A MODEL OF AUTONOMOUS SYSTEM FOR SCIENTIFIC EXPERIMENTS AND SPACECRAFT CONTROL FOR DEEP SPACE MISSIONS

Plamen Angelov, Plamen Hristov

Space Research and Technology Institute – BAS
{pangelov, phristov}@space.bas.bg

Abstract: *The particularities of autonomous control system for deep space missions are described. A new approach for autonomous control system development is proposed and analyzed in details. Some models are analyzed and compared. The general formal model is based on the theory of communicating sequential processes (CSP). Methods for reconfiguration, verification and trace control are described.*

The software that is appropriate not only for the spacecraft flight path control but also for autonomous control of scientific apparatus operation and science experiments parameters is described. The software enables onboard scientific apparatus to autonomously detect and respond to science events

Science algorithms, including onboard event detection, feature detection, change detection, and unusualness detection, are proposed to be used to analyze science data. Thus detecting features of scientific interest these algorithms are used to downlink only significant science data. These onboard science algorithms are inputs to onboard decision-making Replaner that modify the spacecraft observation plan to capture high value science events. This new observation plan is input for the Task execution subsystem of the Autonomous control system (ACS), able to adjust the plan to succeed despite run-time anomalies and uncertainties, and after it is executed by the ACS, which controls onboard scientific apparatus to enable an autonomous goal-directed exploration and data acquisition to maximize science return.

Keywords: *CSP models, autonomous control system, spacecraft, software*

1. Introduction

For future interplanetary space missions the long-term goal is to use such software that is appropriate not only for the spacecraft flight path control but also for

experiments le capture of short-lived science phenomena. In addition, onboard science analysis will enable data be captured at the finest time-scales without overwhelming onboard memory or downlink capacities by varying the data collection rate. Examples include: eruption of volcanoes on planets and its satellites, cloud detection, formation of jets on comets, phase transitions in ring systems and etc. For extended duration missions that study long-term phenomena the generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to an acceptable level. Such long-term phenomena could be atmospheric changes, flexing and cracking of the ice crust on some planets and its satellites.

These future interplanetary missions require a new generation of spacecraft with a new generation of control systems and software. Their most important features are:

- autonomous operation;
- minimal dependence from the Earth-based command center;
- operation by general commands;
- planning and scheduling;
- system dynamic recovering and reconfiguration;
- high reliability of the system hardware and software;

These principles are fully realized in the Remote Agent (RA) model based system for Deep Space One (DS1) spacecraft [1, 2, 3]. In this extremely successful extended mission DS1 encountered comet Borrelly and returned the best images and other science data ever from this comet [2]. Highly advanced technologies, the key to more capable, powerful, and efficient spacecraft and science instruments, are used also in other missions from NASA’s New Millennium Program (Space Technology 5, 6, 7, 8; Earth Observing 1, 3; Deep Space Two) [4].

To build an autonomous control system for interplanetary space missions is difficult due to following reasons:

1. Limited, intermittent communications – for interplanetary space missions the spacecraft must be able to operate for long periods of time without supervision from the Earth-based command center. Some deep space missions only contact the spacecraft once per week, or even rarely.

2. Spacecraft are very complex - a typical spacecraft has thousands of components, each of which must be carefully engineered to survive rigors of space (zero pressure, extreme temperature, radiation, collision with high energy particles, physical stresses).

3. Limited observability - because processing telemetry is expensive, onboard storage is limited, and downlink bandwidth is limited, engineering telemetry is limited. Thus onboard software must be able to make decisions on limited information and the command center team must be able to operate the spacecraft with even more limited information.

4. Limited computing power – because of limited power onboard, spacecraft computing resources are usually very constrained.

5. High stakes – a typical interplanetary space mission costs hundreds of millions of dollars and more, so any failure could have significant economic impact. New space mission can take years to plan, construct the spacecraft, and reach their targets. In addition many mission opportunities are limited by planetary geometries.

In these cases, if a space mission is lost, it may be years before another similar mission can be launched.

The use of highly advanced technologies will lead to self-sufficient autonomous spacecraft systems that can adapt their behavior to complex, rapidly changing, and incompletely understood environments. Areas of research and development of these advanced technologies include adaptive control technologies, control agent architectures, embedded decision systems, evolvable systems, intelligent robotics, adjustable autonomy, distributed and multi-agent systems, goal-level commanding, planning and scheduling.[5, 6, 7, 8]

2. General characteristics of the approach

The authors' approach for creating of an autonomous control system (ACS) is characterized by:

- formal models and methods throughout the system operating cycle are used;
- the system is presented as a set of objects freely configurable and connectible by channels [9, 10];
- Artificial Intelligence (AI) is used in the stage of system reconfiguration or as a part of the control process.

