Causal Mechanism Graph

— A New Notation for Capturing Cause-effect Knowledge in Software Dependability

Fuqun Huang^{*}, Carol Smidts

Department of Mechanical and Aerospace Engineering, The Ohio State University

Columbus, OH 43210, U.S.A

^{*}huang.2315@osu.edu; smidts.1@osu.edu

Abstract: Understanding cause-effect relations between concepts in software dependability engineering is fundamental to various research or industrial activities. Cognitive maps are traditionally used to elicit and represent such knowledge; however they seem incapable of accurately representing complex causal mechanisms in dependability engineering. This paper proposes a new notation called Causal Mechanism Graph (CMG) to elicit and represent the cause-effect domain knowledge embedded in experts' minds or described in the literature. CMG contains a new set of symbols elicited from domain experts to capture the recurring interaction mechanisms between multiple concepts in software dependability engineering. Furthermore, compared to major existing graphic methods, CMG is particularly robust and suitable for mental knowledge elicitation: it allows one to represent the full range of cause-effect knowledge, accurately or fuzzily as one sees fit depending on the depth of knowledge he/she has. This feature combined with excellent reliability and validity poses CMG as a promising method that has the potential to be used in various areas, such as software dependability requirement elicitation, software dependability assessment and dependability risk control.

Keywords: causal mechanism; software dependability; causal mechanism graph; dependability assessment; expert opinion elicitation; cognitive map

1. Introduction

Eliciting and representing cause-effect knowledge in the software dependability domain is significant for the assessment and control of dependability risks. With the increasing dependence on software and the accompanied increased software complexity, software dependability has become the major determinant of the systems' trustworthiness for many safety-critical systems. Software dependability denotes the contribution of the software to the computer-based systems' properties that allow us to trust the system [1]. These properties include such as reliability, safety, security, availability and maintainability [1-3] and different attributes can be emphasized in different contexts or by different stakeholders. Understanding the causal mechanisms [4], the causal factors and the interactions between the factors, underlying software dependability is therefore essential for software practitioners [5, 6]. For instance, causal mechanisms can help design better assessment models and risk control approaches because causal mechanisms capture the causal entities that determine software dependability.

Domain experts are generally considered to possess the most abundant cause-effect knowledge and expert opinion elicitation is a direct way to extract such latent domain knowledge. In software dependability engineering, expert judgment is widely used to obtain information for software dependability assessment [7-11]. There are a variety of graphic methods used in the software dependability domain, such as Bayesian Networks [7, 10, 12, 13], Fault Tree [14] and UML [15]. However, most of these methods (e.g. Fault Trees and UML) focus on representing the accurate relations between components or states of a system; they are not suitable for expressing fuzzy conceptual relations

existing in one's cognitive mental model under uncertainty. The few methods used to elicit experts' mental knowledge such as Bayesian Networks and Cognitive Maps [16] do not provide a sufficient set of notations to represent comprehensive interaction mechanisms between concepts that may exist in the software dependability domain, e.g. Bayesian Networks only allow expression of conditional dependencies and Cognitive maps only allow expressing "influence" relations. A robust notation is required to elicit experts' full range of cause-effect knowledge, accurately or fuzzily as one sees fit depending on the depth of knowledge he/she has. Hence, designing a new notation for capturing domain experts' cause-effect knowledge is a worthy topic of exploration.

This paper provides a new notation, called the *Causal Mechanism Graph (CMG)*, to elicit experts' causal mechanism knowledge in software dependability. Section 2 reviews relevant existing methods. Section 3 describes the design process of the CMG notation. Section 4 presents the CMG notation. Section 5 presents the evaluation of the CMG notation. In Section 6 we discuss the implications of CMG to the research and industrial activities in the software dependability domain and the future work. Section 6 provides the conclusion.

2. Related work

Bayesian Networks are currently widely used to represent expert judgment on risk assessment [7, 10, 12, 13], e.g., dependability assessment [7] and defect prediction [17]. A Bayesian Network (BN) describes the directed dependencies (directed edges) between random variables (nodes). In a Bayesian Network, strength is represented by a conditional probability of occurrence of the dependence as a function of the state of the nodes. Bayesian Networks help people make inferences on the occurrence probability of one event based on its dependence on another event. Fuzzy Bayesian inference was proposed to generalize Bayesian Statistics to fuzzy data [18]. BN is a powerful method for probabilistic inference in domains where expert knowledge is uncertain, ambiguous, or incomplete. However, probabilistic dependencies do not imply causality. Although Causal Bayesian Networks are introduced to represent causal relations by restricting additional semantics [19], the basic notations only contain nodes and edges. BN does not contain the notations required to explicitly capture the real interaction mechanisms between various entities existing in the software dependability domain.

Cognitive maps or causal maps [16, 20-25] are another method used to elicit cause-effect knowledge embedded in human minds. A causal map consists of three elements: nodes (used to represent concepts), edges (the direction of edges denoted by arrows indicates believed causality) and 'influence relationships' (positive or negative) with strengths. Causal maps can be created either directly or indirectly. Previously existing methods generally concentrate on direct elicitation in which causal maps are produced by participants directly, e.g., through ideographic, nomothetic, and hybrid methods [26]. Direct elicitation exempts researchers from interpretation and coding, which inevitably introduces researcher's bias. However, indirect elicitation, which extracts causal maps from documentary sources, is more meaningful in terms of allowing individuals to express thoughts using natural language [26, 27] and therefore use their full range of expression. Indirect elicitation is also non-intrusive to the experts' thinking process. These advantages allow the investigator to gain access to busy individuals who might otherwise be unwilling to participate in more intrusive, interactive procedures. In spite of its numerous applications, causal maps contain limited notations to represent relations between variables. In fact, the only relation contained in a causal map is "influence". Causal maps do not contain logical combinations between various variables, time sequences, and other interaction mechanisms. Thus unfortunately, causal maps are not capable of accurately representing the complex causal mechanisms that exist in engineering domains.

Methods exist in software dependability to accurately model the relations or behaviors of technical systems. Fault Tree [14] analysis is a deductive analysis method for root cause analysis. A fault tree is developed in a top-down way to identify the “basic events” that cause a system failure. Logic notations such as AND, OR, and different types of event notations are provided to construct a fault tree. In a fault tree analysis, a top event is considered as the effect of a set of basic events combined by Boolean logic. Event Tree analysis is performed in an inductive way to analyze the consequences of the primary event of concern using logic notations. Failure mode and effects analysis (FMEA) and its extension method Failure mode effect and criticality analysis (FMECA) [28] are also inductive analyses, in which the causes and effects are enumerated for each failure mode of concern. Fault tree, Event tree and FMEA have no explicit representation on the time element, which is an essential element of causality, whereas Petri nets [29] are used to describe system states and states’ transitions over time. Event Sequence Diagram (ESD) [30] was used to describe dynamic behaviors for probabilistic risk assessment. ESD includes notations to describe time delays and conditions in addition to different types of logic and event notations. These methods are sometimes combined to overcome the shortage of a method in different application contexts [31-33]. However, these methods collectively focus on the relations between components or states in a system, thus corresponding analysis requires the analyst to know a system or component thoroughly. These methods do not include fuzzy conceptual relations (e.g., correlations and contributory causal relations) existing in one’s cognitive mental model under uncertainty. For instance, an expert knows that “program complexity” can be a contributory factor to “faults”, but that does not mean that “complexity” will produce “faults”; how “complexity” interacting with other factors (e.g. the task contexts and programmer’s schemata) [34] leads to “faults” is unknown to the expert. These system-oriented methods do not seem suitable for the elicitation of one’s cognitive knowledge which can be fuzzy, intuitive and/or abstract by nature.

In summary, each of the above methods has its advantages and disadvantages in different application contexts. Considering the various gaps in knowledge of previously used techniques, a set of notations that allows practitioners to straightforwardly represent experts’ knowledge of causal mechanisms in software dependability engineering is required.

3. The design process of the CMG notation

The design process used to develop the CMG notation was based on the Grounded Theory [35]. Grounded Theory was chosen to guide the CMG design process for its three essential principles [36, 37] that are particularly suitable for building new theories. These principles are: using an *inductive* and *iterative* process to generate a new theory from *empirical data*. Deductive research seeks to verify hypotheses that are derived from an existing or pre-conceived theory. By contrast, an inductive process seeks to identify categories and relations from empirical data, and thus build a theory to account for a phenomenon. Developing a new theory is also an interactive process, in which the researchers continuously assimilate and integrate new information into the existing mental model. Thus, the theory evolves during the research process itself. The inductive and iterative features of grounded theory conform well to the cognitive mechanisms underlying humans’ design activities [38, 39]. Therefore, Grounded Theory is widely used to explore new knowledge or construct new theories in various domains, including software engineering [40, 41].

