ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

UDC 004.414.2.052:004.77 DOI: https://doi.org/10.30837/0135-1710.2026.189.072

Vitalii Tkachov

A METHOD FOR LOCAL RECOVERY OF INFORMATION SYSTEM PROCESSES ON A MOBILE PLATFORM

The subject of this study is local disruptions in the internal state of individual processes within an information system on a mobile platform, which prevent the correct continuation of execution without a special recovery intervention. **The aim** of this work is to develop a method for the local recovery of information system processes on a mobile platform, which, in the event of a local disruption of a process's internal state, ensures the continuation of its execution through idempotent re-execution and the deterministic reconstruction of the minimally sufficient local state under conditions of limited resources and intermittent connectivity. The article aims to accomplish the following **tasks**: formalize the local state of the process and the signs of its disruption; establish classes of process steps based on the admissibility of re-execution; develop a rule for choosing between micro-recovery and micro-restart; propose a method for deterministic reproduction of the minimally sufficient local state of the process; conduct a model study of the proposed method's effectiveness compared to a full process restart and deferred service recovery. **Results Achieved.** A formalization of the local process state has been developed in the form of a vector containing the current step number, internal process data, step input data, service execution flags, and the frequency of result fixation outside the process. A local state violation is defined via a vector of inconsistency and a scalar measure of local process state damage. A local recovery rule is proposed that formalizes the choice between micro-recovery and micro-restart, taking into account the expected process state after the recovery action, the trade-off between action requirements and available time and resources, estimates of external result replication, and the compliance of the current state with restart conditions. A method has been developed for the deterministic reconstruction of the minimally sufficient local process state, which determines the selection of the restart reference point and the formation of the set of local state coordinates, input data, and service flags necessary for the correct continuation of execution. The results of the model study showed that the proposed method ensures a faster reduction in the local process state damage index, a lower cumulative recovery deviation index, a higher probability of successful return of the process to correct execution, a shorter average recovery time, a smaller amount of data required for restart, and a lower probability of duplicating the external result. **Conclusions:** The proposed method is appropriate as a separate process mechanism in a system for ensuring the resilience of an information system on a mobile platform, as it allows for reducing the duration of local process execution disruptions and increasing system resilience without resorting to a full restart or service recovery when local recovery is sufficient.

Keywords: information system on mobile platform; process recovery; idempotent re-execution; deterministic state reconstruction; survivability.

1. Introduction

In the classification of information systems, there is a class of systems designed to operate in distributed computing environments. One of the subclasses of this class consists of information systems on mobile platforms (ISMP). For ISMP, the key factors are the dynamic availability of mobile platform resources, the mobility of its nodes, the intermittent connectivity of platform components or nodes, the irregularity of telemetric monitoring of their status, and strict time constraints on decision-making regarding the control of nodes and components. Such operating conditions are often associated with the edge computing environment [1]. In such an environment, these properties are further complicated by the service-segmented architecture of the information system, the need to maintain the continuity of components whose state changes during ISMP operation, and the variability of available resources during short intervals of the main function's execution. Under such conditions, the ISMP's ability to maintain or restore the execution of critical functions is most often associated with the provisioning, migration, or reconfiguration of service instances, although disruptions often occur on a smaller scale – during a single process that loses its correct local state before the ISMP proceeds to a full service or resource reconfiguration [2]–[4]. The availability of the core function in such a case is directly linked to the survivability of the ISMP as a coherent logical unit for executing the core function. The survivability of an information system is viewed as the property of maintaining the execution of target functions or restoring it under conditions of destructive influences, structural

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

disruptions, and resource constraints [5]–[7]. For ISMP, this interpretation is decisive, as it provides grounds for considering the local recovery of an individual process as one of the mechanisms for ensuring the survivability of the information system without resorting to a full service reconfiguration.

Current research in the area of system survivability and operational resilience in edge computing environments is primarily focused on service migration, reconfiguration of their deployment, redundancy, orchestration, restoration of processing sequences, and maintaining the continuity of service operation, the state of which changes during execution, under dynamic platform resource constraints [1, 3, 4]. Such approaches form the basis for adapting an information system to component connectivity failures or platform resource shortages. At the same time, the primary unit of recovery in these approaches is typically a service, a service chain, or a service segment. In cases where the disruption is local in nature and related to damage to the internal state of a single process, a full service reconfiguration proves excessive in terms of time, computational resources, and service data exchange [8]. This necessitates the identification of a separate task for rapid local process recovery without resorting to broader system-wide reconfiguration procedures.

