METHODOLOGY OF THE LEGACY SOFTWARE PORTING

Jurij, Agalakov
FSUE "SRIAE" [1]
Moscow, Russia

Konstantin, Kotov
FSUE "SRIAE" [1]
Moscow, Russia

German, Oganyan
FSUE "SRIAE" [1]
Moscow, Russia

ABSTRACT

In the report is presented the methodology of the legacy software porting which main aspect is the formal verification of the legacy software for the conformance to the standards profile of the application platform. This methodology is applicable to the wide range of the problems frequently comes in practice work and permits well to automates the process of the porting and guarantee his correctness. As a method of the formal verification is proposed to use the method of the bounded model checking added with the technology of the software contracts.

Keywords

porting, legacy software, formal verification, bounded model checking, software contracts

1. INTRODUCTION

Todays situation in the domain of the information technologies is characterized by use of the hardware setting on the base of different processors architectures systems as IBM, Sun, Intel and others. The processors architectures are in process of continuous development and perfection. Recently especially intensively develops the technologies of support of the parallel calculations.

As result of stated tendency is that independently of the domain of application of the information systems more often appears the task for the replacement and modernization of the hardware which its included. And it, in course, leads to the necessity of the porting on the new platforms of the all range of software of these information systems. Software for which is posed the problem of porting will be named legacy. The problem of porting appears both for the middleware and for the application software. In his place systems software in the most of cases strongly dependant of the hardware, easier to develop anew or to change completely, while the redevelopment of all a quantity of application software and middleware for the new platform isnt often efficient which is connected both with the large volume of application software and that the dependence of the application software and middleware on the platform well less that of systems software.

2. STATEMENT OF A PROBLEM OF THE PORTING

To pose exactly the problem of the porting we will examine the Open System Environment Reference Model (OSE RM) [1] which includes 3 basic entities: application software entities, application platform entities and external environment entities. Application platform realizes the set of services for which through the corresponding application programming interface (API) call the application software. The set of services of the OSE RM includes the system services, communication services, information services and human-computer services. Under external environment is realized the set of the external for the application platform entities with which interaction accomplish also through some set of services.

Within the framework of a viewed model under the problem of legacy software porting we will realize the problem of the ensuring of executing of the legacy software in the context of the new application platform and new environment. The platform for which the software was developed first we will determine as a source platform, but the platform on which software should be transferred as a target platform [2].

According to the main subject of the porting mark out its following forms:

- source porting;
- binary porting.

Then under the porting we will realize in general the source porting.

The object of legacy software porting can be determined generally by the one of two methods:

- method of emulation when on the target platform are implemented failing services which have the same interface like on the source platform;
- method of adaptation when the source code of the software unit is modified with the object of the proper use the interfaces of target platforms services.

In view of this the software adaptation is possible only in case if the source or target platform conforms to the same standards profile. Otherwise its necessary, based on the requirements of software, either redevelopment it, or emulate completely the source platform on the target platform.

At the moment are presented the highly developed standards which provide the software portability between the wide set of platforms. The largest expansion received the POSIX (Portable Operating System Interface) standard in which is described the interface of system calls and C library functions, as well the utility programs and the possibilities of command interpreter. POSIX supports applications portability at the source code level that demand the recompiling it on the new platform. More and more popularity comes to the LSB (Linux Standard Base) standard. This standard

[1]Federal State Unitary Enterprise "Scientific Research Institute of Automatic Equipment"

in difference with POSIX at the some cases provide the applications portability on the level of binary code. POSIX as LSB standard develop sufficiently dynamic: the latest version of these standards relates to 2008 years.

So, because of the everywhere use of the standards on the application platform services, the largest interest presents the examination in particular of those cases when the porting of software realizes between the platforms which conform to the same standards profile. In this case the main problem to solve by the porting is the problem of the verification of the legacy software for the conformance of the standards of the application platform. Under the verification generally is realized the examination of the conformance of some created during development and maintenance of software artifacts to another which were created earlier or used as source data, and also the conformance of those artifacts and processes of theirs development for the principles and standards [3]. If by the verification will be determine the discrepancy of the porting software to the standards profile, the methods of verification should fix those fragments of the source code of software, which adaptation will permit to attain this conformance.