The CMG notation is designed using an inductive and interactive process based on data provided by domain experts. We first designed semi-structured questionnaires to obtain the experts’ opinion on software dependability causal mechanisms. Then, the data were iteratively analyzed to discover the main concept categories and relation patterns contained in the answers to the questionnaires. While analyzing the data, the researchers simultaneously designed

the new notations in order to represent the patterns in the data. This process proceeded iteratively until Theoretical Saturation was reached--no new notations were required to represent the information of interest in the data. Finally, using the designed notations, we converted the questionnaire responses to causal mechanism graphs and verified them with the corresponding experts.

3.1 Data collection

We designed semi-structured questionnaires to obtain the experts' opinion on software dependability causal mechanisms. Semi-structured questionnaires are comprised of a mixture of closed (used in structured questionnaires) and open questions (used in unstructured questionnaires). The use of semi-structured questionnaires has the mixed advantages of structured questionnaires and unstructured questionnaires. Such questionnaires can enable exploration of new territories as well as gathering the information that the researcher requires. The questionnaires are "structured" at the high level to ensure that we obtain the necessary variables for describing the cause-effect mechanisms. We have identified six categories of concepts: root causes; intermediate variables; scenarios (assumptions/conditions/circumstances); dependability attributes (i.e. reliability, safety, security, availability and maintainability) and dependability. These categories are designed to prime the experts' thinking process, as well as to bring dependability causal mechanisms into the spotlight for their discussion. Conversely, individual questions are designed as open questions in order to collect a broad range of information.

Since we were concerned about the causal mechanisms that shape software dependability as a general concept, which encompasses the dependencies between various attributes and the causal mechanisms that influence each software dependability attribute, two types of questionnaires were designed: Type I focuses on the relations between two dependability attributes (Question I.1, I.2, II.8-11 in Table 8) and the causal mechanisms for each attribute (Question II.2-7 in Table 8); Type II focuses on the relation between one attribute to another attribute of software dependability (Questions in Part I in Table 9) and the causal mechanisms of a software dependability attribute (Questions in Part II in Table 9). Certain questions are inherently redundant, such as the questions on common causes (Question II.5 in Table 8) and those on specific causes (Question II.6 and II.7 in Table 8). The point here is to verify consistency in the experts' answers. We present the questions on high-level dependencies first, then the questions on the detailed causal factors for each attribute, as it follows the cognitive mechanism of humans retrieving their knowledge (schemata) [42].

The questionnaires were then sent to the experts that had been identified through Google Scholar from May 8th to May 27th 2014. These experts had published at least 5 papers in the paired fields, e.g. "Reliability and Maintainability" or "Dependability and Safety". Since Dependability is considered as a super-set concept that encompasses various attributes, the experts selected in this way are believed to know at least two dependability attributes. We obtained 14 responses from 11 domain experts, among whom three experts answered two questionnaires as they qualified for more than "three dependability attributes" as well as had the willingness to contribute. Since the questionnaires were designed to contain significant redundancies, each attribute was covered by multiple experts, as shown Table 1.

Table 1. Expert opinion coverage

Software dependability and attributes	Covered by # Experts
Software dependability	8
Software reliability	7
Software safety	5

Software security	5
Software availability	5
Software maintainability	4

3.2 Data analysis

The expert answers to the questionnaires, expressed in natural language, constitute the data for designing the CMG notation based on Grounded Theory. The Grounded Theory provides various strategies such as qualitative coding, constant comparison and memoing to identify patterns contained in the data. However, such strategies tend to be very general and the difficulty lies in how to alternatively apply such general strategies in a specific design context to develop a new theory. How the referred strategies were applied in the CMG notation design process is discussed as follows.

Qualitative Coding includes the process of Open Coding to identify the major categories of concepts that can later be used to build the theory, and the process of Axial Coding to strategically relate codes (categories and concepts) to each other [43]. A sample of the concepts and relations extracted during the open coding is shown in Table 2. An example result of axial coding is the notation $\langle \text{arg1}, \text{arg2}, \text{arg3} \rangle$ in Table 3. “Scenario” and “relation type” are included as the two elements of a “constraint”, since “scenario” is always coupled with the type of “influence” (i.e. positive or negative).

Table 2. A sample of the extracted concepts and relations

Relations/Concepts Categories	Common expressions
Influence	Cause, depend on, influence, affect, effect
Scenario	When....., circumstance, situation, context, in....case, given....., if
Conflict	not appropriate for, not satisfy, not meet.
Trigger/Activate	Trigger, activate, provoke, initiate

Constant comparison is the strategy of comparing the new arising codes against the codes from the same questionnaire, and those from other questionnaires [37]. When a new code emerges in front of the researcher, he/she checks related discussions in the same questionnaire and the corresponding places in other questionnaires. This constant comparison is helpful for the researcher to ensure that the generated code is accurate and representative for describing the patterns contained in the questionnaires.

Memoing is the ongoing process of making notes on ideas and questions that occur to the analyst during the process of data analysis. The researcher highlights the key words in different colors as she reads the questionnaires. The researcher also makes notes on the important inferences. These are essential cognitive processes used in building a coherent mental model during one’s text comprehension [44]. The mental model, which is also called situation model, is the model formed in a reader’s mind to represent the situations in the real world, based on the information she obtains from materials like texts [39, 44]. An example memo is shown in Figure 1. The researcher also developed other types of memos during the data analysis--the symbols that are used to represent the patterns discovered in the data. Once the researcher identifies a basic pattern, she designs a symbol to represent the pattern, e.g. using “⊗” to represent the relation “conflict”, shown in Table 3.

Figure 1. An example of notes made during data analysis

3.3 Theoretical refinement and integration

The last stage of the design process is to integrate the drafted notations into a systematic notation system. The experts’ discussion in the questionnaires contains information on cause-effect patterns. We call such patterns “pattern slices”, as they are fragmented patterns expressed by text segments scattered in different places throughout a questionnaire. Experts’ inputs contain no information about how to assemble pattern slices into a coherent graph that can represent the whole mental model embedded in a questionnaire.

A systematic notation system that enables researchers to assemble cause-effect patterns discovered at different places (e.g., discussions under different questions in a questionnaire of this study, or ideally any texts such as academic papers) appeals to us. To achieve this aim, the notation system should contain a set of connection interfaces where users can assemble different notations to construct a large CMG. The draft notations are refined throughout the process of converting each expert response into an individual causal mechanism graph. For instance, an arrow is added to the notation “activate” to indicate the entity that initiates the action of “activation”, shown in Table 3. A bold dot is added to indicate the place of “connection”; while “connection” indicates a relation is present, shown in Table 3. Arrows are added to the directed relations, e.g., notations of “imply”, “cause”, “activate” and “conflict”. Furthermore, letters such as “a₁”, “a₂”, “b” and “S” are used to denote the connection interfaces. The refinement continues until the notation system reaches theoretical saturation.

The theoretical construction process goes further, accurately describing the patterns captured by the notations. Due to the ambiguous nature of natural language, readers or users may fail to fully understand the essence of the notations described in Table 3. Mathematical interpretations are specified for the relational notations, and are shown in Table 4.

4. The CMG notation system

4.1 The basic notation

The basic notations used to represent causal mechanisms are described in Table 3. The notations contain concepts and relationships. Concepts are of three types: root factors, intermediate variables/events and top variables/events. Relationships consist of five groups: logical connections, membership, property, constraints and interactions. The

group of logical connections includes “AND”, “OR” and “NOT”. The membership group contains “Subset” and “Element”. The property group solely includes “Property”. The interactions group includes “Influence”, “Imply”, “Trigger” and “Conflict”. Moreover, constraints are used to restrict the scope of the interaction.

