Technology Aspects of State Explosion Problem Resolving for Industrial Software Design

P.Drobintsev,V.Kotlyarov, I.Nikiforov
Saint-Petersburg State Polytechnic University
e-mail: vpk@spbstu.ru

Abstract – The paper describes technology aspects of proposed solution for resolving of state explosion problem. The main point is in usage of guides which can be created both manually and automatically and allow to reduce state space during trace generation process. The following techniques is described: traceability tracking of requirements, guides generation based on selected criterion, guides analysis in case of problems with traces generation.

Usage of described techniques in verification and testing phases of software development allow to resolve problem of state exposure for industrial projects.

Key words — design specification, state explosion problem, requirements, behavioral tree, guides, guide formalization.

I. INTRODUCTION

One of the main problems in development and testing automation of industrial applications' software is handling of complicated and large scale requirements specifications. Documents specifying requirements specifications are generally written in natural language and may contain hundreds and thousands of requirements. Thereby the task of requirements formalization to describe behavioral scenarios used for development of automatic tests or manual test procedures is characterized as a task of large complexity and laboriousness.

Applicability of formal methods in the industry is determined to a great extent by how adequate is the formalization language to accepted engineering practice which involves not only code developers and testers but also customers, project managers, marketing and other specialists. It is clear that no logic language is suitable for adequate formalization of requirements which would keep the semantics of the application under development and at the same time would satisfy all concerned people [1].

In modern project documentation the formulation of initial requirements is either constructive, when checking procedure or scenario of requirement coverage checking can be constructed from the text of this requirement in natural language, or unconstructive, when functionality described in the requirement does not contain any explanation of its checking method.

For example, behavioral requirements of telecommunication applications in case of described scenario of coverage are constructively specified and assume allow using of verification and testing for realization checking. Non-behavioral requirements are usually unconstructively specified and require additional information during formalization which allows reconstructing the checking scenario, i.e. converting of non-constructive format of requirements specification into constructive one.

II. REQUIREMENT COVERAGE CHECKING

The procedure of requirement checking is exact sequence of causes and results of some activities (coded with actions, signals, states), the analysis of which can prove that current requirement is either covered or not. Such checking procedure can be used as a criterion of coverage of specific requirement, i.e. it can become a so-called criteria procedure. In the text below a sequence or "chain" of events will be used for criteria procedure.

Tracking the facts of criteria procedure coverage in system's behavioral scenario (hypothetical, implemented in the model or real system), it can be asserted that the corresponding requirement is satisfied in the system being analyzed.

Procedure of requirement checking (chain) is formulated by providing the following information for all chain elements (events):

- conditions (causes), required for activating of some activity;
- the activity itself, which shall be executed under current conditions;
- consequences – observable (measurable) results of activity execution.

Causes and results are described with signals, messages or transactions, commonly used in reactive system's instances communications [2], as well as with variables states in the form of values or limitations on admissible values. Tracking states' changes, produced by chains activities, lets observe the coverage of corresponding chains.

Problems with unconstructive formulations of requirements are resolved by development of requirement coverage checking procedures on user or intercomponent interfaces.

Thus, chains containing sequences of activities and events can appear as criteria of requirements coverage; in addition, it is possible that criteria of some requirement coverage is specified not with one, but with several chains.

III. INITIAL DOCUMENTS SPECIFYING APPLICATION REQUIREMENTS

In technical documentation each requirement is usually specified in natural language in one of two ways:

- in form of behavioral requirement, when checking scenario (procedure) of requirement coverage checking can be constructed from the text of this requirement in natural language
- in form of non-behavioral requirement, specifying the contents, structure or some desired feature without explanation of how it can be checked.

Formalization of constructively specified requirements is possible together with effective automated analysis of software requirements and is implemented in VRS/TAT technology [3].

In VRS/TAT technology Use Case Maps (UCM) notation [4] (Fig.1) is used for high-level description of the model, while tools for automation of checking and generation work with model in basic protocols language [5].

