Applying Forensic Analysis Factors to Construct a Systems Dynamics Model for Failed Software Projects

Stephen Kaisler, D.Sc.
SHK & Associates
Laurel, MD 20723
skaisler1@comcast.net

William Money, Ph.D.
The Citadel
Charleston, SC 29409
wmoney@citadel.edu

Stephen Cohen
Microsoft Corporation
Redmond, WA 98052
Stephen.cohen@microsoft.com

Abstract

Forensic analysis of failed software projects can aid in managerial understanding of the issues and challenges of delivering a successful project. The factors and their interrelationships causing software project failure are not well understood or researched with a strong forensic-analytic approach. Previous papers have not adequately explored how dynamic interaction of multiple factors can lead to critical events that ultimately portend eventual failure. This paper proposes the development of a System Dynamics (SD) model that will represent the key factors, their dynamic interactions, and the influence of exogenous events in causing software project failure. Forensic data will be used as inputs to the SD model to assist managers in understanding the factor interactions, the importance of individual factor metrics, as well as the sequence of interactions in causing possible software project failure. Outcomes from the model will include a likelihood of software project failure, possible factor sequences leading to failure, and suggestions of remediation activities that might mitigate eventual failure.

1. Introduction

While numerous technical papers, blogs, and articles have described and assessed failing or failed software projects, few have described these projects based on the component factors that, when evaluated according to scalar performance measures, can be used to assess the probability of failure or the diagnose the underlying reasons for such failure. The factors are composed of a number of different components which can be assessed relative to their importance to a project. Poor scores (high or low depending upon the component) are considered faults which can be used to assess the factors as input to the model. The issues and challenges of assessing software project failures have been described in an initial model [1,2] However, knowing the factors does not answer questions about how and why a software project fails, and no theory of system failures has been proposed or critically examined in the software project failure literature.

1.1 Precious Research

Research addressing software project failure recognizes many different, but interrelated causes of failures, and posits that the failure of systems is a complex and multidimensional problem. The analysis of failures is also difficult because there are different levels of response to perceived impending or detected failure ranging from abandonment to attempting partial or full recovery. An expanded literature review focused on identifying critical factors and their relationships which can constitute an analytical framework for assessing the potential for failure of a software project.

However, the factors associated with failure have been observed and singled out in the practitioner literature, and enumerated in the technical literature as possible software project failure causes.

A research literature survey and analysis was conducted to systematically identify key possible failure factors, and the relationships among the factors. The literature and discussion clearly indicate that a software project failure is not a singular or discrete short term event (e.g., such as falling off a cliff or firing a weapon). It is hypothesized to (more often) encompass a longitudinal sequence of actions and/or accumulation of critically low or poor performance on a number of factors that eventually lead to a software project's failure determination or recognition. This paper reports on and discusses the factors from the literature survey. The number of factors and their interactions suggest that dynamic interactions of

URI: https://hdl.handle.net/10125/79477
978-0-9981331-5-7

multiple factors can ultimately lead to software project failure.

The framework model, initially described in [2], has been revised and enhanced to incorporate additional factors extracted during the literature review to enable researchers to conduct forensic analysis of software project failures. It will be the basis for constructing the System Dynamic model.

1.2 Literature Review

This paper seeks to analyze and explain failure from a broad perspective. Thus, it sought widely published descriptions and discussions of software project failures from the academic and practitioner literatures. Our objective is to avoid attributing software failure causation to only intuitive views, ad hoc rules, and overly simplistic assumptions. We have incorporated defined concepts, and relations between concepts to address this problem by proposing a model based on the literature. We explain our approach with an example to show how forensic analysis is useful because of its reasoning contribution, qualitative distributed attribution to decisions, and holistic approach to the understanding of software project failures.