A comparison between CMG notations and selected competitive notation systems is also provided in Table 3. The selected competitive notation systems include Causal Map, Fault Tree, Event tree, and Bayesian Networks. These methods are selected based on their relevance to cause-effect knowledge representation and the popularity of application in software dependability engineering. The Causal Map is a conventional method, widely used to represent humans’ cognitive knowledge on cause-effect relations, and is the first competitive notation system as it shares the same aim as the CMG. Fault Tree, Event tree, and Bayesian Networks are widely used in software dependability engineering relevant to cause-effect relations; though these methods are not necessarily focused on representing humans’ cognitive knowledge. Compared to the competitive notation systems, the CMG notation system has the following features:

- We differentiate causality and correlation, because correlation does not necessarily imply causality. Causality refers to the relationship between an event and a second event, where the second event is the consequence of the first. The causal factors in this paper include necessary, sufficient and contributory causes (Riegelman 1979) (called “influence shaping factor” in this paper). Correlation in turn refers to a broad relation involving dependency. Time is an essential dimension of Causality (Granger 1980; Pearl 2009), while it is not necessarily the case in correlation. Examples of causality and correlation are provided in Figure 2.
- We introduce the term “scenario” to refer to the assumptions/conditions/circumstances/contexts under which a relation exists. “Scenario” is particularly useful for describing correlations formed by a set of common causal factors. The correlation type (positive or negative) may vary under different circumstances. For example, software safety can be positively correlated with software reliability under the scenario that “higher software reliability implies lower defect rate in general”, which in turn implies lower safety-related defect rate, yielding higher safety. Also software safety can be negatively correlated with software reliability under the scenario that “safety design increases the complexity of a software system thus increases software defect rate”.
- “Trigger/Activate” is introduced to represent a relationship between the presence or occurrence of one entity and how that leads to the occurrence/presence of another entity. “Trigger/Activate” contains the subtle meaning of time sequence that is unable to be represented by common logical symbols (e.g. OR and AND). When we say A AND B cause C, it means: (1) A and B occur concurrently or that the timing does not matter, (2) A and B both precede C. When we say B triggering A causes C, it represents a set of propositions: (1) A is a necessary cause of C, (2) A precedes B, (3) A produces the effect of C only when B happens, and (4) A and B both precede C. So, “Trigger/Activate” is a powerful abstraction to represent many relations commonly existing in software dependability, e.g., operating environments trigger software defects leading to software failures, intrusions exploit vulnerabilities leading to information exposure.
- “Conflict” is designed to represent two factors that *together* produce an effect, where these two factors have different types of influences (e.g., positive versus negative, positive versus neutral or both neutral), and the effect is present only when these two factors are coupled. For instance, one cannot say the architecture of a software system has a positive or a negative influence on maintenance failure without referring to maintenance requirements. Only when the adaptive maintenance requirements are incompatible with the existing architecture, will a maintenance failure ensue. In a conventional causal map,

“maintenance requirements” and “architecture” are two neutral variables, which may be excluded from the list of causal factors. “Conflict” allows us to capture such variables and relations more accurately--when two neutral events collide with each other, they can produce significant new effects.

Figure 2. Examples of causality and correlation

Table 3. The basic notations for causal mechanism graphs

Symbol	Name	Description	Whether contained in other notation systems			
			Causal Map	Fault Tree	Event Tree	Bayesian Networks
Connection	Two entities (say a_1 and a_2) are connected to the entity b , with no information on the connection type. The type of connection is specified by other symbols, i.e. NOT, Subset, Element and Property. “•” denotes the place where the connection is presented.	Implicit	Implicit	Implicit	Implicit
AND	Entity a_1 AND entity a_2 form entity b .	×	√	√	×
OR	Entity a_1 OR entity a_2 form entity b .	×	√	√	×
NOT	Entity a_2 is the complement of entity a_1 at the location b . “•” denotes the place where the connection is presented.	×	Implicit	√	×
Subset	A set a_1 is a subset of a set a_2 , that is, all elements of a_1 are also elements of a_2 . “•” denotes the place where the connection ends, i.e., a_2 around the “•” is the set, while a_1 is the subset	×	Implicit	Implicit	×
Element	An element a_1 is a singleton of the distinct objects that make up that set, S . “•” denotes the place where the connection ends.	×	Implicit	Implicit	×
Property	A property a_1 is special quality or characteristic of an entity, S . “•” denotes the place where the connection ends.	×	×	×	×
Cause	Influence describes the causal relations between two entities. The	√	√	√	Indiscriminate

		strength of the “influence” may vary according to different types of causes (say a_1). For example, when a_1 is a sufficient cause of a_2 , $\rightarrow$ represents “cause” or “lead to” in natural language; when a_1 is a necessary cause of a_2 $\rightarrow$ represents “influence”; a_1 is a contributory cause of a_2 $\rightarrow$ may interpret words such as “affect” or “shape”. Such differences are determined and thus represented by other elements, e.g., <i>strength</i> on the edge. The essence of the symbol $\rightarrow$ is causality.				different types of “dependency”
$a_1 - - - \rightarrow a_2$	Imply	Directed implication, involving a broad class of dependencies except for causality. When one variable implies another variable, it means dependency exists between the two variables (say a_1 implies a_2). Such dependency allows one to make inference about one variable according to another variable. “Imply” is not direct causality, but variables involved in this relation may share some common causal factors.	×	×	×	
< arg1, arg2, arg3 >	Constraints	Describes the scenario for which a relation is present. Arg1 represents the scenario, while arg2 represents the type of the relation. There are three types of relations: positive relation represented by “+”, negative relation represented by “-”, and undefined or neutral relation left as blank. Arg3 describes the strength of the relation by a number. The range of the strength is	×	“Inhibit gate” has the implicit meaning of “scenario”, but it is not used combined with relation types and strength.	√	×

		determined by the application context, e.g., “2” is supposed to be a stronger relation than “1” in a context of a 3-point scale weighting scheme.				
+	Positive relation	A “positive” relation is said to exist from a_1 to a_2 when an increase in a_1 leads to an increase in a_2 . It is one of the values taken by the constraint arg2. Weights are also allowed to be attached to the symbol “+” to describe the strength of the relation.	√	Implicit	Implicit	√
-	Negative relation	A “negative” relation is said to exist from a_1 to a_2 when an increase in a_1 leads to a decrease in a_2 . It is one of the values taken by the constraint arg2. Weights are also allowed to be attached to the symbol “-” to describe the strength of the relation.	√	Implicit	Implicit	√
Conflict	Effect b is present when a_1 is in conflict with a_2 . “Conflict” is designed to represent two factors that <i>together</i> produce an effect, where these two factors have different types of influences (e.g., positive versus negative, positive versus neutral or both neutral), and the effect is present only when these two factors are coupled.	×	×	×	×
Trigger	Effect b is present after event a_2 Triggers or Activates event a_1 . “Trigger” contains the subtle meaning of time sequence that is unable to be represented by common logical symbols. When we say a_2 triggering a_1 causes b , it represents a set of propositions: (1)	×	“Inhibit gate” has implicit meaning of “trigger”.	Implicit	×

		a_1 is a necessary cause of b , (2) a_1 precedes a_2 , (3) a_1 produces the effect of b only when a_2 happens, and (4) a_1 and a_2 both precede b .
Root factors*	The basic factor that affects an event, which could be factors such as sufficient causes, necessary causes and contributory causes (influence shaping factors).	Not able to discriminate	√	√	Not able to discriminate
Intermediate variable/event*	The variable or event that is required to transform the root causes into issues experienced with software dependability attributes.		√	√
Top variable/event*	The variable/event whose causal mechanism is being investigated.		√	√	

* A dashed symbol (e.g., in Figure 4) indicates that the concept is identical to the corresponding concept drawn with a solid line in the same CMG. Such dashed symbols are used to achieve a clearer layout of a CMG.

**A transfer symbol “ ” can be used to split one CMG into several segments when a CMG becomes very large.

4.2 Mathematical interpretations of the CMG notation

All of the relational notations in Table 3 have mathematical interpretations, which are shown in Table 4. A notation contains a set of mathematical constraints on the involved variables. Such constraints specify the quantitative dependencies between the involved variables.

Table 4. Mathematical constraints of the CMG notations

Symbols	Name	Mathematical Constraints
AND	$b = a_1 \cap a_2$
OR	$b = a_1 \cup a_2$
NOT	$a_2 = \bar{a_1}$
Subset	$a_1 \subseteq a_2$
Element	$a_1 \in S$
Cause	<p>Suppose a_1 occurring at the time point t_1, a_2 occurring at the time point t_2, a_1 is said to be a cause of a_2 if and only if [45]:</p> <ul style="list-style-type: none"> (i). $t_1 < t_2$; (ii). $\text{Pr ob}(a_1(t_1)) > 0$; (iii). $\text{Pr ob}(a_2(t_2) a_1(t_1)) > \text{Pr ob}(a_2(t_2))$.
Imply	<p>(1) Positive implication, $a_1 \xrightarrow{+} a_2$:</p> <ul style="list-style-type: none"> (i). $\text{cov}(a_1, a_2) > 0$
		<p>(2) Negative implication, $a_1 \xrightarrow{-} a_2$:</p> <ul style="list-style-type: none"> (i). $\text{cov}(a_1, a_2) < 0$
S	Scenario	<p>(1) Scenario on a causality link, $a_1 \xrightarrow{S} a_2$:</p> <p>Suppose a_1 occurring at the time point t_1, a_2 occurring at the time point t_2, the following rules exist concurrently:</p> <ul style="list-style-type: none"> (i). $t_1 < t_2$; (ii). $\text{Pr ob}(a_1(t_1)) > 0$; (iii). $\text{Pr ob}(a_2(t_2) a_1(t_1)) > \text{Pr ob}(a_2(t_2))$; (iv). $\text{Pr ob}(a_2(t_2) a_1(t_1) S) > \text{Pr ob}(a_2(t_2) a_1(t_1))$.