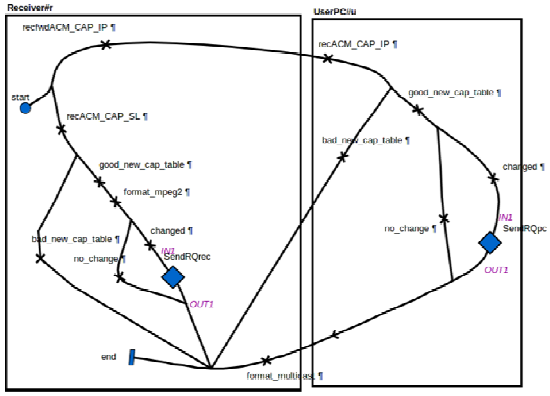


Fig.1 UCM model of two instances: Receiver and UserPC

UCM model (Fig.1) contains two interacting instances model description. Each path on the graph from the event "start" to the event "end" represents one behavioral scenario. Each path contains specified number of events (Responsibilities). Events on the diagram are marked with × symbol, while Stub elements which encode inserted diagram – with ♦ symbol. As a result, each scenario contains specified sequence of events. Variety of possible scenarios are specified with variety of such sequences.

In these terms a chain is defined as subsequence of events which are enough to make a conclusion that the requirement is satisfied. A path on the UCM diagram, containing the sequence of events of some chain, is called trace, covering the corresponding requirement. Based on a trace tests can be generated which are needed for experimental evidence of requirement coverage.

IV. TRACEABILITY MATRIX

Verification project requirements formalization starts with Traceability matrix (TRM) [1] creation (TRM for specific project is presented in table format in Fig.2). "Identifier" and "Requirements" columns contain requirement's identifier, used in the initial document with requirements, and text of the requirement, which shall be formalized. "Traceability" column contains chains of events sufficient for checking of corresponding requirement coverage and "Traces" column – traces or behavioral scenarios used for tests code generation.

Identifier	Requirements	Traceability	Traces
FREQ_GWR.1	Gateway shall transmit ACM_CAP messages repeatedly at an interval of T1 (TBD). This message will carry the ACM Capabilities table.	FREQ_GWR.3	
FREQ_GWR.2	Upon a change in the ACM Capabilities table, the Gateway shall send a new version of ACM_CAP message to the Satellite Terminal.	FREQ_GWR.3	
FREQ_GWR.3	Depending on configuration, ACM_CAP message shall be transmitted either in the MPEG2 private section or in a Multicast message sent across the satellite link.	recACM_CAP_SL _____ recfwdACM_CAP_IP recACM_CAP_IP	FRED_GWR_3-1 FRED_GWR_3-2

Fig.2 TRM – Traceability matrix

For example, in the third row of TRM there are 2 chains in "Traceability" column for covering FREQ_GWR.3 requirement. To satisfy the requirement it is enough to trace ACM_CAP signal sending in one of two possible scenarios:

FREQ_GWR.3-1: start, **recACMCAP_SL**, good_new_cap_table, format_mpeg2, no_chanes, end

FREQ_GWR.3-2: start, **recfwdACM_CAP_IP, recACM_CAP_IP**, good_new_cap_table, format_multicast, end.

It should be noted, that during formulating of criteria chains a model of verified functionality is being created which introduces a lot of state variables, types, agents, instances, etc.

V. DEVELOPING INTEGRAL CRITERIA OF REQUIREMENTS COVERAGE

Mentioned above is disticntive feature of VRS/TAT technology – special criteria of each requirement's coverage checking. Below all criteria related to requirements are listed in ascending order of their strength:

1. Events criterion - coverage level in generated scenarios of subset of events used in criterial chains.
2. Chains criterion - coverage level in generated scenarios of subset of chains (consisting of events and states of variables) with at least one for each requirement.
3. Complex criterion - coverage level in generated scenarios of the whole set of chains specifying integral criteria (combined from criteria 1 and 2) of requirements coverage.

