Вычислительные машины и программное обеспечение

УДК 004.052.3

А.В. Игумнов, С.Э. Сараджишвили

ВОССТАНОВЛЕНИЕ РАБОТОСПОСОБНОСТИ РЕЗЕРВИРОВАННЫХ МНОГОАГЕНТНЫХ СИСТЕМ

A.V. Igumnov, S.E. Saradgishvili

FAULT RECOVERY IN REDUNDANT MULTIAGENT SYSTEMS

Разработана модель резервированной многоагентной системы (МАС) как распределенного программно-аппаратного комплекса, основанная на избыточности задач, а не агентов. Предложена методика восстановления работоспособности резервированной МАС, включающая в себя протокол взаимодействия между компонентами МАС, обеспечивающий выполнение задачи независимо от ее расположения в компонентах МАС, технику поддержания неактивных резервных задач в актуальном состоянии и техники обнаружения отказов и восстановления работоспособности МАС. Достоверность разработанных моделей и методики восстановления работоспособности подтверждена сформулированной и доказанной теоремой о свойстве сохранения работоспособности МАС.

МНОГОАГЕНТНАЯ СИСТЕМА; ИЗБЫТОЧНОСТЬ; РЕЗЕРВИРОВАНИЕ; ОТКАЗ; ВОССТАНОВЛЕНИЕ РАБОТОСПОСОБНОСТИ.

The model of redundant multiagent system (MAS) as distributed hardware-software complex, which is based on replication of tasks instead of replication of agents, is developed. The developed fault recovery methodology presented in the article includes communication protocol for components of MAS, which enables task execution independently of its deployment or belonging to a particular component of MAS, technique for keeping inactive task replicas in actual state and techniques for fault detection and fault recovery. Developed models and methodology are validated by stated and proved theorem on fault tolerance property of MAS.

MULTIAGENT SYSTEM; REDUNDANCY; REPLICATION; FAULT; FAULT RECOVERY.

Применение агентно-ориентированных технологий [1] в промышленных системах определяет необходимость исследования надежности многоагентных систем (МАС) и разработки средств и методов обеспечения отказоустойчивости МАС. Введение избыточности является основным средством обеспечения отказоустойчивости технических систем [2]. Известные методики обеспечения отказоустойчивости МАС, такие как DARX [3], Meta-Agent [4] и Brokered MAS [5], используют избыточность агентов, однако в работе [6] предложено использование избыточности не только агентов, но и задач. Существующие методики обеспечения отказоустойчивости МАС обладают следующими недостатками:

рассматривают МАС только как систему, состоящую из взаимодействующих агентов [7], в то время как современные прикладные и промышленные системы, как правило, являются программно-аппаратными комплексами;

используют резервирование как метод повышения уровня надежности, но не определяют множество отказов и типы отказов, к которым МАС должна быть гарантированно устойчива.

Цель исследования – обеспечить восстановление работоспособности МАС,

реализованной в виде распределенного программно-аппаратного комплекса. В рамках работы решены следующие задачи:

- разработана формальная модель МАС как программно-аппаратного комплекса, учитывающая распределение задач по агентам и агентов по аппаратным компонентам, а также наличие доступа к ресурсам;
- разработана формальная модель резервированной МАС, основанная на введении избыточности задач, а не агентов МАС;
- определена классификация отказов компонентов МАС, устойчивость к которым необходимо обеспечить;
- разработана методика восстановления работоспособности МАС, основанная на модели резервированной МАС, включающая в себя:

протокол взаимодействия компонентов программно-аппаратной МАС, обеспечивающий выполнение задачи определенного типа независимо от ее расположения в компонентах МАС и использование исполнительного ресурса определенного типа;

техники обнаружения функциональных и логических отказов задач, отказов аппаратных компонентов МАС;

техники восстановления работоспособности для каждого из компонентов программно-аппаратной МАС;

технику поддержания неактивных резервных компонентов МАС в актуальном состоянии при изменениях условий выполнения своих действий интеллектуальными агентами;

- сформулирована и доказана теорема о свойстве сохранения работоспособности резервированной МАС, подтвердившая достоверность разработанных моделей и методики восстановления работоспособности.

Формальная модель МАС

Существующие методики обеспечения отказоустойчивости МАС основаны на моделях, представляющих МАС как систему взаимодействующих агентов. Поэтому для разработки методики восстановления работоспособности распределенной программно-аппаратной МАС необходимо разработать ее формальную модель, учитывающую не только взаимодействие отдельных агентов, но и необходимость использования аппаратных компонентов как для исполнения программных агентов, так и для выполнения агентами отдельных задач. Кроме того, т. к. агенты МАС способны модифицировать свои реакции на состояние окружающей среды, т. е. изменять условия запуска своих задач, то формальная модель МАС должна включать в себя компоненты, ответственные за определение данных условий.

Разработанная формальная модель МАС основана на следующих элементах модели, предложенной авторами работы [6]: $A = \{a_i\}$ – множество агентов МАС; $E = \{s_j\}$ – воздействий на систему; $T = \{t_q\}$ – задач МАС; $R = \{r_k\}$ – ресурсов, доступных МАС; $resources(t) \subseteq R$ – множество ресурсов, необходимых для выполнения задачи t; $prect(t) = \{c = s_1 \wedge ... \wedge s_u \mid \forall i \in 1...u : s_i \in E\}$ и $post(t)$ – множества предусловий и постусловий задачи t.