The problem is that simply restarting a process after a failure does not guarantee correct recovery. For processes in the ISMP distributed environment, whose state changes during execution, restarting without accounting for the semantics of actions can cause duplication of external effects, repeated saving of intermediate results, disruption of local state consistency, and accumulation of inconsistent process artifacts [9]. This is why, in modern research in the area of fault tolerance for serverless systems with stateful processes – particularly their process control subsystems – idempotence is considered a fundamental prerequisite for permissible retries [10], while the uniqueness of external effect generation is ensured through special logging schemes, duplicate elimination, and action repeatability verification at the system level [11]. For mobile platforms, the execution of a restart function is considered permissible only for idempotent functional steps, whereas non-idempotent steps require a separate mechanism for coordinated recovery [12].

Another line of research links recovery to the replay of the execution flow, state saving, and recovery from a checkpoint, as well as to the deterministic replay of the execution of a set of processes as a service [13, 14]. Such approaches have demonstrated their suitability for reproducible state reconstruction, fault analysis, and returning the system to coordinated continued operation. However, these approaches are focused on debugging and diagnosing distributed systems, on restoring execution variants, or on cluster and serverless environments. For ISMP, where available resources, time, and connectivity are variable and limited, the task of reconstructing the minimally sufficient local state of a process required for rapid resumption of execution within an acceptable recovery interval has not been formally defined separately.

For ISMP, there are approaches focused on restoring critical services within connectivity windows of variable duration and bandwidth, as well as on analyzing service survivability under resource-time variables. Their application has made it possible to formalize transitions between service profiles, assess the shortfall in meeting threshold requirements for QoS metrics, and perform analytical ranking of services for operational parametric resource management [15, 16]. However, these approaches do not separately address the task of restoring the locally damaged state of an ISMP component or node process, do not establish classes of process steps based on the admissibility of re-execution, and do not define rules for the deterministic reproduction of the minimally sufficient local state to continue execution without transitioning to full service recovery.

Thus, for ISMP, the local recovery of processes after a violation of their internal state under conditions of limited resources and intermittent connectivity remains a distinct scientific problem. Its solution requires the formalization of two interrelated principles. The first is that the re-execution of a process step must be either semantically valid or such that controlled compensation for the re-generated effect is ensured. The second is that the continuation of process execution after a failure must rely on the deterministic reconstruction of a minimally sufficient local state. Within the scope of this task, local process recovery should be implemented in two modes. The first mode consists of micro-recovery, i.e., the local correction of a damaged fragment of the internal state without returning to the re-execution of the preceding fragment of the process. The second mode consists of a micro-restart, in which the process returns to a previously valid state by re-executing only those steps necessary for correct continuation of operation. It is precisely the choice between these two modes

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

and the rules for their implementation that define the essence of the proposed method for local recovery of ISMP processes with idempotent re-execution and deterministic state reproduction.

2. Analysis of Recent Research and Publications

For multi-domain edge-to-cloud environments, special attention has been paid to the problem of idempotency in the service network. In [17], the situation is considered where a repeated HTTP POST request after a failure produces a different external effect than the initial execution. To eliminate this effect, a two-component solution is proposed that combines an idempotency control mechanism with a separate means of resolving functional failures. Within the scope of the subject area, the research aims to confirm the fundamental necessity of controlling the re-execution of actions associated with an external effect. At the same time, the proposed approach operates at the level of network interaction between microservices and the resending of external requests, rather than at the level of restoring the internal local state of an individual process. The approach does not formalize the composition of the minimally sufficient reproducible state of the process and does not implement a function for choosing between local state correction and local restart of the information system process.

The problem of duplicating already executed processes in the context of ensuring service operation after a failure (disruption) is considered in [18] for serverless computing environments. The authors proceed from the assumption that the usual re-execution of a process after a disruption can distort the operation of the entire service, therefore they ensure exactly-once semantics through asymmetric logging, where either read operations or write operations are logged for different load types, and introduce a selection criterion between the two protocols and a mechanism for switching between them without stopping process execution. Experiments have demonstrated a reduction in latency and a decrease in logging overhead compared to previous solutions.