		<p>(2) Scenario on an implication link, $a_1 \xrightarrow{S_1, \pm} a_2$:</p> <p>(i). $cov(a_1, a_2 S) > cov(a_1, a_2)$</p>
Conflict	<p>Suppose a_1 occurring at the time point t_1, a_2 occurring at the time point t_2, b occurring at the time point t_3, the following rules exist concurrently:</p> <p>(i). $t_1 < t_3, t_2 < t_3$;</p> <p>(ii). $Pr ob(b(t_3) a_2(t_2) \bullet a_1(t_1)) > Pr ob(b(t_3))$;</p> <p>(iii). $sgn(\frac{\partial b}{\partial a_1}) \neq sgn(\frac{\partial b}{\partial a_2})$.</p>
Trigger	<p>Suppose a_1 occurring at the time point t_1, a_2 occurring at the time point t_2, b occurring at the time point t_3, the following rules exist concurrently:</p> <p>(i). $t_1 < t_2 < t_3$;</p> <p>(ii). $Pr ob(a_1(t_1)) > 0$; $Pr ob(a_2(t_2)) > 0$;</p> <p>(iii). $Pr ob(b(t_3) a_2(t_2) \bullet a_1(t_1)) > Pr ob(b(t_3))$;</p> <p>(iv). $Pr ob(b(t_3) \overline{a_2(t_2)}) = 0$.</p>

4.3 Causal Mechanism Graph

The symbols in Table 3 are the basic elements used to construct a causal mechanism graph. These symbols can be freely combined to form a powerful function to represent causal mechanisms. A sample of the combination of basic symbols is shown in Figure 3.

Figure 3. Examples of the combination of basic symbols

At the macro level, the CMG is organized in a hierarchical structure. Directed arrows are used to reflect the direction of cause-effect flows: variables at the beginning of a directed symbol are the precedents (causes) of the variable at the end of the same directed symbol (effect). For instance, the diagram at the right side of Figure 3 contains two causality flow paths: Path 1, event b causes event a_1 , which implies that event b occurs before event a_1 ; Path 2, event b triggers event a_2 then causes event a_3 , which implies that event b occurs before event a_2 , and event a_2 occurs before event a_3 . Variables with a logical and membership relation are at the same logical abstraction level in a CMG, for instance, in the left side of Figure 3, a_1 and a_2 are at the same level. Furthermore, the top event symbol is used to

denote the terminal point of all the cause-effect flows in a CMG; the root cause symbol is used to identify a beginning point of a cause-effect flow in a CMG; all other variables and events involved between the beginning points and the top event are represented by the intermediate event/variable symbol. As the size of a CMG becomes extremely large, the transfer symbol “ ” which is the same as that used in fault trees can be used to split one CMG into several segments. A dashed symbol (e.g., in Figure 4) is used to indicate a concept that is identical to the corresponding concept drawn with a solid line in the same CMG. Once a concept is defined as identical, the causal structure that leads to it will not be repeated, so as to achieve a clearer layout of a CMG. These design features enable CMG to represent complex engineering causal mechanisms in a cause-effect network fashion that contains various interacting cause-effect paths. An example CMG segment is shown in Figure 4 and a completed CMG with top event is shown in Figure 6.

Figure 4. An example of causal mechanism graph representing the text segment in Figure 1

4.4 CMG representation capability

Based on the specific features of the CMG basic notations described in Section 3.1, a comparison at the graphical level is made between CMG and the selected competitive notation systems. As the focused application of CMG is to elicit and represent experts' mental knowledge on cause-effect relations in the software dependability domain, the comparison focuses on the robustness for human knowledge expression, capability to represent cause-effect domain knowledge, and the applicable levels of analysis. A set of comparison criteria is proposed for each of these concerns as follows:

Criterion A. Robustness for human knowledge expression. Since the CMG notations aim to represent causal mechanisms that are stored in experts' knowledge base, it is important to allow one to represent their cognitive knowledge flexibly. Due to the variety of their experience, one expert may have detailed causal-mechanism knowledge between two entities (e.g., the relation between software faults and failures), while only having an intuitive judgment on the relations between another two entities (e.g., the relations between software failures and reliability). The CMG notations system should be robust to capture the knowledge at different levels of details. This further requires that the CMG contains notations that allow one to:

- A1, fuzzily represent cause-effect relations when one's knowledge on the relation is incomplete, when the knowledge is one's intuitive judgment, or when one deliberately wants to focus on the abstract level;
- A2, accurately represent cause-effect relations when one's knowledge on the relation is accurate;
- A3, incorporate fuzzy and accurate knowledge in the same CMG.

The robustness for human knowledge expression pertains to the usability characteristics, which are in turn achieved by the capability of the CMG notation system in representing cause-effect knowledge in the software dependability domain (Criterion B).

Criterion B. Capability to represent cause-effect domain knowledge. This includes capability to capture correlations (B1), to capture causality (B2), capability to differentiate causality from implication (B3), capability to capture scenarios under which relations tend to be present (B4), capability to capture interaction mechanisms (B5), and capability to represent the relations hierarchically (B6).

Criterion C. Applicable levels of analysis. Cause-effect relations can exist between components in a technical system. Cause-effect relations can also exist between conceptual entities that are created by the researchers, e.g. software reliability, in order to understand and manage the objects in the real world, i.e. software system. A robust notation system should be capable of representing cause-effect relations at both system level (C1) and conceptual level (C2), as well as allow the flow between these two levels (C3).

The comparison results according to these criteria are summarized in Table 5. Compared to the currently widely used graphic methods, Causal Map, Fault Tree, Event Tree and Bayesian Networks, causal mechanism graphs are very suitable to elicit and represent experts' comprehensive cause-effect knowledge in software dependability engineering.

Table 5. Comparison between CMG and other widely used graphic methods

Comparing criteria			Causal Map	Fault Tree	Event Tree	Bayesian Networks	CMG
A. Robustness for human knowledge expression	A1	Fuzzy expression of cause-effect relations	√			√	√
	A2	Accurate expression of cause-effect relations		√	√		√
	A3	Incorporate fuzzy and accurate knowledge					√
B. Capability to represent cause-effect domain knowledge	B1	Capture Implication				√	√
	B2	Capture Causality	√	√	√	√	√
	B3	Differentiate causality from Implication					√
	B4	Capture scenarios under which relations tend to exist		√	√		√
	B5	Capture interaction mechanisms					√
	B6	Hierarchical representation		√	√		√
C. Applicable levels of analysis	C1	Applicable at system level		√	√		√
	C2	Applicable at conceptual level	√			√	√
	C3	Allow the cause-effect flow between system level and conceptual level		√			√

5. Evaluation of the CMG notation system

We evaluated the inter-coder reliability and the validity of CMG notation system. The inter-coder reliability [46] refers to the degree of consistency different interpreters can achieve when they represent the same textual material using the CMG notation system. The validity [47] denotes the extent to which the CMGs generated by the researchers portray the ideas embedded in the participants' text-based responses.

5.1 Inter-coder reliability evaluation

• Procedures

Two graduate students served as independent coders and constructed the CMGs, in addition to the researchers of this paper. The participants are considered representative, as they have diverse expertise levels: C1 (the second author of this paper) is a professor with 20 years of experience in the software dependability area, C2 (the first author of this paper) is a research scientist with 6 years of experience in this domain, C3 and C4 represent coders with little to no domain knowledge (C3 is a senior PhD student, while C4 is a junior PhD student).

The first author trained the interpreters on the notation system using an interactive presentation. The contents of the training included the framework of the approach, the strategies that can be employed in the interpretation and the meanings of the symbols that can be used to model causal mechanisms. Examples were used for practice. Discussions continued to make certain the other two interpreters had understood the approach and relevant elements.

