Reliability Improvement and Reliability Assessment for Distributed Hardware-Software Multi-Agent Systems

ALEXEI V. IGUMNOV and SERGEY E. SARADGISHVILI
Information and Control Systems Department
Institute of Computing and Control
Saint-Petersburg State Polytechnical University
29 Polytechnicheskaya Street, Saint-Petersburg
RUSSIAN FEDERATION
Alexei.Igumnov@gmail.com, SSaradg@yandex.ru

Abstract: - The problem of reliability of new object of distributed hardware software (DHS) multi-agent system (MAS) is considered. DHS MAS is determined as a system that is based on agent technologies and consists of both agents and hardware components required for execution of agents and for interaction of agents with an environment. Reliability improvement, fault-recovery and several reliability assessment approaches for DHS MAS are presented. The reliability improvement methodology is built upon replication of unique functional components and redundancy of universal components. The fault-recovery methodology defines a set of fault-recovery procedures required for restoration of consistent system configuration after failures of its components. Methodology for operability function formation was developed to enable utilization of logical-and-probabilistic methods for reliability assessment. Another approach for reliability assessment is based on Markovian model and system state graph and was developed to overcome limitations of logical-and-probabilistic methods that are suitable only for systems with hot standby. The state graph based approach allows reliability assessment for DHS MAS with cold standby and different operating modes of system components. The state graph based approach is also applicable for a case when probability of failure of one of components depends on states of other components. New failure model for determination of failure rates of system components in accordance with system state is introduced. Computing experiments described in the article have validated developed methodologies.

Key-Words: - multi-agent system, reliability, fault-recovery, operability function, redundancy, state graph, logical-and-probabilistic method

1 Introduction

An increasing interest in application of multi-agent technologies for development of various industry systems is caused by benefits of multi-agent systems (MAS) declared by an artificial intelligence theory. In [1] MAS is defined as a system in which several agents operate and interact with one another. It’s worth noting that researches name a problem of fault-tolerance as the reason of small amount of deployed real world multi-agent systems [2], [3]. Thus a contradiction between an interest in utilization of agent-oriented technologies and the lack of MAS fault-tolerance defines the problem of MAS reliability considered in the article.

We define a distributed hardware-software (DHS) multi-agent system as an industry system that is based on agent technology. DHS MAS is the object of the presented research. Some examples of industry systems for hospitals that are based on agent technologies are presented in [4], [5]. Whereas typical MAS is defined by a set of agents, DHS MAS also include hosts required for execution of an agent model and actuators required for interaction of agents with an environment. MAS is typically considered as software system, however the necessity of utilization of hardware components for MAS execution and for enabling interaction of MAS with an environment is highlighted in [5]. Thus RFID readers, medical devices and sensors mentioned in [5] as system components may be considered as actuators and embedded computers shall be treated as hosts.

Existing methods for achieving fault-tolerance in MAS such as DarX [6], AAA for broker teams [7], Sentinels approach [8], FATMAS [9] improve fault-tolerance only in cases of occurred faults of individual agents or faults of hosts on that an agent model is deployed for execution because they consider MAS as a set of agents. Failure of each of components of DHS MAS may result in failure of the whole system. Thus we state that new reliability improvement methodology shall be developed taking into account failures of all components of DHS MAS. Whereas it is important to improve fault-tolerance of DHS MAS it is also required to determine a level of this improvement or an assured level of reliability. Such theoretical assessments of assured reliability level are not provided by existing methodologies mentioned above. Moreover effectiveness of existence fault-tolerance methodologies is proved only via experiments.

The objective of the research includes both reliability improvement for DHS MAS in cases of failures of agents, tasks of agents, hosts and actuators and theoretical assessment of assured reliability level in cases of different standby techniques and various operating modes of system components.

2 Methodology for Reliability Improvement

2.1 Distributed Hardware-Software Multi-Agent System Model

A formal model of DHS MAS is required for development of reliability improvement and reliability assessment approaches. Our model of DHS MAS was developed based on following elements of MAS model introduced by FATMAS methodology [9]: agents and tasks supposed to be executed by agents. We have introduced terms of an agent platform (AP) and an actuator (ACT). AP is defined as a system component required for both execution of agents and provision of services required for agent interaction. If MAS is a control system then it shall be situated in some environment and moreover is intended to operate with the said environment. Thus we define an actuator as a system component intended for interaction of agents with an environment in that DHS MAS is situated. Our DHS MAS model is based on theory of sets and predicate logic.

We define DHS MAS as an ordered set *MAS* = *<T, A, HWP, HWR>*, where *T* is a set of tasks, *A* – a set of agents, *HWP* – a set of APs, *HWR* – a set of ACTs. The following set of predicates determines a configuration of DHS MAS:

- *confTaskAgent(t, a)* is intended for determination of deployment of a particular task *t* in a particular agent *a*;
- *confAgentHwp(a, hwp)* is intended for determination of deployment of a particular agent *a* in a particular AP *hwp*;
- *reqHwrTask(hwr, t)* is intended for determination of necessity of utilization of a particular ACT *hwr* for performing of a task *t*;
- *confHwrHwp(hwr, hwp)* is intended for determination of accessibility of a particular ACT *hwr* for a particular AP *hwp*, i.e. for all tasks of all agents deployed in AP *hwp*.

Each task of DHS MAS is supposed to implement some unique functionality in accordance with requirements or specification and is treated as a minimal functional instance. Thus an inability to perform of at least one of tasks leads whole DHS MAS to a failure state. Failures of other system components result in an inability to perform one of more of system tasks and in turn lead DHS MAS to a failure state, e.g.:

- failure of an agent *a* leads to an inability to perform all tasks deployed in this agents, i.e. each task *t* such that *confTaskAgent(t, a) = true*;
- failure of AP *hwp* results in failure of each agent *a* deployed in this AP (i.e. *confAgentHwp(a, hwp) = true*) that in turn causes an inability to perform all agent's tasks;
- failure of ACT *hwr* result in an inability to perform all tasks that require utilization of this ACT, i.e. each task *t* such that *reqHwrTask(hwr, t) = true*.

It's worth noting that DHS MAS model specifies only system structure and components dependencies and does not determine particular operating modes or standby techniques.

2.2 Reorganization Technique

Incorporating redundant copies of system components has been recognized as one of the best methods for improvement of fault-tolerance. FATMAS methodology [9] has introduced duplicating of tasks instead of duplicating of non-critical agents for the first time in the art. However FATMAS also utilizes duplication of critical agents and does not consider ACTs and APs as MAS components.

We suggest distinguishing components of DHS MAS that are unique in terms of implemented functionality such as tasks and ACTs and components that act as universal executive containers. Although we have defined AP as a component responsible for execution of agents we shall explicitly state that each AP is supposed to be able to execute any of system agents. Thus AP can be treated as a universal executive container. Similarly each agent is supposed to be able to perform any of system tasks and therefore is also considered as a universal executive container. Based on these assumptions there longer is no reason to replicate existing agents and APs. Instead it's suggested to introduce redundant sets of universal executive containers that may be considered as a redundant network for deployment of system tasks. Thus we define our methodology for reliability improvement through a reorganization technique that shall be applied to an existing DHS MAS. The reorganization technique shall use both replication of functional components and introduction of redundant sets of universal executive containers. It's assumed that application of our reorganization technique turns an existing DHS MAS to a redundant one.

To enable identification of equivalent functional system components we have introduced terms of task type and actuator type. Let *TT* be a set of task types and *THWR* be a set of actuator types. These sets shall be determined based on the set of tasks *T* and the set of actuator *HWR* of the existing DHS MAS respectively. Moreover the univocal correspondence shall exist between the set of task types *TT* and the set of tasks *T* of the existing system so that all redundant DHS MAS tasks of a similar type are equal to a particular task of the existing DHS MAS and thus can replace each other in case of failures. Similarly the one-to-one mapping shall exist between the set of actuator types *THWR* and the set of actuator *HWR* of the existing DHS MAS. Two actuators of redundant DHS MAS of a similar type are equal in terms of their functionality but they may distinguish in terms of accessibility for various APs. It's worth noting that due to introduction of terms of task type and actuator type the necessity of utilization of a particular ACT for performing of a particular task is turned into the necessity of utilization of ACT of a particular type for performing of a task of a particular type.

The developed reorganization technique that defines our reliability improvement methodology comprises of following steps:

- define a set of task types *TT* and a set of actuator types *THWR* in accordance with respectively the set of tasks *T* and the set of actuator *HWR* of the existing DHS MAS;
- define necessity of utilization of ACT of a particular type for performing of a task of a particular type in accordance with *reqHwrTask()* predicate of the existing DHS MAS;
- introduce replication of tasks and ACTs and define a set of tasks and a set of actuators of a redundant DHS MAS;
- define a redundant set of agents and a redundant set of agent platforms;
- determine configuration of redundant DHS MAS, i.e. deployment of tasks on the redundant set of agents, deployment of agents of the redundant set of APs, accessibility of actuators for APs.

2.3 Redundant Distributed Hardware-Software Multi-Agent System Model

The redundant DHS MAS that is a result of application of our reorganization technique to the existing DHS MAS is defined by an ordered set *RMAS* = <*TT*, *RT*, *RA*, *RHWP*, *THWR*, *RHWR*>, where *TT* is a set of task types, *RT* – a set of tasks, *RA* – a set of agents, *RHWP* – a set of APs, *THWR* – a set of actuator types and *RHWR* is a set of actuators. The following set of predicates determines its configuration:

- *confTypeTask(tt, t)* is intended for determination of a type of a particular task (i.e. the predicate is true if a type of a task *t* is *tt*);
- *confTypeHwr(thwr, hwr)* is intended for determination of a type of a particular ACT (i.e. the predicate is true if a type of ACT *hwr* is *thwr*);
- *confTaskAgent(t, a)* is intended for determination of deployment of a particular task *t* in a particular agent *a*;
- *confAgentHwp(a, hwp)* is intended for determination of deployment of a particular agent *a* in a particular AP *hwp*;
- *reqTHwrTTask(thwr, tt)* is intended for determination of necessity of utilization of an actuator of a particular type *thwr* for performing of a task of a particular type *tt*;
- *confHwrHwp(hwr, hwp)* is intended for determination of accessibility of a particular ACT *hwr* for a particular AP *hwp*.

We have introduced a term of active replica. The active replica of a particular type is defined as one task from a replication group of equivalent tasks of the same type that exert an influence on an environment. It's worth noting that for each task type one and only one active replica shall exist in a particular time instant. Other aspect of the model of a redundant DHS MAS is described in [10], [11].

3. Fault-Recovery Methodology

3.1 Message-Based Communication and Database Schemas

The message based communication protocol is required to ensure that the reorganization technique is applied transparently and is intended to enable execution of an active replica of a particular type as well as utilization of ACT of a particular type independently of its deployment in particular executive container. We suggest that a receiver of a message shall be bounded based on type of originator component. Thus an agent is able to send a message to AP if and only if the said agent is deployed in this AP. Similarly AP is able to send a message to the agent if and only if this AP acts as an executive container for this agent. Also it's assumed that there is no limitation on message based communication between APs. Moreover each AP shall be able to interact with all remote agent platforms via sending a broadcast message.