Recently during the software development more dissemination gets the formal methods of verification. During the verification by the formal methods is realized the comparison between the formal model of tested properties of artifact and of the formal model of the artifact itself. The model of the tested properties is accepted to term specification, and the model of the tested artifact as implementation [3].

The formal verification as compared with the methods of dynamic verification and the examination of the source code usually used by the porting permit well to automates the process of verification and to guarantee the conformance between the implementation and the specification. Accordingly is presented well-grounded to use the formal methods for the software verification by its porting.

3. SOFTWARE VERIFICATION FOR THE CONFORMANCE TO THE STANDARDS PROFILE

After tested the methods of the formal verification, it was achieved that for the solution of the portings problem the method of the model checking is more suitable [4]. This method in contradistinction to the other methods permits to fully automate the process of verification and in case of inconsistency between implementation and specification allow displaying exactly the sequence of the steps of the programs algorithm which may result to the breach of specification. The temporal logic is used in the method for the description of the checking properties and like a model, which properties is tested, the Kripke structure which is a variety of the finite automate [4].

The main problem concerned with an application of model checking is the exponential increase of the number of the Kripke structures states during the extension of the verification programs complication state explosion problem. So because in the report is proposed to use during the porting process the modified method of the model checking method of the bounded model checking (BMC) [5], as its sufficiently effective even in cases of the large number of models states and comes well for verification just of that class of requirements to the software which appear most often by the porting [6].

Bounded model checking is based on the solution of the boolean satisfiability problem (SAT problem) [7]. It means transformation of program model and the specification to propositional logic formula which is checked on satisfiability. The process of construction of the program model in BMC is iterated and is carried out until the resulting model does not become relevant to the initial program [5].

Despite doubtless advantages of the BMC method, its application at the legacy software verification is interfaced to following difficulties:

- the method does not support procedures and functions of programming languages that leads to excessive complexity of verified programs models;
- necessity of construction not only program and specification model, but also environment model;
- necessity of the description is direct in the source code the specification of program.

For overcoming of these difficulties is offered to add a method of verification with the technology of program contracts [8]. The program contract represents the description of functions prototypes, structures of their conditions, and also preconditions and postconditions for each function. The set of all preconditions is treated as the program specification while postconditions set model of a program environment. For use of program contracts at verification is necessary to expand program model logic into logic of the uninterpreted functions possessing property of congruence [7]. Thus it is necessary to notice that programming language functions generally do not possess of the congruences property and therefore at modeling is required to transform function calls in appropriate way. Introduction in language of logic of uninterpreted functions essentially does not complicate the algorithm of verification as from the language of logic with uninterpreted functions is possible to pass always by means of enough simple transformation to equality logic, having kept thus property of satisfiability of formulas in these languages [7].

As the functions are completely defined by the prototypes, at construction of models is possible to compare automatically to each function call of its preconditions and postconditions. Thereby necessity for modification of a source code of porting software disappears.

As to implementation of method BMC is necessary to notice that existing software of verification on the basis of the given method use the built in tools of lexing and parsing of a source code. Thereby the given software enough difficult give in to the integration with widely used in practice development environments and support only limited number of programming languages.

It is represented optimum the approach at which the analysis of a source code is carried out by means of compiler, and the result of this analysis - an abstract syntax tree is represented, is transferred to the verification tools built in the compiler which actually on the basis of a syntax tree generate a resulting propositional logic formula and further transfer it to tools of check of its satisfiability.

4. APPROBATION OF METHODOLOGY

The methodology of the legacy software porting resulted in article has been approved at carrying over of the following, written in C and C++ programming languages, middleware and application software on perspective hardware-software

platforms of the Russian working out:

• failure-safe multiple computer complex software, providing performance of computational process in a duplication mode on the three units of computer complex with synchronisation of results between the core and subordinated computing units and with use of the third computing unit as a hot reserve;
• technological processes management software of enterprise level, proceeding in a real time.

At porting was revealed the incomplete conformity to POSIX of legacy software, in particular the incorrect call a functions of input-output multiplexing. Input-utput multiplexing is used as in the failure-safe multiple computer complex software for the interaction organisation on a network of the program components distributed on three computing units, and in technological processes management software for data exchange between client and server components given software. Implementation of multiplexing, both on source, and on target platforms, is provided with function entering into POSIX `select`. The problem consisted that on a source platform value of the maximum time-out of input-output operations, are given to the function `select`, does not change after function execution, and that time as on a target platform this value decreases. And as legacy software has been written in the assumption of an invariance of a time-out that contradicts POSIX at its start on a target platform astable work of software was observed. Thus it was represented defensible to use formal verification for revealing of those fragments of a source code in which the call a function `select` was incorrect.

