Extended Model of Software Quality Assessment Scenario: Concept, Operations, Application

Oleksandr Gordieiev[1], Daria Gordieieva[1], Vyacheslav Kharchenko[2], Inna Kondius[1] and Ievgen Brezhniev[2]

[1] *Lutsk National Technical University, Lvivska Str., 75, Lutsk, 43018, Ukraine*
[2] *National Aerospace University "Kharkiv Aviation Institute", Chkalov Str.,17, Kharkiv, 61070, Ukraine*

Abstract

A software quality assessment (SQA) is a mandatory process in ensuring the required quality of software as part of the overall software development process. The constant development of existing information technologies and the emergence of new information technologies (artificial intelligence, cloud computing, virtual and augmented reality, etc.) and systems increase the requirements for the assessment process and software quality assurance.
Existing approaches to a quality assessment have significant problems: a weak formalization in the planning SQA tasks; a high degree of uncertainty in decision making by the responsible participants of the process; insufficient or redundant information; determining the number of participants in the software assessment process.
Recent publications in open access, which consider the scenario approach in general and in relation to the tasks of assessing the software quality, were analyzed. More detailed attention was paid to the description of the scenario approach for SQA tasks. Existing approaches to formalizing the scenario approach do not take into account all the features of SQA. The purpose of the article is to develop a model of a software quality assessment scenario.
The article proposes a representation and description of the software quality assessment scenario model, which consists of the following 7 elements: initial conditions, input data, actions, transition data, corrective factors, roles, and results. It has been found that a scenario during its life cycle can be in the following states: scenario on paper, pilot scenario, and real scenario. During the transition to each state, sets of scenario elements can change. To formalize such changes, additional operations on the scenario were introduced and formally described: an operation of exclusion and an operation of inclusion. Variants of inequalities of scenario elements sets were considered for the scenario on paper and the pilot scenario.
As a result, the extended model of the SQA scenario was developed and presented. It can be used for software quality assessment based on the software fault injection. And can be considered as universal model for SQA also for applied intelligent systems.

Keywords

Scenario approach, extended scenario model, software quality, software quality assessment, software fault injection, applied intelligent systems

1. Introduction and related works analysis

A software quality assessment is a mandatory process in ensuring the required software quality as part of the overall software development process. The constant development of existing information technologies and the emergence of new information technologies (artificial intelligence, cloud computing, virtual and augmented reality, etc.) and systems increase the requirements for the assessment process and software quality assurance.

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland.
EMAIL: oleksandr.gordieiev@gmail.com (O. Gordieiev); gordeyeva.daria@gmail.com (D. Gordieieva); v.kharchenko@csn.khai.edu (V. Kharchenko); innastk@ukr.net (I. Kondius); e.brezhnev@csn.khai.edu (I. Brezhniev)
ORCID: 0000-0003-2517-9388 (O. Gordieiev); 0000-0003-0474-3138 (D. Gordieieva); 0000-0001-5352-077X (V. Kharchenko); 0000-0003-3416-7506 (I. Kondius); 0000-0003-2073-9024 (I. Brezhniev)

This evolution among the existing approaches and paradigms of software quality assessment has insufficient dynamics, as there are significant problems, including weak formalization in the planning software quality assessment tasks, a high degree of uncertainty in decision-making by responsible participants of the process, lack or excess of necessary initial information, formation of participants group of software quality assessment process. Especially such problems are clearly expressed in the methods of software quality assessment based on the software fault injection.

One of the existing approaches that can formalize the software quality assessment process to the required level is the scenario approach. The analyzed works on the organization and formalization of the software quality assessment process describe some cases in the development of the software quality assessment process [1-4], and the scenario approach is described in part, at the level of some elements [5-7]. There are works on the scenario approach, but it is considered in general as an approach to management [8-12], without taking into account the specifics of the software quality assessment in general. The scenario approach is not conceptually considered in the works on software quality assessment based on the software fault injection [13, 14]. In some papers [15] on software quality assessment, the scenario model is considered in general, but it does not take into account important features of this process.

Thus, **the aim of the article** is to develop an extended model of software quality assessment scenario, which takes into account its elements, features of state change, etc.

2. The concept of scenario

We will present and formally describe a scenario-oriented approach to software quality assessment. First of all, consider the concept of the scenario. The word «scenario» comes from the Latin word «scaena», which translates as «scene». Initially, the scenario was considered as a literary and dramatic work, written as a basis for the production of film or television, and other events in the theater and elsewhere. In the twentieth century, Herman Kahn, a leading analyst at the RAND Corporation [16], adapted this term for use in writing possible stories of future developments. Kahn is often cited as a father of scenario planning. Oliver Sparrow, one of the founders of the scenario approach at the Royal Dutch Shell Corporation, distinguished four modern interpretations of this term [17]:

- as «a sensitivity analysis» whether in cash flow management, broader risk assessment, or project management;
- as synonymous with the concept of «contingency plan» in military or civilian contingency planning, determining who and what to do in the event of an emergency;
- as synonymous with a contingency plan in corporate or public policy;
- in the sense of «logically agreed assumptions about the future» in decision-making and strategy formation.