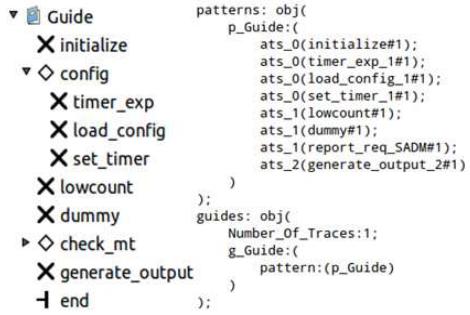
Criteria development shall be adaptive to specific project. Criteria shall be applied flexibly and can be changed according to conditions of scenarios generation.

VI. GENERATING AND SELECTING SCENARIOS WHICH SATISFY TO SPECIFIED INTEGRAL COVERAGE CRITERIA

Trace generation is performed by symbolic and concrete trace generators STG (SimbolicTrace Generator) and CTG (Concrete Trace Generator), which implements effective algorithms of Model Checking. The main problem of trace generation is "explosion" of variants combinations while generating traces from basic protocols, which formalize scenario events, conditions of their implementation and corresponding change of model state after their implementation. Solution here is filtration of generation variants based on numerous limitations specifically defined before trace generation cycle.

There are general and specific limitations. For example, commonly used general limitations are maximum number of basic protocols used in a trace and maximum number of traces generated in a single cycle of generation. Specific limitations are defined by sequences of events in UCM model which guide the process of generation in user-preferable model behavior (so-called Guides). Used are two steps of test scenarios generation by Guides. On the first step guides are created which guarantee specified criteria of system behavior coverage. On the second step guides in UCM notation (Fig.3a) are translated into guides on basic protocols language (Fig.3b) and control trace generation

process. It is important, that only main control points in behavior are specified in guides, while the trace generated from the guide contains detailed sequence of behavioral elements. Such approach to generation significantly reduces the influence of combinatorial explosion on the time of generation during exploring behavioral tree of developed system.



(a) (b)

Fig.3. Guides: (a) in UCM notation, (b) in basic protocols language (VRS Guide Language)

VII. AUTOMATIC AND MANUAL PROCESSES OF GUIDES CREATION

There are two possible approaches to guides creation from high-level system description in UCM language: manual and automatic.

Automatic approach allows to generate numerous guides covering system behavior on branches criteria [6]. Each guide contains information about key points of behavioral scenario, starting from initial model state modeled by StartPoint element and ending in final state modeled by EndPoint element. Process of guides generation is performed by UCM to MSC generator [7].

Automatic approach to guides creation can be considered as fast way to obtain test set which satisfies branches criteria, but such approach is not always suficient. Customer and test engineer may want to check specific scenarios of system behavior for checking some specific requirements. Such scenarios are specified manually by engineer and they are created using UCM Events Analyzer (UCM EVA) tool [8].

In both cases, automatic or manual, problems may occur with guides' coverage by test scenarios.

In automatic mode this is due to the fact that VRS tool not always can successfully generate test scenario form guides because guides are created based on model's control flow and do not consider values of corresponding data flow. At the same time, traces generated based on control flow consider changes in variables values and accordingly model states.

Therefore the actual task is automated analysis of why some guides are not covered by traces and accordingly automation of guides adjusting solved by guides or UCM model modification.

VIII. METHOD OF GUIDES ADJUSTMENT AUTOMATION

Most often reasons of discrepancies are insufficient (not enough detailed) guides specification and mistakes in the sequence of UCM elements in the guide due to incorrect usage of variables, identified as a deadlock on the branch or ramification.

Consider the method of searching of places and reasons of discrepancies between a guide and a trace. Fig.4 shows iterative algorithm of errors searching and fixing automation.