Введем понятия агентной платформы (АП) и исполнительного ресурса. Агентная платформа – это программно-аппаратный компонент системы, предназначенный для исполнения агентов. Введем множество $HWP = \{hwp_i\}$ АП МАС. Пусть предикат $cA\_HWP : A \times HWP \to \{true, false\}$ определяет принадлежность агента АП, тогда функция $AofHWP : HWP \to \varphi(A), \varphi(A) \subseteq A$ определяет множество агентов $AofHWP(hwp) = \{a_i \in A \mid \forall i : cA\_HWP(a_i, hwp) = true\}$, исполняющихся на АП hwp. Так как согласно [6] агент МАС a исполняет ряд задач, то введем предикат $cT\_A : T \times A \to \{true, false\}$, определяющий принадлежность задачи агенту. Тогда функция $TofA : A \to \varphi(T)$ определяет множество задач $TofA(a) = \{t_i \in T \mid \forall i : cT\_A(t_i, a) = true\}$, принадлежащих агенту a, а функция $AofT : T \to \varphi(A)$ определяет агента a, которому принадлежит задача t: $a = AofT(t) \Leftrightarrow cT\_A(t, a) = true$. Введем функцию $TofHWP : HWP \to \varphi(T)$, определяющую множество задач $TofHWP(hwp) = \{t_i \in T \mid \forall i : \exists a \in AofHWP(hwp) : cT\_A(t_i, a) = true\}$, исполняемых АП hwp. Исполнительный ресурс (ИР) – это аппаратный компонент системы, необходимый агенту для выполнения

его задач. Введем множество $HWR = \{hwr_i\}$ ИР МАС. Пусть подмножество ресурсов $\{R \backslash HWR\}$, не являющихся ИР, включает только программные ресурсы, которые могут быть предоставлены любой АП, тогда можно считать, что множество ресурсов МАС содержит только ИР. Доступность отдельного ИР определяется конфигурацией МАС, поэтому пусть предикат $cHWR\_HWP : HWR \times HWP \to \{true, false\}$ определяет доступность ИР hwr для АП hwp. Тогда функция $HWRofHWP : HWP \to \varphi(HWR)$ определяет множество ИР $HWRofHWP(hwp) = \{hwr_i \in HWR \mid \forall i : cHWR\_HWP(hwr_i, hwp) = true\}$, доступных АП hwp. Пусть предикат $cT\_HWR : T \times HWR \to \{true, false\}$ задает необходимость использования ИР задачей, тогда функция $rHWRofT : T \to \varphi(HWR)$ определяет множество необходимых для выполнения задачи t ИР $rHWRofT(t) = \{hwr_i \in HWR \mid \forall i : cT\_HWR(t, hwr_i) = true\}$.

В случае закрытой МАС [8] необходимость выполнения задачи может определяться наблюдаемым состоянием МАС [6]. Так как интеллектуальные агенты способны модифицировать свои реакции на состояние МАС, то необходимость выполнения задачи должна определяться блоком принятия решений (БПР) агента. Будем считать, что каждой задаче $t \in TofA(a)$ агента a можно поставить в соответствие БПР, определяющий необходимость ее выполнения. Введем множество $DM = \{dm_i\}$ БПР, тогда предикат $cT\_DM : T \times DM \to \{true, false\}$ определяет соответствие БПР и задачи.

Исходная МАС задана множеством $MAS = <T, A, HWP, HWR, E, DM>$, где A – множество агентов, DM – БПР, T – задач, E – воздействий на систему, HWR – множество ИР (ресурсов), HWP – АП.

Модель резервированной МАС

Разработанная модель МАС позволяет определить ряд отказов отдельных компонентов, приводящих к отказу МАС:

отказ АП hwp, приводящий к отказу задач $TofHWP(hwp)$ данной АП;

программный отказ агента a, приводящий к отказу задач $TOfA(a)$;

программный отказ задачи;

логический отказ задачи t, т. е. невыполнение хотя бы одного из предусловий из множества $prec(t)$;

функциональный отказ задачи t, т. е. невыполнение хотя бы одного из постусловий из множества $post(t)$;

отказ ИР hwr, приводящий к отказу множества задач $\{t_i \in T \mid \forall i : cT\_HWR(t_i, hwr) = true\}$, использующих данный ИР.

В соответствии с [2] основным средством обеспечения отказоустойчивости системы является введение избыточности ее компонентов. Для обеспечения отказоустойчивости программно-аппаратной МАС будем использовать резервирование задач системы. Кроме того, будем считать, что для обеспечения отказоустойчивости может быть использовано резервирование ИР, необходимых для выполнения задач, а также могут быть введены дополнительные агенты и АП. Для разработки техник обнаружения отказов, восстановления работоспособности, поддержания актуального состояния неактивных резервных компонентов необходимо разработать формальную модель резервированной программно-аппаратной МАС, основанную на резервировании задач и ИР и введении дополнительных агентов и АП.

В работе [6] высказано предположение, что логический отказ задачи может быть преодолен, если воздействия на МАС, определяющие предусловия задачи, подконтрольны МАС, однако механизм контроля не был предложен. Определим эффектор как задачу, способную изменять значение одного из воздействий на МАС. Введем множество $EF = \{ef_i\}$ эффекторов и предикат $cEF\_ST : EF \times E \to \{true, false\}$, задающий соответствие эффектора и воздействия на МАС. Определим внутренний контекст МАС как множество $IC = \{s_i \in E \mid \forall i : \exists ef \in EF : cEF\_ST(ef, s_i) = true\}$ тех воздействий, для которых существуют эффекторы.