To simplify the system we have limited knowledge regarding system configuration that agents and APs shall maintain for successful operation. Each agent *a* shall have a database (DB) *ADB(a)* comprised of following tables: deployment table *ATBT(a)*, required actuators table *ATBR(a)* and active replicas table *ATBAT(a)*. Each AP *hwp* shall have a database *HDB(hwp)* comprised of following tables: deployment table *HTBD(hwp)*, required actuators table *HTBR(hwp)*, available actuators table *HTBAR(hwp)* and active replicas table *HTBAT(hwp)*. Formats of DB tables for agents and APs are defined as follows:

- *ATBT(a)* contains records *(t, tt)* that links the task *t* deployed in the agent and its type *tt*;
- *ATBR(a)* contains records *(tt, {thwr$_i$})* that links a type of task *tt* with required types of actuators for all task types from *ATBT(a)*;
- *ATBAT(a)* contains records *(tt, t)* that links a type of task *tt* and an active replica *t* deployed in the agent in accordance with *ATBT(a)*;
- *HTBD(hwp)* contains records *(a, t, tt)* that links an agent *a* deployed in AP, a task *t* deployed in the agent *a* and its type *tt*;
- *HTBR(hwp)* contains records *(tt, {thwr$_i$})* and acts as an aggregator of *ATBR(a)* tables of all agents deployed in AP;
- *HTBAR(hwp)* contains records *(hwr, thwr)* that links an available ACT *hwr* with its type;
- *HTBAT(hwp)* contains records *(tt, a, 0)* that links the task type *tt* with an agent *a* deployed in this AP if an active replica of type *tt* is deployed in this agent;
- *HTBAT(hwp)* contains records *(tt, 0, rhwp)* that links the task type *tt* with a remote agent platform *rhwp* if an active replica of type *tt* is deployed in one of agents of this remote AP

One of MAS tasks may require execution of another system task. As our reliability improvement methodology utilizes replication of tasks we have replaced a request to execute a particular task with a request to execute a task of particular type. Let's consider a task *t* of the agent *a* deployed in AP *hwp*. If the task *t* requires execution of another task of type *tt* it sends the *pfm(tt)* message to its agent *a*. An active replica of type *tt* may be deployed in the agent *a*, one of other agents of AP *hwp* or in one of agents of one of other APs. Thus the *pfm(tt)* request shall be handled iteratively by the agent that has received it, AP in which the agent is deployed and all remote APs in the following manner:

- if an active replica *t'* of type *tt* is deployed in the agent *a* in accordance with *(tt, t')* record in table *ATBAT(a)* then the agent *a* shall execute it, otherwise the agent *a* shall escalate the processing to AP *hwp* through the *pfm(tt)* message;
- if an active replica of type *tt* is deployed in the agent *a'* of AP *hwp* in accordance with *(tt, a', 0)* record of table *HTBAT(hwp)* then AP *hwp* shall request the agent *a'* to execute an active replica through the *pfm(tt)* message;
- if an active replica of type *tt* is deployed in one of agents of one of remote APs *rhwp* in accordance with *(tt, 0, rhwp)* record of table *HTBAT(hwp)* then AP *hwp* shall escalate the processing to AP *rhwp* though the *pfm(tt)* message.

If a particular task of the redundant DHS MAS requires utilization of an actuator of a particular type *thwr* it sends *use_hwr(thwr)* message to its agent which will route the received message to its AP *hwp* without any further processing. On reception of the *use_hwr(thwr)* message AP will find an accessible actuator of the *thwr* type in accordance with the available actuators table *HTBAR(hwp)* of its DB and will perform the requested action.

3.2 Fault-Recovery Procedures

In section 2.1 we have stated that an inability to perform of at least one of tasks leads whole DHS MAS to a failure state. In a redundant DHS MAS only active replicas are executed or exert an influence on an environment as described in sections 2.3 and 3.1. Thus a redundant DHS MAS is in failure if it is not able to perform an active replica of at least one type. It's worth noting that this inability may be caused by failures of other components of redundant system, e.g. an agent in that the active replica is deployed or an actuator that is required for performing of the active replica. Thus fault-recovery task may be treated as a task of search and activation of new replicas of all required types.

To represent an operable state of a redundant DHS MAS we have introduced a term of a consistent configuration. The consistent configuration of a redundant DHS MAS is defined by a set of conditions that DBs of each agent and AP shall meet for successful operation of whole system.

Conditions for DB *ADB(a)* of each agent *a* are as follows:

- deployment table *ATBT(a)* of each agent *a* shall contain *(t, tt)* record if and only if the task *t* is deployed in the agent *a* (i.e. *confTaskAgent(t, a) = true*) and the task *t* could be executed;
- active replicas table *ATBAT(a)* of each agent *a* shall contain record *(tt, t)* if and only if the task *t* is an active replica of type *tt* and there is a record *(t, tt)* in *ATBT(a)* (i.e. the task *t* could be executed).

DB *HDB(hwp)* of each AP *hwp* shall meet following conditions:

- deployment table *HTBD(hwp)* of each AP *hwp* shall contain *(a, t, tt)* record if and only if the task *t* is deployed in the agent *a* that is deployed in AP *hwp* (i.e. *confAgentHwp(a, hwp) = true & confTaskAgent(t, a) = true*), the agent *a* is not in failure state and the task *t* could be executed;
- available actuators table *HTBAR(hwp)* of each AP *hwp* shall contain record *(hwr, thwr)* if and only if ACT *hwr* is not in failure and is accessible (i.e. *confHwrHwp(hwr, hwp) = true*);
- active replicas table *HTBAT(hwp)* of each AP *hwp* shall contain record *(tt, a, 0)* if and only if the agent *a* is deployed in AP *hwp* (i.e. *confAgentHwp(a, hwp) = true*), the agent *a* is not in failure and there is a record *(tt, t)* in active replicas table *ATBAT(a)*;
- active replicas table *HTBAT(hwp)* of each AP *hwp* shall contain record *(tt, 0, rhwp)* if and only if AP *rhwp* is not in failure and there is a record *(tt, a', 0)* in active replicas table *HTBAT(rhwp)*.

If conditions mentioned above are met then DBs of agents and APs represent the actual configuration of a redundant DHS MAS and do not contain records related to components that are in failure state. However as a consistent configuration shall represent an operable state of the system then for each task type there shall exist an active replica that is not in failure state. Thus DB *HDB(hwp)* of each AP *hwp* shall meet following additional condition: its active replicas table *HTBAT(hwp)* shall contain record *(tt, a, rhwp)* for each task type *tt*.

Fault-recovery procedures shall restore a consistent configuration of redundant DHS MAS that has been broken by an occurred fault of one of system components via excluding records related to failed components and updating records related to locations of new active replicas in databases of agents and agent platforms. All fault-recovery procedures are supposed to be performed iteratively in accordance with components hierarchy, i.e. each procedure starts from the component responsible for failure detection, is escalated though a set of executive containers and may finish in one of remote APs.

Let's consider a failure of a task *t* of the agent *a* deployed in AP *hwp*. Each agent is responsible for detection of failures of its tasks. Thus the agent *a* has detected a failure of the task *t* and shall perform the *a_r_tpfailt(t)* procedure comprised of following steps:

- determine a type *tt* of the task *t* in accordance with the deployment table *ATBT(a)*;
- remove all records related to failed task *t* from its DB and DB of its AP *hwp*:
  - remove a record *(t, tt)* from *ATBT(a)*;
  - remove a record *(tt, t)* from the active replicas table *ATBAT(a)*;
  - send the *a_cmd_rm_task(t)* message to AP *hwp* that shall process this message by excluding *(a, t, tt)* record from the deployment table *HTBD(hwp)*;
- if the task *t* is not an active replica of type *tt* than finish processing;
- initiate search and activation of new replica of type *tt*:
  - if there is a record *(t', tt)* in the deployment table *ATBT(a)* then select *t'* as new active replica, add a record *(tt, t')* to the active replicas table *ATBAT(a)* and finish processing;
  - escalate fault-recovery procedure to AP *hwp* via sending the *req_atask(tt)*.

When AP *hwp* receives the *req_atask(tt)* message from one of its agents *a* it shall continue a process of search and activation of new replica of type *tt* started by the agent *a* via execution of the *h_r_tfail(tt)* procedure comprised of following steps:

- if there is a record *(a', t', tt)* in the deployment table *HTBD(hwp)* then:
  - select the task *t'* deployed in the agent *a'* as new active replica;
  - update the active replicas table *HTBAT(hwp)* to contain a record *(tt, a', 0)* for a type of task *tt*;
  - send the *h_cmd_atask(t')* message to the agent *a'* that shall process this message by adding a record *(tt, t')* to its active replicas table *ATBAT(a)*;
  - finish processing;
- escalate fault-recovery procedure to all other APs of a redundant DHS MAS via sending the broadcast *req_atask(tt)* message.

When AP *rhwp* receives the broadcast *req_atask(tt)* message from AP *hwp* it shall continue a process of search and activation of new replica of type *tt* through execution of the *recv_req_atask(tt)* procedure comprised of following steps:

- if there is a record *(a', t', tt)* in the deployment table *HTBD(rhwp)* then:
  - select the task *t'* deployed in the agent *a'* as new active replica;
  - update the active replicas table *HTBAT(hwp)* to contain a record *(tt, a', 0)* for a type of task *tt*;
  - send the *h_cmd_atask(t')* message to the agent *a'*;
  - send the broadcast *atask_announce(tt, rhwp)* message to all APs.

The broadcast *atask_announce(tt, rhwp)* message is required for update of active replicas tables of other APs, i.e. each AP *hwp* shall ensure that its active replicas table *HTBAT(hwp)* contains a record *(tt, 0, rhwp)* for a task type *tt*.

Each AP is responsible for detection of failures of its agents. If AP *hwp* detects a failure of its agent *a* then it shall perform the *h_r_afailt(a)* procedure comprised of following steps:

- determine a set *TF* of tasks deployed in the agent *a* in accordance with the deployment table *HTBD(hwp)* (i.e. the set *TF* shall contain each task *t* such that there exists a record *(a, t, tt)* in *HTBD(hwp)*);
- determine a set *TTF* of types of active replicas deployed in the the agent *a*:
  - for each task *t* of type *tt* from the set *TF*:
    - ▪ if there is a record *(tt, a, 0)* in the active replicas table *HTBAT(hwp)* then add the task type *tt* to the set *TTF*;
- remove all records related to failed agent *a* from the deployment table *HTBD(hwp)* (i.e. each record that matches a template *(a, x, xx)*)
- for each task type *tt* from the set *TTF* initiate search and activation of new replica of this type via performing the *h_r_tfail(tt)* procedure.

Each AP is also responsible for detection of failures of accessible actuators. When AP *hwp* detects a failure of the accessible actuator *hwr* it shall perform the *h_r_hwrfail(hwr)* procedure comprised of following steps:

- determine a type *thwr* of the actuator *hwr* in accordance with the available actuators table *HTBAR(hwp)*;
- remove a record *(hwr, thwr)* related to the failed ACT from *HTBAR(hwp)*;
- if there is a record *(hwr', thwr)* in the available actuators table *HTBAR(hwp)* then finish processing because another ACT of the same type is available;
- determine a set *TF* of tasks that are in failure caused by a failure of the actuator *hwr* (i.e. the set *TF* shall contain each task *t* of type *tt* such that there is a record *(tt, {thwr$_1$, …, thwr, … thwr$_n$})* in the required actuators table *HTBR(hwp)*);
- determine a set *TTF* of types of active replicas that are in failure caused by a failure of the actuator *hwr* (i.e. the set *TTF* shall contain each task type *tt* such that there is a task *t* of this type in a set *TF* and there exists a record *(tt, x, 0)* in the active replicas table *HTBAT(hwp)*);
- remove all records related to tasks that are in failure caused by a failure of the actuator *hwr* from the deployment table *HTBD(hwp)* and DBs of corresponding agents, i.e. for each task *t* from the set *TF*:
  - remove a record *(a, t, tt)* from *HTBD(hwp)*;
  - send the *h_cmd_rm_task(t)* message to the agent *a* that shall process this message by excluding of *(t, tt)* record from the deployment table *ATBT(a)* and by excluding of *(tt, t)* record from the active replicas table *ATBAT(a)*);
- synchronize all APs;
- for each task type *tt* from the set *TTF* initiate search and activation of new replica of this type via performing the *h_r_tfail(tt)* procedure.