The main differences of ACS from the RA model based system for DS1 [1, 2, 3] are:

1. ACS executes control and development functions - it makes dynamic reconfiguration (analysis, synthesis and etc.) of the execution control system in real time;
2. AI functions are used, both in the control process (if it is necessary) and in the system development;
3. Formal models and methods are used repeatedly – to create and verify formal specification and to control the traces of the system processes (all in real time).

The structure of the ACS system is shown on Fig. 1.

Let's describe the main elements of the proposed approach:

1. Software formal models (specifications): software formal models are a process models type (a sequential process, that can be executed parallel), where the processes are communicating (and are synchronized) by a message exchange.

2. Software system verification, using Hoare's (CSP) specifications and laws: with the CSP theory [9] the software properties can be specified and verified.

3. System operations control, using Hoare's trace models: the traces describe all events that have place, or can have place in the computing process, and can be used for run-time control.

4. System structure and parameters run-time determining on the base of information about control system, controlled objects (spacecraft and scientific apparatus) and background states. (This is a separate subsystem of the ACS.)

5. Dynamic reconfiguration: on the base of the specification, the system structure and parameters are implemented in the general software model frame.

6. Real-time simulation.

Corresponding to this approach the ACS is treated like a set of software and hardware objects that can be configured to exchange information and to synchronize and get access to common system resources.

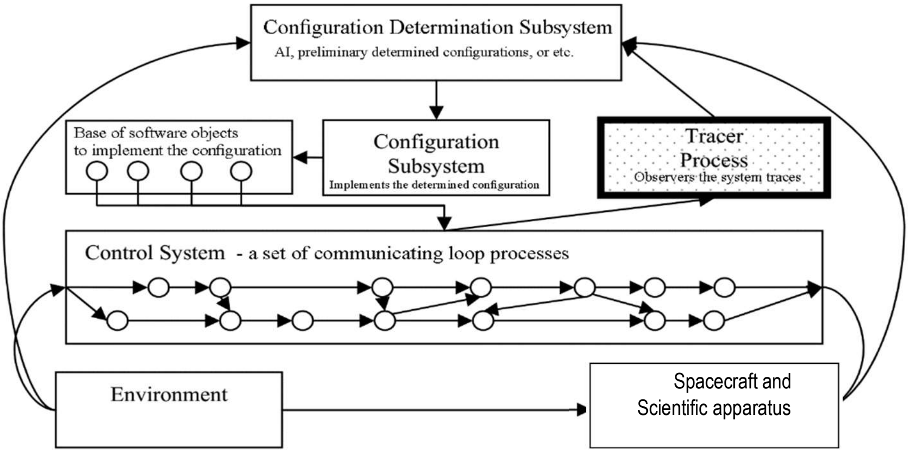
Configuration Determination Subsystem
AI, preliminary determined configurations, or etc.
Base of software objects to implement the configuration
Configuration Subsystem
Implements the determined configuration
Tracer Process
Observers the system traces
Control System - a set of communicating loop processes
Environment
Spacecraft and Scientific apparatus

Fig.1. Structure of ACS as set of objects and subsystems

3. CSP models

The CSP models describe functions and structure of the system by set of communicating sequential processes, that can be executed parallel. The models include CSP-description, alphabet, trace and specification. The interactions are synchronous by one-direction channels.

The models described below represent a RTS system that consists of four main processes

(CONTROL, MEASUREMENT, SIMULATION, MODEL):

MODELING = CONTROL || MEASUREMENT || SIMULATION || MODEL,

where the symbol || indicates parallel processes.

The alphabet of the processes determines a set of events, logically possible for the parallel system:

α(CONTROL || MEASUREMENT || SIMULATION || MODEL) = αCONTROL ∪ αMEASUREMENT ∪ αSIMULATION ∪ αMODEL,

where α is process alphabet.

The traces of the parallel processes are determined by:

trace(CONTROL || MEASUREMENT || SIMULATION || MODEL)={t | (t ↑ αCONTROL) ∈ trace(CONTROL) & (t ↑ αMEASUREMENT) ∈ trace(MEASUREMENT) & (t ↑ αSIMULATION) ∈ trace(SIMULATION) & (t ↑ αMODEL) ∈ trace(MODEL) & t ∈ (αCONTROL ∪ αMEASUREMENT ∪

αSIMULATION ∪ αMODEL)},

where trace is a process protocol, describing the events, which the process has gone to this moment. The symbol ↑ indicates shrink of the trace on the some set, such as the alphabet.