For software verification, according to the technology of program contracts stated above, on the basis of the text of POSIX, was developed the specification of function `select` and environment model which is defined by this function. The specification and environment model were set by following logic formulas for precondition P and postcondition R of function `select`:

$$\begin{aligned} P := \\ (n > 0) \wedge (t_0 \geq 0) \quad \wedge \\ \forall n, set, t, S. \quad \neg t_0 = select^{(2)}(n, set, t, S) \end{aligned} \tag{1}$$

$$\begin{aligned} R := \\ r = select^{(0)}(n, set_0, t_0, S) \quad \wedge \\ set_1 = select^{(1)}(n, set_0, t_0, S) \quad \wedge \\ t_1 = select^{(2)}(n, set_0, t_0, S) \quad \wedge \\ r \geq -1 \quad \wedge \\ r = -1 \Rightarrow (set_1 = set_0) \quad \wedge \\ r > -1 \Rightarrow (\sum_i set_1[i] = r) \quad \wedge \\ t_1 \geq 0 \wedge t_1 \leq t_0 \end{aligned} \tag{2}$$

, where n — the maximum number of descriptor, i.e. the identifier of an input-output device;
r — the returned value characterising result of function execution;
set_0, set_1 — given to and returned by function list (a bit array) descriptors;
t_0, t_1 — given to and returned by function time-out value;
S — the set of conditions, not presented to the prototype, influencing result of function execution;
$select^{(0)}$, $select^{(1)}$, $select^{(2)}$ — the functions calculating returned value r, the list of descriptors and the time-out value accordingly.

Further, by support of the instrumental tools of the verification implemented a method of bounded model checking, for each legacy software unit on the basis of its source code and a postcondition (2) the model of the given unit M has automatically been constructed. The algorithm of models construction was taken from [9]. The model M represents the formula over the bit vectors and uninterpreted functions united with the specification, i.e. with the formula of a precondition (1), in a resulting formula of a following kind [5]:

$$M \wedge \neg P \tag{3}$$

After formula (3) transformation in the equivalent formula of propositional logic for resulting formula the SAT problem solved. The satisfiability of the formula testified that the software unit described by model M, does not correspond to specification P and for the successful porting requires adaptation. In this case the source code of the software unit was modified and repeatedly passed verification process. In a situation of an unsatisfiability of the formula (3) it was guaranteed that the problem of incorrect use of multiplexing function, in the software unit, is absent.

Similarly, the verification process resulted above was carried out and for all other POSIX functions the using which of the legacy software units didnt conform to the standard.

5. CONCLUSION

Summing up it is possible to allocate following high lights of offered methodology of the legacy software porting:

• at porting it is necessary to use formal verification of the legacy software for the conformance to the standards profile of the application platform;
• as the basic method of verification in the porting process the use of a method of bounded model checking, added with the technology of the software contracts, is justified;
• it is expedient to integrate verification tools in widely used development environments, in particular, into compilers.

The approbation of the methodology of porting on a wide range of software has shown that the formal verification methods essentially reduce the laboriousness of the porting process and provide high degree of reliability of software on target platform.

REFERENCES

[1] *ISO/IEC TR 14252:1996. Information technology - Guide to the POSIX Open System Environment (OSE).*

[2] James D. Mooney. Developing portable software. In *IFIP Congress Tutorials*, pages 55–84, 2004.

[3] Kuliamin Victor. Methods of software verification. Technical report, Institute for System Programming, Russian Academy of Sciences, 2007.

[4] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, 2000.

[5] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in Computers*, 58:118–149, 2003.

[6] Vijay D'Silva, Daniel Kroening, and Georg Weissenbacher. A survey of automated techniques for formal software verification. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(7):1165–1178, 2008.

[7] Daniel Kroening and Ofer Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 2008.

[8] Bertrand Meyer. Applying "design by contract". *IEEE Computer*, 25(10):40–51, 1992.

[9] Edmund M. Clarke, Daniel Kroening, and Karen Yorav. Behavioral consistency of c and verilog programs using bounded model checking. In *DAC*, pages 368–371, 2003.