All the main definitions are summarized by the Dutch researcher Philip Van Notten in the following definition [18]: a scenario is a consistent description of alternative hypothetically possible variants for future events, which reflects different perspectives on the past, present, and future, and which can be the basis for action planning.

3. Formalized representation of the model

Adapting the presented definition of the scenario for software quality assessment, we obtain the following interpretation: a software quality assessment scenario is a product of planning and describing a (continuous) sequence of actions aimed at software quality assessment, which includes a description of initial conditions, inputs, expected results (hypotheses), corrective factors and distribution of participant roles in software quality assessment process. Among the process participant roles are the following: organizer (research engineer) of the software quality assessment process (scenario developer), head of the software testing team (quality team leader), tester (quality engineer), user. Thus, the software quality assessment scenario includes the following 7 elements: actions, transition data transmitted from stage to stage, roles, input data, corrective factors, initial conditions, expected result, or hypothesis. Elements of the scenario are presented in general form: graphic (Fig. 1) and formal form:

- $INCONSCE = \{inconsce_k\}_{k=1}^{p}$ – a set of initial conditions of the software quality assessment scenario (*INCONSCE* – INitial CONditions of SCEnario), inconsce – an initial condition of the software quality assessment scenario;
- $INDASCE = \{indasce_j\}_{j=1}^{f}$ – a set of input data of the software quality assessment scenario (*INDASCE* – INput DAta of SCEnario), indasce – input data of the software quality assessment scenario;
- $ACTSCE = \langle actsce_i \rangle_{i=1}^{q}$ – a set of actions of the software quality assessment scenario (*ACTSCE* – ACTions of SCEnario), actsce – an action of the software quality assessment scenario;

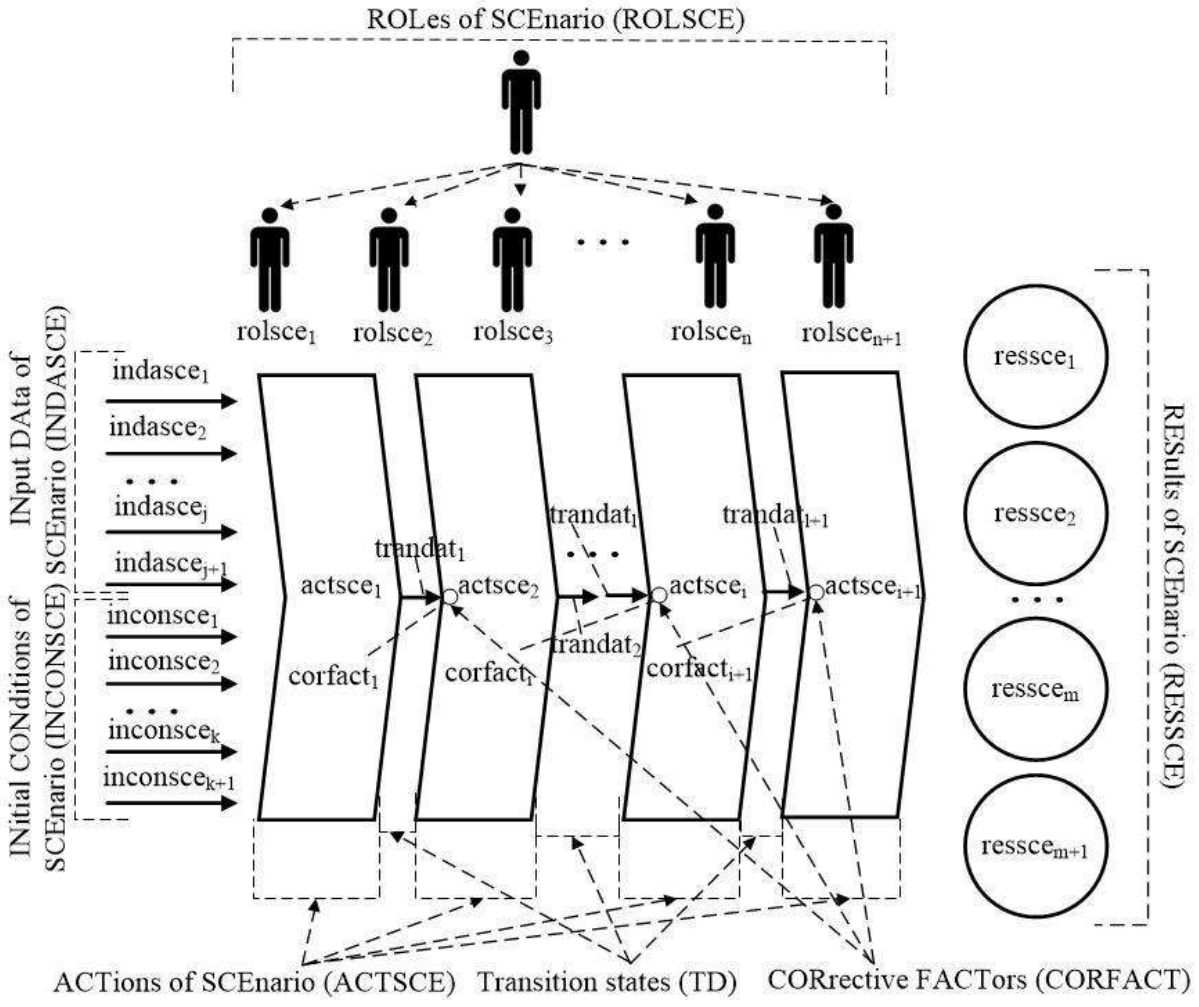


Figure 1: Graphical representation of an extended model of software quality assessment scenario