In [19], the features of using a state-saving and restoration mechanism in user space for migrating a containerized application between nodes by saving its current state and subsequently restoring it in a different runtime environment were investigated. It is shown that the time required to create a checkpoint is primarily determined by the amount of memory used, whereas the restoration time has a different dependency related to the procedure for reconstructing the saved state. The results obtained confirm the technical feasibility of this approach for resource-constrained infrastructures. This work focuses on the scenario of transferring process execution between nodes and includes local process recovery after a disruption of its internal state. The work does not formalize the division of process steps based on re-execution feasibility and does not define rules for constructing the minimally sufficient local state required to resume execution after a failure.

Maintaining delay-sensitive services in edge computing is the subject of [20], which examines the robust migration of edge microservices whose state changes during execution. The main focus is on two interrelated tasks: the migration of a containerized microservice and the preservation of its connection with a mobile user. To this end, a network solution based on OvS using Podman and CRIU is proposed for experimental evaluation of migration duration and service downtime. Additionally, an analytical model of the upper bounds of these metrics has been constructed. The significance of this work lies in the transition from a general concept of migrating a service whose state changes during execution to a concrete instrumental and analytical description. The subject of the study is the migration of a microservice between nodes. Consequently, local damage to the process state and the admissibility of re-executing a single step sufficient to continue process execution without full migration between information system components remain outside the scope of consideration.

For the tasks of migrating video analytics applications at edge nodes, an approach that accounts for the heterogeneity of their state is proposed in [21]. It is shown that existing solutions focused on cloud scenarios do not ensure proper migration of such applications in an edge environment, since different components of their state have distinct natures and require different processing methods. In view of this, the authors applied separate processing of three types of state using pre-processing, synchronization, and replay, and introduced a state store and an auxiliary control component, which allows for porting with minimal changes to the application code. Thus, the study confirms the feasibility of structured, rather than complete, state reproduction. At the same time, process replay is a component of the service migration procedure between nodes that does not account for local process

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

recovery following a disruption of its internal state. The aforementioned work does not address the choice between local state correction and micro-restart, nor does it formalize the verification of the validity of re-executing individual process steps.

Of particular interest is the work [22], which proposes a method for deterministic parallel execution of processes for modern data center applications (DORADD), in which the result depends on the current state of the processed data and the sequence of operations. The approach is based on ensuring the same result for the same input data by ordering data access and constructing a dynamic acyclic graph of dependencies between operations. It is shown that this approach allows combining high throughput with low latency and can be used for deterministic replay of a logging scheme during rapid recovery after failures. The authors confirm the feasibility of deterministic process replay in information systems where result reproducibility is of fundamental importance. At the same time, DORADD is focused on high-performance execution of ordered operations in data center systems and does not address the problem of local process recovery in subsystems. It does not formalize the locally damaged state of a process, does not establish classes of process steps based on the admissibility of re-execution, and does not define rules for constructing a minimally sufficient local state for rapid process recovery in the event of failures.

An analysis of works [17]–[22] shows that current research covers individual aspects related to the problem of this study, specifically, control of query replay at the network level, ensuring the uniqueness of result recording through external state logging, state preservation and recovery during process execution transfer, structured processing of various components of the process state, as well as deterministic process replay in distributed information systems. At the same time, these results do not solve, within a single formulation, the problem of local recovery of ISMP processes under resource constraints and intermittent connectivity of its components and nodes. In particular, the works under consideration lack a formalization of damage to the internal state of the process, do not establish classes of process steps based on the admissibility of re-execution, do not define the conditions for choosing between local state correction and micro-restart, and do not construct a method for deterministic reproduction of the minimally sufficient local state to continue process execution after a failure. It is precisely this unresolved part of the problem that determines the feasibility of developing a method for local recovery of ISMP processes with idempotent retry and deterministic state reconstruction.