Sixty (60) papers (available in a separate bibliography) from the technical literature were identified using search terms including "project failure", "software project failure", "failed projects", etc. The searches produced several hundred papers that matched one or more of these terms from the Xplor Digital Library (http:\\ieeexplore.ieee.org/), IEEE CDSL, the ACM Digital Library (http://portal.acm.org), Elsevier Science Direct (www.sciencedirect.com), Springer Link (http://link.springer.com), the AIS Repository, and the CEUR-WS Repository.

An initial review of the papers led to an identification of 14 categories based on the frequency of mention of relevant terms. Papers were selected based on their mention of a term from the categories and their discussion of success or failure of a project. A fault was indicated by a negative statement regarding the potential for success of the project. These categories were assigned to one or more causal factors.

Table 1 presents a summary of the literature review used to develop a forensic analysis framework.

Table 1. Literature Review Analysis

Elements	Count
Project Visibility	2
Management Tools & Methods	24
System Life Cycle	5
Time	12
Schedule	25
Risk	27
Personnel Resources	36
Communication	20
Cost & Budget	25
Technology	25
Stakeholders	33
Project Complexity	36
Project Management	44
Software	12

This review and analysis revealed that most papers utilized anecdotal data, that understanding of critical factors was not present in the literature, and that measurements of such factors and their relationships to each other were not well understood or specified. Moreover, it often seemed that the software project failure was recognized in the eye of the beholder.

Analysis of the data derived from the literature survey provided one perspective on the potential causes of failure of software projects. The distribution of causes depicted in Table 1 and the analysis of the reviewed papers – many of which reviewed several projects – indicated that most projects fail for multiple reasons.

The complexity and difficult of assessing failures as a grouping of phenomena can be seen with an example. One can consider a system that was delivered late or over budget or lacking some functionality. If the system was still useful to the customer because it provided some benefits, it might be hard to classify as a failure. Indeed, the Standish Group CHAOS 2020 report [3] termed 50% of IT projects as *challenged* or impaired and 19% as failed, e.g., almost 3/4ths of IT projects did not achieve their initial goals. This observation occurs over multiple industry sectors.

Potter [4] suggested that "success and failure are two sides of the same coin". Al-Ahmad. Fagih, et al. [5] constructed a taxonomy of IT Project Failure reasons. Ralph and Kelly

[6] described the dimensions of software engineering success. However, their work does not suggest that failure is the absence of success. Different types of failure have also been explored from a developer's and manager's perspective by Lundin and Wictorin [7].

[A comprehensive discussion of the varying reasons for project failure is beyond the scope of this paper. A separate bibliography provides a longer list of relevant papers.]

1.3 Case Study: Failing and Recovery

Cohen [9] has led many efforts to analyze potential failure and develop recovery strategies for these projects. This section presents a synopsis of one such effort which would help to drive our System Dynamics model development.

A large institution operating globally needed to modernize their legacy system. Having been built to conform to specific needs over decades, the system had grown into a patchwork of technologies, inconsistent implementations, and ill-defined standards and was run by a network of local, independent, administrators who had absolute control and unfettered access to any component on their segment of the network. Like many non-IT focused organizations, they went through an extensive contracting award process and eventually selected a very large, capable, well-established firm to lead and deliver an $80M, 4-year, mission critical program.

3.5 years and ~$70M later, they had:

- followed a rigorous methodology;
- staffed with senior program leads having a wealth of experience to draw upon;
- engaged representative users, legacy system administrators, and organizational leads;
- adopted the key mission standards and sought more recent standards to migrate to;
- produced a small mountain of documentation;
- created a structured meeting rhythm to ensure engagement and transparency;
- created a huge set of tests to verify the solutions quality; and
- Maintained a voluminous risk register that was managed weekly and referenced by all.

Yet the program was facing what seemed to be certain failure. There was no software. Contracts were reviewed and discussed, logical and clear reasons were provided describing how this project state was reached. A small group of project participants sought a way to move the project forward and deliver a working solution out to the users. A recovery lead was identified and with no time and little funding asked to deliver a working system.