When AP *hwp* detects a failure of another AP *rhwp* it shall perform the *h_r_hwpfail(rhwp)* procedure comprised of following steps:

- determine a set *TTF* of types of active replicas that are located in AP *rhwp* (i.e. the set *TTF* shall contain each task type *tt* such that there exists a record *(tt, 0, rhwp)* in the active replicas table *HTBAT(hwp)*);
- remove all records related to active replicas that are in failure caused by a failure of AP *rhwp* from the deployment table *HTBAT(hwp)* (i.e. each record that matches a template *(xx, 0, rhwp)*);
- for each task type *tt* from the set *TTF*:
  - initiate search and activation of new replica of this type via performing the *h_r_tfail(tt)* procedure;
  - if there is a record *(tt, a, 0)* in the active replicas table *HTBAT(hwp)* (i.e. after execution of the *h_r_tfail(tt)* procedure new active replica is located in the agent *a* of AP *hwp*) then send the broadcast *atask_announce(tt, hwp)* message to all APs.

It's worth noting that in case of failure of AP *rhwp* the *h_r_hwpfail(rhwp)* procedure shall be performed only by one of remote APs.

4 Validation of Fault-Tolerance Property of Redundant Multi-Agent System

To define conditions that are required for the success of fault-recovery procedures we have introduced following additional functions and predicates in accordance with the formal model of a redundant DHS MAS:

- function *agentOfT(t)* is intended for determination of an agent in that a particular task is deployed:

$$agentOfT(t) = a \Leftrightarrow confTaskAgent(t,a) = true, \tag{1}$$

  where *t* is a task, *a* – an agent;
- function *hwpOfT(t)* is intended for determination of AP such that a particular task belongs to one of its agents:

$$\begin{aligned} &hwpOfT(t) = hwp \Leftrightarrow (\exists a \in RA: \\ &confTaskAgent(t,a) = true \wedge \\ &confAgentHwp(a,hwp) = true), \end{aligned} \tag{2}$$

  where *t* is a task, *hwp* is AP, *a* – an agent, *RA* – a set of agents;
- function *reqTHwr(tt)* determines a set of actuator types required for performing of a task of a particular type:

$$\begin{aligned} &reqTHwr(tt) = \{thwr \in THWR \mid \\ &reqTHwrTTask(tt,thwr) = true\}, \end{aligned} \tag{3}$$

  where *tt* is a task type, *thwr* – an actuator type, *THWR* – a set of actuator types;
- function *tOfTask(t)* is required for determination of a type of a particular task:

$$tOfTask(t) = tt \Leftrightarrow confTypeTask(tt,t) = true, \tag{4}$$

  where *t* is a task, *tt* is a task type;
- function *tOfHwr(hwr)* is intended for determination of a type of a particular actuator:

$$\begin{aligned} &tOfHwr(hwr) = thwr \Leftrightarrow \\ &confTypeHwr(thwr,hwr) = true, \end{aligned} \tag{5}$$

  where *hwr* is an actuator, *thwr* is an actuator type;
- predicates *failT(t), failA(a), failP(hwp), failR(hwr)* determine respectively states of failure of the task *t*, the agent *a*, AP *hwp* and ACT *hwr;*
- predicate *atType(tt, t)* determines whether the task *t* is an active replica of type *tt.*

In accordance with section 2.1 failure of each of system components my lead to an inability to perform of one or more of system tasks. Let the predicate *failTG(t)* be true if and only if the task *t* could not be performed. Thus an operable state of the task *t* is defined as follows:

$$\begin{aligned} &failTG(t) = false \Leftrightarrow (failT(t) = false) \wedge \\ &(failA(agentOfT(t)) = false) \wedge \\ &(failP(hwpOfT(t)) = false) \wedge \\ &(\forall thwr \in reqTHwr(tOfTask(t)), \exists hwr \in RHWR: \\ &tOfHwr(hwr) = thwr \wedge failR(hwr) = false \wedge \\ &confHwrHwp(hwr, hwpOfT(t)) = true), \end{aligned} \tag{6}$$

where *t* is a task, *thwr* – an actuator type, *hwr* – an actuator, *RHWR* – a set of actuators.

Let's define conditions that are required for success of fault-recovery procedures:

- in case of failure of an active replica *t* of type *tt* there shall exist another task *t'* of type *tt* that is in an operable state:

$$\begin{aligned} &(failtT(t) = true \wedge atType(tOfTask(t),t) = true) \Rightarrow \\ &(\exists t' \in RT: (t' \neq t \wedge tOfTask(t') = tOfTask(t) \wedge \\ &failTG(t') = false)). \end{aligned} \tag{7}$$

- in case of failure of an agent *a* for each active replica t_i of type tt_i which is deployed in the agent *a* there shall exist another task t_i' of type tt_i that is in an operable state:

$$\begin{aligned}&\forall t_i \in RT : (\,failA(a) = true \wedge \\ &confTaskAgent(t_i, a) = true \wedge \\ &atType(tOfTask(t_i), t_i) = true) \Rightarrow \\ &(\exists t_i' \in RT : (t_i' \neq t_i \wedge tOfTask(t_i') = tOfTask(t_i) \wedge \\ &failTG(t_i') = \ false)).\end{aligned} \tag{8}$$

- in case of failure of an agent platform *hwp* for each active replica t_i of type tt_i that is deployed in one of agents of AP *hwp* there shall exist another task t_i' of type tt_i that is in an operable state:

$$\begin{aligned}&\forall t_i \in RT : (\,failP(hwp) = true \wedge \\ &hwpOfT(t_i) = hwp \wedge \\ &atType(tOfTask(t_i), t_i) = true) \Rightarrow \\ &(\exists t_i' \in RT : (t_i^{'} \neq t_i \wedge tOfTask(t_i') = tOfTask(t_i) \wedge \\ &failTG(t_i') = \ false)).\end{aligned} \tag{9}$$

- in case of failure of an actuator *hwr* of type *thwr* for each task type tt_i which requires utilization of an actuator of type *thwr* there shall exists a task t_i' of type tt_i that is in an operable state:

$$\begin{aligned}&\forall tt_i \in TT : (tOfHwr(hwr) \in reqTHwr(tt) \wedge \\ &failR(hwr) = true) \Rightarrow \\ &(\exists t_i' \in RT : (tOfTask(t_i') = tOfTask(t_i) \wedge \\ &failTG(t_i') = \ false)).\end{aligned} \tag{10}$$

Theorem. If the redundant DHS MAS have a consistent configuration it will recover:

- from detected failure of a particular active replica if condition (7) is met;
- from detected failure of a particular agent if condition (8) is met;
- from detected failure of a particular agent platform if condition (9) is met;
- from detected failure of a particular actuator if condition (10) is met.

Proof is presented in [19] and is as follows. Let's consider a processing of a request to execute an active replica of a particular type by redundant DHS MAS with a consistent configuration. Let the *pfm(tt)* request be received by an agent a_r deployed in AP hwp_r from its task t_r. As configuration is consistent there is a task t_a that is an active replica of type *tt*. If the task t_a is located in the agent a_r then due to consistent configuration there is a record *(tt, t_a)* in *ATBAT(a_r)* and processing of the *pfm(tt)* request by the agent a_r will result in execution of the task t_a. Otherwise the agent a_r will send the *pfm(tt)* message to AP hwp_r. If the task t_a is located in an agent a_1 of the AP hwp_r then there is a record *(tt, a_1, 0)* in *HTBAT(hwp_r)* and a record *(tt, t_a)* in *ATBAT(a_1)*. Thus the processing of the *pfm(tt)* message by AP hwp_r will result in execution of the task t_a by the agent a_1. Otherwise AP hwp_r will send a broadcast *pfm(tt)* message to other APs. If the task t_a is located in some agent a_2 of remote AP hwp_2 then the processing of the *pfm(tt)* message by AP hwp_2 will result in execution of task t_a by the agent a_2 similarly to the case of deployment of the task t_a in the agent a_1 of AP hwp_r.

As the requested active replica is executed independently of its deployment while DHS MAS configuration is consistent and DHS MAS is supposed to be in non-failure state if it is able to execute an active replica of each type we shall proof that fault-recovery procedures restore a consistent configuration.

Let's consider a failure of the task *t* of type *tt*. Let *a* be an agent in that the task *t* is deployed, *hwp* be AP in that the agent *a* is deployed. In accordance with the conditions of theorem there is a task *t'* that is not in failure state. In accordance with *a_r_tpfail(t)* procedure records *(t, tt)* of *ATBT(a)* and *(a, t, tt)* of *HTBD*(*hwp)* will be removed. At this step a consistent configuration is partially restored in terms of conditions related to deployment tables. If the task *t* is not an active replica then processing will be finished, otherwise a record *(tt, t)* of *ATBAT(a)* will be removed.

If the task *t'* is deployed in the agent *a* (i.e. a record *(t', tt)* exists in *ATBT(a)*) then it will be selected as new active replica. In this case a record *(tt, a, 0)* of *HTBAT(hwp)* remains valid as well as a record *(tt, 0, hwp)* of *HTBAT(rhwp)* of each remote AP $rhwp \neq hwp$. Thus including a record *(tt, t')* in *ATBAT(a)* will restore a consistent configuration.

If the task *t'* is deployed in the agent $a_1 \neq a$ of the same AP *hwp* then it will be selected as new active replica. In accordance with the *h_r_fail(tt)* procedure a record *(tt, a, 0)* will be changed to a record *(tt, a_1, 0)* in *HTBAT(hwp)* and a record *(tt, t')* will be added to *ATBAT(a_1)*. A record *(tt, 0, hwp)* of *HTBAT(rhwp)* of each remote AP $rhwp \neq hwp$ remains valid. Thus a consistent configuration is restored.

If the task *t'* is deployed in the agent $a_2 \neq a$ of the remote AP $rhwp \neq hwp$ then it will be selected as new active replica. In accordance with *recv_req_atask(tt)* procedure a record *(tt, a_2, 0)* will be added to *HTBAT(rhwp)* and a record *(tt, t')* will be added to *ATBAT(a_2)*. A record *(tt, 0, hwp)* of *HTBAT($rhwp_i$)* of each remote AP $rhwp_i \neq rhwp$ will be changed to a record *(tt, 0, rhwp)* in accordance with processing of the broadcast *atask_announce(tt, rhwp)* message. Thus a consistent configuration is restored. Consequently consistent configuration of redundant DHS MAS will be restored independently of deployment of the task *t'*.

In case of a failure of the agent *a* deployed in AP *hwp* each record *(a, t, tt)* will be removed from *HTBD(hwp)* in accordance with the *h_r_afail(a)* procedure and a consistent configuration will be partially restored in terms of conditions related to deployment tables. As was proved for a case of task failure the *h_r_tfail(tt)* procedure will activate new replica of the required type and will restore a consistent configuration in terms of conditions related to active replicas tables. The *h_r_tfail(tt)* procedure will be performed by AP *hwp* for each task type *tt* such than an active replica of this type were deployed in the agent *a*. In accordance with conditions of the theorem all *h_r_tfail(tt)* procedures will succeed. Thus a consistent configuration of DHS MAS will be restored.

Cases of failures of actuators and APs are considered in a similar manner. Simplified proofs for these cases were described in [10], [12].

5 Logical-and-Probabilistic Approach for Reliability Assessment

Utilization of multi-agent technologies in industry systems requires obtaining of a guaranteed reliability level. Thus there is a need for methods for theoretical assessment of redundant DHS MAS reliability level achieved by application of our reorganization technique and fault-recovery procedures. Existing approaches described in [13], [14] enable calculation of probability of survival only in case of failures of hosts of a network on which MAS is deployed. Moreover as mentioned in [15] generic techniques developed for software systems usually are not suitable for agent-based systems. In this section the adaptation of logical-and-probabilistic methods that enable synthesis of a reliability function of a system in an analytic form [16] for redundant DHS MAS is discussed.