3. Research Objectives and Tasks

The objective of the research is to develop a method for the local recovery of ISMP processes that, in the event of a local violation of the process's internal state, ensures the continuation of its execution through idempotent re-execution and the deterministic reconstruction of a minimally sufficient local state under conditions of limited resources and intermittent connectivity of components or nodes.

To achieve this goal, the following tasks must be solved:

– formalize the local state of the ISMP process, identify signs of its violation, and establish classes of process steps based on the admissibility of re-execution, which will allow defining the process-level formulation of the local recovery problem;

– develop a local process recovery rule that determines the choice between micro-recovery of a local state fragment and micro-restart of the process with re-execution of a subset of steps, ensuring a well-founded selection of recovery actions within a single method;

– develop a procedure for deterministically reproducing the minimally sufficient local state of the process after a local failure, taking into account the idempotent semantics of retry and the resource- and time-based constraints of ISMP operation, which will ensure the continuation of process execution without transitioning to full service recovery;

– develop a set of metrics and conduct an experimental study of the proposed method's effectiveness on representative scenarios of local process state violations, which will enable the evaluation of its performance based on temporal, resource, and functional characteristics.

4. Materials and Methods

The subject of the study is local disruptions in the internal state of individual processes within the ISMP, which do not necessarily trigger the ISMP's transition to service reconfiguration but prevent the correct continuation of the target function's execution without special recovery intervention. Such disruptions include loss of consistency in intermediate data, disruption of the

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

internal sequence of execution steps, partial fixation of service data or other input information, and a mismatch between the current local state of the process and the state required for its further execution.

The generalized research framework encompasses event and data sources, computational nodes, the process execution environment, local logs of service data (markers), results of process execution steps, and local recovery mechanisms. Within this framework, each process receives input data, forms a sequence of execution steps, modifies its own local state, and, if necessary, generates a result for recording outside the process. Local recovery mechanisms use current service records, available local data, and process service markers, an assessment of the platform's available resources, and parameters of the current connectivity interval to determine whether a damaged state fragment can be corrected or a local process retry can be initiated. Thus, the proposed method is intended for ISMPs in which process recovery is an internal operation within the operational interval of functioning and does not amount to a complete migration of the process (or the service within which it may run), a transition between service profiles, or a general reconfiguration of the ISMP.

The research materials consist of time series of telemetric indicators of mobile platform resource usage, process execution logs, service records of completed and incomplete steps, and a set of situations involving local violations of the internal state of processes that ensure the functioning of the information system. Within these situations, the following are taken into account: partial execution of a step, failure after recording an intermediate result, loss of part of local data and service markers, re-arrival of an input event, and interruption of available communication between system components. For each situation, the following are recorded: available unoccupied resources of the mobile platform, permissible local recovery time, the volume of service data required to reproduce the state, and the presence or absence of results from the previous execution of the process recorded outside its boundaries.

Several interrelated approaches were used to solve the problem:

– the process behavior is described as a sequence of steps with local data and service completion flags;

– to analyze the admissibility of re-execution, the principles of idempotent semantics were used, according to which repeating the same step should not change the final result compared to a single execution. If an external result is present, its controlled consideration or compensation must be ensured;

– for recovery after a local failure, a deterministic replay approach is used, whereby the continuation of process execution is based on the reconstruction of a minimally sufficient local state, ensuring the uniqueness of the subsequent process flow under identical input data, service markers, and the order of replay;

– The effectiveness of the proposed method was evaluated based on a simulation study comparing local recovery time, the proportion of successfully recovered processes, the scope of the reconstructed state, the number of permissible re-executions, and the frequency of re-executing actions whose results had already been recorded outside the process.

5. Research Results and Discussion

5.1. Formalization of the local process state and the semantics of re-execution

In previously proposed approaches to analyzing the process level of ISMP, the process state space was defined through invariants of the queue, time buffer, data validity, and causal-temporal order, which is sufficient for detecting violations at the level of the entire process environment [15]. In service-oriented approaches, the service profile served as the primary unit of evaluation, and recovery was described as a transition between profiles with feasibility checks within connectivity windows [16]. For the problem considered in this study, this is insufficient, since the object of recovery is a single process with a locally damaged internal state, for which it is necessary to determine the possibility of safe continuation of execution.