Эффекторы и задачи исходной МАС образуют расширенное множество задач МАС $ET = T \cup EF$. Для обеспечения отказоустойчивости предлагается вводить

избыточность задач и соответствующих им БПР. Введем множество типов задач $TT = \{tt_i\}$ так, чтобы каждой задаче $et_i \in ET$ соответствовал один тип задачи. Для эффекторов соответствие типа задачи элементу внутреннего контекста задается функцией $TTofST : IC \to \varphi(TT)$. Пусть $RT = \{t_i\}$ – множество задач резервированной МАС, тогда функция $TYofT : RT \to \varphi(TT)$ определяет тип задачи. Из группы задач-реплик одного типа только одна реплика является активной, а агент МАС исполняет только БПР, соответствующие активным репликам. Пусть предикат $cATofTT : RT \times TT \to \{true, false\}$ определяет, является ли задача активной репликой. В резервированной МАС вместо запроса на исполнение задачи вводится запрос на исполнение задачи определенного типа, обработка которого приводит к выполнению активной реплики.

После введения избыточности задач необходимо определить структуру резервированной МАС, при этом должны быть решены следующие задачи: введение избыточности ИР, введение дополнительных агентов и АП, распределение задач по агентам и агентов по АП. Решение поставленных задач должно быть обеспечено некоторой методикой синтеза структуры резервированной МАС, не описанной в данной статье. Так как допускается использование избыточности ИР, то введем множество типов ИР $HWRT = \{hwrt_i\}$ и множество ИР резервированной МАС $RHWR = \{hwr_i\}$, тогда функция $TYofHWR : RHWR \to \varphi(HWRT)$ определяет тип ИР. Пусть предикат $cTT\_HWRT : TT \times HWRT \to \{true, false\}$ задает необходимость использования ИР определенного типа задачей определенного типа, тогда функция $rHWRTofTT : TT \to \varphi(HWRT)$ определяет множество типов ИР $rHWRTofTT(tt) = \{hwrt_i \in HWRT \mid \forall i : cTT\_HWRT(tt, hwrt_i) = true\}$, необходимых для выполнения задачи типа *tt*.

Резервированная МАС задана множеством $RMAS = <RT, RA, RHWP, RHWR, E, DM, IC, TT, HWRT>$, где RA – множество агентов; DM – множество БПР; RT – задач; E – воздействий на систему; IC – внутренний контекст; $RHWR$ – множество ИР; $RHWP$ – АП; TT – множество типов задач; $HWRT$ – типов ИР. Конфигурация резервированной МАС задана предикатами и функциями: $cT\_A : RT \times RA \to \{true, false\}$ определяет распределение задач по агентам; $cA\_HWP : RA \times RHWP \to \{true, false\}$ – распределение агентов по АП; $cHWR\_HWP : RHWR \times RHWP \to \{true, false\}$ – доступность отдельного ИР для АП; cTT_HWRT – необходимость использования ИР определенного типа задачей определенного типа; $cT\_DM : RT \times DM \to \{true, false\}$ определяет соответствие БПР и задачи; $TTofST$ определяет тип эффектора, контролирующего элемент внутреннего контекста; $TYofT$ и $TYofHWR$ определяют тип задачи и ИР; $precTT : TT \to \{c = s_1 \wedge ... \wedge s_u \mid \forall i \in 1...u : s_i \in E\}$ – предусловия для задачи определенного типа; $cATofTT$ – активную реплику для каждого типа задачи.

Протокол взаимодействия компонентов МАС

Для обеспечения взаимодействия компонентов резервированной МАС необходимо разработать соответствующий протокол взаимодействия. Протокол взаимодействия должен обеспечивать прозрачность резервирования, т. е. возможность запуска задачи определенного типа без необходимости явного выбора конкретной задачи из группы реплик, а также независимо от размещения активной реплики данного типа в агентах и АП МАС. Аналогично протокол взаимодействия должен обеспечивать использование ИР определенного типа без явного указания конкретного ИР. Также протокол взаимодействия компонентов МАС необходим для поддержания актуального состояния неактивных резервных компонентов, обнаружения отказов и восстановления работоспособности системы.

Введем протокол обмена сообщениями. Пусть функция *send*(*msg*(*args*), *dest*) означает посылку сообщения *msg* с параметрами *args* получателю *dest*. Передача сообщений возможна между АП, задачей и агентом, агентом и АП. Введем особые значения параметра *dest*: *up* – получателем является вышестоящий компонент МАС, т. е.

агент для задачи и АП для агента; *bcst* – сообщение отправлено всем АП. Функция *dest:recv*(*msg*(*args*), *src*) означает обработку компонентом *dest* сообщения, полученного от отправителя *src*.

Агент МАС должен обладать базой данных (БД), содержащей информацию о задачах, необходимых типах ИР и наличии активных реплик. БД может быть реализована в виде таблиц, содержащих записи вида: (*t*, *tt*), связывающие идентификатор задачи и тип задачи; (*tt,* {*hwrt*}), связывающие тип задачи и необходимые типы ИР; (*tt*, *t*), определяющие наличие активных реплик. БД агента обеспечивает реализацию функций *TofA*, *TYofT*, *rHWRTofTT* и позволяет ввести функцию $ATofTTinA : RA \times TT \to \varphi(RT)$, определяющую задачу, являющуюся активной репликой определенного типа. Агент должен реализовывать процедуры для модификации БД: *a_setATofTT*(*t*, *tt*) обеспечивает обновление таблицы активных реплик так, чтобы типу *tt* соответствовала реплика *t*; *a_rm_t*(*t*) – удаление всех записей о задаче *t*.