After several interviews, shadowing program leadership though various meetings, and reading the small mountain of documents, what was presented did *not* identify responsibility instead focusing on causes of the failure. As a result, the recovery team quickly provided an actionable plan to remediate issues as well as build and deploy a solution to the field. A summary of that plan is presented in the following table.

Table 2. Case Study: Issues and Actions

Issue	**Action**
Stakeholders	
Critical goals were not resolved to an architecture or design. Goal: System must support the mission part of which supports life/safety workloads. Goal: The system must be usable side by side with the legacy system as the transition will be lengthy.	Baseline the Architecture to meet goals.
Constraint: there will be no more money or time for extensive design.	Revise staffing to focus on experience & speed.
Dependencies	
Inter-organizational coordination blocking deployment planning. interorganizational acceptance / sign-off	Replace meeting rhythm with communication and approval discussions
Use of tools	
No promotion environments (dev>test>build)	Build needed environments.
Bespoke components with limited knowledgeable staff	Align tools to new team skills.
Reliance on a new tool to solve complexity without staff experienced on the tool	Remove tool dependencies.
Key Skills	
No User Liaison proactively informing the global population.	Create liaisons and fund global "tours".

No staff experience in complex near real time architecture or design.	Include needed experience in new staffing.
No experience doing rigorous process-based development on large systems.	Include needed experience in new staffing.
Management	
Owning organization was not "respected" by the management in the user community	Leverage team reputation and conduct extensive outreach programs.
Business outcomes had been abdicated to technical goals	
Technology	
Nearly religious belief that customizing a commercial product would solve things, it didn't.	Remove technology dependence.
* With little to no implemented software, technology was a limited source of concern	

Note: This organization cannot be named due to the sensitivity of the events.

The resulting solution was ready for production deployment on time and within available funds. New builds were being released to the client and back-end system with minimal disruption. Full global roll out completed within 2 years after which the solution entered a devops life cycle and maintained continuous release for more than 10 years.

2. Methodology

This paper views software project failures as the outcome of a system of dynamic factors that include context, components, inflection points, events, and decisions. The literature review identified causal factors from which we developed a forensic analysis framework. This framework, through causal mapping will drive the development of a system dynamics model that will enable the evaluation of software project failure based on data collected from the forensic analysis of a software project.

2.1 Research Questions

We identified two research questions which led to the definition of our technical approach:
RQ1: *What are the components of the factors that impact and influence major software project failures, and the faults that can occur for the components making up the factors?*
RQ2: *How can we model failure factors and their relationships to understand their impact on project success or failure and estimate the likelihood of project failure?*

2.2 Forensic Analysis Model

The forensic analysis model (FAM), depicted in Figure 1, is a multilevel, accumulative model consisting of events, decisions, causal factors, project metrics, and a figure of merit (FoM) as described in Table 2. This forensic analysis model is an enhanced version of the model originally proposed in [1].

Table 3. Forensic Analysis Model Elements

Event: an action that is associated with the technology, such as "module A failed to meet requirement 1.3".
Decision: an action taken by project personnel or external actors. For example, a project manager may decide "not to hire Joe because his salary requirements exceeds the budget for that position".
Fault: an event or decision receiving a low metric score that is believed to affect the ability of an organization to complete a software project because it exploits a vulnerability in software project management.
Fault Class: a set of faults that affects one or more causal factors.
Causal Factor: an opportunity in a software project for assessing whether a project is succeeding or failing.
Inflection Point: where critical decisions have to be made about the potential success or failure of a software project and the application of recovery procedures [1].
Project Metric: a weighted perspective on the effect of causal factors.
Figure of Merit: a single number on a predetermined scale that indicates potential success or failure.

Events and decisions are statements about a software project.

A *catastrophic fault* may cause a software project to fail suddenly. Anecdotal evidence suggests that failure is an accumulation of faults (over some time period). The range of the metric scale and the threshold(s) at which a fault is determined depends on a project and its domain.