Loop process, describing the system functions by separate loops:

LPi = {PR, CH, PAR, ARG},

where: PR – procedure, implementing the process functions; CH – set of information channels; AGR – set of aggregates; PAR – parameters.

CSP model is:

LoopProcess = c[0]?x → AgregatesList; c[n]!y → ENDprocess; ModelProcesses,

where: c[i] indicate channels with numbers; AgregatesList – linear aggregates list;

?x – CSP procedure to receive a value from channel c and assumes it to the x variable;

!y – CSP procedure to transmit the value of y variable to the channel;

ModelProcesses – modeling processes.

The traces are:

LoopProcess=P;Q;ModelProcesses;

P=c[0]?x→AgregatesLIst; Q=c [n]!y→ENDprocess;

trace(P) = {t|t=<>V(t0=c[0]?x & t’∈ trace(AgregatesList))} = {<>} U {<c[0]?x>∧t|t

∈ trace(AgregatesList)};

trace(Q)={t|t=<> V (t0 = c[n]!z & t’∈ trace(ENDprocess))} = {<>} U {<c[n]!z>∧t|t

∈ trace(ENDprocess)};

trace(LoopProcess)= {s;t|s ∈ trace(P)& t ∈ trace(Q); r|s ∈ trace(P)& t ∈ trace(Q)& r ∈ trace(ModelProcesses)}.

where the symbol ^ indicate after (between traces); t0 – beginning of the trace; t' – end of the trace; <> – empty trace.

A list of aggregates CSP model is:

AgregatesList = (c[i]?z →AGREGATE[i]; AgregatesList | c[i]!z → AGREGATE[i]; AgregatesList) | (next →AGREGATE[i]; AgregatesList) | (end_list →ENDprocess),

where END process is a special process, the alphabet of which has only one event, that indicate successful end.

AGREGATE=input(x) →AgrTransferFunction(x:y); output(y) →ENDprocess;
AgrTransferFunction = (f1→ENDprocess) | (f2→ENDprocess) |...| (fn→ENDprocess);

where f1,...,fn – functions, doing the transformation x→y.

There are two traces:

AgregatesList
(c[i]?z →AGREGATE[i]) = P1; (c[i]!z →AGREGATE[i]) = P2 ;
(next →AGREGATE[i] = P3;(end_list→ENDprocess) = P4 ;
trace(P1)={<>} U {<c[i]?z>∧t|t ∈trace(AGREGATE[i])};
trace(P1;AgregatesList)={s;t|s ∈trace(P1) &t ∈trace(AgregatesLIst)};
trace(P2)={<>} U {<c[i]!z>∧t|t ∈trace(AGREGATE[i])};
trace(P2;AgregatesList)={s;t|s ∈trace(P2) &t ∈trace(AgregatesLIst)};
trace(P3)={<>} U {<next>^t|t ∈trace(AGREGATE[i])};
trace(P3;AgregatesList)={s;t|s ∈trace(P3) & t ∈trace(AgregatesLIst)};
trace(P4)={<>} U {<end_list>^t|t ∈trace(ENDprocess)};
trace(P4;AgregatesList)={s;t|s ∈trace(P4)&t ∈trace(AgregatesLIst)};
trace(AgregatesList)= {t|t=<> V (t0 ∈B & t' ∈trace(P(t0)},
where B = {c[i]?z,c[i]!z,next,end_list}; P(t0)= (P1|P2|P3|P4).
– AGREGATE
P = (input(x)→AgrTransferFunction);
Q = (output(y)→ENDprocess); AGREGATE=(P;Q);
trace(P)={<>} U {<input(x)>^t|t ∈trace(AgrTransferFunction)};
trace(Q)={<>} U {<output(y)>^t|t ∈trace(ENDprocess)};
trace(AGREGATE)= {s;t|s ∈trace(P) & t ∈trace(Q)}.
– AgrTransferFunction
trace(AgrTransferFunction)= {t|t=<> V (t0 ∈B & t'∈trace(P(t0)},

where B = {f1,f2,.., fn} alternative
P(t0)=f1→ENDprocess|f2→ENDprocess|...| fn→ENDrocess);

In a similar way the other system objects are defined.

