[5] Ye, Cang, and Johann Borenstein. "Characterization of a 2D laser scanner for mobile robot obstacle negotiation." Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. IEEE, 2002. - Vol. 3.

[6] Duchon, F., Dekan M., Jurisica, L. and Vitko A. "Some applications of laser rangefinder in mobile robotics." Journal of Control Engineering and Applied Informatics 14.2 (2012): 50-57.

[7] Laser distance sensors optoNCDT ILR 1030/1031. Micro-Epsilon Messtechnik. Available on-line: <http://www.micro-epsilon.com/ displacement-position-sensors/-laser-distance-sensor/optoNCDT_ILR_ 1030_1031 /index.html>

LANGUAGE OF DESIGN OF MODEL OF BEHAVIOR OF COMPLEX SYSTEM

Harchenko S.L., Bagaev D.V.
Kharkov National University of Radioelectronics
Kharkov, Ukraine
Kovrov State Technological Academy named after V.A. Degtyarev
Kovrov, Russia
khsln@yandex.ru, dmitrybag@gmail.com

Abstract – The paper deals with creating a complex pattern of behavior controls technical system based on the use of the formal language of design. Which provided visualization used in the model of concurrency control system behavior and the possibility of action sequences (the track) with the performance. The resulting pattern of behavior may be useful in the verification and validation of the control program of the technical system.

Keywords – pattern of behavior, a process, asynchronous parallel processes, multiprocessor control system, the graph model of behavior.

I. INTRODUCTION

One of the urgent tasks of software development - is the creation of programs for complex technical systems, such as robot control systems. Of the total list of works carried out in the solution of this problem in the early stages of design, it is possible to allocate the task of creating a model of the future behavior of the robot control system. Perform extensive quality indicators such work can be substantially increased if the design of the management model to use formal language that allows to visualize the execution trace, organization and interaction of parallel processes, using a multiprocessor system.

Besides, complexity of modern control systems demands to provide application of the multiuser operating mode over the project. That provides possibility of simultaneous work on the project to several groups of the independent developers united by one ultimate goal.

Arising needs for resources when performing such works, indicate the need of application of a various automation equipment without which application time implementation of the project much more exceeds admissible intervals of expectation of the end result.

Searches of the complex solution of such complex problems of design led to that one of possible versions of the solution of a task is realization of methodology of design which is based on creation of model of behavior of the control system which description can be automatically transformed to sequence equivalent to it (or to some equivalent sequences) operators of one of programming languages.

As the consensus about methodology of creation of the software product in which decisions with use of series-parallel algorithm or functioning are applied as asynchronous parallel processes isn't present, it is possible to choose representation of model of behavior in the form of the count.

Such approach will allow to display through columns all options of branching in algorithm of model of behavior, and action which are correlated to edges of the count, can be written down in any formal language. The record form a file script can serve one of such examples of record.

It is obvious that realization of such model of behavior imposes restrictions on a choice of an operating system. If to consider definition "action" as any sequence of executed operators, such approach is characteristic for UNIX of similar operating systems in which such sequence of operators is treated as process. For such OS, performance of process can be considered as execution of operators by virtual car. In this case the virtual cars created by an operating system have opportunity to communicate among themselves through the general resources of the environment of performance of processes.

II. PROBLEM DEFINITION

For today software creation for difficult control systems is one of actual tasks. In the general list of works which are performed at the solution of such task at early design stages, there is a creation of model of behavior of future system. Performance of this work can baht is significantly accelerated if to use formal language at design of model of behavior of a difficult control system. Especially it should be noted need for creation of the mechanism of visualization of designs of language of design which display overlapping at realization of interactions a component of a difficult control system.

One of the main requirements to language of design of model of the behavior, difficult control system, will be that the created designs of language of design of model of behavior have to be suitable for machining. It will allow to transform further by formal rules the description of the designed model of behavior of a difficult control system to sequence of operators of one of programming languages.