АП должна обладать БД, содержащей информацию об агентах, доступных ИР, а также о расположении активных реплик всех типов как в агентах данной АП, так и в агентах удаленных АП. БД АП может быть реализована в виде таблиц, содержащих записи вида: (*a*, *t*, *tt*), определяющие распределение задач (и их типы) по агентам; (*hwr*, *hwrt*), определяющие доступные ИР и их типы; (*tt*, {*hwrt*}), определяющие необходимые типы ИР в соответствии с типом задачи; {*tt*, *hwp*, *a*}, определяющие расположение активных реплик. Пусть функции $AHWPofTT : TT \to \varphi(RHWP)$ и $AAofTT : TT \to \varphi(RA)$ определяют (согласно БД АП) АП и агента, которым принадлежит активная реплика определенного типа. БД АП *hwp* должна содержать информацию о размещении активных реплик каждого типа для обеспечения взаимодействия между любыми задачами и агентами, т. е. должно быть выполнено условие $\forall tt \in TT : \exists hwp^* = AHWPofTT(tt) \wedge ((hwp^* = hwp) \Rightarrow \exists a = AAofTT(tt))$. Информация об агенте и размещенных в нем активных репликах вносится в БД при получении сообщения *a_cfg_anc*, которое посылается агентом при размещении на АП и изменениях БД агента. Информация о размещении активных реплик в удаленных АП – при получении сообщения *arepl_anc*(*tt*, *hwp*), которое посылается каждой АП при активизации реплики. БД АП обеспечивает реализацию функций *AofHWP*, *TofHWP*, *TYofT*, *TofA*, *AofT*, *rHWRTofTT*, *HWRofHWP*, *TYofHWR*. АП должна реализовывать процедуры для модификации БД: *hwp_rm_t*(*t*) обеспечивает удаление записей о задаче *t*; *hwp_rm_hwr*(*hwr*) – удаление записи о ИР *hwr* из таблицы доступных ИР; *hwp_setAAofTT*(*tt*, *a*) – обновление таблицы активных реплик так, чтобы типу *tt* соответствовал агент *a*; *hwp_rmATofTT*(*tt*) – удаление из таблицы активных реплик записи о типе *tt*. АП может инициировать изменение конфигурации любого из ее агентов, т. е. запросить выполнение агентом *a* процедур *a_setATofTT* и *a_rm_t* через отправку сообщений *a_cmd_AT*(*t*) и *a_cmd_rm*.

Задачи МАС могут требовать исполнения других задач, что в рамках модели резервированной МАС означает требование исполнения задачи определенного типа. Агент МАС *a* может запустить выполнение задачи $t \in TofA(a)$ при помощи процедуры *a_pfm_t*(*t*). Запрос на исполнение задачи определенного типа реализуется путем посылки сообщения *pfm*. Сообщение *pfm,* полученное от задачи, обрабатывается агентом: *a:recv*(*pfm*(*tt*), *t*) {if $(\exists t^* = ATofTTinA(a, tt))$ then *a_pfm_t*(*t**) else *send*(*pfm*(*tt*), *up*)}. Обработка сообщения *pfm,* полученного от агента, осуществляется АП: *hwp:recv*(*pfm*(*tt*), *a*) {$hwp^* = AHWPofTT(tt)$; if ($hwp^* == hwp$) then *send*(*pfm*(*tt*), *AAofTT*(*tt*)) else *send*(*pfm*(*tt*), *hwp**)}. Обработка сообщения *pfm*, полученного от удаленной АП, осуществляется АП: *hwp:recv*(*pfm*(*tt*), *hwp**) {*send*(*pfm*(*tt*), *AAofTT*(*tt*)}. Сообщение *pfm*, полученное от АП, обрабатывается агентом: *a:recv*(*pfm*(*tt*), *hwp*) {*a_pfm_t*(*ATofTTinA*(*a*, *tt*))}.

Запрос задачи на использование ИР определенного типа реализуется путем посылки сообщения *send*(*use_hwr*(*hwrt*), *up*), которое пересылается агентом вышестоящей АП: *a:recv*(*use_hwr*(*hwrt*), *t*) {*send*(*use_hwr*(*hwrt*, *up*)}. Так как АП имеет непосредственный

доступ к ИР, то АП *hwp* может использовать ИР $hwr \in HWRofHWP(hwp)$ посредством процедуры *hwp_use_hwr*(*hwr*), возвращаемое значение которой указывает на успешное использование ИР или его отказ. При обработке сообщения *use_hwr* АП должна обеспечить использование ИР или зафиксировать его отказ: *hwp:recv*(*use_hwr*(*hwrt*), *a*) {if not (($\exists hwr \in HWRofHWP(hwp)$: $TYofHWR(hwr) = hwrt$) AND (*hwp_use_hwr*(*hwr*))) then {*hwp_rm_hwr*(*hwr*); *hwp_hwr_fail*(*hwrt*)}.

Поскольку необходимость выполнения задачи определяется БПР, который может быть модифицирован интеллектуальным агентом, и только БПР, соответствующий активной реплике, исполняется агентом, необходимо обеспечить возможность модификации БПР неактивных реплик. Если агент *a* фиксирует изменение БПР *dm*, соответствующего задаче *t* (т. е. $cT\_DM(t, dm) = true$), то он, определив тип задачи $tt = TYofT(t)$, выполняет процедуру *update_dm*(*tt*, *dm*) и посылает сообщение *send*(*upd_dm*(*tt*, *dm*), *up*). Процедура *update_dm*(*tt*, *dm*) заключается в модификации БПР всех задач агента определенного типа $\{t \in TofA(a) \mid TYofT(t) = tt\}$. АП пересылает сообщение *upd_dm* всем своим агентам и удаленным АП: *hwp:recv*(*upd_dm*(*tt*, *dm*), *a*) {foreach ($a^* \in AofHWP(hwp)$) *send*(*upd_dm*(*tt*, *dm*), *a*); *send*(*upd_dm*(*tt*, *dm*), *bcst*)}. Согласно алгоритму обработки сообщения *upd_dm*, оно будет доставлено всем агентам МАС, которые при его получении выполнят процедуру *update_dm*. Этот механизм позволяет поддерживать актуальное состояние всех неактивных реплик, т. к. такие задачи информируются об изменениях в БПР активной реплики, т. е. об изменениях условий своего запуска.