Let a component or node of the ISMP implement a set of processes:

$$P = \{ p_i \mid i = 1, \ldots, n \}. \quad (1)$$

For each process p_i, an ordered set of its steps is specified:

$$A_i = \{ a_{i,r} \mid r = 1, \ldots, m_i \}, \quad (2)$$

where: $a_{i,r}$ – the r-th step of the process p_i; m_i – the total number of steps in the local execution scenario of the process p_i.

The local state of the process p_i at discrete time t is given by the vector:

$$\vec{s}_i(t) = \left(r_i(t), \vec{d}_i(t), \vec{u}_i(t), \vec{v}_i(t), \mu_i(t) \right)^{\mathrm{T}}, \quad (3)$$

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

where $r_i(t)$ – the step number of the process at time t; $\vec{d}_i(t)$ – a vector of internal process data generated in previous steps and required to execute the current step without recalculating the entire preceding portion of the process; $\vec{u}_i(t)$ – a vector of input data for the current step, i.e., the values, events, and parameters with which this step is to be executed; $\vec{v}_i(t)$ – a vector of status flags indicating which internal operations of the current step have already been completed, which intermediate results have already been recorded, and which service actions have already been performed; $\mu_i(t)$ – the number of times the result of the current step has already been recorded outside the process (entry to an external log, message transmission, update of a shared repository, sending of a control command, etc.).

The vector $\vec{s}_i(t)$ (3) specifies the minimum sufficient description of the process for the local recovery task. The component $r_i(t)$ determines the process's position in the execution sequence; $\vec{d}_i(t)$ – the presence of internal data without which it is impossible to proceed to the next action; $\vec{u}_i(t)$ – the completeness of input data for the current step; $\vec{v}_i(t)$ – the degree of completion of internal operations and service updates; $\mu_i(t)$ – whether the result of the current step has been updated once or repeatedly outside the process. It is the combination of these components that makes it possible to determine whether the process can continue directly, whether a local correction of the state is needed, or whether some steps need to be re-executed.

To formally identify a local disturbance, the misalignment vector is introduced:

$$\vec{\delta}_i(t) = \left(\delta_{i,1}(t), \delta_{i,2}(t), \delta_{i,3}(t), \delta_{i,4}(t)\right)^{\mathsf{T}}, \qquad (4)$$

where its components are defined by the following relations:

$$\delta_{i,1}(t) = \left\| \vec{u}_i(t) - \hat{\vec{u}}_{i,r_i(t)} \right\|,$$

$$\delta_{i,2}(t) = \left\| \vec{d}_i(t) - \hat{\vec{d}}_{i,r_i(t)} \right\|,$$

$$\delta_{i,3}(t) = \left| r_i(t) - \nu_i\left(\vec{v}_i(t)\right) \right|,$$

$$\delta_{i,4}(t) = \max\left\{0, \mu_i(t) - \sigma_{i,r_i(t)}\right\}$$

where $\hat{\vec{u}}_{i,r_i}$ – reference vector of input data for step $a_{i,r}$; $\hat{\vec{d}}_{i,r_i}$ – reference vector of internal data for which the execution of step $a_{i,r}$ is valid; $\nu_i\left(\vec{v}_i(t)\right)$ – operator for determining the number of the last step, the completion of which is confirmed by service flags; $\sigma_{i,r_i}(t) \in \{0,1\}$ – indicator of the validity of recording the result of step $a_{i,r}$ outside the process.

The components of $\vec{\delta}_i(t)$ (4) have the following meanings:

– $\delta_{i,1}(t)$ describes the deviation between the actual input data of the current step and the input data under which the execution of this step is correct. In other words, this component indicates whether the current set of input values matches the set with which the step $a_{i,r}$ is intended to be executed. For example, if the current step is supposed to process a telemetry packet with a specific source ID, timestamp, and set of measured parameters, then $\delta_{i,1}(t)$ increases if some of this data is lost, corrupted, or if an outdated data packet is received again;

– The $\delta_{i,2}(t)$ characterizes the discrepancy between the actual internal data of the process and the internal state required to execute the current step. This component indicates whether the process has retained the intermediate data, results of previous calculations, and control values without which the current step cannot proceed correctly. For example, if a processing buffer, an intermediate feature vector, or an indicator of successful completion of the previous stage should already have been generated before executing the step, then corruption or loss of even a portion of this data leads to an increase in $\delta_{i,2}(t)$;