Formal language of design of model of behavior the count for visualization of interactions uses a component and branchings of sequences of executed actions in a control system. In such count this or that action of model of behavior is associated with each edge, and tops of the count display current state of process of modeling. On the other hand, the model of behavior needs to be considered as component structure which can декомпозировать any element to an indivisible condition (component). And everyone to a component, at all levels of decomposition, model of behavior it is possible to consider as process in a control system.

I ruled records of action which is associated with an edge or top of the count, are regulated by the interpreter applied in the project for which the set of rules of analysis and interpretation of an entrance stream is in advance defined.

III. Process in behavior model

If we call the count of model of behavior as *P*, and in this count of an edge of the count will designate actions, and condition tops. In this case functioning of model of behavior can be considered as system transitions on actions from one condition (s_i) in another. Generally the model of behavior can be presented as a set of actions of *Act(P)* which *P* process, thus, for any process can execute

$$Act(P) \subseteq Act,$$

where *Act* set of all possible actions.

The choice of a set of actions of model of behavior depends on a situation for which some set of actions is characteristic. If to divide a set of actions on entrance $\alpha?$ days off $\alpha!$ and internal τ i.e. which aren't connected with environment, it is possible to designate an infinite set of names of objects of process (*Names*) which can be entered or removed. Then the set of actions can be defined as

$$Act = \{\alpha? | \alpha \in Names\} \cup \{\alpha! | \alpha \in Names\} \cup \{\tau\}.$$

In this case process in model of behavior can be defined as

$$P = (S, s^0, R),$$

where S – a set of conditions of *P*;

s^0 – an initial condition of *P*, s^0, *S*;

R – look transitions (s_1, α, s_2), where subset of a type of $R \subseteq S \times Act \times S$.

And process of *P*, in behavior model, it is possible to consider as passing of a set of transitions of a type of $s^0 \xrightarrow{a0} s_1 \xrightarrow{a1} s_2$ and performance of actions of *a0*, *a1*, ... Process will continue work until then while there is a transition from *R* and will stop work in the absence of that.

Let's define a set of all actions of process of *P* as *Act*(*P*) from *Act*\{τ}, i.e.

$$Act(P) \overset{def}{=} \{\alpha \in Act \setminus \{\tau\} | \exists (s_1 \xrightarrow{a} s_2) \in R\}.$$

Process will be final, if sets of *S* and *R* have the termination.

The final sequence of *a0, a1, ...* set of *Act* for which there is a sequence of conditions of *s0, s1, s2, ... P* process will be the route of process of *P*. In this case the set of all routes of process can be designated *P* models of behavior as *Tr(P)*.

If to remove all unattainable conditions and all transitions at which there are unattainable conditions, from process of model of behavior the turned-out process of *P'* (the achievable part of process of *P*) will have the same behavior, as well as *P* process therefore they can be considered as identical.

During the work with model of behavior replacements of conditions in the course of *P* are possible. If we replace $s \in S$ with any $s' \in S$ and we will designate process as *P'* which turns out from *P* by replacement of *s* by *s'* in sets of *S* and *R*, transition of a type of $s^a \to s_1$ can be replaced with transition of a type of $s'^a \to s_1$. In this case *P'* process will possess the same behavior as *P*. If to make multiple replacement of conditions in the course of *P*, and to consider it as replacement of a subset of conditions of process, display of this process can be presented as $f: S \to S'$ and result will be

$$P' = (S'', (s')^0, R'),$$

where $(s')^0 \stackrel{def}{=} f(s^0)$ and for everyone vapors $s_1, s_2 \in S$ and $\alpha \in Act$

$$(s_1^a \to s_2) \in R \Leftrightarrow (f(s_1)^a \to f(s_2)) \in R'$$

In this case behavior of processes identical, therefore, processes are identical.