5.1 Methodology for Operability Function Formation

Logical-and-probabilistic methods are based on transformation of a logical operability function that determines a state of the system based on states of its components to such form which allows replacement of logical operators with arithmetical operators and logical variables with corresponding probabilities of no-failure [16]. Therefore we shall develop a methodology for formation of an operability function of redundant DHS MAS.

In section 3.2 we have argued that a redundant DHS MAS is in failure if it is not able to perform an active replica of at least one type. Thus in accordance with definition of an operable state of a particular task (6) the criterion of serviceability of a redundant DHS MAS is as follows:

$$\forall tt \in TT, \exists t \in RT : failTG(t) = false, \tag{11}$$

where *tt* is a task type, *t* – a task, *RT* – a set of tasks.

It's well known that an operability function may be formed as a disjunction of all conjunction defined by the shortest ways of successful operation [16]. To define our methodology for formation of an operability function we have introduced terms of minimal functional configuration (MFC) and minimal operable configuration (MOC). Each MFC shall represent one of the shortest paths of successful operation [16] without considering the necessity of actuators utilization. Thus MFC shall contain a minimal set of tasks required for successful operation of the redundant DHS MAS as well as sets of agents and APs required for execution of these tasks. We define MFC as an ordered set *<MT, MA, MHWP>*, where *MT* is a set of tasks, *MA* – a set of agents, *MHWP* – a set of APs. Based on each MFC one or more minimal operable configurations could be formed. Each MOC shall represent the shortest path of successful operation in accordance with logical-and-probabilistic method. We define MOC as an ordered set *<MFC, MWR>*, where *MFC* is a minimal functional configuration, *MWR* – a set of actuators of MOC.

The operability function of the redundant MAS shall be determined in the following form:

$$\bigvee_{\forall MOC} \Big[\big(\bigwedge_{\forall t \in MT} w(t) \big) \wedge \big(\bigwedge_{\forall a \in MA} w(a) \big) \wedge \big(\bigwedge_{\forall hwp \in MHWP} w(hwp) \big) \wedge \big(\bigwedge_{\forall hwr \in MWR} w(hwr) \big) \Big]. \tag{12}$$

In (12) *MOC* is a minimal operable configuration, *MT*, *MA*, *MHWP* are respectively sets of tasks, agents and APs of MFC used for construction of MOC, *MWR* is a set of actuators of MOC, *t*, *a*, *hwp*, *hwr* are respectively a task, an agent, AP and an actuator, *w()* is a logical function representing a state of particular component.

Determination of all MOCs is required for formation of an operability function in form (12).

To define the methodology for formation of a set of all MOCs we have introduced a set of additional functions:

- *tasksOfTT(tt)* determines a set of tasks of a particular type:

$$tasksOfTT(tt) = \{t \in RT \mid confTypeTask(tt,t) = true\}, \tag{13}$$

  where *tt* is a task type, *t* – a task, *RT* – a set of tasks;
- *rTHwrOfP(MFC, hwp)* determines a set of actuator types required for performing tasks of MFC that are deployed in agents of AP *hwp*:

$$\begin{aligned}&rTHwrOfP(MFC,hwp) = \{thwr \in THWR \mid \\ &\exists t \in MT \in MFC, \exists a \in MA \in MFC : \\ &confTaskAgent(t,a) = true \wedge \\ &confAgentHwp(a,hwp) = true \wedge \\ &thwr \in reqTHwr(tOfTask(t))\},\end{aligned} \tag{14}$$

  where *hwp* is AP, *thwr* – ACT type, *THWR* – a set of ACT types, *t* – a task, *MT* – a set of tasks of MFC, *a* – an agent, *MA* – a set of agents of MFC;
- *avTHwrOfP(MFC, hwp, thwr)* determines a set of actuators of a type *thwr* that are accessible for AP *hwp*:

$$\begin{aligned}&avTHwrOfP(MFC,hwp,thwr) = \{hwr \in RHWR \mid \\ &confHwrHwp(hwr,hwp) = true \wedge \\ &tOfHwr(hwr) = thwr\},\end{aligned} \tag{15}$$

  where *hwp* is AP, *thwr* – ACT type, *hwr* – ACT, *RHWR* is a set of ACTs.

The developed methodology for formation of a set of all MOCs comprises of following steps.

At the first step a set *MTA* of all sets of tasks *MT* that could act as a base of MFC is formed as follows:

$$\begin{aligned}&MTA = \prod_{\forall tt \in TT} tasksOfTT(tt) = \{MT_k = (t_1,...,t_n) \mid \\ &n = |TT|, \forall i, \forall j \neq i : tOfTask(t_i), \neq tOfTask(t_j)\},\end{aligned} \tag{16}$$

where *tt* is a task type, *TT* is a set of task types, t_i is a task.

At the second step a set of all MFCs is formed as follows based on a set *MTA* (16):

$$\begin{aligned}&MFCA = \{MFC_k = (MT_k, MA_k, MHWP_k) \mid MT_k \in MTA, \\ &MA_k = \{a \in RA \mid \exists t \in MT_k : confTaskAgent(t,a) = true\}, \\ &MHWP_k = \{hwp \in RHWP \mid \exists a \in MA_k : \\ &confAgentHwp(a,hwp) = true\}\},\end{aligned} \tag{17}$$

where *a* is an agent, *RA* – a set of agents, *t* – a task, *hwp* is AP, *RHWP* – a set of APs.

The third step is required to determine a set *MRA* of all sets of ACTs *MR* that could be used for formation of MOC based on each MFC from the set *MFCA* (17). This step for each MFC comprises of following sub-steps:

- determine whether MFC could be used for formation of MOC in terms of availability of actuators of all required types based on following condition:

$$\begin{aligned}&\forall hwp \in MHWP \in MFC, \\ &\forall thwr \in rTHwrOfP(MFC,hwp) : \\ &avTHwrOfP(MFC,hwp,thwr) \neq \varnothing,\end{aligned} \tag{18}$$

  where *hwp* is AP, *MHWP* is a set of APs of MFC, *thwr* is ACT type;
- for each AP of MFC determine a set of all sets of ACTs *MRH* that include one and only one ACT of each ACT type required for performing of tasks of MFC deployed in agents of this AP:

$$\begin{aligned}&MRHA(MFC,hwp) = \{MRH = \{hwr\}\} = \\ &\prod_{\forall thwr \in rTHwrOfP(MFC,hwp)} avTHwrOfP(MFC,hwp,thwr),\end{aligned} \tag{19}$$

  where *hwp* is AP, *hwr* is ACT, *thwr* is ACT type;
- determine a set *MRAP* of all sets of ACTs *MR* that could be used to build MOC based on this MFC:

$$\begin{aligned}&MRAP(MFC) = \{MR_i = \\ &\bigcup_{\forall MRH_k \in MRU_i} MRH_k \mid \forall MRU_i \in MRAU(MFC)\}, \\ &MRAU(MFC) = \prod_{\forall hwp \in MHWP \in MFC} MRHA(MFC,hwp) \\ &= \{MRU_i = \{MRH_k\}\},\end{aligned} \tag{20}$$

  where *hwp* is AP, *MHWP* is a set of APs of MFC, *MRHA* is determined in accordance with (19);
- determine a desired set *MRA* by excluding from a set *MRAP* (20) such sets MR_i that do not meet the condition required for treatment of MOC as the shortest way of successful operation:

$$\begin{aligned}&MRA(MFC) = \{MR \in MRAP(MFC) \mid \\ &\forall hwr \in MR, \exists hwp \in MHWP \in MFC : \\ &[\exists a \in MA, \exists t \in MT : agentOfT(t) = a \wedge \\ &confAgentHwp(a,hwp) = true \wedge \\ &tOfHwr(hwr) \in reqTHwr(tOfTask(t)] \wedge [\forall hwr' \in MR : \\ &(hwr' \neq hwr \wedge confHwrHwp(hwr',hwp) = true) \Rightarrow \\ &(tOfHwr(hwr') \neq tOfHwr(hwr)]\},\end{aligned} \tag{21}$$

where *hwr*, *hwr'* are ACTs, *hwp* is AP, *a* is an agent, *t* is a task, *MHWP*, *MA*, *MT* are respectively sets of APs, agents and tasks of MFC.

At the final step a set *MOCA* of all MOCs of redundant DHS MAS is determined as follows:

$$MOCA = \{MOC = (MFC_k, MR_{ki}) \mid \forall MFC_k \in MFCA, \forall MR_{ki} \in MRA(MFC_k), \quad (22)$$

where *MOC* is minimal operable configuration, MFC_k is minimal functional configuration, MR_{ki} is a set of ACTs, *MFCA* is determined in accordance with (17), *MRA* is determined in accordance with (21).

Based on a set of all MOCs (22) the logical operability function of a redundant DHS MAS could be formed in accordance with (12). In accordance with existing logical-and-probabilistic methods this logical operability function could be transformed to the reliability function.

5.2 Limitation of Logical-and-Probabilistic Approach

Main advantage of logical-and-probabilistic approach is synthesis of an operability function in an analytic form independently from particular reliability indexes and parameters of system components. However logical-and-probabilistic methods are suitable only for systems with persistent connections between all components [17]. In other words each component of a redundant system may fail independently from states of other components.

In section 2 we have mentioned that our redundant DHS MAS model does not specify particular operating and standby modes of system components. However for utilization of logical-and-probabilistic approach redundant DHS MAS shall meet following conditions:

- all duplicate components operates in hot standby mode;
- reliability function of each component is persistent, i.e. probability of no-failure of a particular component does not depend on neither state of this component nor states of other system components, for example:
  - probability of no-failure of a particular agent does not depend on number of deployed tasks and number of tasks that act as active replicas;
  - probability of no-failure of a particular AP does not depend on number of deployed agents and tasks situated in these agents;
  - probability of no-failure of a particular ACT does not depend on availability of this ACT for an active replica that requires its utilization;
  - probability of no-failure of a particular tasks does not depend on state of this task (i.e. whether the task is an active replica or not).

It's worth noting that if probability of no-failure of at least one of system components depends on states of other components (e.g. a task that is not an active replica could not fail) then logical-and-probabilistic approach is not suitable for this redundant DHS MAS.

6 State Graph Based Approach for Reliability Assessment

As was mentioned above logical-and-probabilistic approach is limited to redundant DHS MAS which components operate only in hot standby mode and have persistent reliability function. However in real redundant system probability of no-failure of a particular component may depend on either its state of states of other system component. To overcome mentioned above limitation of logical-and-probabilistic approach we have decided to utilize state graph based approach [18]. State graph based approach uses Markovian model which requires that failure rate of each component shall be persistent only while whole system is in one of its states. Each system state is defined as some aggregate of states of its components. Thus state graph based approach allows reliability assessment for systems that components operate in cold standby mode, reduced reserve mode or in combined reserve mode. In accordance with [18] generic state graph based method comprises of following steps:

- determine all system states;
- determine nodes of state graph such that each node represent one of system states
- determine edges of state graph such that each edge represents a transition from one state to another that is related to failure of one or more system components;
- mark each edge of state graph with transition intensity (transition intensity is determined as sum of failure rates of system components that shall fail to initiate the transition);
- work out a Kolmogorov system of differential equations based on state graph in the following form:

$$\frac{dP_i(t)}{dt} = -\sum_{j\in L_{2i}} \lambda_{ij} P_i(t) + \sum_{k\in L_{1i}} \lambda_{ki} P_k(t), i \in E, \quad (23)$$

where $P_i(t)$ is a probability that system is in state i at time instant t, λ_{ij} – a transition intensity from state i to state j, L_{1i} – a subset of system states such that there is a transition from each state of the subset to the state i, L_{2i} – a subset of system states such that there exists a transition from the state i to each state of the subset, E – a set of system states.