Practice tasks were embedded in this training to enhance the interpreters' understanding of the approach as well as to ensure that the training was effective. At the beginning of the presentation, the text segment shown in Figure 1 and the notations in Table 3 were presented to the audience. The interpreters were required to draw a CMG to represent the expert's thoughts as accurately as possible based on what the expert expressed by the text segment.

After the training, the interpreters were required to draw a CMG again using the same material. Obvious evolutions were observed from simple two-level structures (e.g., Figure 5) to structures very close to the CMG in Figure 4.

Figure 5. An initial CMG drawn by an interpreter before the training

After the training, a test was conducted to assess the inter-coder reliability of the approach. The authors along with the other two interpreters produced CMGs independently based on the same texts. Four text segments were randomly chosen from the responses to the questionnaires. The interpreters were asked to convert the four text segments into four CMGs, named Task 1, Task 2, Task 3 and Task 4. Therefore, we obtained 16 CMGs from 4 interpreters based on 4 texts.

• Measures

Inter-coder reliability is a measure of agreement among multiple coders on how they apply codes to text data. Various coefficients have been developed to calculate inter-coder reliability, e.g., S [48], p [49] and K [50]. The concepts on which these coefficients are based are similar: observe agreement/disagreement corrected by the estimated agreement/disagreement that can occur solely by chance. The differences between the coefficients lie in the different assumptions used to calculate chance correlation [51].

The inter-coder reliability assessment of CMG is deemed to be much more complex than what is currently used for content analysis or discourse comprehension. The current methods generally take two forms: (1) participants assigning a theme to a segment of text by choosing from a fixed theme list, and (2) rating an item according to a fixed scale (e.g., 5-point scale). The inter-coder reliability assessment is much more difficult for CMG, as the coders are required to extract concepts and construct the relations between these concepts. The inter-coder reliability should be based on a comprehensive measure of the difference between CMGs. Again such measure is even more difficult to define than that used for conventional causal maps [21, 22]. The inter-coder reliability assessment of CMG based on difference measures lends itself to a follow-up investigation. Here we propose a simple measure for pair-wise agreement:

$$h_{c_1 c_2} = 1 - \frac{DE_{c_1 c_2}}{E_{c_1} + E_{c_2}} \quad (1)$$

where $h_{c_1c_2}$ is the agreement between coder c_1 and c_2 . E_{c_1} represents the total number of elements contained in the CMG produced by the coder c_1 , including nodes, edges, relations and constraints defined in Table 3. E_{c_2} represents the total number of elements contained in the CMG produced by the coder, c_2 . DE indicates the number of elements in disagreement, and is proposed to capture all of the following differences:

- Existence or non-existence of nodes (variables and events);
- Existence and non-existence of relations between the existing nodes;
- Types of the existing relations and the constraints on the relations;
- The relative position of the nodes involved in the same relation type.

When two segments are deemed equivalent, the elements that constitute these segments are excluded from the calculation of DE .

Extending the agreement between two coders to multiple coders, we get the overall agreement for a text segment i , shown in formula (2).

$$h_i = \frac{1}{\binom{c}{2}} \sum_{m=1}^{c-1} \sum_{n=m+1}^c h_{mn} \quad (2)$$

where h_{mn} is the agreement between the coder m and the coder n on the text segment i , c is the total number of coders.

The inter-coder reliability is estimated by the average agreement on the randomly selected text segments, calculated by formula (3).

$$R = \frac{1}{N} \sum_{i=1}^N h_i \quad (3)$$

where R is the inter-coder reliability, N is the number of text segments.

• Data and Results

The inter-coder reliability data for the case study is summarized in Table 6. Numbers in diagonal cells indicate the number of elements contained in the coder's CMG, e.g., the "43" with grey background in Table 6 indicates that the CMG produced by coder C2 for Task 2 contained 43 elements. The cells to the upper right of the diagonal cells contain the numbers of elements in disagreement (i.e., DE_{mn}) between two coders (i.e., m and n), e.g., "13" with the grey background indicates there were 13 elements in disagreement between the CMGs produced by C2 and C3 for Task 2. The cells to the lower left of the diagonal cells indicate the calculated pair-wise agreement (i.e., h_{mn}) between two coders (i.e., m and n), e.g., "0.84" in the grey background shows that the pair-wise agreement between C2 and C3 is "0.84" for Task 2.

Table 6. Inter-coder reliability data and results

DE_{mn} E_c h_{mn}	Task1				Task 2				Task 3				Task 4			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
C1	34	3	8	6	43	16	19	16	37	0	4	3	33	3	8	18
C2	0.96	37	5	5	0.81	43	13	13	1.00	37	4	3	0.96	38	5	14
C3	0.90	0.94	49	16	0.77	0.84	40	9	0.95	0.95	42	1	0.88	0.93	34	11
C4	0.90	0.92	0.79	28	0.78	0.82	0.87	29	0.96	0.96	0.99	40	0.74	0.76	0.80	20
h_i	$h_1 = 0.90$				$h_2 = 0.82$				$h_3 = 0.97$				$h_4 = 0.85$			
R	0.89															

Based on the data in Table 6 and Equation (3), we calculated the overall inter-coder reliability: 0.89. The result indicates that inter-coder reliability of the CMG is excellent according to the standardized classification given by Cicchetti [52]. However, the inter-coder reliability tends to decrease with the increase in the variations of expertise levels and task complexity (minimum inter-coder reliability is 0.74), as expected by the theory that experts generally agree more with each other than with novices [34], which implies that basic domain experience is preferred when one uses CMG.

Our sample size (the number of elements contained in the causal map segments used to calculate inter-coder

reliability) is $Samplesize = \sum_{i=1}^4 \frac{\sum_{c=1}^4 E_{c_i}}{4} = 146$, which is 24.5% of the total number of elements contained in the final

CMGs in Table 7, twice more than the general sample size requirements (10%) proposed by Lombard et al [53].

5.2 Validity evaluation

• Procedures

We were concerned with the accuracy with which CMGs are able to portray the ideas/concepts/ thoughts embedded in the text-based responses of the participants. This concern is labeled as *interpretive validity*, which refers to the degree to which the research participants' viewpoints and thoughts are accurately understood by the qualitative researcher and portrayed in the research report [47]. Participant feedback is considered as the most important strategy to assess and improve interpretive validity [47].

The first author constructed the individual CMGs based on the text responses to the questionnaires. An example CMG is shown in Figure 6. Correlations exist between availability and security, and the correlations can be positive as well as negative, depending on the scenarios (contexts or assumptions). Furthermore, correlations between availability and security are formed due to the existence of shared causal factors and mechanisms, e.g. design faults. Finally, the CMG describes the causal mechanisms of software availability and security failures, e.g., the chain of how various root factors (e.g. poor management and complexity) influence availability faults. These faults are then activated under specific conditions and transformed into software availability failures (the software system becomes unavailable) when the tolerant design is unable to tolerate the faults.

Such CMGs provide the fundamental models to assess software dependability failures at different stages of the software development lifecycle. For instance, in Figure 6 the **path** from “software security failure” to “software security” allows one to *estimate* Security based on “security failure” data. This estimation is performed after relevant data is collected during software system operation. The deeper path from “vulnerabilities” to “software security” allows one to *predict* “security” based on “vulnerability” data; such predictions can be started at an earlier stage of the software development lifecycle, e.g. the software design stage. The causal map also provides the factors that should be controlled to prevent security failure, e.g. careful use of COTS at the beginning, and improving the design of “intrusion detection and prevention mechanisms” at the design stage. We can see that such causal mechanism graphs contain information sufficiently rich to predict, estimate, and prevent the events of interest, as well as to direct various research activities related to the events of interest.

These CMGs were then double checked by the second author. Disagreements were discussed and revised until a consensus was reached. Thus, each response was converted to a CMG that was unanimously agreed upon by the researchers. These CMGs were then sent back to the experts to assess their validity in the form of confirmatory questionnaires.

Each confirmatory questionnaire contained the CMG produced by the researchers. In the confirmatory questionnaires, we employed a triangulation strategy to collect different sources of evidence [54] to assess validity. We asked the expert to identify the possible misinterpreted and missing elements, and also to estimate the extent (in the form of percentage) to which the causal map was representative of their responses to the questionnaire. A sample of the questions contained in the confirmatory questionnaires to validate CMGs is provided as follows (take Figure 6 for example):

- *Question 1.* Please estimate the extent to which the causal map is representative of your responses to the questionnaire: _____%.
- *Question 2.* Are there any incorrect or missing concepts/reasons in Figure 6? Please provide specific details below (for a detailed description of the notations used in the causal maps please refer to Table 3).