IV. OPERATIONS WHICH CAN BE CARRIED OUT OVER PROCESS IN BEHAVIOR MODEL

By consideration of process of modeling it is possible to come to conclusion that over it the limited number of actions can be executed. Now we will consider those actions which can be realized in process.

In case for process in model of behavior there are no transitions, and there is one entrance condition - it call empty and designate as *0*.

Let's consider a case of addition to process of *P* prefix actions. In this case to a set of conditions of process the condition of s which is an initial condition of new process is added. It leads to that to a set of transitions $s^a \to s^0$ transition is added, and it is possible to call the process which has turned out in this case as $\alpha.P$.

If there is an alternative composition, that couple of processes of P_1 and P_2 needs to construct process of *P* which will function also as P_1 or P_2 is supposed. The choice, what branch will be used by *P* process, depends on a choice of the process or on an environment choice.

So, if $P_1 = \alpha?.P_1'$ and $P_2 = \beta?.P_2'$ and the environment can enter the α, but can not enter the β, then *P* must select only possible behavior - P_1. Thereafter, the process *P* can not change its decision. Then the alternative formulation is as follows

$$Pi = (S_i, s_i^0, R_i) \ (i = 1, 2)$$

and a set of conditions of S_1 and S_2 in this case have no general elements.

In this case alternative composition will be

$$P_1 + P_2 = (S, s^0, R),$$

where: - S is $S_1 \in S_2$ to which the new condition of s^0 which will be initial for $P_1 + P_2$ is added;

- R contains all transitions from R_1 and R_2;
- for each transition from R_i $(i = 1, 2)$ a type of $s^0_i, a \to s$, R contains $s^0, a \to s$ transition.

If sets of S_1 and S_2 have the general elements, for $P_1 + P_2$ need to be replaced those elements which enter into S_1 in S_2 and as appropriate to execute S_2 and s^0_2 modification.

As the model of behavior is difficult system, in it can be used and parallel compositions from several interacting a component. If to consider two Sys_1 and Sys_2 systems (subsystems) which are components of one Sys system, i.e.

$$Sys \overset{def}{=} \{Sys_1, Sys_2\}.$$

That behavior of systems can be presented by P_1 and P_2 processes respectively, and behavior of Sys_i $(i = 1, 2)$ as a part of Sys system will be presented to the corresponding P_i. Let's designate $\{P_1, P_2\}$ as process which describes behavior of system. It can be treated as round column P, we will consider thus all transitions from a condition in a condition in the column as instant. The fact of performance of action, in this case, we will fix at the time of transition.

Each entrance or output action of P_i $(i = 1, 2)$ represents result of interaction of P_i with process not entering into set $\{P_1, P_2\}$, or as result of interaction with P_j, where $j \in \{1,2\} \setminus \{i\}$, or it internally process action. If internally process action that P_i $(i = 1, 2)$ transfers it a certain object, and P_j it accepts $P_j (j \in \{1,2\} \setminus \{i\})$.

Each possible option of behavior of process of P_i (i = 1, 2) can compare a thread (in treatment of UNIX OS is a process, i.e. sequence of operators), which we will designate as i. It will allow to define option of behavior of process of i $(i = 1, 2)$, for P_i as a part of processes $\{P_1, P\}$.

If to designate set of all options of behavior of processes as $Beh\{P_1, P_2\}$, each of which corresponds to one of options of functioning $\{P_1, P_2\}$. That can be assumed that process $\{P_1, P_2\}$ functions consistently, i.e. that at any option $\{P_1, P_2\}$ form the linear ordered sequence of $tr = (act_1, act_2, ...)$ actions which are ordered on performance time. If to designate With as one of sequences, the set of indexes of elements of sequence can be designated as $Ind(tr)$, and a set of points as $Points(C)$.