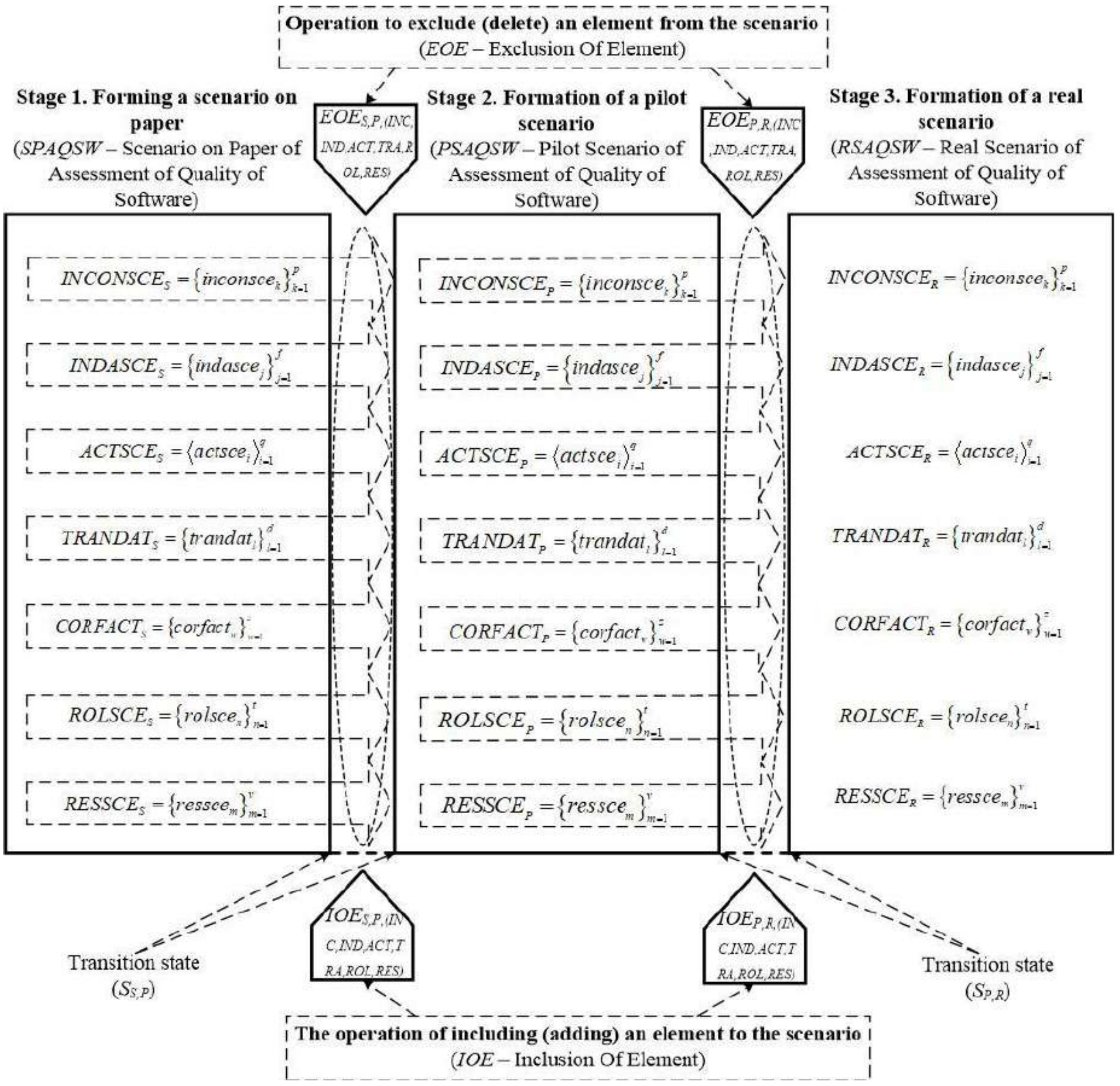
- $TRANDAT = \{trandat_l\}_{l=1}^{d}$ – a set of transition data that is transmitted from action to action, i.e. the output data transit and become the input data for the next action (*TRANDAT* – TRANsition DATa), trandat – transition data that is transmitted from action to action;
- $CORFACT = \{corfact_w\}_{w=1}^{z}$ – a set of corrective factors that clarify the actions in the transition from action to action (*CORFACT* – CORrective FACTors), corfact – a corrective factor;

- $ROLSCE = \{rolsce_n\}_{n=1}^{t}$ – a set of roles of the software quality assessment scenario (*ROLSCE* – ROLes of SCEnario), rolsce – a role of the software quality assessment scenario;
- $RESSCE = \{ressce_m\}_{m=1}^{v}$ – a set of results of the software quality assessment scenario (*RESSCE* – RESults of SCEnario), ressce – a result of the software quality assessment scenario.