Causal factors represent opportunities for taking remediative action if a software project appears to be failing as discussed in [1]. In any software project, some actions will be more important than others. Determining relative importance is key to determining if recovery is possible, developing a plan for recovery, and applying remediative or recovery procedures.

Project metrics directly affect the confidence that project management, organizations, and customers will have in potential project success because they can be direct indicators of potential failure.

The Figure of Merit is determined on a project-by-project basis for a particular domain. Projects falling significantly below the threshold lead to a *terminal inflection point*, which is a decision to terminate the project [1].

The Forensic Analysis Model, developed from our literature survey, satisfies RQ1. Continued review of the technical literature and case studies will be used to refine this model further in conjunction with experiments performed using the SD model.

3. Technical Approach

The goal of this research is to develop a System Dynamics (SD) model that will allow assessment of the potential for success or failure of a software project based on data collected on the events and decisions made during the project. To construct the SD model, we will use Causal Mapping [10] as suggested by our reviewer as this will enable us to translate instances of the elements into the components of the SD model.

We are also considering using Sowa's Conceptual Graphs (CG) [11] as a complementary technique to assist in eliminating possible ambiguity due to natural language. We are also considering Unified Modeling Language (UML) [12] to provide formal descriptions of the design.

3.1 Causal Mapping

Causal Mapping was developed by Eden [13] to handle multiple causal flows in decision-making processes. It was intended to facilitate the understanding of how events occurring in one area could impact events occurring in or more other areas. It emphasizes developing a holistic or systemic view of what has occurred or is occurring, in our case, within a project.

Causal Mapping (CM) has been used to explore human decision making processes in a variety of disciplines. We intend to use its principles to not only capture decisions, but also to capture events occurring in systems that may not be the direct result of human decisions, but indirectly result from human decisions, as described in selected technical literature.

3.1.1 CM Advantages & Disadvantages

Powell [14] has developed a *Guide to Causal Mapping*, which draw inspiration from Pearl and Mackenzie's [15] *The Book of Why*. He has identified several advantages (A) and disadvantages (D) of Causal Mapping, which are presented in Table 4.

Table 4. CM Advantages & Disadvantages

(A) Identifies and elaborates domain structures through chains of argumentation.
(A) Captures networks of effects for events and decisions.
(A) Directly understand causality based on narrative
(D) Lack of ability to recognize significant changes in the environment
(D) Lack of longitudinal data and perspective
(D) Failure to recognize the creation/insertion of previously unknown causative factors
(D) Failure to determine weights of causal factors

3.1.2 Applying Causal Mapping

We will apply CM to analyze the relevant technical literature that was identified through our literature search. We will extract events and decisions, their relationships, and their effects through our analysis.

An example of direct human decision making is illustrated by the hiring of a new programmer for a project. The multiple effects of this decision include: need to educate the programmer in the system structure, delaying some aspects of software development, drawing upon existing team members time to train the new person, etc. One decision and its associated event have multiple (potentially negative) impacts on various aspects of the project. Table 5 identifies events and decisions.

Table 5. CM Example

Events	Identify need for new programmer Hire new programmer
Decisions	Train new programmer Divert Staff to training task

	Delay SW development to train new programmer
Possible Indirect Decision	Delay SW module delivery Reschedule SW module development and/or delivery

An example of an indirect effect is an error in stating certain requirements that leads to errors in design which leads to errors in software development which leads to errors occurring in software testing. One error can lead to a sequence of errors or a spreading wavelet of errors if multiple elements of the system are affected.

3.1.3 Applying CG and UML

As seen in Table x, the statements reflect high-level events. Detailed analysis will reveal discrete subordinate variables to be considered which must be included in the SD model. CG provides techniques for decomposing activities to reveal interdependencies in elements of an activity. UML provides a formal modeling mechanism with temporal aspects that can assist in identifying concurrency issues.