– $\delta_{i,3}(t)$ describes the discrepancy between the "step number at which the process is currently located and the number of the last step whose completion has been confirmed by system logs. This component indicates whether the process's current position in the execution sequence aligns with the entries actually recorded in the system log. For example, if the process has already moved to step $r_i(t) = 5$, but the status marks confirm the completion of only the third step, then $\delta_{i,3}(t)$ indicates a discrepancy in the execution order and shows that the process has progressed further than is confirmed by its status;

– $\delta_{i,4}(t)$ indicates that the permissible number of times the current step's result has been recorded outside the process has been exceeded. This component is zero if the current step's result has not yet been recorded or has been recorded within the permissible

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

number of times, and takes on a positive value if the re-execution has already led or may lead to excessive recording of that result. For example, if the current step has already generated a record in an external log, sent a message to another component, or saved a command to an external storage, then re-executing this same step without accounting for the value of $\mu_i(t)$ may lead to duplication of the result, which is reflected by the $\delta_{i,4}(t)$ component.

These four components represent the main types of local inconsistency that are critical to the process recovery problem: input data mismatch, corruption of internal process data, violation of the confirmed execution order, and excessive recording of the current step's result outside the process.

The set of locally valid process states p_i is defined as follows:

$$\Omega_i = \left\{ \vec{s}_i(t) \middle| \vec{\delta}_i(t) = \vec{0} \right\}. \quad (5)$$

Thus, a process state belongs to the set Ω_i (5) if and only if the input data of the current step, the internal process data, the order of step execution, and the fact of recording the result outside the process are consistent.

To quantitatively assess the extent of local damage to the condition, a scalar index is introduced:

$$\theta_i(t) = \vec{\omega}_i^{\mathrm{T}} \vec{\delta}_i(t), \quad \vec{\omega}_i = \left(\omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \omega_{i,4}\right), \quad \sum_{j=1}^{4} \omega_{i,j} = 1, \quad \omega_{i,j} \geq 0. \quad (6)$$

The metric $\theta_i(t)$ (6) is a scalar measure of the current degree of damage to the local state of the process. Its value is zero for a locally valid state and increases as the inconsistency grows across one or more components of the vector $\vec{\delta}_i(t)$. The coefficients $\omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \omega_{i,4}$ determine the contribution of, respectively, input data mismatch, damage to the process's internal data, violation of the confirmed step execution order, and re-fixation of the result outside the process to the final value $\theta_i(t)$. Subsequently, $\theta_i(t)$ (6) is used for a quantitative comparison of possible recovery actions.

The change in the local state of the process at step $a_{i,r}$ is represented as a transition operator:

$$\vec{s}_i(t+1) = \hat{f}_{i,r_i(t)}\left(\vec{s}_i(t), \vec{x}_i(t), \xi_i(t)\right), \quad (7)$$

where $\vec{x}_i(t)$ is the vector of actual input data for the current step, i.e., the set of values, events, and parameters with which this step is actually executed at time t; $\xi_i(t)$ is the execution result code for the current step, distinguishing between successful completion, partial completion, and abnormal termination.

Equation (7) specifies the local state to which the process transitions after executing the current step based on the actual input data and the actual execution result.

To isolate the part of the process state that corresponds to the result already recorded outside the process, the following statement is introduced:

$$\vec{y}_i(t) = \pi_i\left(\vec{s}_i(t)\right). \quad (8)$$

The vector $\vec{y}_i(t)$ (8) contains only those execution results that have an external fixation and are therefore relevant for the analysis of re-execution. Such results may include input data, a record in an external log, a transmitted message, an update to a shared store, or a command sent to another ISMP component.