Figure 6. An example CMG built based on an expert's questionnaire response

• Measures

We obtained 8 CMGs that have been validated by the experts. Among these 8 confirmatory questionnaires, 7 experts provided direct estimations on the interpretive validity, while one expert identified the contested location in the CMG. He provided comments, "I think you have captured the gist of what I was trying to say though you have not expanded the recovery aspect. I assume you will reference the relevant papers I had sent." The "recovery aspect" was represented as a root factor in the corresponding CMG. Based on his comments, we calculated the validity using equation (4).

$$V_c = (1 - \frac{DE_e}{E_c}) \bullet 100\% \quad (4)$$

where V_c represents the validity of the CMG, DE_e is the number of elements that the expert disagrees with—elements that were missing, incorrect or should be extended, E_c indicates the total number of elements in the CMG.

• Data and Results

We obtained 8 groups of CMGs validated by the experts. The other 6 CMGs were not provided with validation comments either because the CMG was too small (the validity would be extremely high) or because of the unavailability of the experts. Among these 8 confirmatory questionnaires, 7 experts provided direct estimations on the interpretive validity, while one expert identified the contested location in the CMG. The validity data is summarized in Table 7.

Note that since this study required an expert to have published at least 5 papers in two fields (e.g., reliability and safety), the available number of experts is limited due to the fact that experts who meet these interdisciplinary requirements are few. However, the total number of validated CMG elements is 588, which is not a small sample and covers the full range of notation elements required to describe the knowledge of experts interviewed, though the CMG elements are categorized into 8 groups.

Table 7. The data for the validity of ACMG

CMG Group #	CMG1	CMG2	CMG3	CMG4	CMG5	CMG6	CMG7	CMG8
Size of the CMG*	66	58	126	81	95	41	69	52
Validity rating	90%	95%	95%	95%	95%	98%	100%	100%
Validity assessment type	Type I**	Type I	Type I	Type I	Type I	Type II***	Type I	Type I
Descriptive Statistics for CMG Size	Min=41, Max=126, Average=73.5, Standard Deviation=27.0							
Descriptive Statistics for Validity	Min=90%, Max=100%, Average=96%, Standard Deviation=3%							
Total number of validated CMG elements	588							

* Measured by the number of elements constituting the CMG

** Type I: Experts' direct estimation

*** Type II: Calculation based on experts' comments and formula (4)

From Table 7 we can see that the experts rated our CMGs very highly in terms of representing their text responses (Average=96%, Standard Deviation=3%). We performed a Wilcoxon signed-rank test [55] to examine the significance of the CMG validity data. The Wilcoxon signed-rank test was chosen because it is a non-parametric statistical test especially robust for small sample sizes and because requires no assumptions on the distribution of the data. The results show that CMG's interpretive validity is significantly higher than 0.9 (Median=0.95, $z=2.32$, $p < 0.05$).

6. Discussion

Understanding the causal mechanisms between concepts in software dependability engineering is a fundamental subject of interest to various research and industrial activities. The proposed CMG notation system was proven to be particularly robust, reliable and valid to elicit and represent experts' mental knowledge on causal mechanisms in software dependability engineering. The following sections will discuss the potential application of CMG, the possible limitations and future studies.

6.1 Potential Applications

CMG was primarily aimed to provide a robust notation to elicit and represent cause-effect knowledge embedded in experts' minds in software dependability engineering, though it possesses symbols that can be used to model a system's characteristics related to software dependability. In extreme cases, a CMG can be reduced to a fault tree, a Bayesian Network, or a conventional causal map. Thus, CMG can be used in all the situations when the referred three methods apply.

The biggest strength of CMG is that it provides the bridge between complex mental knowledge, representation of natural language, representation of mathematical language and visual representation with respect to cause-effect knowledge. Therefore, it provides an easy-to-use instrument for software dependability researchers and practitioners to communicate and merge cause-effect knowledge in the software dependability engineering domain. The produced CMGs are essential to promoting software dependability research and industrial activities. For instance, CMG can be potentially used to elicit software dependability requirements, perform risk analysis, assist in the dependability design and promote management strategies.

An example of using CMG to promote the elicitation of software dependability requirements is shown in Figure 7 and Figure 8. The example is based on a published case of system dependability requirement elicitation for the software system Tactical Separation Assisted Flight Environment (TSAFE) developed by NASA [56]. We chose this published example because it allows us to demonstrate the capability of CMG in extracting cause-effect knowledge contained in the general literature, as well as examine CMG's potential for representing causal mechanisms that influence system dependability, since this example shows the elicitation of TSAFE's system dependability requirements, rather than solely software dependability requirements.

The fundamental step necessary to elicit dependability requirements is to determine how various factors influence dependability. Let us assume for now that we are researchers and practitioners who do not know the causal mechanisms of dependability, we are going to extract such knowledge from the published literature, i.e. the paper in question. Using the CMG notation, the natural language representing the knowledge of interest, Table 2 "UMD customization for TSAFE" in the paper [56], shown in the left side of Figure 7 can be easily converted into CMG segments shown in the right side of Figure 7.

The advantage of CMG is not just to allow practitioners to extract cause-effect knowledge of interest from the literature, and represent it in a concise and easy-to-understand way, but also to allow practitioners to merge the knowledge and check the coherence and consistency of such knowledge. By only reading the text in the left side of Figure 7, we are unable to know whether the TSAFE requirements are complete and coherent. From the right side of Figure 7 we can easily see what we obtained in Figure 7 are just the isolated fragments of a full dependability requirement elicitation model. By assembling these CMG segments, we are able to identify the missing concepts (e.g., faults/vulnerabilities in Figure 8) and the missing dependencies (e.g., the relation between "event" and "failure", the dependency between "failure type" and "hazard type", and the correlation between "availability" and

“hazards”). It is notable that we are interested in what CMG can do as a representation method on the selected literature, rather than in developing a complete and accurate dependability model here. Therefore, Figure 8 only includes the concepts defined and the missing or ambiguous concepts and relations (red portion) that were simply the result of the CMG segments merging process, rather than those that come from the researchers’ own knowledge base or from the experts’ inputs obtained in our dependability study. In practice, once practitioners identify the missing or contradicting requirements, they may use their own knowledge base or more literature review to develop a complete and coherent dependability model. For instance, based on the researchers’ knowledge base, “attack” is a term related to “security”; the researchers may go further to identify security requirements in Figure 8. In the same fashion, hazards are related to safety and safety requirements should be defined. Using the CMG notation, we are able to systematically assemble various concepts to form a coherent model, which is essential to assess and control dependability risks, e.g. failures propagate through different paths in Figure 8 and can have significantly different probabilities and lead to various hazards.

The conceptual knowledge represented in Figure 8 is certainly beneficial to dependability requirement elicitation, as it provides stakeholders a comprehensive view on “what aspects constitute a dependability requirement” and “how these aspects relate to each other”, as well as “how a requirement should be described”. Essentially, Figure 8 can be used as a template (compared to a Class in object-oriented programming) for one to define a dependability requirement (compared to an Object that is an initiation of the Class). Take the requirement related to the failure “system crash” in the referred paper [56] for example, this requirement can be directly described in Figure 9, which is produced by assigning a set of specific values that relate TSAFE to the entities described in Figure 8.

Figure 7 Applying CMG to extract cause-effect knowledge embedded in the literature

Figure 8 Applying CMG for software dependability requirement analysis and model checking

Figure 9 A specific dependability requirement represented by CMG template

6.2 Limitations and Future Studies

The limitation of this study is that the CMG notation is validated in the scope of limited experts' responses to questionnaires on cause-effect knowledge in software dependability engineering. Threats to external validity may exist in terms of the extent to which CMG can be used for representing the whole set of cause-effect knowledge existing in software dependability engineering. The current maximum size of the validated CMGs contains 126 elements (see CMG3 in Table 7), while the representative validity may decrease when the size of a CMG becomes extremely large. It is also notable that the CMG is a method to elicit and represent cause-effect knowledge in software dependability engineering, however, the CMG itself does not provide a completed set of cause-effect domain knowledge.