3.2 Analysis using System Dynamics

Ackerman, Eden and Williams [16] applied System Dynamics (SD) to examine dynamic causality in decision-making within a project to understand and quantify the resulting effects. They explored interrelationships among causal effects that led to system problems leading to project failure. Using this approach, they identified disruptive actions in the complex, interacting parts of a project and followed them to the resulting outcomes. They conducted "what-if" analyses by varying the model parameters to assess possible remediation actions.

Other researchers have also used SD models to examine decision threads in complex projects to assess effects and outcomes. But, few SD models have explicitly focused on determining how projects fail as the result of interrelationships among these threads.

As Ackermann and Eden note, SD models accommodate multiple causality and feedback. Since causality is the basis for understanding project success or failure, causal mapping is a good technique for helping to identify the concepts and relationships to be implemented in an SD model because its output can be directly translated to the components of an SD model.

3.2.1 SD Rationale

A systems dynamics approach employs differential equations as a mathematical tool to understand the nonlinear behavior of complex systems and assess the rates of change of causal factors in a software development project. It incorporates state variables as objects to represent the state of a system. The model has a state variable representing the current state of a component (success or failure). An SD model uses derivatives to define rates of change in state variables that specify the tendency to be successful or to fail over time based upon the measured change in the component values. The derivative aggregates all changes to show the net change in the state variable over time. The structure of the model will describe the effects of state variables, their relationships, remediative actions, and the feedback from such actions in affecting recovery or minimizing the degree of failure.

3.2.2 Key SD Concepts

An SD model [17] is based on several key concepts which are described in Table 6.

Table 6. SD Key Concepts

Stocks are an accumulation of material, information, or other resources in a system over time. The quantity of a stock reflects the net changes in its inflows and outflows.
Flows are transfers of material, information, or other resources between stocks and/or the environment.
Sources represent the evaluated data collected about events and decisions that are inflows to stocks.
Sinks represent the repository of data at the conceptual boundary of the model that are outflows from stocks.
Rates are variables that control the flows of information into and out of stocks.
Auxiliaries are variables that modify information as it passes from stocks to rates.
Feedback loops can amplify or modify the quantity of stocks over time and support the implementation of iterative decision making in SD models.

3.2.3 Model Building

An SD model for forensic analysis of software projects will proceed through several stages as briefly described in Table 7.

Table 7. SD Model Development

Step	Description
1	Identify Input/Output Variables: Some of these are derived directly from the FAM; some will be exogeneous. These will suggest the initial model boundary. They may also supply weights to internal variables.
2	Review literature for endogeneous variables representing internal states of the model (rates, auxiliaries).
3	Define flows that represent the causality using causal loop diagrams (CLDs).
4	Define equations for stocks, rates, and auxiliaries.
5	Implement the SD model using VENSIM; develop visualizations of model behavior.
6	Develop test suites for several scenarios. Perform sensitivity and validation studies. Conduct “What-If” experiments.

3.2.4 Proposed Model Structure

Events, decisions, causal factors, and project metrics will be represented as nodes in the SD model. The value of each node will be dynamically computed based on the inflows and outflows to each node.

Some inflows will be generated by exogenous factors that exist outside of the project, such as weather, budget reductions, and supply delays. For experimentation purposes, exogeneous factors can be generated by lookup tables or random variables.

Values assigned to each node can be adjusted based on the perceived evaluation by the users (project managers, stakeholders, etc.). Links will connect nodes to represent relationships and to the causal factors to which they are associated. Causal factor nodes will be connected to the project metric nodes which will connected to a summary FoM node.

3.2.5 The Benefit of System Dynamics

Several benefits accrue to using system dynamics as our modeling technology as described in Table 8.

Table 8. Benefits of System Dynamics

Ability to simulate the effects of events and decisions in a model of a complex system over time.
Specific events can be activated or terminated
Recurring events can have their values adjusted within a specified range
Variables are recalculated at each time step to reflect their current values.
Yields a deeper level of understanding the interdependencies of elements than textual descriptions.
Provides a clear structural representation of the problem or process.
Provides a “hands-on” tool to conduct “what if” experiments.