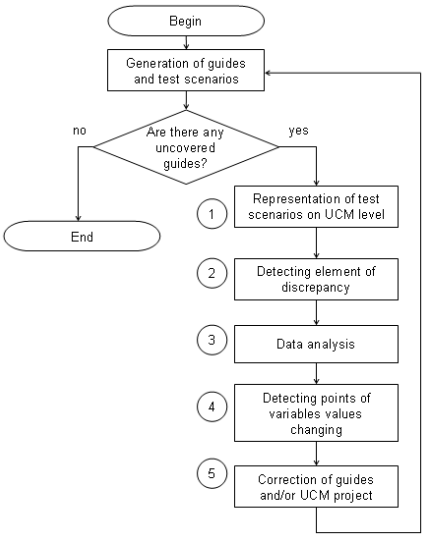


Fig.4. Algorithm of searching and fixing errors in guides

(1) Guides and traces generated in VRS are presented as MSC diagrams containing sequences of basic protocols application, thus the first step of the algorithm is mapping basic protocols names on UCM elements.

(2) Comparing the guide and the trace in terms of UCM elements it is possible to define the last trace element which satisfies the sequence of UCM elements specified in the guide. The next uncovered element of the guide will be referred to as the element of discrepancy.

(3) For the element of discrepancy uncovered in the symbolic trace it is possible to explore corresponding data and precondition.

(4) Then variables of precondition shall be singled out into separate list and those places on the UCM diagram where variables of this list are changed shall be analyzed. The analysis shall be performed from the bottom up, starting with the events closest to the element of discrepancy.

(5) After revealing the reason of discrepancy, the guide shall be corrected or the UCM model shall be changed.

The steps above shall be repeated until all guides will be covered by traces.

Consider the process of searching for discrepancy on the example of telecommunication project (Fig. 5).

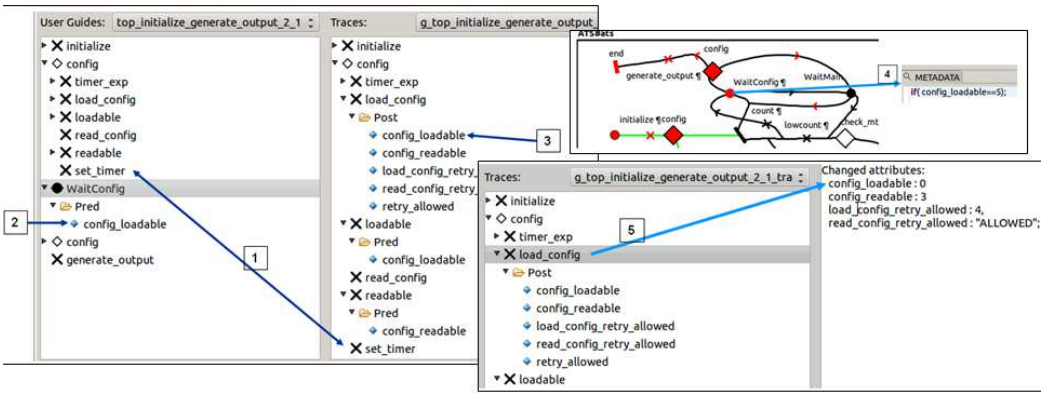


Fig.5. Revealing the reason of discrepancy between the guide and the trace due to variables values

While searching for the reason of guide's non-coverage it is firstly required to find the last element of coincidence between the guide and the trace (1) – **set_timer** in this example. Then the guide element which can not be achieved in the trace (the discrepancy element) shall be found – **WaitConfig** in this example. Analyzing its metadata (2), detect the variable which affects the trace generation (3) and can be the reason of discrepancy - **config_loadable** in this example. After analysis of this variable's values in the UCM model (4) draw a conclusion that in order to apply this element the variable's value shall be **5**. The analysis is held only for those points in the trace where **config_loadable** is used. In current case such point is **load_config** element (5), where **0** is assigned to **config_loadable** variable. Thus the conclusion can be made that in order to create a guide which will be successfully covered by a trace it is required to assign value 5 on **load_config** element or change the guide.