6.1 State Graph Based Approach Definition

To enable formation of state graph of a particular redundant DHS MAS we assume that fault-recovery procedures are deterministic, i.e. for each system state and for each system component failure of this component will always switch a system to one and only one state. In other words for a particular system state and a particular failed component new system state to which the system will transition in accordance with fault-recovery procedures shall be determined explicitly.

As was mentioned in section 2 agents and APs are treated as universal executive containers and failure of agent of APs leads to inability to perform one or more of MAS tasks, i.e. tasks deployed in failed agent or tasks deployed in agents of failed AP. Thus we suggest that states of agents and APs shall be excluded from definition of redundant DHS MAS state definition. So redundant DHS MAS state is defined as an aggregate of states of all system tasks and all system ACTs.

We suppose that each task and ACT may be in one of following states:

- active state denoted as *[a]*;
- standby state denoted as *[s]*;
- failure state denoted as *[f]*.

It's supposed that a particular system component may fail only while it is in active state (i.e. standby state corresponds to cold standby mode).

We define a state of redundant DHS MAS as a following ordered set comprised of states of each system task and actuators:

$$\begin{aligned} SMAS = \{ & \{s(t_i) \mid \forall i \in 1...|RT|, t_i \in RT\}, \\ & \{s(hwr_j) \mid \forall j \in 1...|RHWR|, hwr_j \in RHWR\}\}, \end{aligned} \quad (24)$$

where $s(t_i)$ is a state of task t_i, RT – a set of tasks, $s(hwr_j)$ is a state of ACT hwr_j, $RHWR$ – a set of ACTs.

We consider redundant DHS MAS as a non-repairable system (i.e. each component that has failed could not be repaired). Thus we do not distinguish system states in which redundant DHS MAS is not operable in accordance with the criteria of serviceability (11). So we assume that there exists one and only one system state in which the system is not operable and this state is denoted as $SMAS_F = \{\varnothing\}$.

6.2 State Graph Formation

To enable formation of state graph of redundant DHS MAS we have developed new iterative algorithm. To enable algorithm definition we have introduced following additional sets and functions:

- set of already formed system states $SMASA = \{SMAS_i\}$, where $SMAS_i$ is a system state;
- set of already formed graph edges:

$$\begin{aligned} & SEDGA = \{SEDG_m = (SMAS_i, SMAS_j, F_{ij} = \{c_k\}) \mid \\ & SMAS_i \in SMASA, SMAS_j \in SMASA, \\ & c_k \in RT \cup RA \cup RHWR \cup RHWP\}, \end{aligned} \quad (25)$$

  where $SEDG_m$ is an edge that represents transition from state $SMAS_i$ to state $SMAS_j$ initiated by failure of any of system components c_k from set F_{ij}, RT – a set of tasks, RA – a set of agents, $RHWR$ – a set of ACTs, $RHWP$ – a set of APs;
- function *cstate(c, SMAS)* is intended for determination of a state of a particular ACT or a particular task in one of systems states:

$$cstate : (RT \cup RHWR) \times SMASA \to \{[a],[s],[f]\}, \quad (26)$$

  where RT is a set of tasks, $RHWR$ – a set of ACTs, $SMASA$ – a set of system states.

Our methodology for state graph formation is defined by following iterative algorithm:

- let n be a number of system state under consideration;
- determine a set of tasks FET that may fail in system state $SMAS_n$:

$$FET = \{t_i \in RT \mid cstate(t_i, SMAS_n) = [a]\}, \quad (27)$$

  where t_i is a task, RT – a set of tasks;
- determine a set of actuators FEA that may fail in system state $SMAS_n$:

$$FEA = \{hwr_i \in RHWR \mid cstate(hwr_i, SMAS_n) = [a]\}, \quad (28)$$

where hwr_i is ACT, *RHWR* – a set of ACTs;

- for each system component *cmp* that may fail (i.e. each task from a set *FET*, each ACT from a set *FEA*, each agent from a set of agents *RA*, each AP from a set of APs *RHWP*) new system state $SMAS_q$ is determined in following manner:
  - let new system state $SMAS_q$ be equal to the system state under consideration $SMAS_n$;
  - if considered component *cmp* is a task *t*:
    - mark the task *t* as failed, i.e. set a corresponding element *s(t)* of a set $SMAS_q$ to *[f]* value;
    - determine a set of unused ACTS *FUA* such that each ACT *hwr* from the set *FUA* is not in failure state and is not accessible for at least one task which is not in failure state and requires its utilization:

$$\begin{aligned}&FUA=\{hwr\in RHWR\mid(cstate(hwr,SMAS_q)\neq[f])\wedge\\&(\forall t'\in RT:(cstate(t',SMAS_q)\neq[f]\wedge\\&tOfHwr(hwr)\in reqTHwr(tOfTask(t'))\Rightarrow\\&confHwrHwp(hwr,hwpOfT(t'))=false)\},\end{aligned}\qquad(29)$$

where *hwr* is ACT, *RHWR* – a set of ACTs,, *t'* – a task, *RT* – a set of tasks.

    - mark each unused ACT *hwr* from the said set *FUA* as failed, i.e. set a corresponding element *s(hwr)* of a set $SMAS_q$ to *[f]* value;
  - if considered component *cmp* is an actuator *hwr*:
    - mark the actuator *hwr* as failed, i.e. set a corresponding element *s(hwr)* of a set $SMAS_q$ to *[f]* value;
    - determine a set of tasks *FETH* such that each task *t* from the set *FETH* is not in failure state, requires utilization of ACT of the type *thwr = tOfHwr(hwr)* and has no access to other not failed ACTs of the type *thwr*:

$$\begin{aligned}&FETH=\{t\in RT\mid(cstate(t,SMAS_q)\neq[f])\wedge\\&(tOfHwr(hwr)\in reqTHwr(tOfTask(t))\wedge\\&(\forall hwr'\in RHWR:(cstate(hwr',SMAS_q)\neq[f]\wedge\\&tOfHwr(hwr')=tOfHwr(hwr))\Rightarrow\\&confHwrHwp(hwr',hwpOfT(t))=false)\},\end{aligned}\qquad(30)$$

where *t* is a task, *RT* – a set of tasks, *hwr'* – an actuator, *RHWR* – a set of ACTs;

    - for each task *t* from the said set *FETH* process its failure as described in steps for the case of failed task, i.e. mark the task and all unused actuators as failed;
  - if considered component *cmp* is an agent *a*:
    - determine a set of tasks *FETA = {t | agentOfT(t) = a}* such that each task *t* from the said set *FETA* is deployed in agent *a*;
    - for each task *t* from the said set *FETA* process its failure as described in steps for the case of failed task;
  - if considered component *cmp* is an agent platform *hwp*:
    - determine a set of tasks *FETP = {t | hwpOfT(t) = hwp}* such that each task *t* from the set *FETP* is deployed in one of agents of AP *hwp*;
    - for each task *t* from the said set *FETA* process its failure as described in steps for the case of failed task;
  - update system state *SMASq* in accordance with fault-recovery procedures (i.e. some components that are in standby state will be switched to an active state);
  - if redundant DHS MAS is not operable in state $SMAS_q$ in accordance with the criteria of serviceability (11), then set $SMAS_q$ to empty set;
  - if generated state $SMAS_q$ to which the system is switched from the state $SMAS_n$ due to a failure of component *cmp* is a new state (i.e. it is not included in the set *SMASA* of already formed states), then update the set *SMASA* to include new state $SMAS_q$;
  - if graph edge related to transition from the state $SMAS_n$ to the state $SMAS_q$ already exists in the set of graph edges *SEDGA*, then update the element $SEDG = (SMAS_n, SMAS_q, F_{nq} = \{c_k\})$ of the set *SEDGA* to include the system component *cmp* to the set F_{nq} of system components that failures initiate the transition, otherwise update the set *SEDGA* to include new edge $SEDG' = (SMAS_n, SMAS_q, \{cmp\})$.

The iterative algorithm for state graph formation stops when all states from the set *SMASA* of generated state are processed, i.e. when after the performed iteration the number of system state under consideration *n* is equal to a number of elements in the set *SMASA*.

6.3 Extended Redundant Distributed Hardware-Software Multi-Agent System Model

As was mentioned above state graph based approach allows assessment of reliability for systems that components operate in cold standby mode. In case of redundant DHS MAS we have developed a methodology for state graph formation based on the assumption that a particular functional component (i.e. task or ACT) may fail only if it is in an active state. State graph based approach is also suitable in cases of advanced operating modes of system components. For example a task in an active state may be executed periodically and may fail only during its execution. Similarly an actuator in an active state may fail only when it is actually utilized by one of tasks. To consider different operating modes of MAS components we have extended our DHS MAS model.

We have introduced a term of task class to describe operating mode of a particular task. We suppose that each MAS task may belong to either event-driven or periodical class. A task of periodical class is executed periodically while it is in an active state with fixed time intervals between executions. Moreover duration of single execution of a task of periodical class is fixed. Execution of event-driven task is initiated by a particular accidental event in an environment. Duration of single execution of event-driven task is also fixed.

A term of actuator class was introduced to define various operating modes of a particular ACT. Each ACT may belong to either regular service class or task-driven class. An actuator of regular service class operates continuously while it is in an active state. Consequently an actuator of regular service class may fail at any time instant while it is active. A task-driven actuator operates only during its utilization by one or more of system tasks. Thus a task-driven actuator could not fail while it is not utilized even if it is in an active state.

Terms of operating modes for agents and agent platforms we introduced to deal with different failure behaviors of these components. If an agent operates in regular service mode then its failure rate does not depend on states of tasks deployed in the agent. If an agent operates in task service mode then its probability of failure increases during execution of one or more of its tasks. Similarly probability of failure of a particular AP that operates in task service mode increases when one of agents situated in this AP is executing its tasks.

To adopt new terms we have extended our redundant DHS MAS model with following new elements:

- set of task classes *CT = {[ed], [pd]}*, where *[ed]* denotes event-driven class, *[pd]* denotes periodical class;
- set of actuator classes *CHWR = {[rs], [td]}*, where *[rs]* denotes regular service class, *[td]* denotes task-driven class;
- set of agent operating modes *OA = {[rs], [ts]}* and set of AP operating modes *OHWP = {[rs], [ts]}*, where *[rs]* denotes regular service mode, *[ts]* denotes task service mode;
- predicate *confClTTask(ct, tt)* is true if a task of type *tt* belongs to class *ct*;
- predicate *confClTHwr(chwr, thwr)* determines whether an actuator of type *thwr* belongs to class *chwr*;
- predicate *confOmA(oa, a)* determines whether an agent *a* operates in mode *oa*;
- predicate *confOmHwp(ohwp, hwp)* determines whether AP *hwp* operates in mode *ohwp*;
- function *timed(tt)* determines a duration of single execution of a task of type *tt* that belongs to event-driven or periodical class;
- function *time(tt)* determines a fixed time interval between executions of a task of type *tt* that belongs to periodical class;
- function *stream(tt)* determines a rate of occurrence of accidental events that initiates execution of a task of type *tt* that belongs to event-driven class.

6.4 Failure Model for State Graph Based Approach

Reliability assessment requires information of failure behavior of each system component. We suppose that failure behaviors of components of redundant DHS MAS are defined by following functions:

- function *qft(tt)* determines a probability of failure during single execution for a task of type *tt* that belongs to either event-driven or periodical class;
- function *fra(a)* determines a failure rate for an agent *a* that operates in either regular service or task service mode;
- function *rda(a)* determines a reduction factor for an agent *a* that operates in task service mode;
- function *qfa(a)* determines a probability of failure during single execution of one of active tasks for an agent *a* that operates in task service mode;
- function *frp(hwp)* determines a failure rate for AP *hwp* that operates in either regular service or task service mode;
- function *rdp(hwp)* determines a reduction factor for AP *hwp* that operates in task service mode;
- function *qfp(hwp)* determines a probability of failure for AP *hwp* during single execution of one of active tasks via one of agents situated in this AP;

- function *frr(thwr)* determines a failure rate for an actuator of type *thwr*.