Обнаружение отказов компонентов МАС

Для восстановления работоспособности системы необходимо зафиксировать состояние отказа. Поэтому определим техники обнаружения отказов, которым подвержена программно-аппаратная МАС, заданная предложенной нами формальной моделью.

Для обнаружения программных отказов задач и агентов предлагается использовать существующие техники сторожевых watchdog-таймеров [9] в агентах и АП. Для обнаружения функционального отказа задачи агент должен контролировать выполнение постусловий после завершения задачи. Функциональный отказ задачи определяется условием $\exists c = s_1 \wedge ... \wedge s_k \in post(t) : (\forall i \in 1...k : s_i \in E) \quad \wedge(\exists u \in 1...k : s_u = false)$. Для обнаружения логического отказа задачи агент должен контролировать выполнение предусловий перед запуском задачи. Логический отказа задачи определяется условием $\exists c \in precTT(TYofT(t)) : c = false$. Отказ ИР фиксируется АП, взаимодействующей с этим ИР, в процессе обработки сообщения *use_hwr*. Условием отказа ИР является отсутствие доступного АП ИР запрошенного типа $\forall hwr \in HWRofHWP(hwp) : TYofHWR(hwr) \neq \neq hwrt$ или отрицательный результат выполнение процедуры *hwp_use_hwr*().

Для обнаружения отказа АП необходимо использовать протокол рукопожатия. Пусть каждая АП *hwp* МАС сообщает о своей жизнеспособности путем посылки сообщения *send*(*alv*(*hwp*), *bcst*). Если сообщения *alv* посылаются каждой АП с периодичностью *t_a* и моменты посылки сообщений не синхронизированы, то любая АП, анализируя принятые сообщения в течение периода времени $t\_a + \Delta t$, может определить множество АП *ALV*, с которыми имеется связь, и множество АП *DEAD*, с которыми связь отсутствует. Пусть в начале цикла обнаружения отказов АП выполняет процедуру *hwp_startd* {$ALV = 0$}, а в течение цикла происходит следующая обработка *alv* сообщений: *hwp:recv*(*alv*(*hwp**), *hwp**) {$ALV+=hwp^*$}, тогда при завершении цикла может быть определено множество $DEAD = (RHWP \backslash ALV) \backslash \{hwp\}$. Если АП *hwp* фиксирует отсутствие связи с АП *hwp**, то возможен либо отказ *hwp**, либо отказ одной из линий связи между *hwp* и *hwp**. Будем считать, что все АП МАС объединены сетью, организованной согласно топологии звезда или шина [10]. Тогда отказ линии связи между АП *hwp* и центральным звеном сети (общей шиной или коммутатором) означает, что *hwp* не получит *alv* сообщений ни от одной другой АП МАС, а ни одна другая АП МАС *hwp** не получит *alv*

сообщения от *hwp*. Если АП *hwp* фиксирует отсутствие связи со всеми удаленными АП, что определяется условием $DEAD = RHWP \setminus \{hwp\}$, то АП не может определить находятся ли все удаленные АП в состоянии отказа или отсутствие связи обусловлено отказом линии связи между самой АП и центральным звеном сети, поэтому АП фиксирует свой собственный отказ и останавливает работу всех размещенных в ней агентов. Если АП фиксирует отказ части удаленных АП, то она должна выполнить процедуру *hwp_rhwp_fail* для восстановления, значит, процедура окончания цикла обнаружения отказов должна быть реализована следующим образом: *hwp_endd*() {$DEAD = (RHWP \setminus ALV) \setminus \{hwp\}$; if ($DEAD \neq RHWP \setminus \{hwp\}$) then foreach ($hwp^* \in DEAD$) *hwp_rhwp_fail*($hwp^*$)}.

Восстановление работоспособности резервированной МАС

Для восстановления работоспособности резервированной МАС, заданной разработанной нами формальной моделью, необходимо разработать процедуры, которые должны быть выполнены различными компонентами МАС в случае обнаружения отказов. При этом будем считать, что процесс восстановления работоспособности должен быть организован иерархически. Так, первая попытка восстановления работоспособности предпринимается компонентом, зафиксировавшим отказ. В случае неудачи задача восстановления работоспособности делегируется компоненту, в котором размещен компонент, зафиксировавший отказ. Если восстановление работоспособности невозможно в рамках одной АП, то решение данной задачи обеспечивается совместными действиями всех АП МАС.

Обнаруженный логический отказ задачи может быть преодолен, если невыполнение предусловий обусловлено воздействиями на МАС $s_i \in IC$, принадлежащими внутреннему контексту. Множество воздействий, обусловливающих невыполнение предусловий задачи *t*, определяется как $SF = \{sf_i \mid \forall i : sf_i = false \wedge (\exists c = s_1 \wedge \ldots \wedge sf_i \wedge \ldots \wedge s_k \in precTT(TYofT(t)))\}$. Определим условие восстановления работоспособности после логического отказа задачи: $\forall sf \in SF : sf \in IC$. При обнаружении логического отказа задачи *t* агент *a* выполняет следующую процедуру для вызова необходимых эффекторов *a_t_lfail*(*t*) {if ($\forall sf \in SF : sf \in IC$) then foreach ($sf \in SF$) {$tt = TTofST(sf)$; if ($\exists t^* = ATofTTinA(a, tt)$) then *a_pfm_t*($t^*$) else *send*(*pfm*(*tt*), *up*)}}.

