R. Hoffman, Eds. Cambridge Univ. Press, 2006, ch. 1, pp. 3–20.

[47] K.A. Ericsson and N. Charness, "Expert performance—its structure and acquisition," *American Psychologist*, vol. 49, pp. 725–747, 1994.

[48] D.N. Ford and J.D. Sterman, "Expert knowledge elicitation to improve formal and mental models," *System Dynamics Review*, vol. 14, no. 4, pp. 509–340, 1998.

[49] N.R. Franks, A. Dornhaus, J.P. Fitzsimmons, and M. Stevens, "Speed versus accuracy in collective decision making," *Proc. R. Soc. Lond B*, vol. 270, no. 1532, pp. 2457–2463, 2003.

[50] N.R. Franks and R. T., "Teaching in tandem-running ants," *Nature*, vol. 439, no. 7073, 2006.

[51] F. Fransella, R. Bell, and D. Bannister, *A Manual for Repertory Grid Technique*. John Wiley & Sons, Ltd., 2004.

[52] D. Gentner, K.J. Holyoak, and B.N. Kokinov, Eds., *The Analogical Mind: Perspectives from Cognitive Science*. MIT Press, 2001.

[53] D. Gentner and A.L. Stevens, Eds., *Mental Models*. Lawrence Erlbaum Associates, Inc., 1983.

[54] G. Gigerenzer, *Gut Feelings. The Intelligence of the Unconscious*. Viking, Penguin, Ltd., 2007.

[55] G. Gigerenzer and D.G. Goldstein, "Reasoning the fast and frugal way: Models of bounded rationality," *Psychological Review*, vol. 103, no. 4, pp. 650–669, 1996.

[56] G. Gigerenzer and D.G. Goldstein, "Betting on one good reason," in *Simple Heuristics that Make Us Smart*, G. Gigerenzer and P.M. Todd, Eds. Oxford University Press, 1999, ch. 4, pp. 75–95.

[57] G. Gigerenzer and P.M. Todd, Eds., *Simple Heuristics that Make Us Smart*. Oxford University Press, 1999.

[58] F. Gobet and N. Charness, "Expertise in chess," in *The Cambridge Handbook of Expertise and Expert Performance*, K.A. Ericsson, N. Charness, P.J. Feltovich, and R.R. Hoffman, Eds. Cambridge Univ. Press, 2006, ch. 30, pp. 523–538.

[59] S. Gregor, "The nature of theory in information systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, Sept. 2006.

[60] S. Grimstad and M. Jørgensen, "Inconsistency in expert judgment-based estimates of software development effort," *J. Systems and Software*, vol. 80, no. 11, pp. 1770–1777, 2007.

[61] T.M. Gruschke and M. Jørgensen, "Assessing uncertainty of software development effort estimates: Learning from outcome feedback," in *Proc. Eleventh Int'l Symp. Software Metrics*, 2005, p. 4.

[62] T. Hærem, "Task complexity and expertise as determinants of task perceptions and performance: Why technology-structure research has been unreliable and inconclusive," Ph.D. dissertation, Norwegian School of Management BI, 2002.

[63] T. Hærem and D. Rau, "The influence of degree of expertise and objective task complexity on perceived task complexity and performance," *J. Applied Psychology*, vol. 92, no. 5, pp. 1320–1331, 2007.

[64] T. Halkjelsvik and M. Jørgensen, "From origami to software development: A review of studies on judgment-based predictions of performance time," *accepted to Psychological Bulletin*, 2011.

[65] T. Halkjelsvik, M. Jørgensen, and K.H. Teigen, "To read two pages, I need 5 minutes, but give me 5 minutes and I will read four: How to change productivity estimates by inverting the question," *Applied Cognitive Psychology*, 2010.

[66] K.R. Hammond and T.R. Stewart, *The Essential Brunswik*. Oxford University Press, 2001.

[67] J.E. Hannay, "Personality, intelligence, and expertise: The impact on software development," in *Making Software: What Really Works and Why We Believe It*, A. Oram and G. Wilson, Eds. O'Reilly, 2010, ch. 6.