Using GUI-based systems, such as VENSIM [18], a user can adjust the values of the components, weights and the exogenous factors using controls (e.g., like rheostats). Controls can also be used to activate or deactivate components of the SD model to explore “what-if” scenarios. One can also visualize the change in causal factors, project metrics, and the FoM using graphs or other tools, and assess the limits and strengths of factors with simulations.

3.5.2 Uncertainty

A significant factor is uncertainties: the “known unknowns and the unknown unknowns”. Known unknowns, as Islam et al. [19] note, are “related to time-to-market, budget and schedule estimation, technology evolution, and stakeholders’ expectations”. With project management experience, some estimates of the impact, if not the severity, of these unknowns can be made and factored into the planning process. It is unknown unknowns for which no estimates nor reasonable guesses can be made.

It is noted that risk and uncertainty are related, but are not the same concept. *Uncertainty* is the unknown, whereas *risk* is a recognized element that can go wrong or fail. Risks can be managed, but uncertainty can only be reduced. One can assess and assert the likelihood of a risk and measure its perceived value fluctuations during a project. One can reduce uncertainty through application of various managerial strategies [20], and the understanding that as time progresses uncertainty is reduced.

4. Assessing Events and Decisions

The stocks and flows will be represented as numeric values. Each event and decision in the model will be evaluated according to risk and reward tables developed by Cohen [2] shown in Tables 9 and 10. Values for the Causal Factors, Project Metrics, and the Figure of Merit will be “rolled up” by the SD equations.

Table 9. Risk Table

0	No impact
-1	Negligible impact, easily resolved
-2	Likely to create additional tasks
-3	Expands the scope of the issue
-4	Degrades team morale and/or communication
-5	Requires a notable response
-6	Increases cost or delays schedule
-7	Injects current and future issues
-8	Degrades solution quality
-9	Shows critical feature failure
-10	Prevents product/milestone delivery

Table 10. Reward Table

0	No impact
1	Improve poor performance to nominal
2	Reduce isolate effort of effort
3	Broadly improves things
4	Improves team morale/communications
5	Yields measurable improvement
6	Reduces cost or schedule
7	Resolves current issues or reduces severity of future issues
8	Improves overall quality significantly
9	Ensures feature delivery
10	Ensures total product/milestone delivery

The tables apply weighted criteria to assess risk which may significantly affect the potential for success and reward associated with remediation activities which may significantly affect the potential for partial or whole recovery. Using the model, a qualitative review of project activities is converted to a quantitative assessment by this model.

Risks, representing negative events and decisions, due to events and decisions flow into and out of stocks. Rewards, due to positive events and decisions, also flow into and out of stocks.

Traditional approaches to risk management often focus on single agents, single inflows, or single outflows, and linear progression. As noted above, there are many competing and, possibly, correlated risks that can affect project success or failure. SD supports the notion of multiple causality by allowing stocks to have multiple inflows and to affect multiple stocks via multiple outflows. Using feedback loops, SD supports the concept of iterative cause and effect.

This risk-reward model was used previously [2] was applied to a limited description of the Advanced Automation System (AAS) project of the Federal Aviation Administration [21]. The assessment was presented in Table 7 at the end of that paper. Of the 21 statements evaluated, only three had a positive impact on the program. The description clearly identified a significant negative impact that portended the likely failure of the project. The methodology for applying this assessment process will be described in a forthcoming book [9].

5. Observation

The inevitability of project failure seems to be mythical based on the accumulated anecdotal evidence, but unsupported by actual metrics and analysis. Software projects can and should be successful. Successfully managing projects and developing software is described by the plethora of articles, books, and conferences. Further, the potential pitfalls and problem areas are known, and guidelines for avoiding them are well described. A system dynamics assessment can assist project managers and stakeholders in understanding where, when, and perhaps why a project is succeeding or failing, and, if failing, how serious the problem is.