Generic state graph based approach is built upon Markovian model. Thus it’s assumed that failure rate of each system component is persistent while the system is in one of its states and time to failure of each system component is a random variate of exponential distribution [18]. We also assume that for each task of type that belongs to event-driven class random stream of accidental events that initiates execution of this task is a simple or Poisson stationary stream with exponentially distributed random intervals between events.

Let’s assume that the state graph of redundant DHS MAS is already formed in accordance with methodology described in section 6.2. To work out a system of differential equations in form (23) it’s required to determine transition intensities for each graph edge. As was noted above a transition intensity for a particular edge is calculated as a sum of failure rates of such system components that shall fail to initiate a transaction described by this edge. Because in accordance with extended redundant DHS MAS model failure rate and probability of failure of a particular component depends on its state as well as on states of other components, we shall define techniques for determination of failure rates for DHS MAS components in a particular system state.

Let’s consider a redundant DHS MAS state *SMAS*. A failure rate for a particular task *t* that is in an active state is determined as follows:

$$fratet(t) = \begin{cases} stream(tt)\cdot qft(tt),\ if\ \ confClTTask([ed],tt)=true \\ -\dfrac{\ln(1-qft(tt))}{time(tt)},\ if\ \ confClTTask([pd],tt)=true \end{cases}, \qquad (31)$$

where *tt* = *tOfTask(t)* is a type of task *t*, *[ed]* denotes event-driven task class, *[pd]* denotes periodical task class.

As an actuator of task-driven class operates and thus may fail only when it’s utilized by a particular task that is in an active state, let’s determine a rate of failures of a particular actuator that are caused by utilization of this actuator by a particular active task *t* as follows:

$$fratehwrtt(hwr,tt) = \begin{cases} stream(tt)\cdot(1-e^{frr(thwr)\cdot timed(tt)}), \\ \quad if\ \ confClTTask([ed],tt)=true \\ -frr(thwr)\cdot\dfrac{timed(tt)}{time(tt)}, \\ \quad if\ \ confClTTask([pd],tt)=true \end{cases}, \qquad (32)$$

where *thwr* = *tOfHwr(hwr)* is a type of an actuator *hwr, tt = tOfTask(t)* – a type of task *t* that utilize the actuator.

A failure rate for a particular actuator *hwr* that is in an active state is determined as follows:

$$fratehwr(hwr) = \begin{cases} frr(thwr),\ if\ \ confClTHwr([rs],thwr)=true \\ \sum\limits_{\substack{\forall t\in RT: cstate(t,SMAS)=[a]\wedge \\ confHwrHwp(hwr,hwpOfT(t))=true\wedge \\ thwr\in reqTHwr(tOfTask(t))}} fratehwrtt(hwr,tOfTask(t)), \\ \quad if\ \ confClThwr([td],thwr)=true \end{cases}, \qquad (33)$$

where *thwr* = *tOfHwr(hwr)* is a type of an actuator *hwr*, *t* – a task, *RT* – set of tasks, *SMAS* – a system state.

Let’s consider a particular agent that operates in a task service mode. Failure of this agent may be caused either by agent itself or by execution of one of its active tasks. A rate of failures of an agent that are caused by execution of a particular task *t* is determined in a following manner:

$$frateatt(a,tt) = \begin{cases} stream(tt)\cdot qfa(a),\ if\ \ confClTTask([ed],tt)=true \\ -\dfrac{\ln(1-qfa(a))}{time(tt)},\ if\ \ confClTTask([pd],tt)=true \end{cases}, \qquad (34)$$

where *a* is an agent, *tt* = *tOfTask(t)* is a type of task *t*.

Finally a failure rate for a particular agent *a* in DHS MAS state *SMAS* is determined as follows:

$$fratea(a) = \begin{cases} fra(a), \\ \quad if\ \ confOmA([rs],a)=true \\ fra(a)\cdot rda(a)+ \\ +\sum\limits_{\substack{\forall t\in RT: \\ cstate(t,SMAS)=[a]\wedge \\ agentOfT(t)=a}} frateatt(a,tOfTask(t)), \\ \quad if\ \ confOmA([ts],a)=true \end{cases}, \qquad (35)$$

where *a* is an agent, *t* – a task, *RT* – set of tasks, *SMAS* – a system state.

A rate of failures of an agent platform operating in a task service mode that are caused by execution of a particular task *t* situated in one of its agents is determined similarly to (34):

$$fratehwptt(hwp, tt) = \begin{cases} stream(tt) \cdot qfp(hwp), \ if \ confClTTask([ed], tt) = true \\ -\dfrac{\ln(1 - qfp(hwp))}{time(tt)}, \ if \ confClTTask([pd], tt) = true \end{cases}, \qquad (36)$$

where *hwp* is AP, *tt = tOfTask(t)* is a type of task *t*.

A failure rate for a particular AP in DHS MAS state SMAS is determined as follows:

$$fratehwp(hwp) = \begin{cases} frp(hwp), \\ \qquad if \ confOmHwp([rs], hwp) = true \\ frp(hwp) \cdot rdp(hwp) + \\ + \sum\limits_{\substack{\forall t \in RT: \\ cstate(t, SMAS) = [a] \wedge \\ hwpOfT(t) = hwp}} fratehwptt(hwp, tOfTask(t)), \\ \qquad if \ confOmHwp([ts], hwp) = true \end{cases}, \qquad (37)$$

where *hwp* is AP, *t* – a task, *RT* – set of tasks, *SMAS* – a system state.

It's worth noting that for a particular redundant DHS MAS state failure rates for tasks and ACTs that are not in active state are not determined as these components could not fail because of cold standby mode.

7 Experiments

The hypothesis to be verified though a set of experiments is as follows: the reliability level achieved though utilization of the developed reorganization technique described in section 2.2 and fault-recovery methodology described in section 3 is equal to the theoretical assessments

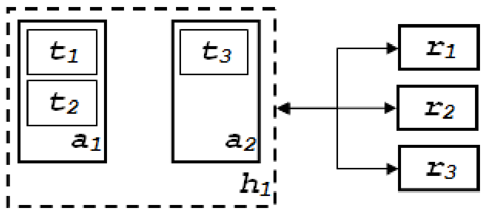

Fig. 1. Existing DHS MAS (MAS$_1$)

determined via logical-and-probabilistic and state graph based approaches.

7.1 Examples of Distributed Hardware-Software Multi-Agent Systems

The first existing DHS MAS (MAS$_1$) is presented on Fig. 1. MAS$_1$ is defined by following sets: a set of tasks $T = \{t_1, t_2, t_3\}$, a set of agents $A = \{a_1, a_2\}$, a set of *APs* $HWP = \{h_1\}$, a set of ACTs $HWR = \{r_1, r_2, r_3\}$. Its configuration is defined as follows:

- *confTaskAgent* is true on a set $\{(t_1, a_1), (t_2, a_1), (t_3, a_2)\}$;
- *confAgentHwp* is true on a set $\{(a_1, h_1), (a_2, h_1)\}$;
- *confHwrHwp* is true on a set $\{(r_1, h_1), (r_2, h_1), (r_3, h_1)\}$;
- *reqHwrTask* is true on a set $\{(r_1, t_1), (r_2, t_1), (r_2, t_2), (r_3, t_3)\}$.

Application of our reliability improvement methodology has transformed the existing DHS MAS MAS$_1$ to a redundant DHS MAS RMAS$_1$ that is presented on Fig. 2. RMAS$_1$ is defined by following sets: a set of task types $TT = \{tt_1, tt_2, tt_3\}$, a set of tasks $RT = \{t_{11}, t_{12}, t_{21}, t_{22}, t_{31}, t_{32}\}$, a set of agents $RA = \{a_1, a_2, a_3\}$, a set of APs $RHWP = \{h_1, h_2\}$, a set of ACT types $THWR = \{rt_1, rt_2, rt_3\}$ and a set of ACTs $RHWR = \{r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}\}$. The configuration of RMAS$_1$ is as follows:

- *confTypeTask* is true on a set $\{(tt_1, t_{11}), (tt_1, t_{12}), (tt_2, t_{21}), (tt_2, t_{22}), (tt_3, t_{31}), (tt_3, t_{32})\}$;
- *confTypeHwr* is true on a set $\{(rt_1, r_{11}), (rt_1, r_{12}), (rt_2, r_{21}), (rt_2, r_{22}), (rt_3, r_{31}), (rt_3, r_{32})\}$;

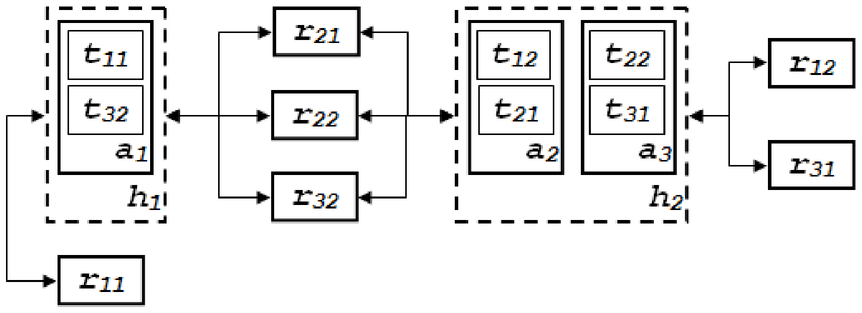

Fig. 2. Redundant DHS MAS (RMAS$_1$)

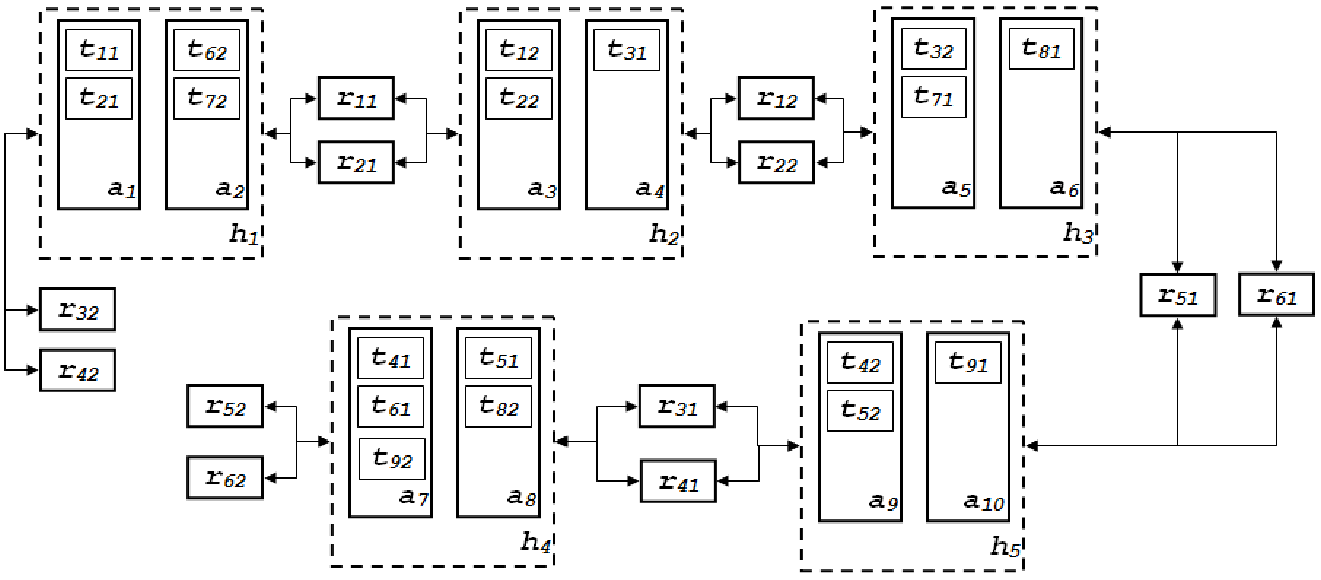

Fig. 3. Redundant DHS MAS (RMAS$_2$)

- *confTaskAgent* is true on a set $\{(t_{11}, a_1), (t_{12}, a_2), (t_{21}, a_2), (t_{22}, a_3), (t_{31}, a_3), (t_{32}, a_1)\}$;
- *confAgentHwp* is true on a set $\{(a_1, h_1), (a_2, h_2), (a_3, h_2)\}$;
- *reqTHwrTTask* is true on a set $\{(rt_1, tt_1), (rt_2, tt_1), (rt_2, tt_2), (rt_3, tt_3)\}$;
- *confHwrHwp* is true on a set $\{(r_{11}, h_1), (r_{12}, h_2), (r_{21}, h_1), (r_{21}, h_2), (r_{22}, h_1), (r_{22}, h_2), (r_{31}, h_2), (r_{32}, h_1), (r_{32}, h_2)\}$.