[68] J.E. Hannay, E. Arisholm, H. Engvik, and D.I.K. Sjøberg, "Personality and pair programming," *Transactions on Software Engineering*, vol. 36, no. 1, pp. 61–80, January/February 2010.

[69] J.E. Hannay and H.C. Benestad, "Perceived productivity threats in large agile development projects," in *Proc. 4th Int'l Symp.Empirical Software Engineering and Measurement (ESEM)*. IEEE Computer Society, 2010, pp. 1–10.

[70] J.E. Hannay, D.I.K. Sjøberg, and T. Dybå, "A systematic review of theory use in software engineering experiments," *IEEE Trans. Software Eng.*, vol. 33, pp. 87–107, Feb. 2007.

[71] R. Hertwig, U. Hoffrage, and L. Martignon, "Quick estimation: Letting the environment do the work," in *Simple Heuristics that Make Us Smart*, G. Gigerenzer and P.M. Todd, Eds. Oxford University Press, 1999, ch. 10, pp. 75–95.

[72] R. Hertwig and P.M. Todd, "More is not always better: The benefits of cognitive limits," in *Thinking: Psychological Perspectives on Reasoning, Judgment and Decision Making*, D. Hardman and L. Macchi, Eds. John Wiley & Sons, Ltd, 2003, pp. 213–231.

[73] R.M. Hogarth, *Educating Intuition*. The University of Chicago Press, 2001.

[74] R.M. Hogarth, "When simple is hard to accept," in *Ecological Rationality: Intelligence in the World*, P.M. Todd, G. Gigerenzer, and T.A.R. Group, Eds. Oxford University Press, 2006.

[75] R.M. Hogarth and N. Karelaia, "On heuristic and linear models of judgment: Mapping the demand for knowledge," June 2006, available at Social Science Research Network ssrn.com/abstract=1002514.

[76] M. Holm, "Construction and evaluation of a tool for quantifying uncertainty of software cost estimates," Master's thesis, Institutt for Informatikk, Universitetet i Oslo, 2011.

[77] K.J. Holyoak, "Analogy," in *The Cambridge Handbook of Thinking and Reasoning*, K.J. Holyoak and R.J. Sternberg, Eds. Cambridge Univ. Press, 2005, pp. 117–142.

[78] J. Horn and H. Masunaga, "A merging theory of expertise and intelligence," in *The Cambridge Handbook of Expertise and Expert Performance*, K.A. Ericsson, N. Charness, P.J. Feltovich, and R.R. Hoffman, Eds. Cambridge Univ. Press, 2006, ch. 34, pp. 587–612.

[79] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—a comparative study of students and professionals in lead-time impact assessment," *Empirical Software Engineering*, vol. 5, no. 3, pp. 201–214, Nov. 2000.

[80] P. Jarvis, *The Practitioner-Researcher*. Jossey-Bass Publishers, 1999.

[81] E.J. Johnson, "Expertise and decision under uncertainty: Performance and process," in *The Nature of Expertise*, M.T.H. Chi, R. Glaser, and M.J. Farr, Eds. Lawrence Erlbaum Associates, Inc., 1988, pp. 209–228.

[82] P.N. Johnson-Laird, *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge Univ. Press, 1983.

[83] P.N. Johnson-Laird, "The history of mental models," in *Psychology of Reasoning: Theoretical and Historical Perspectives*, K. Manktelow and M.C. Chang, Eds. Psychology Press, 2004, pp. 179–212.

[84] P.N. Johnson-Laird, "Mental models and thought," in *The Cambridge Handbook of Thinking and Reasoning*, K.J. Holyoak and R.J. Sternberg, Eds. Cambridge Univ. Press, 2005, pp. 185–208.

[85] M. Jørgensen, "A review of studies on expert estimation of software development effort," *J. Systems and Software*, vol. 70, no. 1–2, pp. 37–60, 2004.

[86] M. Jørgensen, "Top-down and bottom-up expert estimation of software development effort," *Information and Software Technology*, vol. 46, no. 1, pp. 3–16, 2004.