4. ACS operation control method based on CSP trace model

The CSP theory proposes appropriate means for preliminary specification of the system functions and structure and for the following operation control. These means are specifications, alphabets and traces of the processes [9, 10].

The general algorithm of the control method is shown on Fig.2:

- the system start;
- the software specification is defined, including the set of system objects;

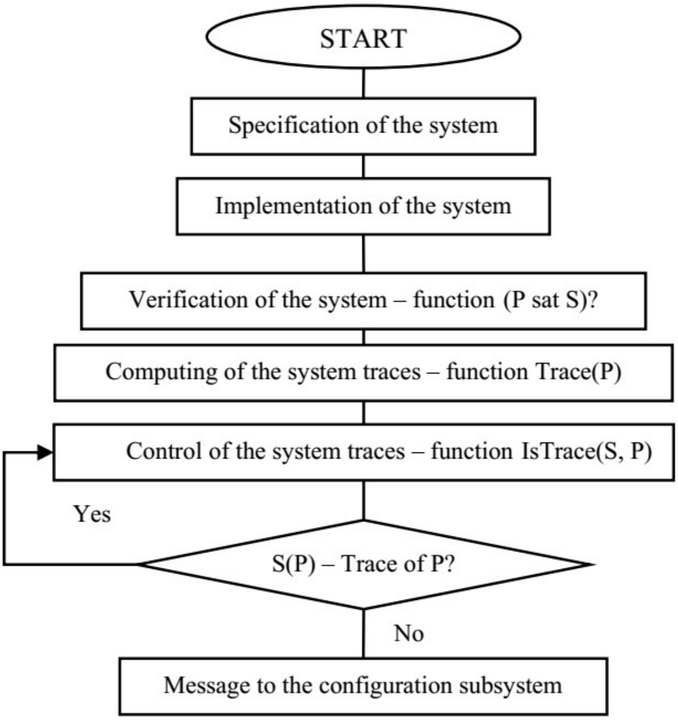
START
Specification of the system
Implementation of the system
Verification of the system – function (P sat S)?
Computing of the system traces – function Trace(P)
Control of the system traces – function IsTrace(S, P)
Yes
S(P) – Trace of P?
No
Message to the configuration subsystem

Fig.2. Flowchart of the system traces control method

– the program system implementation and verification (CSP specifications verification method);

– all possible traces computing for the specified system (procedure trace (P));

– the special process (tracer) observes and registers the system traces and determines if they are valid;

– when an invalid trace is registered (emergency situation).

5. Verification method

The method's idea is to be satisfied the relation P sat S (process P satisfies the specification S). It must satisfy preliminary determined specifications of the traces.

This is method that works with a limited, preliminary determined set of objects and standard structures. The method sequence is:

1. process (subsystem) type (structure) determining;
2. traces computation by the function trace(p);
3. traces verification (choice from a set of standard traces) by the function SAT(P,S(np)), represented below, where symbol (np) is any process trace.

The function SatP,S) that decides whether the system implementation satisfies the system specification:

```
      Sat(P,S(np)) =
     if P = STOP then if np = <> then return TRUE;
else
   return FALSE; end;
elsif P sat S(np) & (c ∏P) then
    if (np = <> V (np0 = c & S(np'))) then return TRUE;
    else return FALSE; end;
    elsif P sat S(np) & (c ∏d ∏P) then
       if (np ≤c,d> V (np ≥<c,d> & S(np")))
            then return TRUE; else return FALSE;
      elsif P sat S(np) & Q sat T(np) & (c∏P|d∏Q)
      then
if (np = <> V (np0 = c & S(np')) V (np0 = d & T(np')) /*S(np') & T(np') are the
specifications of the chosen alternative */
     then return TRUE;
     else return FALSE;
      elsif ∇x ∈B.(P(x) sat S(np,x)) & (x:B ∏P(x)) then
if (np = <> V (np0 ∈B & S(np',np0)))
     then return TRUE; (*a process with choice from a set*)
else return FALSE;
    elsif P sat S(np) & Q sat T(np) & P || Q then if (P||Q) sat (S(np |' ∝P) & T(np |'
∝Q)) then return TRUE; (* parallel processes *)
else return FALSE;
——— other standard structures ——-
else return FALSE; End;
——— other standard structures ——
    else return FALSE;
End;
```