The second test redundant DHS MAS RMAS$_2$ is presented on Fig. 3 and is defined by following sets: a set of task types $TT = \{tt_1, tt_2, tt_3, tt_4, tt_5, tt_6, tt_7, tt_8, tt_9\}$, a set of tasks $RT = \{t_{11}, t_{12}, t_{21}, t_{22}, t_{31}, t_{32}, t_{41}, t_{42}, t_{51}, t_{52}, t_{61}, t_{62}, t_{71}, t_{72}, t_{81}, t_{82}, t_{91}, t_{92}\}$, a set of agents $RA = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$, a set of APs $RHWP = \{h_1, h_2, h_3, h_4, h_5\}$, a set of ACT types $THWR = \{rt_1, rt_2, rt_3, rt_4, rt_5, rt_6\}$ and a set of ACTs $RHWR = \{r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}, r_{41}, r_{42}, r_{51}, r_{52}, r_{61}, r_{62}\}$. The configuration of DHS MAS RMAS$_2$ is as follows:

- *confTypeTask* is true on a set $\{(tt_1, t_{11}), (tt_1, t_{12}), (tt_2, t_{21}), (tt_2, t_{22}), (tt_3, t_{31}), (tt_3, t_{32}), (tt_4, t_{41}), (tt_4, t_{42}), (tt_5, t_{51}), (tt_5, t_{52}), (tt_6, t_{61}), (tt_6, t_{62}), (tt_7, t_{71}), (tt_7, t_{72}), (tt_8, t_{81}), (tt_8, t_{82}), (tt_9, t_{91}), (tt_9, t_{92})\}$;
- *confTypeHwr* is true on a set $\{(rt_1, r_{11}), (rt_1, r_{12}), (rt_2, r_{21}), (rt_2, r_{22}), (rt_3, r_{31}), (rt_3, r_{32}), (rt_4, r_{41}), (rt_4, r_{42}), (rt_5, r_{51}), (rt_5, r_{52}), (rt_6, r_{61}), (rt_6, r_{62})\}$;
- *confTaskAgent* is true on a set $\{(t_{11}, a_1), (t_{12}, a_3), (t_{21}, a_1), (t_{22}, a_3), (t_{31}, a_4), (t_{32}, a_5), (t_{41}, a_7), (t_{42}, a_9), (t_{51}, a_8), (t_{52}, a_9), (t_{61}, a_7), (t_{62}, a_2), (t_{71}, a_5), (t_{72}, a_2), (t_{81}, a_6), (t_{82}, a_8), (t_{91}, a_{10}), (t_{92}, a_7)\}$;
- *confAgentHwp* is true on a set $\{(a_1, h_1), (a_2, h_1), (a_3, h_2), (a_4, h_2), (a_5, h_3), (a_6, h_3), (a_7, h_4), (a_8, h_4), (a_9, h_5), (a_{10}, h_5)\}$;
- *reqTHwrTTask* is true on a set $\{(rt_1, tt_1), (rt_1, tt_3), (rt_2, tt_2), (rt_2, tt_3), (rt_3, tt_4), (rt_3, tt_6), (rt_4, tt_5), (rt_4, tt_6), (rt_5, tt_7), (rt_5, tt_9), (rt_6, tt_8), (rt_6, tt_9)\}$;
- *confHwrHwp* is true on a set $\{(r_{11}, h_1), (r_{11}, h_2), (r_{12}, h_2), (r_{12}, h_3), (r_{21}, h_1), (r_{21}, h_2), (r_{22}, h_2), (r_{22}, h_3), (r_{31}, h_4), (r_{31}, h_5), (r_{32}, h_1), (r_{41}, h_4), (r_{41}, h_5), (r_{42}, h_1), (r_{51}, h_3), (r_{51}, h_5), (r_{52}, h_1), (r_{52}, h_4), (r_{61}, h_3), (r_{61}, h_5), (r_{62}, h_4)\}$.

Other example of redundant DHS multi-agent system is presented in [19].

7.2 Experiments for Logical-and-Probabilistic Approach

Experiments for logical-and-probabilistic approach are based on statistical modelling of time to failure of all components and further processing of the obtained sequence of failures on the imitation model of test redundant DHS MAS.

As logical-and-probabilistic approach is utilized for formation of analytic reliability function it's assumed that all components of redundant DHS MAS operates in hot standby mode.

All minimal functional configurations of redundant DHS MAS RMAS$_1$ determined in accordance with methodology described in section 5.1 are presented in Table 1. Several minimal operable configurations of RMAS$_1$ are presented in Table 2. Total number of MOCs of RMAS$_1$ is 24.

Experimental assessments of probability of no-failure *PSR(t)* as well as analytic reliability function *PAR(t)* for test redundant DHS MAS RMAS$_1$ and reliability function *PA(t)* for existing DHS MAS MAS$_1$ are presented on Fig. 4.

Experimental assessments for other test redundant DHS MASs are presented in [19].

7.3 Experiments for State Graph Based Approach

Experiments for state graph based approach are distinguished from experiments for logical-and-probabilistic approach that are described in section 7.2. It's worth noting that failure rates determined in accordance with (31) – (37) are used only for theoretical reliability assessment and are not utilized in experiments. Statistical modeling of time to failure values is used only for components which failure rates do not depends on states of other components, e.g. agents, agent platforms and actuators of regular service class. For tasks of both event-driven and periodical classes simulation of system operation is performed. For each single execution of a particular task following values are determined:

- next time instant of task execution (statistical modelling is used for tasks of event-driven class);
- state of an agent operating in a task service mode in which the task is deployed (i.e. whether the agent has failed due to task execution);
- state of AP operating in a task service mode in which an agent of the task is deployed;
- state of an actuator that is in an active state and is utilized during task execution.

For determination of mentioned above states statistical modelling of random variate of uniform distribution is utilized.

Table 1. Minimal functional configurations (RMAS$_1$)

MFC	MT	MA	MHWP
MFC$_1$	{t_{11}, t_{21}, t_{31}}	{a_1, a_2, a_3}	{h_1, h_2}
MFC$_2$	{t_{11}, t_{21}, t_{32}}	{a_1, a_2}	{h_1, h_2}
MFC$_3$	{t_{11}, t_{22}, t_{31}}	{a_1, a_3}	{h_1, h_2}
MFC$_4$	{t_{11}, t_{22}, t_{32}}	{a_1, a_3}	{h_1, h_2}
MFC$_5$	{t_{12}, t_{21}, t_{31}}	{a_2, a_3}	{h_2}
MFC$_6$	{t_{12}, t_{21}, t_{32}}	{a_1, a_2}	{h_1, h_2}
MFC$_7$	{t_{12}, t_{22}, t_{31}}	{a_2, a_3}	{h_2}
MFC$_8$	{t_{12}, t_{22}, t_{32}}	{a_1, a_2, a_3}	{h_1, h_2}

Table 2. Minimal operable configurations (RMAS$_1$)

MOC	MFC	MWR
MOC$_1$	MFC$_1$	{r_{11}, r_{21}, r_{31}}
MOC$_2$	MFC$_1$	{r_{11}, r_{21}, r_{32}}
MOC$_3$	MFC$_1$	{r_{11}, r_{22}, r_{31}}
MOC$_4$	MFC$_1$	{r_{11}, r_{22}, r_{32}}
MOC$_5$	MFC$_2$	{r_{11}, r_{21}, r_{32}}
MOC$_6$	MFC$_2$	{r_{11}, r_{22}, r_{32}}
MOC$_7$	MFC$_6$	{r_{12}, r_{21}, r_{32}}
MOC$_8$	MFC$_6$	{r_{12}, r_{22}, r_{32}}

Processing of each occurred failure by the imitation model of redundant DHS MAS in accordance with fault-recovery procedures described in section 3.2 results in switching DHS MAS to a new state though replacement of some system components with duplicates.

Several states and edges of state graph for redundant DHS MAS RMAS$_1$ are presented in Table 3 and Table 4 respectively. State graphs for RMAS$_1$ and RMAS$_2$ contains 43 and 3131 states.

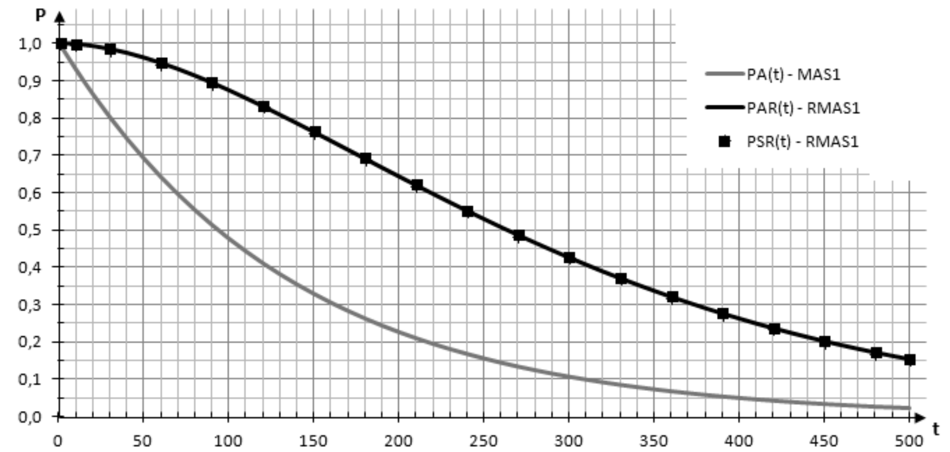

Fig. 4. Reliability functions and experimental assessments of probability of no-failure (MAS$_1$ and RMAS$_1$, logical-and-probabilistic approach)

Table 3. Redundant DHS MAS states ($RMAS_1$)

State	States of MAS components											
	t_{11}	t_{12}	t_{21}	t_{22}	t_{31}	t_{32}	r_{11}	r_{12}	r_{21}	r_{22}	r_{31}	r_{32}
1	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]
2	[f]	[a]	[a]	[s]	[a]	[s]	[f]	[a]	[a]	[s]	[a]	[s]
3	[a]	[s]	[f]	[a]	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]
4	[a]	[s]	[f]	[a]	[a]	[s]	[a]	[s]	[a]	[s]	[f]	[a]
5	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]	[f]	[a]	[a]	[s]
6	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]	[a]	[s]	[f]	[a]
7	[f]	[a]	[a]	[s]	[a]	[f]	[f]	[a]	[a]	[s]	[a]	[s]
8	[a]	[f]	[f]	[a]	[a]	[s]	[a]	[f]	[a]	[s]	[a]	[s]
9	[a]	[s]	[a]	[f]	[f]	[a]	[a]	[s]	[a]	[s]	[f]	[a]
10	Failure state - $SMAS_F$											

Table 4. Edges of state graph ($RMAS_1$)