The sequence of tr is linearization With (it is supposed that in parallel processes carrying-out operators aren't synchronized on time, therefore, it is possible to construct some ordered sequence) if there is a display

$$Lin: Points \to (Ind\ (tr).$$

In this case the answering condition can define the description of process $\{P_1, P_2\}$ as construction P

$$Tr(P) = \bigcup_{C \in Beh\{P_1, P_2\}} Lin(C)$$

with $Beh\{P_1, P_2\}$

In this case in the course of P all linearizations of processes of P_1 and P_2 answering to any their joint behavior are presented.

If processes look like $P_i = (S_i, s^0_i, R_i)$ $(i = 1, 2)$, and tr is the route (S, s^0, R) which components can be defined as

$$-S \overset{def}{=} S_1 \times S_2 \overset{def}{=} \{(s_1, s_2) | s_1 \in S_1, s_2 \in S_2\}$$

$$-s^0 = (s^0_1, s^0_2)$$

- for each transition $s_1 \overset{\alpha}{\rightarrow} s'_1$ of R_1 and each state of $s \in S_2$. R comprises a passage $(s_1, s) \overset{\alpha}{\rightarrow} (s'_1, s)$. And for each transition $s_2 \overset{\alpha}{\rightarrow} s'_2$ of R_2 and each state $s \in S_1$, R comprises a transition $(s, s_2) \overset{\alpha}{\rightarrow} (s, s_2)$.

For each pair of transitions $s_1 \overset{\alpha}{\rightarrow} s'_1 \in R_1$ and $s_2 \overset{\alpha}{\rightarrow} s'_2 \in R_2$, R contains the transition $(s_1, s_2) \overset{\tau}{\rightarrow} (s'_1, s'_2)$.

It is possible to claim that this process of P is parallel composition of processes of P_1 and P_2 which can be designated as $P_1 | P_2$.

During the work with parallel processes restrictions which follow from logic of functioning of processes of P_1 and P_2 are applied. If the L any subset of a set of *Names*, in this case restriction on L looks like $P\backslash L = (S, s^0, R')$ which turns out by removal of all transitions having tags from a subset of L, i.e.

$$R' \overset{def}{=} \{(s^\alpha \rightarrow s') \in R | \alpha = \tau \text{ or } name(\alpha) \in L\}.$$

It is necessary to remember and that during the work the set of conditions of process of P decides on parallel compositions as work of conditions of processes of P_1 and P_2 entering into this process.

As the model of behavior is under construction on the component principle, it can lead to repeated use of the same component in model. It leads to that tags of transitions and names of actions repeat. For permission of the such conflicts renaming of names f is carried out: *Names* → *Names* and change of tags $\alpha?, \alpha!$ on $f(\alpha)?$ and $f(\alpha)!$. The turned-out process can be designated as $P[f]$.

At nonidentical renaming when names from the list $\alpha_1, \ldots, \alpha_n$ also occurs displays in names $\beta_1, \ldots, \beta_n$. In this case process will be designated as $P[\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n]$ [1].

V. TREATMENT OF DESIGNS OF THE COUNT OF MODEL OF BEHAVIOR

Recognizing that we consider behavior model, and it means work with such categories as a condition of system, action, an event, transition from one condition in another, transition conditions, sequence of events, from column P it is possible to allocate information (J) and the managing director (G) of the column. If to compare

transitions in columns J, G and transitions in the column of the description of model of behavior, it is possible to receive the statement about their route equivalence of Tr (P) = Tr(G) = Tr(J) that will allow to operate with each of counts at the solution of specific problems of creation of the software.

So to column J it is possible to refer all actions which change internal state of model of behavior. Now we will define that action - any operation which makes signing up in port of input-output is meant the term or carries out an splitting of new process in system. It is necessary to remember that the declaration of assignment need to carry to operations of change of internal state of process.

For all these listed elements it is possible to execute operation of replacement of action on the corresponding time equivalent on each layer of decomposition in the operating count of model of behavior. Such action, further, will simplify a task of the analysis and verification of the operating count.