6. Conclusions

The technical literature review and anecdotes demonstrated that there is limited understanding from a project management perspective about what are significant causes of software project failure, and how to recognize them in real time. There are few tools and methods and little understanding of how to perform ongoing assessments and forensic analysis to determine the likelihood of project failure during the execution of the project.

This suggests that (1) systems are not designed and developed with assessable metrics and data, and (2) that projects are not structured to address system assessment as a continuous process rather than as a culminating activity when potential failure is impending. This is a different metric from performance on isolated software project descriptive and reporting measures. Assessability is an objective that means the ability to determine whether or not the totality of a software project has achieved its interim and final goals during system development.

Computing assessability requires metrics for each of the causal factors. Many have been suggested in the technical literature, but there is no consensus on a preferred subset. This paper has begun to identify a set of components that can lead to a process for evaluating causal factors.

This paper has proposed using a System Dynamics model to provide a continuous assessment tool for evaluating project success or failure and indicating what events and decisions support these outcomes.

7. Future Work

This paper has not addressed the different methodologies used in project management nor the use of formal methods. While formal methods can increase confidence that a project may not fail or may fail gracefully, they cannot "prove or not prove" that a software project will not fail.

Previous work identifies a set of activities required to address software project failures [2}. It is suggested that these efforts include two additional tasks:

1. *Develop Models of software Failure*: There are many models of IT system success, such as Delone and McLean [22], but few models of software failure. Forensic analysis can provide the data to construct and validate such models. Through a case study of a particular project, we will test, refine, and validate the SD model.

2. *Extend Model to Identify Remediation and Recovery Mechanisms*: Our eventual goal is not only to be able to predict potential failure, but then to examine and suggest possible remediative actions to lead to recovery.

Acknowledgement

The authors gratefully thank several colleagues and the anonymous reviewer who have provided critique, suggestions, and additional insights into the concepts and analysis presented in this paper.

Note: An extensive bibliography has been prepared for this project and is available from the authors.

References

[1] Cohen, S., S. Kaisler, and W. Money. 2020. "Forensic Analysis of Failed Software Projects", Tutorial at the 53rd Hawai'i International Conference on System Sciences, Maui, HI

[2] Kaisler, S., W. Money, and S. Cohen. 2021. "Forensic Analysis of Failed Software Projects: Issues and Challenges", *54th Hawai'i International Conference on Systems Sciences* (Virtual).

[3] Standish Group. 2020, "*CHAOS Manifesto. Beyond Infinity*". Boston, MA.

[4] Potter, S. 1987. *On the Right Lines: The Limits of Technological Innovation*. W. Frances Publishing Co., London, England.

[5] Al-Ahmad, W., K. Fagih, K. Khanfar, et al. 2009. "A Taxonomy of IT Project Failure: Root Causes", *International Management Review*, 5(1):93-104.

[6] Ralph, P. and P. Kelly. 2014. "The Dimensions of Software Engineering Success", *International Conference on Software Engineering*, ACM Press. New York, NY

[7] Lundin, M. and E. Wictorin. 2020. "The Subjectivity of Failure", School of Economics and Management, Lund University, Lund, Sweden.

[8] Munns, A.K. and B.F. Bjeimi. 1996. "The Role of Project Management in Achieving Project Success", 14(2):81-87, Scotland, UK

[9] Cohen, S. 2022. *Lessons Learned from Recovery Procedures for Failing Software Projects*, [In Preparation].