Achieved is the reduction of laboriousness in searching for the reasons of errors due to decreasing of the number of points being analyzed which is actual for large industrial projects created in accordance with considered technology.

IX. FORMAL DEFINITIONS OF THE MODEL AND GUIDES LANGUAGE

In [9] the guided search method is described which is used for generation of test scenarios satisfying to specified coverage criteria. Used coverage criteria (Guides) are specified in the form of special regular expressions over the model's transition names alphabet. Guides in abstract way specify the sought-for behaviors in terms of model and simultaneously restrict the number of analyzed states by cutting off the behavioral branches which do not satisfy to specified criteria.

Definition 1. Transitive system M is the tuple <Q, q0, T, P, f>, where Q – the set of states, $q_0 \in Q$ – the initial state, T – the set of names of parameterized transitions, P – the set of agents, f:Q $\rightarrow$ P – the map, specifying the actual set of agents in Q state.

To keep connection with initial UCM model of the system being analyzed associate the model's events with the names of its transitions. Parameterization is especially actual for distributed systems composed of parallel asynchronous processes which can be generated and eliminated dynamically. The agent can be presented by one or a set of processes [10].

Definition 2. A path in M from qi state to qj state is such sequence of states and transitions

$q_i \xrightarrow{t_i(a_i)} q_{i+1} \xrightarrow{t_{i+1}(a_{i+1})} q_{i+2} \ldots q_j$, that $q_k \in Q \wedge t_k \in T \wedge a_k \in f(q_k)$ for each $k \in i..j$.

Definition 3. A trace in M is the sequence t0(a0), t1(a1), … tn(an)… such that the path $q_0 \xrightarrow{t_0(a_0)} q_1 \xrightarrow{t_1(a_1)} \ldots \xrightarrow{t_n(a_n)} q_n \ldots$ exists

Definition 4. The language associated with M is denoted as $L(M) \subset L^*$ – this is the set of all traces coming from initial state q0.

Definition 5. Guide a.n is the transition a on maximal distance n, which allows the set of traces {a, X1a, …, X1…Xn a}, where X1,…,Xn – any non-empty symbols from {L\a},

~ a – restriction of a transition, allows any symbol from {L\a},

a; b – (where a, b – guides) guides concatenation, allows the set of traces {ab},

a ∨ b – (where a, b – guides) nondeterministic selection of guides, allows the set of traces {a, b},

a || b – parallel composition of the guide a, which represents Za language over the X set alphabet, and the guide b, which represents Zb language over the Y set alphabet, herewith $X \cap Y = \varnothing$, because the sets of agents do not intersect in X and Y. Then the parallel composition of guides is the set represented by $Z^{a\|b} = Z^{a}_{\Uparrow Y} \cap Z^{b}_{\Uparrow X}$ language

join(a1,…,an) – the set {Sn} of all combinations of guides a1,…,an,

loop(a) – iteration of the guide a, i.e. $\{aa^*\}$.

The features of the operations listed above are described in details in [11].

Semantically guides specify the "control points" in model's behavior and also specify the criteria (a chain of events in model's behavior) of selection of generated traces for their further usage as test scenarios [12, 13]. Supposed scenarios of modeled system's behavior are checked for admissibility, simultaneously restricting their search.

X. RESULTS OF DEPLOYMENT IN PILOT PROJECTS

Table 1 contains the results of deployment of design and testing integrated technology in the area of wireless telecommunication applications development. As a result, a

significant reduction of laboriousness and increase of software quality were obtained.

Table 1. RESULTS OF TECHNOLOGY DEPLOYMENT IN PILOT PROJECTS.