As the form of record of the information count is suitable for machining, further it is possible to use the interpreter which in an automatic mode will replace the action elements of the information count with the corresponding sequences of operators of the chosen programming language.

Having carried out the analysis of the managing director column G it is possible to allocate some moments, so transition from a condition in a condition can be unconditional or it can be limited to performance of some condition. Such condition influences a choice of the route and depends or on a condition of objects connected with internal or entrance actions. Most conveniently for routes defining a choice to write down conditions over an edge of the count. It will significantly improve visualization of a condition of transition. The type of such record assumes the following options:

- lack of record over an edge, unconditional transition;

- record existence over an edge, transition will be executed if expression over an edge is true. Especially it is necessary to emphasize that in the column surely there has to be an alternative route which will be realized in case of condition non-performance;

- existence over an edge of «*» symbol means that unconditional performance of transition (it is used only in an alternative design) is realized.

The following situation in the operating count needs to be considered when the multiprocessor platforms on which it is realized parallel asynchronous processes are used. In this case it is necessary to consider parallel processes as two or more components. Symbols "&&>" will be a sign of the organization of parallel processes, and as a condition of end of overlapping symbols ">&&" or ">//" will serve. These designs have to be connected with a condition, i.e. top of the count.

It should be noted that the organization of parallel processes is possible only according to one scenario, and end of a parallel segment assumes two scenarios. In the first case the scheme "and" when further work will be possible at completion of

parallel processes is realized. In the second case, the competitive scheme is realized "or" - process which came to the end earlier has to destroy competing parallel process with release of its resources.

As the branch condition and performance of cyclic operators to implement with the selection of the most optimal design, it is at the moment the developer is recommended to indicate the most appropriate statement of branching, for example, "*>if/case*", which is associated with a vertex of realizing input action. Failing this, the interpreter will implement the scheme "*if*". Similar rules apply to the choice of the operator of the cycle. If there is no clear indication to choose the type of cycle, such as "*>for/while*", the interpreter selects the implementation of the design "*>while*".

VI. RULES OF DESIGNING OF THE COUNT AND PERFORMANCE OF INSCRIPTIONS

For convenience of reading columns it is designed at the left on the right and from top to down. All elements of the count (tops and edges) have unique marking. As one top (condition) can leave some edges (actions), the generating top of the count is duplicated with the indication of a sign of circulation – « ' » in the top identifier. The similar design arises and in a situation when some edges enter into top of the count. At transfer of elements for "*a new line*" identifiers of tops repeat with the indication of operation of transfer « '' ».

At design of parallel structures, cycles and branchings on a condition it is offered to use the component principle of creation of structure with further carrying out decomposition, i.e. to project model by the principle "*from above – down*". It will increase "*readability*" of model of behavior. If these elements have "indivisible" character – simply them to write down that in the corresponding places of the count. As the general recommendation, at creation of model, it is necessary to consider performance of the principle of layer-by-layer design and "*not to overload*" each layer of decomposition of model of behavior.

Concerning inscriptions on a process graph. It is necessary to allocate some types, these are recommendatory, performance and action conditions. Their general characteristic will be that they are characterized as a stream of symbols written down by some rules. Now we will consider them one after another.

The group of recommendatory inscriptions treats only "*entrance*" tops of the count (conditions). Their characteristic feature is existence of a pair symbol «"», as restrictions of a stream of symbols. These are "entrance" recommendatory designs: "*>if*", "*>case*", "*>for*", "*>while*". If they are absent as recommendatory, the interpreter is obliged to realize situationally standard designs of *if* and *while*.

There is one more design which belongs to "entrance" - start of parallel processes "&&>". This design is the managing director and assumes existence of several parallel processes. In other situations use of this design is forbidden. In model of behavior designs ">&&" and ">||" which define the mechanism of completion of parallel processes. By record rules these designs are associated with top of the count for which the scenario transition is realized.