This paper leads to four main avenues for future research. One direction is to merge different experts' knowledge based on the CMG. This will include proposing a set of formal merging rules to merge different CMGs without losing individual CMG information. Methods are also required for CMG analysis and comparison, e.g., methods to identify consensus knowledge between different experts, and methods to compare the differences between different CMGs. These methods will enable practitioners to combine different experts' knowledge and identify the most important consensus knowledge. Such knowledge can be further used in various ways, e.g., improving software dependability risk estimation, prediction, prevention and management. A follow-up paper will address these issues.

The second direction for future study is to extend CMG's capability to model system dynamic behaviors. Currently CMG focuses on representing the cause-effect mental knowledge that exists in domain experts' minds. Such knowledge tends to be conceptual knowledge, rather than particular to the structure or behavior of a software system. The CMGs notation can be used to represent cause-effect relations in software systems, in which case CMG provides a more comprehensive set of notations than Fault Tree or Bayesian Networks. The notation such as "activation" and "conflict" have rich semantics that should be able to capture complex dynamic behaviors of software systems. More notations and constraints are required in order to integrate conceptual knowledge representation with software system modelling. For instance, the strengths on the links in a CMG should vary over time for a dynamic system. To represent this feature, a current strength represented by a static number should be updated to a strength represented as a function of time. Moreover, this system *time* will be measured differently from that used in conceptual cause-effect relations where time is implied as precedents to posteriors. Such issues need to be resolved when extending CMG to dynamic system modeling.

The third future study direction is to build integrated software dependability assessment models, which include estimation and prediction models for various dependability attributes, and conduct corresponding empirical studies based on the consensus CMGs obtained from the domain experts. The consensus cause-effect knowledge shared by multiple domain experts represents the most important knowledge for software dependability assessment. Such causal mechanism graphs similar to Figure 6 will tell researchers what entities should be measured. Furthermore, these graphs provide the framework to construct the assessment models using the results of the measurements, i.e., the influence types (positive or negative) between two variables and more accurate quantification dependencies (e.g. "conflict") between multiple variables provide the quantitative constraints on the assessment models. However, the strength of the relation between variables varies from systems to systems. Historic data or measurement data for the specific system under evaluation will be required for building a comprehensive accurate dependability assessment model for the system.

The fourth future direction is to conduct empirical case studies on the dependability assessment models for the aimed software systems. CMG provides a framework to build software assessment models, but the strengths of the links in the CMG vary from system to system. For instance, how likely a proportion of faults are triggered by a set of operating conditions is determined by the specific faults, faults' properties (e.g. locations), and specific operation profile for the particular system of interest. This paper focuses on the design of the CMG notations to fully represent the knowledge possessed by experts. Therefore, the validation focuses on the consistency between the mental models represented by causal mechanism graphs and the mental models possessed by experts in their minds, rather than validate the correctness of the experts' mental model. The correctness of experts' knowledge will need to be addressed in the future studies that aim to build dependability assessment models. These quantification models will need to be validated by assessing whether the CMG models elicited from experts correctly represent the relations existing in the object software systems.

7. Conclusion

This paper designed a notation system called Causal Mechanism Graph (CMG) to represent complex cognitive knowledge of causal mechanisms in software dependability. A Causal Mechanism Graph constructed based on this notation system is capable of capturing logic, time and scenario features, which are essential to the description of interactions between various factors to produce an effect. These notations allow researchers to model causal mechanisms more accurately: *logic* symbols allow various logical combinations between causes or effects; *scenario* enables the identification of situations when a relation is likely to exist; *time flow* allows a number of cause-effect units to develop into a cause-effective chain. Moreover, notations were designed to capture the recurrent patterns of comprehensive causal mechanisms existing in the software dependability domain, i.e., *activate* and *conflict*. Furthermore, each relational notation in CMG has a set of quantitative rules. CMG is especially suitable to represent one's cognitive knowledge, as it allows one to model the dynamic causal mechanisms in a robust way, i.e. both accurately and/or fuzzily at the same time depending on the depth of knowledge one has. Finally, this notation system is proven to possess high interpretive validity and inter-coder reliability by the domain experts and analysts respectively.

This causal mechanism graph notation system provides practitioners with a useful method to elicit domain cause-effect knowledge. The generated causal mechanism graphs can be further used in various ways, i.e. develop integrated software dependability assessment models, identify software dependability risks and provide guidance for software dependability management.

Acknowledgements

This research is funded by the U.S. Department of Energy (grant DE-NE0000709). We acknowledge the experts serving on the expert panel (in alphabetical order): Victor R. Basili, Norman Fenton, John de Grosbois, Myron Hecht, Erland Jonsson, Mohamed Kaâniche, Karama Kanoun, Michael R Lyu, Yashwant K. Malaiya, Daniel P. Siewiorek, Kishor S. Trivedi.

References

[1] Avizienis A, Laprie J-C, Randell B. Fundamental concepts of dependability: University of Newcastle upon Tyne, Computing Science; 2001.

- [2] Littlewood B, Strigini L. Software reliability and dependability: a roadmap. Proceedings of the Conference on the Future of Software Engineering: ACM; 2000. p. 175-88.
- [3] Basili V, Donzelli P, Asgari S. A unified model of dependability: Capturing dependability in context. Software, IEEE. 2004;21:19-25.
- [4] Machamer P, Darden L, Craver CF. Thinking about mechanisms. Philosophy of science. 2000;1-25.
- [5] Huang F, Liu B. Systematically Improving Software Reliability: Considering Human Errors of Software Practitioners. 23rd Psychology of Programming Interest Group Annual Conference (PPIG 2011). York, UK2011.
- [6] Huang F. Post-completion Error in Software Development. The 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2016), ICSE 2016. Austin, TX, USA: ACM; 2016. p. 108-13.
- [7] Neil M, Littlewood B, Fenton N. Applying Bayesian belief networks to system dependability assessment. Safety-Critical Systems: The Convergence of High Tech and Human Factors: Springer; 1996. p. 71-94.
- [8] Rosqvist T, Koskela M, Harju H. Software quality evaluation based on expert judgement. Software Quality Journal. 2003;11:39-55.
- [9] Li M, Smidts CS. A ranking of software engineering measures based on expert opinion. Software Engineering, IEEE Transactions on. 2003;29:811-24.
- [10] Walls L, Quigley J. Building prior distributions to support Bayesian reliability growth modelling using expert judgement. Reliability Engineering & System Safety. 2001;74:117-28.
- [11] Huang F, Liu B, Wang S, Li Q. The impact of software process consistency on residual defects. Journal of Software: Evolution and Process. 2015;27:625-46.
- [12] Van Koten C, Gray A. An application of Bayesian network for predicting object-oriented software maintainability. Information and Software Technology. 2006;48:59-67.
- [13] Langseth H, Portinale L. Bayesian networks in reliability. Reliability Engineering & System Safety. 2007;92:92-108.
- [14] Lee W-S, Grosh DL, Tillman FA, Lie CH. Fault Tree Analysis, Methods, and Applications & A Review. Reliability, IEEE Transactions on. 1985;34:194-203.
- [15] Rumbaugh J, Jacobson I, Booch G. Unified Modeling Language Reference Manual, The: Pearson Higher Education; 2004.
- [16] Huff AS. Mapping strategic thought: John Wiley & Sons; 1990.
- [17] Fenton N, Neil M, Marsh W, Hearty P, Marquez D, Krause P, et al. Predicting software defects in varying development lifecycles using Bayesian nets. Information and Software Technology. 2007;49:32-43.
- [18] Frühwirth-Schnatter S. On fuzzy Bayesian inference. Fuzzy Sets and Systems. 1993;60:41-58.
- [19] Pearl J. Causality: models, reasoning and inference. England2000.
- [20] Bougon MG. Congregate cognitive maps: a unified dynamic theory of organization and strategy. Journal of management studies. 1992;29:369-87.
- [21] Langfield-Smith K, Wirth A. Measuring differences between cognitive maps. Journal of the Operational Research Society. 1992;1135-50.
- [22] Markíczy L, Goldberg J. A method for eliciting and comparing causal maps. Journal of management. 1995;21:305-33.
- [23] Eden C. Analyzing cognitive maps to help structure issues or problems. European Journal of Operational Research. 2004;159:673-86.
- [24] Scavarda AJ, Bouzdine - Chameeva T, Goldstein SM, Hays JM, Hill AV. A Methodology for Constructing Collective Causal Maps*. Decision Sciences. 2006;37:263-83.
- [25] Nelson KM, Nadkarni S, Narayanan V, Ghods M. Understanding software operations support expertise: a revealed causal mapping approach. Mis Quarterly. 2000;475-507.