Thus, the software quality assessment scenario (*SAQSW* – Scenario of Assessment of Quality of Software) is described as a set of sets (1):

$$SAQSW = \left\{ \begin{matrix} INCONCE, INDASCE, ACTSCE, TRANDAT, \\ CORFACT, ROLSCE, RESSCE \end{matrix} \right\}. \tag{1}$$

Experimentally, it was found that the scenario during its life cycle (Fig. 2) evolves and is presented in the following three states:



Operation to exclude (delete) an element from the scenario
(EOE – Exclusion Of Element)
Stage 1. Forming a scenario on paper
(SPAQSW – Scenario on Paper of Assessment of Quality of Software)
EOE S,P,(INC,IND,ACT,TRA,ROL,RES)
Stage 2. Formation of a pilot scenario
(PSAQSW – Pilot Scenario of Assessment of Quality of Software)
EOE P,R,(INC,IND,ACT,TRA,ROL,RES)
Stage 3. Formation of a real scenario
(RSAQSW – Real Scenario of Assessment of Quality of Software)
INCONSCE_S = {inconsce_k}_{k=1}^p
INDASCE_S = {indasce_j}_{j=1}^f
ACTSCE_S = ⟨actsce_i⟩_{i=1}^q
TRANDAT_S = {trandat_l}_{l=1}^d
CORFACT_S = {corfact_u}_{u=1}^z
ROLSCE_S = {rolsce_n}_{n=1}^t
RESSCE_S = {ressce_m}_{m=1}^v
INCONSCE_P = {inconsce_k}_{k=1}^p
INDASCE_P = {indasce_j}_{j=1}^f
ACTSCE_P = ⟨actsce_i⟩_{i=1}^q
TRANDAT_P = {trandat_l}_{l=1}^d
CORFACT_P = {corfact_u}_{u=1}^z
ROLSCE_P = {rolsce_n}_{n=1}^t
RESSCE_P = {ressce_m}_{m=1}^v
INCONSCE_R = {inconsce_k}_{k=1}^p
INDASCE_R = {indasce_j}_{j=1}^f
ACTSCE_R = ⟨actsce_i⟩_{i=1}^q
TRANDAT_R = {trandat_l}_{l=1}^d
CORFACT_R = {corfact_u}_{u=1}^z
ROLSCE_R = {rolsce_n}_{n=1}^t
RESSCE_R = {ressce_m}_{m=1}^v
Transition state (S_S,P)
IOE S,P,(INC,IND,ACT,TRA,ROL,RES)
IOE P,R,(INC,IND,ACT,TRA,ROL,RES)
Transition state (S_P,R)
The operation of including (adding) an element to the scenario
(IOE – Inclusion Of Element)

Figure 2: Software Quality Assessment Scenario Lifecycle

- «scenario on paper» (*SPAQSW* – Scenario on Paper of Assessment of Quality of Software). This is the first state of the scenario developed by the organizer of the assessment process. To indicate this state of the scenario for each set an index «S» is added (2):

$$SPAQSW = \begin{Bmatrix} INCONCE_S, INDASCE_S, ACTSCE_S, TRANDAT, \\ CORFACT_S, ROLSCE_S, RESSCE_S \end{Bmatrix}; \tag{2}$$

- «pilot scenario» (*PSAQSW* – Pilot Scenario of Assessment of Quality of Software). This is a scenario on paper that runs in test mode. Such a scenario is needed to work out and refine the scenario on a real test case. As a rule, the number of participants involved in the scenario is minimal. Typically, this scenario differs from the scenario on paper by refining the scenario elements. To indicate this state of the scenario for each set an index «P» is added (3):

$$PSAQSW = \begin{Bmatrix} INCONCE_P, INDASCE_P, ACTSCE_P, TRANDAT_P, \\ CORFACT_P, ROLSCE_P, RESSCE_P \end{Bmatrix}; \tag{3}$$

- «real scenario» (*RSAQSW* – Real Scenario of Assessment of Quality of Software). This state of the scenario is used to assess the software quality for the actual object of research. As a rule, it can differ from the pilot scenario due to the clarifications that are made to it in the process of implementation. To indicate this state of the scenario for each set an index «R» is added (4):

$$RSAQSW = \begin{Bmatrix} INCONCE_R, INDASCE_R, ACTSCE_R, TRANDAT_R, \\ CORFACT_R, ROLSCE_R, RESSCE_R \end{Bmatrix}. \tag{4}$$