[87] M. Jørgensen, "The "magic step" of judgment-based software effort estimation," in *Int'l Conf. Cognitive Economics*. New Bulgarian University, 2005, pp. 105–114.

[88] M. Jørgensen, "Practical guidelines for expert-judgment-based software effort estimation," *IEEE Software*, vol. 22, no. 3, pp. 57–63, 2005.

[89] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *Int'l J. Forecasting*, vol. 23, pp. 449–462, 2007.

[90] M. Jørgensen, "How should we compare forecasting models with expert judgement?" *Int'l J. Forecasting*, vol. 23, pp. 473–474, 2007.

[91] M. Jørgensen, "Selection of strategies in judgment-based effort estimation," *J. Systems and Software*, vol. 83, no. 6, pp. 1039–1050, 2010.

[92] M. Jørgensen, B. Boehm, and S. Rifkin, "Software development effort estimation: Formal models or expert judgment?" *IEEE Software*, vol. 26, no. 2, pp. 14–19, Mar/Apr 2009.

[93] M. Jørgensen and G.J. Carelius, "An empirical study of software project bidding," *IEEE Trans. Software Eng.*, vol. 30, no. 12, pp. 953–969, 2004.

[94] M. Jørgensen and S. Grimstad, "Avoiding irrelevant and misleading information when estimating development effort," *IEEE Software*, pp. 78–83, May/June 2008.

[95] M. Jørgensen and S. Grimstad, "The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment," *IEEE Trans. Software Eng.*, vol. 37, no. 5, pp. 695–707, 2011.

[96] M. Jørgensen and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation," *J. Systems and Software*, vol. 83, no. 1, pp. 29–36, 2010.

[97] M. Jørgensen and D.I.K. Sjøberg, "Impact of effort estimates on software project work," *Information and Software Technology*, vol. 43, no. 15, pp. 939–948, Dec. 2001.

[98] M. Jørgensen and D.I.K. Sjøberg, "Generalization and theory building in software engineering research," *J. Systems and Software*, vol. 70, no. 1–2, pp. 37–60, 2004.

[99] M. Jørgensen and D.I.K. Sjøberg, "The impact of customer expectation on software development effort estimates," *J. Project Management*, vol. 22, no. 4, 2004.

[100] M. Jørgensen, D.I.K. Sjøberg, and U. Indahl, "Effort estimation: Software effort estimation by analogy and "regression toward the mean"," *J. Systems and Software*, vol. 68, no. 3, pp. 253–262, 2003.

[101] M. Jørgensen, K.H. Teigen, and K.J. Moløkken-Østvold, "Better sure than safe? over-confidence in judgement based software development effort prediction intervals," *J. Systems and Software*, vol. 70, no. 1–2, pp. 79–93, 2004.

[102] M. Jørgensen, "Individual differences in how much people are affected by irrelevant and misleading information," in *2nd European Conf. Cognitive Science*. Hellenic Cognitive Science Society, 2007, pp. 347–352.

[103] D. Kahneman and S. Frederick, "A model of heuristic judgment," in *The Cambridge Handbook of Thinking and Reasoning*, K.J. Holyoak and R.G. Morrison, Eds. Cambridge Univ. Press, 2004, pp. 267–294.

[104] A.R. Kearney and S. Kaplan, "Toward a methodology for the measurement of knowledge structures of ordinary people: The Conceptual Content Cognitive Map (3CM)," *Environment and Behavior*, vol. 29, no. 5, pp. 579–617, Sept. 1997.

[105] G. Klein, "Developing expertise in decision making," *Thinking & Reasoning*, vol. 3, no. 4, pp. 337–352, 1997.

[106] M.J. Kolkman, N. Kok, and A. van der Veen, "Mental model mapping as a new tool to analyse the use of information in decision-making in integrated water management," *Physics and Chemistry of the Earth*, vol. 30, pp. 317–332, 2005.

[107] L.L. Levesque, J.M. Wilson, and D.R. Wholey, "Cognitive divergence and shared mental models in software development project teams," *J. Organizational Behavior*, vol. 22, pp. 135–144, 2001.

[108] J. Li, G. Ruhe, A. Al-Emran, and M.M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Software Engineering*, vol. 12, no. 1, pp. 1382–3256, 2007.