- [26] Hodgkinson GP, Clarkson GP. What have we learned from almost 30 years of research on causal mapping. *Causal mapping for research in information technology*. 2005:46-79.
- [27] Laukkanen M. Comparative cause mapping of organizational cognitions. *Organization science*. 1994;5:322-43.
- [28] Defense USDo. Mil-Std-1629a: 1980: Procedures for Performing a Failure Mode, Effects and Criticality Analysis: Department of Defense; 1980.
- [29] Murata T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*. 1989;77:541-80.
- [30] Swaminathan S, Smidts C. The event sequence diagram framework for dynamic probabilistic risk assessment. *Reliability Engineering & System Safety*. 1999;63:73-90.
- [31] Bucci P, Kirschenbaum J, Mangan LA, Aldemir T, Smith C, Wood T. Construction of event-tree/fault-tree models from a Markov approach to dynamic system reliability. *Reliability Engineering & System Safety*. 2008;93:1616-27.
- [32] Lee BH. Using Bayes belief networks in industrial FMEA modeling and analysis. *Reliability and Maintainability Symposium, 2001 Proceedings Annual: IEEE; 2001*. p. 7-15.
- [33] Bluvband Z, Polak R, Grabov P. Bouncing failure analysis (BFA): the unified FTA-FMEA methodology. *Reliability and Maintainability Symposium, 2005 Proceedings Annual: IEEE; 2005*. p. 463-7.
- [34] Huang F, Liu B, Song Y, Keyal S. The links between human error diversity and software diversity: Implications for fault diversity seeking. *Science of Computer Programming*. 2014;89, Part C:350-73.
- [35] Glaser BG, Strauss AL. *The discovery of grounded theory: Strategies for qualitative research*: Chicago, Aldine; 1967.
- [36] Glaser BG. *Emergence vs forcing: Basics of grounded theory analysis*: Sociology Press; 1992.
- [37] Corbin J, Strauss A. *Basics of qualitative research: Techniques and procedures for developing grounded theory*: Sage publications; 2014.
- [38] Visser W. Dynamic aspects of design cognition: Elements for a cognitive model of design. *INRIA, Rapport de recherche n. 2004:2004-116*.
- [39] Huang F, Liu B, Huang B. A Taxonomy System to Identify Human Error Causes for Software Defects. In: Pham H, editor. *Proceedings 18th Issat International Conference on Reliability & Quality in Design*. Boston, USA: Int Soc Sci Appl Technol; 2012. p. 44-9.
- [40] Sjøberg DI, Dybå T, Anda BC, Hannay JE. Building theories in software engineering. *Guide to advanced empirical software engineering*: Springer; 2008. p. 312-36.
- [41] Coleman G, O'Connor R. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology*. 2007;49:654-67.
- [42] Huang F, Liu B, Wang Y. Review of Software Psychology. *Computer Science*. 2013;40:1-7.
- [43] Saldaña J. *The coding manual for qualitative researchers*: Sage; 2012.
- [44] Graesser AC, Singer M, Trabasso T. Constructing inferences during narrative text comprehension. *Psychological review*. 1994;101:371.
- [45] Granger CW. Testing for causality: a personal viewpoint. *Journal of Economic Dynamics and control*. 1980;2:329-52.
- [46] Miles MB, Huberman AM. *Qualitative data analysis: An expanded sourcebook*: Sage; 1994.
- [47] Johnson B, Christensen L. *Educational research: Quantitative, qualitative, and mixed approaches (5th Edition)*. London: Sage; 2008.
- [48] Bennett EM, Alpert R, Goldstein A. Communications through limited-response questioning. *Public Opinion Quarterly*. 1954;18:303-8.
- [49] Scott WA. Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly*. 1955;19:321-5.

- [50] Cohen J. A coefficient of agreement for nominal scales. Educational and psychological measurement. 1960;20:37-46.
- [51] Artstein R, Poesio M. Inter-coder agreement for computational linguistics. Computational Linguistics. 2008;34:555-96.
- [52] Cicchetti DV. Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. Psychological assessment. 1994;6:284.
- [53] Lombard M, Snyder-Duch J, Bracken CC. Practical resources for assessing and reporting intercoder reliability in content analysis research projects. Retrieved April. 2004;19:2004.
- [54] Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering. 2009;14:131-64.
- [55] Wilcoxon F. Individual comparisons by ranking methods. Biometrics bulletin. 1945:80-3.
- [56] Donzelli P, Basili V. A practical framework for eliciting and modeling system dependability requirements: Experience from the NASA high dependability computing project. Journal of Systems and Software. 2006;79:107-19.
- [57] Carol Smidts, Fuqun Huang, Xiang Li, Mutha C. A Method for Quantifying the Dependability Attributes of Software-Based Safety Critical Instrumentation and Control Systems in Nuclear Power Plants. 9th international conference on Nuclear Plant Instrumentation, Control & Human-Machine Interface Technologies. Charlotte, NC, USA: ANS; 2015.

Appendix. Two types of questionnaires used for data collection

Table 8. Template of the questionnaires Type I (paired attributes)[57]

Questions		Motivations for the question design	
Part I. Detailed Discussions	How do Attribute A and Attribute B Influence Each Other? I.1 <i>How does</i> Attribute A Influence Attribute B? I.2 <i>How does</i> Attribute B Influence Attribute A?	Obtain an overview of the relations between two attributes.	
Part II. Identification of Dependence Mechanisms	II.1 What are the root causes of Attribute A problems?	Obtain the root causes of Attribute A problems.	Obtain the mechanisms underlying Attribute A problems
	II.2 What conditions, intermediate variables or events are required to transform the root causes into an Attribute A problem?	Obtain intermediate variables or events underlying Attribute A problems.	
	II.3 What are the root causes of Attribute B problems?	Obtain the root causes of Attribute B problems.	Obtain the mechanisms underlying Attribute B problems
	II.4 What conditions, intermediate variables or events are required to transform the root causes into an Attribute B problem?	Obtain intermediate variables or events underlying Attribute B problems.	
	II.5 What are the common causes for Attribute A problems and Attribute B problems?	Redundant questions that help the respondent differentiate the root causes of the two attributes and improve the confidence in the corresponding causal map.	
	II.6 What causes specifically lead to Attribute A problems, but do not lead to Attribute B problems?		
	II.7 What causes specifically lead to Attribute B problems, but do not lead to Attribute A problems?		
	II.8 What conditions, intermediate variables, states or events are required to transform an Attribute A problem into an Attribute B problem?	Obtain explicit information on the scenarios under which Attribute A problems are transformed into Attribute B problems, and vice versa.	
	II.9 What conditions, intermediate variables, states or events are required to transform an Attribute B problem into a Attribute A problem?		
	II.10 The relation from Attribute A to Attribute B tends to be (multiple choices): () a. Attribute A has positive effects on Attribute B, given the following assumptions/conditions/circumstances	Obtain explicit information on how Attribute A influences Attribute B.	

	<p>b. Attribute A has negative effects on Attribute B, given the following assumptions/conditions/circumstances _____.</p> <p>c. No relation, they are independent.</p>	
	<p>II.11 The relation from Attribute B to Attribute A tends to be (multiple choices): ()</p> <p>a. Attribute B has positive effects on Attribute A, given the following assumptions/conditions/circumstances _____.</p> <p>b. Attribute B has negative effects on Attribute A, given the following assumptions/conditions/circumstances _____.</p> <p>c. No relation, they are independent.</p>	Obtain explicit information on how Attribute B influences Attribute A.

Table 9. Template of the questionnaires Type II (attribute – dependability) [57]

<i>Type 2: Attribute- dependability</i>	Part I. Identification of Key Attributes	Please check the attributes (double click the appropriate box) that you think can directly influence <i>Software Dependability</i>	Identify the attributes that constitute dependability.
		Among the attributes that you chose above, please identify the attributes that interact directly with Attribute C.	Identify the attributes that interact directly with Attribute C.
		Among the attributes you identified in Part I, Question 2, which attributes should be involved in a detailed discussion allowing you to meaningfully characterize the relationship between Attribute C and Dependability. This discussion will be the object of Part II.	Identify the attributes that interact directly with Attribute C and the expert is comfortable discussing.
		Please discuss the <i>interaction mechanisms</i> between Attribute C and the attributes identified in Part I, Question 3. Further, discuss how these attributes and their interactions <i>influence Dependability</i> .	Obtain explicit information on how Attribute C-related attributes influence dependability.
	Part II. Identification of Dependence Mechanisms	Questions similar to Part II of the “Paired attributes” questionnaire.	Obtain information on the interaction mechanisms between the attributes.