Project	Number of requirements	Number of basic protocols	Requirements coverage level	Number of found and fixed errors (overall/critical)	Efforts (human-weeks)
Module 1 of wireless network (WN)	400	127	75%	142/11	5.6
Module 2 of WN	730	192	80%	106/18	12
High-level module of WN	148	205	100%	68/23	11
Clients and administrator connection module of WN	106	163	100%	42/8	6
Mobile phone software module	200	170	100%	96/10	7

XI. CONCLUSION

The result of the work is improved integrated technology of verification and testing of software projects which provides:

1. Full automation of industrial software product development process with requirements semantics implementation control.
2. Generation of application's model and symbolic behavioral scenarios, which cover 100% of application's behavioral features.
3. Automated concretization of symbolic traces in accordance with test plan.
4. High level of software development and quality management process automation.

REFERENCES

[1] S.Baranov, V.Kotlyarov, A.Letichevsky. Industrial technology of mobile devices testing automation based on verified behavioral models of requirements project specifications// «Space, astronomy and programming» – SpbSU, Spb. – 2008. – pp. 134–145. (in Russian)

[2] Z.Manna, A.Pnueli.: The Temporal Logic of Reactive and Concurrent Systems. Springer-Verlag, 1992.

[3] S.Baranov, V.Kotlyarov, A.Letichevsky, P.Drobintsev. The technology of Automation Verification and Testing in Industrial Projects. / Proc. of St.Petersburg IEEE Chapter, International Conference, May 18-21, St.Petersburg, Russia, 2005 – pp. 81-86

[4] Recommendation ITU-T Z.151. User requirements notation (URN), 11/2008

[5] A. Letichevsky, J. Kapitonova, A. Letichevsky Jr., V. Volkov, S. Baranov, V. Kotlyarov, T. Weigert. Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. Proc of ISSRE04 Workshop on Integrated-reliability with Telecommunications and UML Languages (ISSRE04:WITUL), 02 Nov 2004: IRISA Rennes France.

[6] P.Drobintsev, V.Kotlyarov, I.Chernorutsky. Test automation based on user scenarios coverage. "Scientific and technical sheets", SpbSTU, vol.4(152)-2012, pp.123-126 (in Russian)

[7] I.Anureev, S.Baranov, D.Beloglazov, E.Bodin, P. Drobintsev, A.Kolchin,, V.Kotlyarov, A. Letichevsky, A. Letichevsky Jr., V.Nepomniashiy, I.Nikiforov, S.Potienko, L.Priyma, B.Tytin. Tools for support of integrated technology for analysis and verification of specifications telecom applications // SPIIRAN proceedings- 2013-№1-28P.

[8] I.Nikiforov, A.Petrov, V.Kotlyarov. Static method of test scenarios adjustment generated from guides // "Scientific and technical sheets", SpbSTU, vol.4(152)-2012, pp. 114-119 (in Russian)

[9] A.Kolchin, V.Kotlyarov, P. Drobintsev. A method of the test scenario generation in the insertion modelling environment // "Control systems and computers", Kiev: "Akademperiodika", vol.6-2012, pp.43-48 (in Russian)

[10] A.A. Letichevsky, J.V. Kapitonova , V.P. Kotlyarov, A.A. Letichevsky Jr., N.S.Nikitchenko, V.A. Volkov, and T.Weigert. Insertion modeling in distributed system design // Programming problems. – 2008. – pp. 13–38

[11] A. Letichevsky Jr., A. Kolchin. Test scenarios generation based on formal model // Programming problems. – 2010. – № 2–3. – pp. 209–215 (in Russian)

[12] V.P. Kotlyarov. Criteria of requirements coverage in test scenarios, generated from applications behavioral models // "Scientific and technical sheets", SpbSTU. – 2011. – vol.6.1(138). – pp.202–207. (in Russian)

[13] Baranov S., Kotlyarov V., Weigert T. Varifiable Coverage Criteria For Automated Tesdting. SDL2011: Integrating System and Software Modeling // LNCS. –2012. –Vol.7083. – P.79–89.