[109] R.A. Marshall, "The contribution of earned value management to project success on contracted efforts," *J. Contract Management*, pp. 21–33, summer 2007.

[110] S. McConell, *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006.

[111] P.E. Meehl, *Clinical versus Statistical Prediction: A Theoretical Analysis and a Review of the Evidence*. University of Minnesota Press, 1954.

[112] K. Moløkken-Østvold and M. Jørgensen, "A review of surveys on software effort estimation," in *Proc. Int'l Symp. Empirical Software Engineering (ISESE)*. IEEE Computer Society, 2003, pp. 223–230.

[113] M.G. Morgan, B. Fischhoff, A. Bostrom, and C.J. Atman, *Risk Communication: A Mental Models Approach*. Cambridge Univ. Press, 2002.

[114] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the feasibility of a case-based reasoning model for software effort estimation," *Management Information Systems Quarterly*, vol. 16, no. 2, pp. 155–171, June 1992.

[115] A.H. Murphy, "Skill scores based on the mean square error and their relationships to the correlation coefficient," *Monthly Weather Review*, vol. 116, pp. 2417–2424, 1988.

[116] T. Mussweiler, "Comparison processes in social judgment: Mechanisms and consequences," *Psych. Review*, vol. 110, no. 3, pp. 472–489, 2003.

[117] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: An empirical case study," in *Proc. 30th Int'l Conf. Software Engineering (ICSE)*. ACM Press, 2008, pp. 521–530.

[118] D.A. Norman, "Some observations on mental models," in *Mental Models*, D. Gentner and A.L. Stevens, Eds. Lawrence Erlbaum Associates, Inc., 1983, ch. 1, pp. 7–14.

[119] J.D. Novak and D.B. Gowin, *Learning how to Learn*. Cambridge Univ. Press, 1984.

[120] M. Poppendieck and T. Poppendieck, *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006.

[121] M.J. Prietula, S.S. Vicinanza, and T. Mukhopadhyay, "Software-effort estimation with a case-based reasoner," *J. Experimental and Theoretical Artificial Intelligence*, vol. 8, pp. 341–363, 1996.

[122] "PS2000 standard contract for iterative development," www.dataforeningen.no/index.php?cat=134112, accessed 2010.

[123] S. Ramanujan, R.W. Scamell, and J.R. Shah, "An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort," *J. Systems and Software*, vol. 54, no. 2, pp. 137–157, Oct. 2000.

[124] F. Ravary, E. Lecoutey, G. Kaminski, N. Châline, and P. Jaisson, "Individual experience alone can generate lasting division of labor in ants," *Current Biology*, vol. 17, no. 15, pp. 1308–1312, 2007.

[125] W. Reitman, *Cognition and Thought*. Wiley, 1965.

[126] H. Rognerud and J.E. Hannay, "Challenges in enterprise software integration: An industrial study using repertory grids," in *Proc. 3rd Int'l Symp.Empirical Software Engineering and Measurement (ESEM)*. IEEE Computer Society, 2009, pp. 11–22.

[127] W.C. Salmon, "Four decades of scientific explanation," in *Scientific Explanation*, ser. Minnesota Studies in the Philosophy of Science, P. Kitcher and W.C. Salmon, Eds. Minnesota Press, 1989, vol. XIII, pp. 3–219.

[128] F.L. Schmidt and J.E. Hunter, "The validity and utility of selection methods in personnel psychology: Practical and theoretical implications of 85 years of research findings," *Psychological Bulletin*, vol. 124, no. 2, pp. 262–274, 1998.

[129] F.L. Schmidt, J.E. Hunter, and A.N. Outerbridge, "Impact of job experience and ability on job knowledge, work sample performance, and supervisory ratings of job performance," *J. Applied Psychology*, vol. 71, no. 3, pp. 432–439, 1986.

[130] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.

[131] W.R. Shadish, T.D. Cook, and D.T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.

[132] T.M. Shaft and I. Vessey, "The relevance of application domain knowledge," *J. Management Information Systems*, vol. 15, no. 1, pp. 51—78, 1998.