Так как любой из отказов компонентов МАС приводит к отказу одной или нескольких активных реплик, то восстановление работоспособности сводится к нахождению и активизации задачи-реплики соответствующего типа. Если реплика не найдена в рамках агента, зафиксировавшего отказ задачи, то запрос на поиск реплики передается АП, в которой размещен агент, в виде сообщения *req_repl*. При обнаружении функционального или программного отказа задачи *t* агент *a* выполняет процедуру *a_t_fail*(*t*) {$tt = TYofT(t)$; *a_rm_t*(*t*); if ($\exists t^* \in TofA(a) : TYofT(t^*) = tt$) then *a_setATofTT*($t^*$, *tt*) else *send*(*req_repl*(*tt*), *up*)}. При получении от агента сообщения *req_repl* АП выполняет процедуру поиска и активизации реплики: *hwp:recv*(*req_repl*(*tt*), *a*) {*hwp_t_fail*(*tt*)}. Если реплика не найдена в рамках АП, то запрос на поиск реплики передается всем удаленным АП: *hwp_t_fail*(*tt*) {*hwp_rmATofTT*(*tt*); if ($\exists t \in TofHWP(hwp) : TYofT(t) = tt$) then {$a = AofT(t)$; *hwp_setAAofTT*(*tt*, *a*); *send*(*a_cmd_AT*(*t*), *a*)} else *send*(*req_repl*(*tt*), *bcst*)}. Поиск и активизация реплики в удаленных АП осуществляется при приеме сообщения: *hwp:recv*(*req_repl*(*tt*), *hwp**) {*hwp_rmATofTT*(*tt*); if ($\exists t \in TofHWP(hwp) : TYfT(t) = tt$) then {$a = AofT$ (*t*); *hwp_setAAofTT*(*tt*, *a*); *send*(*a_cmdAT*(*t*), *a*); *send*(*arepl_anc*(*tt*, *hwp*), *bcst*)}}.

Программный отказ агента *a* является причиной отказов активных задач-реплик, размещенных в данном агенте. При обнаружении программного отказа агента *a* АП *hwp* определяет множество типов отказавших активных реплик $TTF = \{tt \in TT \mid AAofTT(tt) = a\}$ и выполняет процедуру, обеспечивающую поиск и активизацию соответствующих задач-реплик: *hwp_a_fail*(*a*) {foreach ($t \in TofA(a)$) *hwp_rm_t*(*t*); foreach ($tt \in TTF$) *hwp_t_fail*(*tt*);}.

Отказ ИР $hwr \in HWRofHWP(hwp)$ при отсутствии других доступных АП *hwp* ИР типа $hwrt = TYofHWR(hwr)$ приводит к от-

казам активных задач-реплик, для выполнения которых необходимо использование ИР типа *hwrt*. Если АП *hwp* фиксирует отказ ИР типа *hwrt*, то АП определяет множество типов отказавших активных задач-реплик $TTFR = \{tt \in TT \mid hwrt \in rHWRTofTT(tt) \wedge \exists a \in AofHWP(hwp) : AAofTT(tt) = a\}$, а также множество задач АП $TFR = \{t \in TofHWP(hwp) \mid$ которые требуют наличия ИР типа *hwrt*. Так как задачи из *TFR* не могут быть выполнены на данной АП в силу недоступности необходимого ИР, то следует удалить информацию о них из БД агентов и АП, чтобы эти задачи не рассматривались в процессе поиска и активизации реплик. Восстановление работоспособности обеспечивается процедурой: *hwp_hwr_fail*(*hwrt*) {foreach ($t \in TFR$) {*hwp_rm_t*(*t*); *send*(*a_cmd_rm_t*(*t*), *AofT*(*t*));}; foreach ($tt \in TTFR$) *hwp_t_fail*(*tt*)}.

Отказ АП *hwp** приводит к отказам всех активных задач-реплик, размещенных на данной АП. Если АП *hwp* фиксирует отказ удаленной АП *hwp**, то необходимо определить множество типов отказавших активных задач-реплик $TTFH = \{tt \in TT \mid AHWPofTT(tt) = hwp^*\}$. Поиск и активизация соответствующих задач-реплик осуществляется процедурой *hwp_rhwp_fail*(*hwp**) {foreach ($tt \in TTFH$) {*hwp_t_fail*(*tt*); if ($\exists a = AAofTT(tt)$) then *send*(*arepl_anc*(*tt*, *hwp*), *bcst*)}}.

Доказательство свойства сохранения работоспособности

Покажем, что использование разработанных процедур, составляющих методику восстановления работоспособности МАС, и разработанного протокола взаимодействия компонентов МАС обеспечивает сохранение работоспособности резервированной программно-аппаратной МАС, заданной предложенной нами формальной моделью.

Сформулируем теорему. При функциональных, логических и программных отказах задач, программных отказах агентов, отказах АП и ИР резервированная МАС сохраняет работоспособность, т. е. обеспечивает выполнение задачи типа *tt*, если после обнаружения отказа в системе существует задача-реплика типа *tt*, имеющая доступ к необходимым для ее выполнения ИР, и логический отказ задачи обусловлен только воздействиями, принадлежащими внутреннему контексту МАС.

Доказательство. Выполнение задачи типа *tt* может быть обусловлено реакцией МАС или запросом другой задачи. В первом случае задача будет выполнена, если существует активная реплика данного типа, т. к. реакция МАС определяется БПР, а агенты МАС исполняют только БПР, соответствующие активным репликам. Во втором случае согласно протоколу взаимодействия компонентов МАС обработка сообщения *pfm*(*tt*) обеспечивает исполнение активной реплики типа *tt* независимо от ее размещения в компонентах МАС. Следовательно, МАС обеспечивает выполнение задачи типа *tt*, если существует активная реплика данного типа t^*: $cATofTT(t^*,tt) = true$.