Thus, we will refine the general record for the software quality assessment scenario based on the state of the scenario and add a «VOS» index for each set, which indicates the state of the scenario (VOS - Variant Of Scenario). Thus, the index «VOS» can have the following values: *S – SPAQSW* – Scenario on Paper of Assessment of Quality of Software, *P – PSAQSW* – Pilot Scenario of Assessment of Quality of Software, *R – RSAQSW* – Real Scenario of Assessment of Quality of Software) (5):

$$SAQSW = \begin{Bmatrix} INCONCE_{VOS}, INDASCE_{VOS}, ACTSCE_{VOS}, TRANDAT_{VOS}, \\ CORFACT_{VOS}, ROLSCE_{VOS}, RESSCE_{VOS} \end{Bmatrix}. \tag{5}$$

4. Operation on the scenario

During its life cycle, the scenario can be refined, i.e. modified. The article does not consider and analyze examples and reasons in which cases the scenario may change, because such material requires more volume of the article and may claim a separate article. It was found that such changes can be reduced to the following two operations on scenario elements:

- exclusion (deleting) a scenario element ($EOE_{VOS,VOS,TEOS}$ – Exclusion Of Element);
- inclusion (adding) a scenario element ($IOE_{VOS,VOS,TEOS}$ – Inclusion Of Element).

A scenario element conversion operation is also possible, but it is not considered because a pair of exclusion-inclusion operations can represent it. When entering additional sets for each of them, the index «TEOS» was added, which indicates the variant of the scenario (*TEOS* – Type of Elements Of Scenario). Thus, the index «TEOS» can take the following values: *INC – INCONSCE* – INitial CONditions of SCEnario, *IND – INDASCE* – INput DAta of SCEnario, *ACT – ACTSCE* – ACTions of SCEnario, *TRA – TRANDAT* – TRANsition DATa, *COR – CORFACT* – CORrective FACTors, *ROL – ROLSCE* – ROLes of SCEnario, *RES – RESSCE* – RESults of SCEnario).

For a more formal description of such changes in the scenario, we introduce additional notation – $S_{VOS,VOS}$, which can be of two types: $S_{S,P}$ – a transition state of the scenario on paper to the pilot scenario, $S_{P,R}$ – a transition state of the pilot scenario to the real scenario. Consider possible variants of inequalities of scenarios and their elements for transition states (Fig. 2).

Formally, we present a description of the operations of exclusion ($EOE_{VOS,VOS,TEOS}$) and inclusion ($IOE_{VOS,VOS,TEOS}$). To do this, enter the following additional sets:

- a set of initial elements from the corresponding set (SOE_{TEOS} – Set of Original Elements). This set includes all elements of the initial scenario to which the corresponding operation will be applied;
- a set of elements that are excluded from the corresponding set ($SEXE_{TEOS}$ – Set of Excluding Elements). Because only one element can be deleted from the set of initial scenario elements when

using an exclusion operation, such set will consist of one element. Although such elements in the set can accumulate when the exclusion operation for the initial scenario is used repeatedly;

- a set of excluded elements from the corresponding set (SEE_{TEOS} – Set of Excluded Elements). Because only one element can be deleted from a set of initial scenario elements when using an exclusion operation, such set will consist of one element, although such elements in the set can accumulate when the exclusion operation for the initial scenario is used repeatedly. That is, the element of the scenario when using the exclusion operation transits from the set of excluding elements to the set of excluded elements;
- a set of resulting elements from the corresponding set (SRE_{TEOS} – Set of Resulting Elements). Such set is formed as the difference between the set of initial elements and the set of excluding scenario elements, or as the sum of the set of initial elements and the set of included elements;
- a set of included elements from the corresponding set (SIE_{TEOS} – Set of Included Elements). This is a set that consists of an element or elements that will be added to the set of initial elements, and such combination of sets forms the resulting set of scenario elements.

In general, the operation of exclusion (deleting) an element ($EOE_{VOS,VOS,TEOS}$) for each state of the scenario is written as follows (6):

$$EOE_{VOS,VOS,TEOS} = \left\{ \begin{array}{l} SRE_{TEOS} = SOE_{TEOS} \setminus SEXE_{TEOS} \\ SEE_{TEOS} = SOE_{TEOS} \cap SEXE_{TEOS} \\ SIE_{TEOS} = \varnothing \end{array} \right\}; \tag{6}$$

and the operation of inclusion (adding) an element ($IOE_{VOS,VOS,TEOS}$) for each state of the scenario is written as follows (7):

$$IOE_{VOS,VOS,TEOS} = \left\{ \begin{array}{l} SRE_{TEOS} = SOE_{TEOS} \cup SIE_{TEOS} \\ SOE_{TEOS} \cap SIE_{TEOS} = \varnothing \\ SEE_{TEOS} = SEXE_{TEOS} = \varnothing \end{array} \right\}. \tag{7}$$

The life cycle of the software quality assessment scenario includes 3 stages, which correspond to its three states (Fig. 2):

1. Stage of forming the «scenario on paper»;
2. Stage of forming the «pilot scenario»;
3. Stage of forming the «real scenario».