The function trace(P) that computes the system traces:

```
    trace(P)= (μR)* &
if P = (STOPP) then trace(P)= \<>\;
elsif P = (c→P) then trace(P)= \<> U \<c> ∧t|t ∈trace(P)\;
elsif P = (c →P | d →Q)then trace(P)= \t|t = <> V (t0 = c & t'∈trace(P)V
(t0 = d & t'∈trace(Q))\;
elsif P = (x:B →P(x)) = \t|t = <> V (t0 ∈B & t'∈trace(P(t0)))\
elsif P = (mX : A.F(x))) = U trace(Fn(STOPA));
elsif P = (P || Q)then trace(P)= trace(P)Ç trace(Q);
else t = trace(P||Q) then
trace(P||Q)=\t|(t |`μR) ∈trace(P)& (t |`μQ) ∈trace(Q)& t ∈(aR U aQ)*\
...........—other structures--.....................
else
end.
```

The function IsTrace(s,p) that decide whether the trace s is valid for the process P:

```
        IsTrace(s,P)=
        if s = NIL then TRUE;
        elsif P(s0) = “BLEEP then FALSE;
        else IsTrace(s', P(s0));
        end.
```

6. Sofware archtecture

The software architecture is shown on Fig.3.

The concept for intelligent autonomous observation is applied as follows:

1. A list of science targets to monitor as well as the programs of the spacecraft flight path and orientation is stored onboard or that have been sent as high-level goals from the Earth-based command center. The model-based planning algorithms used by Flight and experiments planner enable rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including

faster recovery from spacecraft anomalies. This onboard planner accepts as inputs the science and engineering goals and ensures high-level goal-oriented behavior.

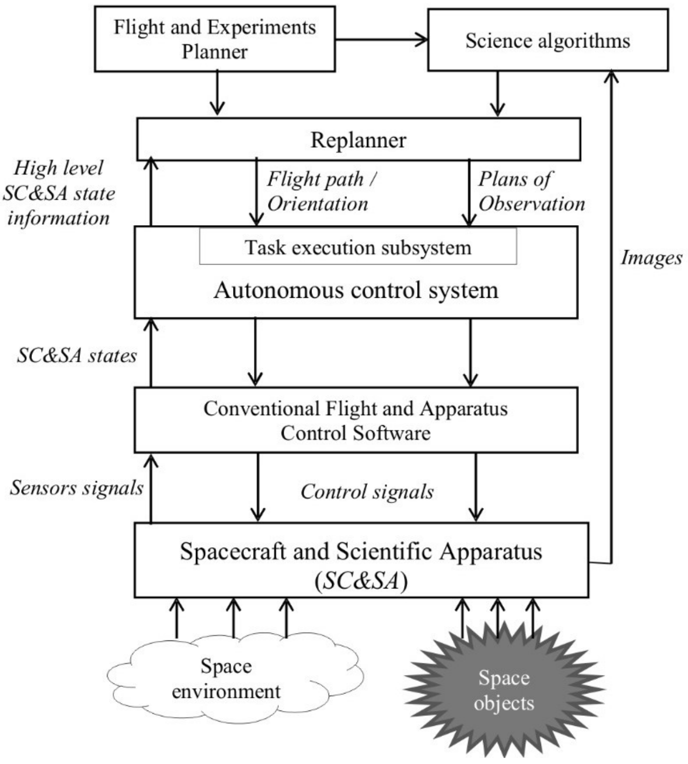
Flight and Experiments Planner
Science algorithms
Replanner
High level SC&SA state information
Flight path / Orientation
Plans of Observation
Images
Task execution subsystem
Autonomous control system
SC&SA states
Conventional Flight and Apparatus Control Software
Sensors signals
Control signals
Spacecraft and Scientific Apparatus (SC&SA)
Space environment
Space objects

Fig.3. Software architecture

2. The software generates an observation plan to monitor the targets on this list by using different scientific apparatus, e.g. for volcanic studies, the infra-red and near infra-red bands are used. It addition it can generate new plans for changing the spacecraft flight path and orientation if needed.

3. The task execution subsystem accepts the new observation plan as an input, reconfigured the control system (using CSP models) and expands the plan into low-level commands. During execution of the observation plan by generation commands and control signals (using conventional software), different scientific apparatus are activated. Specialized unit in the subsystem monitors the execution of the plan and has the flexibility and knowledge to perform event driven commanding to enable local improvements in execution as well as local responses to anomalies.