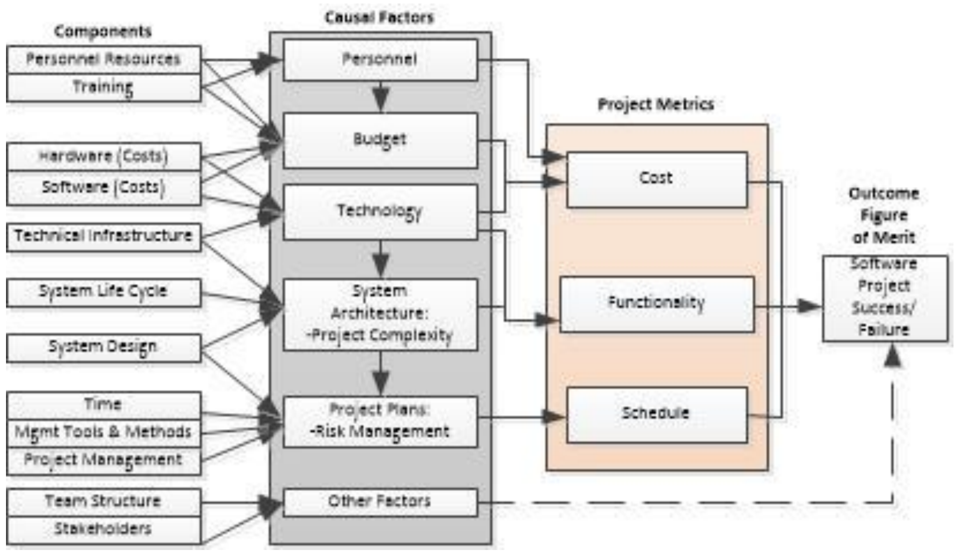
[10] Ackerman, F. and C. Eden. 2005. "Using Causal Mapping with Group Support Systems to Elicit an Understanding of Failure in Complex Projects: Some Implications of Organization Research", *Group Decision and Negotiation*, 14:355-376

[11] Sowa, J.F. 1984. *Conceptual Structures: Information Processing in Mind and Machine*, Addison Wesley, Reading, MA

[12] Armour, F. and G. Miller. 2001. *Advanced Use Case Modeling: Software Systems*, Addison-Wesley, Reading, MA.

[13] Eden, C. 1988. “Cognitive Mapping: A Review,” *European Journal of Operational Research* 36, 1-13.
[14] Powell, S. 2021. Guide to Causal Mapping, Causal Map, LTD., Bath, England
[15] Pearl, J. and D. Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect,* Basic Books, NY.
[16] Ackermann, F., C. Eden, and T. Williams. 1997. “Modelling for Litigation: Mixing Qualitative and Quantitative Approaches”. *Interfaces*, 27():48-65.
[17] Forrester, J. 2009. Some Basic Concepts in Systems Dynamics, D-4894, Sloan School of Management, MIT, Boston, MA
[18] Ventana Systems, Inc. 2015. VENSIM Brochure.
[19] Islam, S., H. Mouratidis, E.R. Weippl, and J. Jurgens. 2014. “An Empirical Study on the Implementation and Evaluation of a Goal-driven Software Development Risk Management Model”, *Information and Software Technology*, 56(2):117-133
[20] Marinho, M., S. Sampaio, T. Lima, and H. de Moura. 2014. “A Guide to Deal with Uncertainties in Software Project Management”, *International Journal of Computer Science & Information Technology* (IJCSIT), 6(5).
[21] U.S. Government Accountability Office (GAO). 1994. Advanced Automation System: Implications of Problems and Challenges, https://www.gao.gov/products/T-RCED-94-188.
[22] DeLone, W.H. and McLean, E.R. 2003. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems*. 19, 4, 9–30.

Figure 1. Kaisler, Money, Cohen Forensic Analysis Framework


Components
Personnel Resources
Training
Hardware (Costs)
Software (Costs)
Technical Infrastructure
System Life Cycle
System Design
Time
Mgmt Tools & Methods
Project Management
Team Structure
Stakeholders
Causal Factors
Personnel
Budget
Technology
System Architecture: -Project Complexity
Project Plans: -Risk Management
Other Factors
Project Metrics
Cost
Functionality
Schedule
Outcome Figure of Merit
Software Project Success/ Failure