Such stages of forming a software quality assessment scenario can be performed only sequentially: at the beginning the stage of forming a scenario on paper, then the pilot and real scenarios.

Moving from stage to stage, the software quality assessment scenario can change during its life cycle. Such changes are a consequence of scenario element exclusion and (or) inclusion operations.

For the transition state $S_{S,P}$ there are two variants of inequalities. The first is when the «scenario on paper» is not equal to the pilot scenario, i.e. $SPAQSW \neq PSAQSW$. If we consider such inequality at the level of elements, then there are variants for equality and inequality of such elements. Consider the variants for inequalities at the level of the elements of the «scenario on paper» and «pilot scenario» and present them as an exhaustive simple search, excluding the variant of complete equality (Table 1). For example, one such variant of inequalities (Table 1, line 22) will be described in more detail. This variant consists of the following ratios: $INCONCE_S \neq INCONCE_P$, $INDASCE_S \neq INDASCE_P$, $ACTSCE_S = ACTSCE_P$, $CORFACT_S \neq CORFACT_P$, $TRANDT_S = TRANDT_P$, $ROLSCE_S \neq ROLSCE_P$, $RESSCE_S = RESSCE_P$. Since among the inequalities there are equations that indicate the identity of the elements, we will consider and describe only the following inequalities: $INCONCE_S \neq INCONCE_P$, $INDASCE_S \neq INDASCE_P$, $CORFACT_S \neq CORFACT_P$, $ROLSCE_S \neq ROLSCE_P$. Since the inequalities of the scenarios indicate the use of the operation of inclusion or exclusion, we will describe in more detail the following inequalities for both operations:

Table 1
Variants of inequalities of sets of elements for «scenario on paper» and «pilot scenario»

№	$INCONCE_{S(P)}$, $INCONCE_{P(R)}$	$INDASCE_{S(P)}$, $INDASCE_{P(R)}$	$ACTSCE_{S(P)}$, $ACTSCE_{P(R)}$	$CORFACT_{S(P)}$, $CORFACT_{P(R)}$	$TRANDAT_{S(P)}$, $TRANDAT_{P(R)}$	$ROLSCE_{S(P)}$, $ROLSCE_{P(R)}$	$RESSCE_{S(P)}$, $RESSCE_{P(R)}$
1.	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
2.	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$=$
3.	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$\neq$
4.	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$=$
5.	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$\neq$
6.	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$=$
7.	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$=$	$\neq$
8.	$\neq$	$\neq$	$\neq$	$\neq$	$=$	$=$	$=$
9.	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$\neq$
10.	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$=$
11.	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$=$	$\neq$
12.	$\neq$	$\neq$	$\neq$	$=$	$\neq$	$=$	$=$
13.	$\neq$	$\neq$	$\neq$	$=$	$=$	$\neq$	$\neq$
14.	$\neq$	$\neq$	$\neq$	$=$	$=$	$\neq$	$=$
15.	$\neq$	$\neq$	$\neq$	$=$	$=$	$=$	$\neq$
16.	$\neq$	$\neq$	$\neq$	$=$	$=$	$=$	$=$
17.	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$\neq$	$\neq$
18.	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$\neq$	$=$
19.	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$=$	$\neq$
20.	$\neq$	$\neq$	$=$	$\neq$	$\neq$	$=$	$=$
21.	$\neq$	$\neq$	$=$	$\neq$	$=$	$\neq$	$\neq$
22.	$\neq$	$\neq$	$=$	$\neq$	$=$	$\neq$	$=$
				…			
125.	$=$	$=$	$=$	$=$	$=$	$\neq$	$\neq$
126.	$=$	$=$	$=$	$=$	$=$	$\neq$	$=$
127.	$=$	$=$	$=$	$=$	$=$	$=$	$\neq$
128.	$=$	$=$	$=$	$=$	$=$	$=$	$=$

- if an exclusion operation was applied to elements of the set of initial conditions, actions, corrective factors and roles for the scenario (8-11):

$$EOE_{S,P,INC} = \begin{cases} SOE_{INC} = INCONCE_S \\ SRE_{INC} = SOE_{INC} \setminus SEXE_{INC} = INCONCE_P \\ SEE_{INC} = SOE_{INC} \cap SEXE_{INC} \\ SIE_{INC} = \varnothing \end{cases}; \tag{8}$$

$$EOE_{S,P,IND} = \begin{cases} SOE_{IND} = INDASCE_S \\ SRE_{IND} = SOE_{IND} \setminus SEXE_{IND} = INDASCE_P \\ SEE_{IND} = SOE_{IND} \cap SEXE_{IND} \\ SIE_{IND} = \varnothing \end{cases}; \tag{9}$$

$$EOE_{S,P,COR} = \begin{cases} SOE_{COR} = CORFACT_S \\ SRE_{COR} = SOE_{COR} \setminus SEXE_{COR} = CORFACT_P \\ SEE_{COR} = SOE_{COR} \cap SEXE_{COR} \\ SIE_{COR} = \varnothing \end{cases}; \tag{10}$$

$$EOE_{S,P,ROL} = \begin{Bmatrix} SOE_{ROL} = ROLSCE_S \\ SRE_{ROL} = SOE_{ROL} \setminus SEXE_{ROL} = ROLSCE_P \\ SEE_{ROL} = SOE_{ROL} \cap SEXE_{ROL} \\ SIE_{ROL} = \varnothing \end{Bmatrix}; \quad (11)$$