[133] J. Shanteau, "Competence in experts: The role of task characteristics," *Organizational Behavior and Human Decision Processes*, vol. 53, pp. 252–266, 1992.

[134] H.A. Simon, *The New Science of Management*. Harper & Row, 1960.

[135] H.A. Simon, "The structure of ill-structured problems," *Artificial Intelligence*, vol. 4, pp. 181–201, 1973.

[136] H.A. Simon, *The Sciences of the Artificial*, 3rd ed. MIT Press, 1996.

[137] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanović, N.K. Liborg, and A.C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Trans. Software Eng.*, vol. 31, no. 9, pp. 733–753, Sept. 2005.

[138] M. Sliger and S. Broderick, *The Software Project Manager's Bridge to Agility*. Addison Wesley, 2008.

[139] S. Sonnentag, "Expertise in professional software design," *J. Applied Psychology*, vol. 83, no. 5, pp. 703–715, 1998.

[140] E. Stensrud and I. Myrtveit, "Human performance estimating with analogy and regression models: An empirical validation," in *METRICS. 5th Symposium, March 20–21, 1998*, 1998, pp. 205–213.

[141] R.J. Sternberg, "Cognitive conceptions of expertise," *Int'l J. Expert Systems*, vol. 7, no. 1, pp. 1–12, 1994.

[142] R.J. Sternberg, "Intelligence," in *The Cambridge Handbook of Thinking and Reasoning*, K.J. Holyoak and R.J. Sternberg, Eds. Cambridge Univ. Press, 2005, ch. 31, pp. 751–774.

[143] T.R. Stewart, "A decomposition of the correlation coefficient and its use in analyzing forecasting skill," *Weather and Forecasting*, vol. 5, pp. 661–666, 1990.

[144] T.R. Stewart, "Improving reliability of judgmental forecasts," in *Principles of Forecasting: A Handbook for Researchers and Practitioners*, J.S. Armstrong, Ed. Kluwer Academic Publishers, 2001, pp. 81–106.

[145] T.R. Stewart and C.M. Lusk, "Seven components of judgmental forecasting skill: Implications for research and the improvement of forecasts," *J. Forecasting*, vol. 13, no. 7, pp. 579–599, 1994.

[146] F. Strack and T. Mussweiler, "Explaining the enigmatic anchoring effect: Mechanisms of selective accessibility," *J. Personality and Social Psychology*, vol. 73, no. 3, pp. 437–446, 1997.

[147] I. Stuart and D.F. Prawitt, "The influence of audit structure on auditors' performance in high and low complexity task settings," June 2004, available at Social Science Research Network ssrn.com/abstract=569871.

[148] T. Sulaiman, B. Barton, and T. Blackburn, "AgileEVM— earned value management in scrum projects," in *Proc. IEEE AGILE 2006*. IEEE Computer Society, 2006, pp. 7–16.

[149] S.S. Vicinanza, T. Mukhopadhyay, and M.J. Prietula, "Software-effort estimation: An exploratory study of expert performance," *Information Systems Research*, vol. 2, no. 4, pp. 243–262, 1991.

[150] J.F. Voss and T.A. Post, "On the solving of ill-structured problems," in *The Nature of Expertise*, M.T.H. Chi, R. Glaser, and M.J. Farr, Eds. Lawrence Erlbaum Associates, Inc., 1988, pp. 261–285.

[151] W.W. Wittmann, "Brunswik-symmetry: A key concept for successful assessment in education and elsewhere," 2004, paper presented at the 4th Spearman Conference, Philadelphia.

[152] W.W. Wittmann and H.M. Süß, "Investigating the paths between working memory, intelligence, knowledge, and complex problem solving performances via Brunswik symmetry," in *Learning and Individual Differences: Process, Traits, and Content Determinants*, P.L. Ackerman, P.C. Kyllonen, and R.D. Roberts, Eds. American Psychological Association, 1999, pp. 77–108.

[153] R.E. Wood, "Task complexity: Definition of the construct," *Behaviour and Human Decision Processes*, vol. 37, pp. 60–82, 1986.