4. The collected images are sent to Science algorithms block, where they analyzed and based on this analyze the data is downlinked. Had no new scientific event detected, the science software would generate a goal for the Replanner to acquire the next highest priority target in the list of targets. Such way the system modify the current operations plan to include numerous new activities in order to enable the new science observation.

5. The software executes the new generated plans in conjunction with several autonomy elements.

6. This cycle is then repeated on subsequent observations.

The developed software enables to autonomously detect and respond to science events. Such science events can be: volcanic eruptions on planets and its satellites, cloud detection, growth and retreat of ice caps, formation of jets on comets and etc. Classification algorithms are used to analyze imagery and other signal to detect change and science events (thus to downlink science data only on change and when detect features of scientific interest). Detection of these events is then used to trigger scientific apparatus. Onboard Replanner then develops a response plan that accounts for target visibility and operations constraints. This plan is then executed using a task execution subsystem that can deal with run-time anomalies.

Conclusion

The proposed integrated approach allow to design and study an spacecraft autonomous control system based on the CSP models, appropriate for scientific experiments and spacecraft flight path and orientation control during interplanetary missions. This approach is implemented by the methods for verification and control. The models describe in general the functions of the system by using a rigorous mathematical theory (CSP theory). A new software model (process-aggregate) is represented. It is a high-efficient model with high degree of structural accordance to real systems. The control and verification methods allow creation of high-reliability systems by preliminary verification and run-time system observation. The control method, based on a trace model, is a new in this type of systems. The main advantage of the approach is the completeness and the possibility to achieve some degree of assurance in the system quality before the flight tests.

The software architecture appropriate for the spacecraft flight path and orientation control and for autonomous control of scientific apparatus operation and science experiments parameters are proposed. The developed software enables to autonomously detect and respond to science events. Such science events can be: volcanic eruptions on planets and its satellites, cloud detection, growth and retreat of ice caps, formation of jets on comets and etc. The onboard analysis using science algorithms enables to capture even short-lived science phenomena. Thus the data could to be captured at the finest time-scales without overwhelming onboard memory or downlink capacities by varying the data collection rate on the fly. In addition by generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to a manageable level for extended duration missions that study long-term phenomena such as atmospheric changes and flexing and cracking of the ice crust on some planets and its satellites.

Bibliography

1. D. Bernard, G. Dorais, E. Gamble, B. Kanefsky J. Kuriena, G. K. Man, W. Millar, N. Muscettola P. Nayak, K. Rajant,N. Rouquette, B. Smith, N. W. Taylor, Yu-Wen Tung, Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment, Proceedings of the AIAA Space Technology Conference & Exposition, Albuquerque, NM, Sept. 28-30, 1999. AIAA-99-4512.
2. NASA, Remote Agent Experiment Home Page
https://ti.arc.nasa.gov/tech/asr/planning-and-scheduling/remote-agent/ [retrieved June 12, 2017]
3. NASA, Jet Propulsion Laboratory
https://www.jpl.nasa.gov/nmp/ds1/ [retrieved June 12, 2017]
4. NASA, Jet Propulsion Laboratory, https://www.jpl.nasa.gov/nmp/ [retrieved June 12, 2017]
5. NASA, Jet Propulsion Laboratory
https://www.jpl.nasa.gov/nmp/TECHNOLOGY/missions.php/ [retrieved June 12, 2017]
6. S. Chien, B. Engelhardt, R. Knight, G. Rabideau, R. Sherwood, E. Hansen, A. Ortiviz, C. Wilklow, S. Wichman, "Onboard Autonomy on the Three Comer Sat Mission," Proc.SAIRAS 2001, Montreal, Canada, June 2001.
7. N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt, "IDEA: Planning at the Core of Autonomous Reactive Agents", Proceedings of the Workshops at the AIPS-2002 Conf., Tolouse, France, April 2002.
8. M. Griffin, H. Burke, D. Mandl, & J. Miller, "Cloud Cover Detection Algorithm for the EO-1 Hyperion Imagery," Proceedings of the 17th SPIE AeroSense 2003, Orlando, FL, April 21-25, 2003.
9. C.A.R. Hoare, Communicating Sequential Processes. London, Prentice-Hall International, London, UK, 1985, 2015 (http://www.usingcsp.com/cspbook.pdf).
10. Hristov P.L., P.S. Angelov, Autonomous Onboard Computer Systems Using Real Time Trace Models, IEEE Proceedings of 2nd International Conference on Recent Advances in Space Technologies: Space in the Service of Society RAST'2005, June 9-11, 2005, Istanbul, Turkey, pp.189-194.