- if the inclusion operation was applied to elements of the set of initial conditions, actions, corrective factors and roles for the scenario (12-15):

$$IOE_{S,P,INC} = \begin{Bmatrix} SOE_{INC} = INCONCE_S \\ SRE_{INC} = SOE_{INC} \cup SEXE_{INC} = INCONCE_P \\ SOE_{INC} \cap SIE_{INC} = \varnothing \\ SEE_{INC} = SEXE_{INC} = \varnothing \\ SIE_{INC} = \varnothing \end{Bmatrix}; \quad (12)$$

$$IOE_{S,P,IND} = \begin{Bmatrix} SOE_{IND} = INDASCE_S \\ SRE_{IND} = SOE_{IND} \cup SEXE_{IND} = INDASCE_P \\ SOE_{IND} \cap SIE_{IND} = \varnothing \\ SEE_{IND} = SEXE_{IND} = \varnothing \\ SIE_{IND} = \varnothing \end{Bmatrix}; \quad (13)$$

$$IOE_{S,P,COR} = \begin{Bmatrix} SOE_{COR} = CORFACT_S \\ SRE_{COR} = SOE_{COR} \cup SEXE_{COR} = CORFACT_P \\ SOE_{COR} \cap SIE_{COR} = \varnothing \\ SEE_{COR} = SEXE_{COR} = \varnothing \\ SIE_{COR} = \varnothing \end{Bmatrix}; \quad (14)$$

$$IOE_{S,P,ROL} = \begin{Bmatrix} SOE_{ROL} = ROLSCE_S \\ SRE_{ROL} = SOE_{ROL} \cup SEXE_{ROL} = ROLSCE_P \\ SOE_{ROL} \cap SIE_{ROL} = \varnothing \\ SEE_{ROL} = SEXE_{ROL} = \varnothing \\ SIE_{ROL} = \varnothing \end{Bmatrix}. \quad (15)$$

The second variant of inequalities, when the «scenario on paper» is identical to the «pilot scenario», i.e. $SPAQSW = PSAQSW$. Thus, for each set of «scenario on paper» corresponds to the equivalent set of «pilot scenario», i.e. $INCONSCE_S = INCONSCE_P$, $INDASCE_S = INDASCE_P$, $ACTSCE_S = ACTSCE_P$, $TRANDAT_S = TRANDAT_P$, $CORFACT_S = CORFACT_P$, $ROLSCE_S = ROLSCE_P$, $RESSCE_S = RESSCE_P$.

For the transition state $S_{P,R}$ there are the following two variants of inequalities. The first is when the «pilot scenario» is not equal to the «real scenario», i.e. $PSAQSW \neq RSAQSW$.

Such inequality in the set of variants at the level of scenario elements is similar to the inequality of «scenario on paper» and «pilot scenario», given that as the coefficients for the scenario elements the coefficients in parentheses are considered, i.e. instead of the index «S» index «P» is considered, and instead of the index «P» the index «R» is considered (Table 1).

The second variant of inequalities, when the «pilot scenario» is identical to the «real scenario», i.e. $PSAQSW = RSAQSW$. Thus, each of the sets of elements of one scenario is equal to the corresponding set of another scenario, i.e. $INCONSCE_P = INCONSCE_R$, $INDASCE_P = INDASCE_R$, $ACTSCE_P = ACTSCE_R$, $CORFACT_P = CORFACT_R$, $TRANDAT_P = TRANDAT_R$, $ROLSCE_P = ROLSCE_R$, $RESSCE_S = RESSCE_P$.

5. Application of the model

The proposed model can be used to assess the software quality using software fault injection [19]. In particular, in the development and implementation of Fault Injection Testing (FIT) [20], which is used in the Research-and-Production Corporation «RADIY», different fault injection scenarios have been applied to assess the functional safety of FPGA projects and safety related FPGA based information and control systems of the nuclear power plant. Besides, different fault injection techniques and scenario based tools were used during successful certification of FPGA platform RadICS against requirements of Nuclear Regulatory Committee (US NRC).

To perform FIT, various profiles of defects are formed, which are injected in the electronic project, physical module, top-level software in the form of single and multiple defects. This diversity of profiles gives rise to a variety of FIT and quality assessment scenarios.

It should also be noted scenario approach application for software user interfaces usability assessment [21] with use eye-tracking. Elements of such process (including different scenarios) are input data, initial conditions, actions, transition states, corrective factors, results and participants according to them roles.

Extended model of the software quality assessment scenario can be used for applied intelligent systems as to class information systems.