Рассмотрим случай программного или функционального отказа задачи *t*. В соответствии с условием теоремы существует задача t^*: $TYofT(t^*) = TYofT(t)$, которая должна быть активирована для сохранения работоспособности. Отказ задачи *t* фиксируется агентом *a*: $cT\_A(t,a) = true$. Если задача t^* принадлежит агенту *a*, то ее активизация обеспечивается процедурой *a_t_fail*(*t*). В противном случае агент *a* передает запрос на активизацию АП, в которой он размещен. АП согласно процедуре *hwp_t_fail*(*tt*) обеспечивает активизацию задачи t^*, если она принадлежит данной АП, или передает запрос на активизацию всем удаленным АП. Если задача t^* принадлежит одной из удаленных АП, то она будет активирована при получении запроса *req_repl*. Значит, при обнаружении программного или функционального отказа задачи будет активирована соответствующая реплика, т. е. сохранена работоспособность МАС. Программный отказ агента *a* приводит к отказам всех активных задач-реплик, размещенных в агенте. Отказ агента фиксируется АП, в которой он размещен. Согласно процедуре *hwp_a_fail*(*a*) АП определяет множество отказавших активных реплик и выполняет для каждого из них процедуру *hwp_t_fail*, которая, как было доказано, обеспечивает активизацию задачи t^* типа *tt* независимо от ее размещения в компонен-

тах МАС. Следовательно, если выполнено условие теоремы, при отказе агента *a* для каждой из размещенных в нем активных реплик будет найдена и активирована новая реплика, т. е. будет восстановлена работоспособность МАС. Отказ АП *hwp** приводит к отказам всех активных задач-реплик, размещенных в агентах данной АП. АП *hwp* согласно протоколу рукопожатия фиксирует отказ АП *hwp**, согласно процедуре *hwp_rhwp_fail*(*hwp**) определяет множество типов отказавших активных реплик в соответствии с БД и для каждого типа выполняет процедуру *hwp_t_fail*. Если выполнено условие теоремы, то данная процедура обеспечивает активизацию задачи *t** типа *tt*, следовательно при отказе АП *hwp** будет восстановлена работоспособность МАС. Отказ ИР фиксируется АП и приводит к отказу активных задач-реплик, размещенных в агентах данной АП, которым для их работы необходим данный ИР. Согласно процедуре *hwp_hwr_fail* АП определяет множество типов отказавших активных реплик и для каждого из них выполняет процедуру *hwp_t_fail*, которая при выполнении условия теоремы обеспечивает активизацию задачи соответствующего типа, следовательно, при отказе ИР будет восстановлена работоспособность МАС. Логический отказ задачи фиксируется агентом *a*, в котором задача размещена. Так как согласно условию теоремы логический отказ задачи вызван воздействиями, принадлежащими внутреннему контексту, то согласно процедуре *a_t_lfail* для каждого из воздействий, определяющих невыполнения предусловий задачи, будет вызвана задача-эффектор. Поскольку было доказано, что при отказах задач, агентов и АП, в МАС будут активированы реплики соответствующих типов, то все эффекторы будут выполнены, что согласно определению эффектора обеспечивает выполнение предусловий задачи, находящейся в состоянии логического отказа, следовательно, работоспособность МАС будет восстановлена. Теорема доказана.

Сравнение методик обеспечения отказоустойчивости МАС

Сравнение существующих методик обеспечения отказоустойчивости МАС DARX [3], Meta-Agent [4], Brokered MAS [5] и техники, предложенной в [6], с разработанными методикой восстановления работоспособности МАС и моделью резервированной МАС выявило следующие преимущества:

- в отличие от существующих методик разработанная модель резервированной МАС учитывает распределение задач по агентам и агентов по аппаратным компонентам, а также наличие доступа к исполнительным ресурсам для аппаратных компонентов;
- модель резервированной МАС основана только на избыточности задач и допускает избыточность ресурсов и введение дополнительных агентов и аппаратных компонентов, в то время как DARX, Meta-Agent и Brokered MAS основаны только на избыточности агентов, а техника, предложенная в [6], хоть и предлагает использовать избыточность отдельных задач, но вводит также и избыточность агентов;
- разработанная методика восстановления работоспособности МАС позволяет преодолевать логические отказы задач, в то время как только техника [6] допускает такую возможность;
- в отличие от существующих методик разработанная методика восстановления работоспособности обеспечивает обнаружение функциональных отказов задач и отказов исполнительных ресурсов, а также обеспечивает поддержание неактивных резервных задач в актуальном состоянии в случае изменения условий их выполнения;
- протокол взаимодействия между компонентами МАС обеспечивает возможность вызова задачи независимо от знания о ее расположении в компонентах МАС, что уменьшает сложность реализации агента.