Index of initial state	Index of final state	Set of failed components
1	2	$\{t_{11}, r_{11}\}$
1	3	$\{t_{21}\}$
1	4	$\{t_{31}\}$
1	5	$\{r_{21}\}$
1	6	$\{r_{31}\}$
1	7	$\{a_1, h_1\}$
1	8	$\{a_2\}$
1	9	$\{a_3\}$
1	10	$\{h_2\}$
2	10	$\{t_{12}, r_{12}, a_2, h_2\}$
3	10	$\{t_{22}, a_3, h_2\}$
4	10	$\{t_{32}, r_{32}, a_1, h_1, h_2\}$
5	10	$\{r_{22}, h_2\}$
6	10	$\{ r_{32}, h_2\}$
7	10	$\{ t_{12}, t_{31}, r_{12}, a_2, a_3, h_2\}$
8	10	$\{ t_{11}, t_{22}, r_{11}, a_1, a_3, h_1, h_2\}$
9	10	$\{ t_{21}, t_{32}, r_{32}, a_1, a_2, h_1, h_2\}$

We assume that tasks and actuators that are not in an active state do not operate and consequently could not fail. State graph based approach allows consideration of different operating modes of system components. Particular operating modes of agents and APs as well as classes of tasks and ACTs are defined by following MAS model extensions for both $RMAS_1$ and $RMAS_2$ redundant systems:

$$\begin{aligned}
&confClTTask([ed], tt_i) = true \Leftrightarrow i = 2n,\\
&confClTTask([pd], tt_i) = true \Leftrightarrow i = 2n+1,\\
&confClThwr([rs], thwr_i) = true \Leftrightarrow i = 2n,\\
&confClThwr([td], thwr_i) = true \Leftrightarrow i = 2n+1,\\
&\forall a : confOmA([ts], a) = true,\\
&\forall hwp : confOmHwp([ts], hwp) = true,
\end{aligned} \tag{38}$$

where tt_i is a task type, $thwr_i$ – a type of ACT, a – an agent, hwp – AP, *[ed]* denotes event-driven task class, *[pd]* denotes periodical task class, *[rs]* denotes regular service ACT class, *[td]* denotes task-driven ACT class, *[ts]* denotes a task service operating mode.

On Fig 5, we can see experimental assessments of probability of no-failure $PSR_1(t)$ and analytic reliability function $PAR_1(t)$ for test redundant DHS MAS $RMAS_1$, reliability function $PA_1(t)$ for

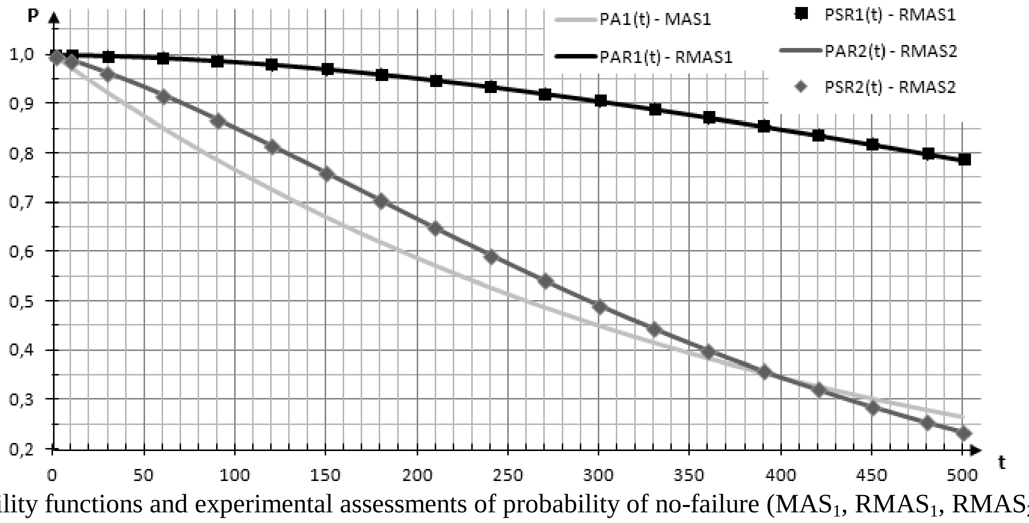

Fig. 5. Reliability functions and experimental assessments of probability of no-failure (MAS_1, $RMAS_1$, $RMAS_2$, state graph based approach)

existing DHS MAS MAS_1, experimental assessments of probability of no-failure $PSR_2(t)$ and analytic reliability function $PAR_2(t)$ for test redundant DHS MAS $RMAS_2$.

8 Conclusion

A distributed hardware-software multi-agent system has been considered as the object of the research. Reliability improvement, fault recovery and reliability assessment approaches were developed for the new object of DHS MAS.

This paper is an extended version of our conference paper [19] and is based on results of our previous studies [10], [11], [12]. We have started with a problem of reliability assessment for MAS that is intended for control of a set of actuators combined into executive conveyers and have suggested utilization of logical-and-probabilistic methods in [11]. The initial version of our fault-recovery methodology for MAS is presented in [10]. In [12] we have proved that our fault-recovery procedures ensure a level of fault-tolerance that corresponds to a theoretical assessment determined in accordance with logical-and-probabilistic method.

In this paper we finalize our research in the domain of fault-tolerance in MAS. Results includes definition of new object of DHS MAS, reliability improvement and fault-recovery techniques, MAS failure behavior and operation model, several reliability assessment approaches. Our fault-recovery procedures [10] were extended via strict definition of database schemas for MAS components that are required for an implementation of our fault-recovery methodology. Our previous suggestion of utilization of logical-and-probabilistic methods [11] was adopted for DHS MAS with independent actuators and a full methodology for operability function formation is presented. Due to highlighted disadvantages of logical-and-probabilistic methods we have developed a set of extensions for DHS MAS model that determines various failure behaviors and operating modes of redundant system. Moreover a new approach for reliability assessment of redundant DHS MAS that is based on state graph based method and Markovian model is presented. This approach includes new iterative methodology for state graph formation.

The methodology for reliability improvement is based on new reorganization technique that distinguish functional system components such as tasks and actuators and components that acts as universal executive containers such as agents and agent platforms. The reorganization technique improves reliability of DHS MAS via replication of all functional components and via introduction of a redundancy of universal executive containers.

The presented fault-recovery methodology was developed to deal with failures of particular components in redundant DHS MAS. The methodology consists of database schemas and fault-recovery procedures that are required for restoration of consistent system configuration after detected occurred failures of tasks, agents, agent platforms and actuators. The theorem on fault-tolerance property of redundant DHS MAS defines a conditions required for success of fault-recovery procedures.

Two different approaches were suggested for assessment of assured reliability level of redundant DHS MAS. The developed methodology for operability function formation enables utilization of known in the art logical-and-probabilistic methods for reliability assessment. However as logical-and-probabilistic approach is limited only for systems with hot standby, the second approach for reliability assessment was suggested based on state graph based method and Markovian model. For formation of state graph of redundant DHS MAS new methodology defined by an iterative algorithm was developed. Existing model of redundant DHS MAS was extended to consider different operating modes and failure behaviors of system components. New failure model for determination of failure rate of a particular component in accordance with states of other system components was introduced.

Performed computing experiments have validated that a reliability level assured though application of the methodology for reliability improvement and fault-recovery methodology corresponds to theoretical assessments obtained via logical-and-probabilistic and state graph based approaches.

References:

[1] L. C. Lee, H. S. Nwana, D. T. Ndumu, P. De Wilde, The stability, scalability and performance of multi-agent systems, *BT Technology Journal*, vol. 16(3), pp. 94–103, July 1998.

[2] H. F. Ahmad, A. Ali, Z. A. Khan, S. Shahid, H. Suguri, Decentralized architecture for fault tolerant multi agent system, in *Proceedings of the ISADS'2005 Autonomous Decentralized Systems*, pp. 167–174, 2005.

[3] R. Deters, A. Fedoruk, Improving fault-tolerance by replicating agents, in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pp. 737–744, 2002.

[4] A. Dakno, H. Zgaya, S. Hammadi, H. Hubert, Toward a multi-agent model for the care of patients at the emergency department, in *Proceedings of WSEAS International Conference on Mathematics and Computers in Science and Engineering*, pp. 264–269, 2008.

[5] Cristina Turcu, T. Cerlinca, Cornel Turcu, M. Cerlinca, R. Prodan, An RFID and multi-agent based system for improving efficiency in patient identification and monitoring, *WSEAS Transactions on Information Science and Applications*, vol. 6(11), 2009, pp. 1792–1801.

[6] M. Bertier, O. Marin, P. Sens, DARX – a framework for the fault-tolerant support of agent software, in *ISSRE'03 Proceedings of the 14th International Symposium on Software Reliability Engineering*, pp. 406–416, 2003.

[7] P. R. Cohen, S. Kumar, H. J. Levesque, The adaptive agent architecture: achieving fault-tolerance using persistent broker teams, in *Proceedings of Fourth International Conference on MultiAgent Systems*, pp. 159–166, 2000.

[8] S. Haegg, A sentinel approach to fault handling in multi-agent systems, *Multi-Agent Systems Methodologies and Applications*, pp. 181–195, 1997.

[9] S. Mellouli, A reorganization strategy to build fault-tolerant multi-agent systems, *Advances in Artificial Intelligence : Lecture Notes in Computer Science*, vol. 4509, pp. 61–72, 2007.

[10] A. V. Igumnov, S. E. Saradgishvili, Fault recovery in redundant multiagent systems, *St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems*, vol. 193(2), 2014, pp. 99–109.

[11] A. V. Igumnov, S. E. Saradgishvili, Reliability assessment for redundant multi-agent systems, *Electronic Journal "Science and Education: Electronic Scientific and Technical Periodical"*, vol. 1, 2014. Available: http://dx.doi.org/10.7463/0114.0696290).

[12] A. V. Igumnov, Fault-tolerance in redundant distributed hardware-software multi-agent systems, in *Proceedings of COMOD-2014 International Conference on Computer Modeling and Simulation*, pp. 155–161, 2014.

[13] S. Kraus, V. S. Subrahmanian, N. C. Tas, Probabilistically survivable mass, in *Proceedings of IJCAI'2003 International Joint Conference on Artificial Intelligence*, vol. 3, pp. 789–795, 2003.

[14] S. Kraus, E. Neisterski, V. S. Subrahmanian, D. Peleg, Y. Zhang, Computing the fault tolerance of multi-agent deployment, *Artificial Intelligence*, vol. 173(3), pp. 437–465, 2009.

[15] A. Chella, M. Cossentino, L. Sabatucci, Tools and patterns in designing multi-agent systems with PASSI, *WSEAS Transactions on Communications,* vol. 3(1), 2004, pp. 352–358.

[16] I. A. Ryabinin, G. N. Cherkesov, *Logiko-veroyatnostnye metody issledovaniya nadezhnosti strukturno-slozhnykh system [The logic-probabilistic research methods of structure-complex systems reliability]*, Moscow: Radio i svyaz' Publ., 1981, 264 p. (in Russian).

[17] A. M. Polovko, S. V. Gurov, *Osnovy teorii nadezhnosti [Reliability theory fundamentals]*, St.Petersburg: BHV-Piter, 2006, 703 p. (in Russian)

[18] G. N. Cherkesov, *Nadezhnost apparatno-programmnykh kompleksov [Reliability of hardware-software systems]*, St.Petersburg: Piter, 2005, 479 p. (in Russian)

[19] A. V. Igumnov, S. E. Saradgishvili On improvement of fault-tolerance in distributed hardware-software multi-agent systems and assessment of assured reliability, in *Recent Advances in Mathematical Methods in Applied Science, Proceedings of the 2014 International Conference on Mathematical Models and Methods in Applied Sciences (MMAS'14)*, pp. 295–302, 2014.