6. Conclusions

An extended model of software quality assessment scenario is presented and formally described in the article. Its using will formalize planning (initial conditions, input data, corrective factors, actions, transition states, roles and results) and scenario execution. These processes take into account possible features of scenario states, transition of scenario from state to state considering possible changes of sets of scenario elements.

Further research should be focused on the development and automation of realization of detailed scenarios for the quality assessment of software and FPGA projects considering cyber security issue and possibilities of injecting vulnerability (similar design faults/defects). Such approach will provide more complete assessing safety of FPGA platform based information and control systems in conditions of threats and insider intrusions. Another direction for research is a combination of profile-oriented and scenario-oriented approaches to a single paradigm and more detail adaptation such the model for applied intelligent systems.

References

[1] Yan, M., Xia, X., Zhang, X. et al. (2019). Software quality assessment model: a systematic mapping study. Science China Information Sciences Journal, Vol. 62, P.1-18. https://doi.org/10.1007/s11432-018-9608-3

[2] Miguel J., Mauricio D. and Rodriguez G. (2014). A Review of Software Quality Models for the Evaluation of Software Products. International Journal of Software Engineering & Applications (IJSEA), Vol.5(6), P. 31-53.

[3] Brandtner, M. (2013) Fostering software quality assessment. In Proceedings of 35th International Conference on Software Engineering (ICSE'2013). P.1393-1396.

[4] Issa, T., Isaias, P. (2022). Usability and Human–Computer Interaction (HCI). In: Sustainable Design. Springer, London. P. 23-40. https://doi.org/10.1007/978-1-4471-7513-1_2

[5] Deissenboeck F, Juergens E, Lochmann K, et al. (2020). Software quality models: purposes, usage scenarios and requirements. In proceedings of the ICSE Workshop on Software Quality, P. 9-14.

[6] Briggs, Ch. M., Matejova, M. (2019). Scenario Planning and Complex Scenario Approach. Disaster Security. P. 38-60.

[7] Koppen, P. J., Mackor, A. R. (2019). A Scenario Approach to the Simonshaven Case. Wiley Online Library. Topics in Cognitive Science. P. 1-20. URL: https://onlinelibrary.wiley.com/doi/10.1111/tops.12429.

[8] Campi, M. C., Garatti, S. (2018). Introduction to the Scenario Approach: book. Society for Industrial and Applied Mathematics and the Mathematical Optimization Society. 116 p.

[9] Ramponi, F. A. (2018). Consistency of the Scenario Approach. Journal on Optimization. Vol. 28(1). P.135-162.

[10] Garatti, S., Campi, M. C. (2019). Learning for Control: a Bayesian Scenario Approach. In proceedings of the IEEE 58th Conference on Decision and Control (CDC'2019). P.1772-1777.

[11] Shumkov, Y.A., Vidovskiy, L.A. (2017). Scenario approach to project management. Polythematic Online Scientific Journal of Kuban State Agrarian University. Vol. 134(10). P. 1-9.

[12] Kononenko I., Sushko H. (2021). Mathematical model of software development project team composition optimization with fuzzy initial data. Radioelectronic and computer systems journal. Vol. 3(99). P. 149-159. URL:https://doi.org/10.32620/reks.2021.3.12.

[13] Soumya, S., Baiju, A. (2014). Software Faults Emulation by Software Fault Injection. International Journal of Computer Applications. Vol. 97. P.9-11.

[14] Cotroneo, D., Madeira, H. (2013). Introduction to Software Fault Injection. Innovative Technologies for Dependable OTS-Based Critical Systems / ed. D. Cotroneo. Springer. P. 1-15.

[15] Oleksandr Gordieiev, Konstantin Leontiev (2020). Software quality assessment scenario model. Technical sciences and technologies Journal. Vol. 3(21), P. 209-219. URL: https://doi.org/10.25140/2411-5363-2020-3(21)-209-219.

[16] Kahn, H. (1967). The Year 2000: A framework for speculation on the next thirty-three year. NY: Macmillan Publishing Company. 432 p.

[17] Sparrow, O. (2000). Making use of scenarios – from the vague to the concrete. Scenario & Strategy Planning. Vol. 2(5). P. 18-21.

[18] Van Notten, Ph. (2005). Writing on the wall: scenario development in times of discontinuity. Florida: Boca Raton. 209 p.

[19] Kooli M., Bosio A., Benoit P., Torres L. (2015). Software testing and software fault injection, 10th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS 2015), P. 1-6, doi: 10.1109/DTIS.2015.7127370.

[20] Kharchenko V., Sklyar V., Odarushchenko O. and Ivasuyk A. (2013). Fault-injection testing: FIT-ability, optimal procedure and tool for FPGA-based systems SIL certification. In proceedings of the East-West Design & Test Symposium (EWDTS 2013). P. 1-5, doi: 10.1109/EWDTS.2013.6673129.

[21] Purwaningsih R., Yenifi I. (2015). Usability Assessment of International Office Website of Diponegoro University with Scenario-Based Usability Evaluation Method and Wammi Method. Internatioanl Journal ComTech: computer, mathematics and engineering applications Vol. 6 (3). P. 329-342.