В статье представлены разработанные нами формальная модель МАС как программно-аппаратного комплекса и модель резервированной МАС, основанная на введении избыточности отдельных задач, а не агентов. На основе разработанных моделей предложены протокол взаимодействия между компонентами МАС, техники обнаружения отказов и восстановления работоспособности и техника поддержания резервных задач МАС в актуальном состоя-

нии, образующие методику восстановления работоспособности резервированной МАС. Научная новизна и практическая ценность результатов исследования заключаются в следующем:

модель МАС позволяет рассматривать МАС как распределенный программно-аппаратный комплекс, конфигурация которого задана распределением задач по агентам и агентов по агентным платформам, а также доступностью отдельных ресурсов для агентных платформ МАС;

модель резервированной МАС основана на использовании только избыточности задач вместо избыточности агентов и при этом допускает использование избыточности ресурсов, а также введение дополнительных агентов и агентных платформ;

методика восстановления работоспособности включает в себя техники обнаружения отказов аппаратных компонентов, исполнительных ресурсов и логических и функциональных отказов задач;

методика восстановления работоспособности обеспечивает работоспособность МАС после программных и функциональных отказов задач, программных отказов агентов, отказов аппаратных компонентов и исполнительных ресурсов, а также после логических отказов задач, если в МАС существуют эффекторы, способные удовлетворить предусловия задачи, находящейся в состоянии отказа;

методика восстановления работоспособности обеспечивает поддержание неактивных резервных задач в актуальном состоянии, что позволяет снизить вычислительную нагрузку на систему и обеспечить восстановление работоспособности при изменении условий выполнения отдельных задач, характерном для интеллектуальных агентов;

протокол взаимодействия между компонентами МАС обеспечивает выполнение задачи независимо от ее расположения в компонентах системы, что уменьшает сложность реализации агента, т. к. агент не должен обладать информацией о фактической конфигурации МАС для запроса исполнения определенной задачи;

сформулирована и доказана теорема о свойстве сохранения работоспособности резервированной МАС, подтвердившая достоверность разработанных моделей и методики восстановления работоспособности.

СПИСОК ЛИТЕРАТУРЫ

1. **Jennings N.R.** On agent-based software engineering // Artificial intelligence. 2000. Vol. 117. No. 2. Pp. 277–296.

2. **Pullum L.L.** Software fault tolerance techniques and implementation. Artech House, 2001. 360 p.

3. **Guessoum Z., Briot J.P., Faci N.** Towards fault-tolerant massively multiagent systems // Massively Multiagent Systems I. Springer Berlin Heidelberg, 2005. Pp. 55–69.

4. **Serugendo G.D.M., Romanovsky A.** Designing fault-tolerant mobile systems // Scientific Engineering for Distributed Java Applications, International Workshop. Springer Berlin Heidelberg, 2003. Pp. 185–201.

5. **Kumar S., Cohen P.R.** Towards a fault-tolerant multiagent system architecture // Proc. of the 4th Internat. Conf. on Autonomous Agents, ACM. 2000. Pp. 459–466.

6. **Mellouli S.** A reorganization strategy to build fault-tolerant multiagent systems // Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2007. Pp. 61–72.

7. **Wooldridge M., Jennings N.R.** Intelligent agents: Theory and practice // Knowledge engineering review. 1995. Vol. 10. No. 2. Pp. 115–152.

8. **Faci N., Guessoum Z., Marin O.** DimaX: a fault-tolerant multiagent platform // Proc. of the 2006 Internat. Workshop on Software Engineering for Large-Scale Multiagent Systems, ACM. 2006. Pp. 13–20.

9. **Chen X., Feng J., Hiller M., Lauer V.** Application of software watchdog as a dependability software service for automotive safety relevant systems // Dependable Systems and Networks, 2007. 37th Annual IEEE/IFIP Internat. Conf. on, IEEE. 2007. Pp. 618–624.

10. **Олифер В.Г., Олифер Н.А.** Компьютерные сети. Принципы, технологии, протоколы: Учебн. пособие. 4-е изд. СПб.: Питер, 2013. 944 с.

REFERENCES

1. **Jennings N.R.** On agent-based software engineering, *Artificial intelligence*, 2000, Vol. 117, No. 2, Pp. 277–296.

2. **Pullum L.L.** *Software fault tolerance techniques and implementation*, Artech House, 2001, 360 p.

3. **Guessoum Z., Briot J.P., Faci N.** Towards

fault-tolerant massively multiagent systems, *Massively Multiagent Systems I*, Springer Berlin Heidelberg, 2005, Pp. 55–69.

4. **Serugendo G.D.M., Romanovsky A.** Designing fault-tolerant mobile systems, *Scientific Engineering for Distributed Java Applications, International Workshop*, Springer Berlin Heidelberg, 2003, Pp. 185–201.

5. **Kumar S., Cohen P.R.** Towards a fault-tolerant multiagent system architecture, *Proceedings of the 4th International Conference on Autonomous Agents*, ACM, 2000, Pp. 459–466.

6. **Mellouli S.** A reorganization strategy to build fault-tolerant multiagent systems, *Advances in Artificial Intelligence*, Springer Berlin Heidelberg, 2007, Pp. 61–72.

7. **Wooldridge M., Jennings N.R.** Intelligent agents: Theory and practice, *Knowledge engineering review*, 1995, Vol. 10, No. 2, Pp. 115–152.

8. **Faci N., Guessoum Z., Marin O.** DimaX: a fault-tolerant multiagent platform, *Proceedings of the 2006 International Workshop on Software Engineering for Large-scale Multiagent systems*, ACM, 2006, Pp. 13–20.

9. **Chen X., Feng J., Hiller M., Lauer V.** Application of software watchdog as a dependability software service for automotive safety relevant systems, *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*, IEEE, 2007, Pp. 618–624.

10. **Olifer V.G., Olifer N.A.** *Kompyuternye seti. Printsipy, tehnologii, protokoly*, Uchebn. posobie, 4-e izd., St. Petersburg: Piter Publ., 2013, 944 p. (rus)

ИГУМНОВ Алексей Владимирович – *аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: Alexei.Igumnov@gmail.com

IGUMNOV, Alexei V. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: Alexei.Igumnov@gmail.com

САРАДЖИШВИЛИ Сергей Эрикович – *доцент кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: SSaradg@yandex.ru

SARADGISHVILI, Sergey E. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: SSaradg@yandex.ru