Now consider an item such as the execution condition. This stream of characters written on the edge of the graph, and involves the ability to record any kind of logical expression, including the composite. The result of the logical expression is obtaining certification - the "**true**" or "**false**". This design is most applicable in the implementation cycle of the while statement and the statement of branching if.

However, in a situation where design case used to compare the value of the variable pattern and the comparison result decision of the possibility of performing an action associated with the edge of the graph . Since in this case must be provided in "**template**", then the value of the variable with which comparison is made, must be determined by the operator in case, therefore, a complete record of the design suggested should be represented as ">case" $ **<variable name>** . Between constructs ">*case*" and $ **<variable name>** use spaces that are ignored when parsing the string. When writing templates themselves must use the construct " $ **<variable name>**, with the last template in the operator always has to be a template $*. His performance is the lack of coincidence of all the previous template with the value stored in the variable associated with the job conditions of the design case.

Considering the design conditions for the situation "> *for*", it should be noted that in this case given a range of values of the variable changes and which tend to cyclically change its value . Therefore, the condition that is written on the edge of the graph should be shaped sequence of groups of characters separated by spaces or other special characters. An example of the formation of such a stream of characters:

$ <variable name> = <the minimum value of the variable (constant)> $ <variable name> = $ <variable name> <type of operation> {$ <variable name> or <constant>} <maximum value of the variable (constant)>

All elements of the text structures must be separated by spaces or special characters. Attention should be paid to such a structure as a text element **<type of operation>**. In this case, should be considered only arithmetic operations. For the $ **<variable name>** = $ **<variable name>** **<type of operation>** {$ **<variable name>** or **<constant>**} can be used and unary operations with the construction of a suitable design.

Now consider the character stream that carries the information about the action. It is assumed that the number of elements to the edges of which are characterized as "**action**" is not limited. Therefore, the separator of elements should act or a sign of the completion of the flow (push it «button» Enter) or a special character, " ; " that separates the operation effect. For the structure itself "action" there is only one rule - the use of space characters, spaces or special characters for the separation of the structure.

It should be noted, and feature the use of variable names, as in the elements of "condition" and in the elements of the "**action**". A common feature of the variable name is the symbol "$" is in the first position of the sequence of characters.

The general rules should include the fact that the elements of the text constructs do not use special characters if their application not stated in special agreements for the interpreter.

VII. CONCLUSIONS

Design of model of behavior of a control system is a first step to achievement of a goal. And if when performing this work it is possible to reach reduction of quantity of mistakes of "a human factor" and it can happen only at introduction in process of design of formal language to high extent of visualization, as shows the solution proposed in article.

As the created model has all signs of a scripting programming language that obviously that with use of such representation of model of behavior it is possible to check correctness of implementation of key decisions in a control system.

When obtaining such conclusion, it is possible to start the following phase of implementation of the project – generation of a code of the operating program according to the formal description of model of behavior of a control system.

It is supposed that the solution proposed in article will allow to introduce new approaches to design of the software of control systems, and it will allow to reduce significantly as terms of performance of work, and will increase reliability of a created product.

References

[1] A.M.Mironov A.A. Theory processes / Internet http://www.twirpx.com/file/617525/ resource /

REVIEW OF GROUP CONTROL ALGORITHMS

Norseev S.A., Bagaev D.V.
KSTA named Degtyarev V.A.
Kovrov, Russia
Norseev@gmail.com, dmitrybag@gmail.com

Problem of distributing task between some systems and coordination interaction this systems is actual problem. In this article will review main methods of task distribution between different system elements and adjustment interaction these elements.

algorithm; behavior; swarm; multiagent system

I. MAIN TERMS

Agent – it is object that solves narrow range of specific tasks. In this role can be: processor, microcontroller, computer, robot and other. Typically, single agent performs a specific simple operation. For example, defining readings of sensor and sending these