For the step $a_{i,r}$, single and double executions from the same specially fixed initial state $\bar{\vec{s}}_i(t)$ are considered:

$$\bar{\vec{s}}_{i,r}^{(1)} = \hat{f}_{i,r_i}\left(\bar{\vec{s}}_i, \bar{\vec{x}}_i, 1\right), \quad \bar{\vec{s}}_{i,r}^{(2)} = \hat{f}_{i,r_i}\left(\bar{\vec{s}}_{i,r}^{(1)}, \bar{\vec{x}}_i, 1\right) \quad (9)$$

In formula (9), the state $\bar{\vec{s}}_{i,r}^{(1)}$ corresponds to a single execution of the step $a_{i,r}$, whereas the state $\bar{\vec{s}}_{i,r}^{(2)}$ corresponds to a repeated execution of the same step with the same input data $\bar{\vec{x}}_i$. Comparing these two states is necessary to determine whether the repeated execution changes the result already recorded outside the process.

Process steps with idempotent semantics of re-execution are defined by the set:

$$\Pi_i^{(1)} = \left\{ a_{i,r} \in A_i \middle| \pi_i\left(\bar{\vec{s}}_{i,r}^{(2)}\right) = \pi_i\left(\bar{\vec{s}}_{i,r}^{(1)}\right) \right\}. \quad (10)$$

Thus, a step $a_{i,r}$ belongs to the set $\Pi_i^{(1)}$ (10) if its repeated execution does not change the result recorded outside the process compared to a single execution. For example, this class may include steps involving the recalculation of an intermediate value, the re-evaluation of a logical condition, or the re-generation of internal data, provided that such re-execution does not produce a new record, message, or other additional result outside the process.

Steps that can be repeated to produce a one-time result through a specific compensatory action are defined as follows:

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

$$\Pi_i^{(2)} = \left\{ a_{i,r} \in A_i \,\middle|\, \exists\, \bar{g}_{i,r} : \pi_i\left(\bar{g}_{i,r}\left(\vec{\bar{s}}_{i,r}^{(2)}\right)\right) = \pi_i\left(\vec{\bar{s}}_{i,r}^{(1)}\right),\ a_{i,r} \notin \Pi_i^{(1)} \right\} . \quad (11)$$

The set $\Pi_i^{(2)}$ (11) includes steps for which re-execution in itself changes the result recorded outside the process; however, there is an operator $\bar{g}_{i,r}$ that eliminates redundant recording and reduces the result of two executions to that of a single execution. For example, if a repeat execution creates a duplicate entry in an external log, a duplicate service message, or a second instance of an already recorded result, and this duplicate can be detected and eliminated by a special procedure, then the corresponding step belongs to the set $\Pi_i^{(2)}$.

Finally, steps that cannot be repeated without a separate change in the process state are defined as follows:

$$\Pi_i^{(3)} = A_i \setminus \left(\Pi_i^{(1)} \Pi_i^{(2)}\right) . \quad (12)$$

The set $\Pi_i^{(3)}$ (12) encompasses steps for which re-execution results in a change to the result recorded outside the process, and for which there is no $\bar{g}_{i,r}$ operator capable of mapping the result of two executions to the result of a single execution. In other words, this class includes steps whose re-execution is not permitted without first changing the state of the process. For example, this could be the formation of a unique external command, an irreversible update to a shared repository, or the re-transmission of an action that cannot be duplicated.

Thus, the relations A_i (2), $\vec{s}_i(t)$ (3), $\vec{\delta}_i(t)$ (4), Ω_i (5), $\theta_i(t)$ (6), and the classes $\Pi_i^{(1)}$, $\Pi_i^{(2)}$, $\Pi_i^{(3)}$ (10)–(12) form a formal description of the local state of the process, the conditions for its validity, and the semantics of step repetition. This makes it possible to distinguish a locally valid state of the process from states in which the consistency of input data, internal process data, the confirmed execution order, or the recording of results outside the process is violated. Furthermore, the introduced classification of process steps based on the admissibility of re-execution establishes a formal basis for distinguishing between steps for which direct re-execution is admissible, admissible with a compensation operator, or inadmissible without a prior change in the process state.

5.2. The rule of local process recovery

Modern approaches to the local recovery of processes whose state changes during execution are based on the premise that the decision to restart cannot be made solely upon the occurrence of a failure. Works on local component restart and controlled recovery of microservice systems [8, 10] emphasize the need to verify the boundaries of recovery and the ability to compensate for results already recorded outside the process. In studies focused on permissible request retries and the singleness of result recording [17, 10], it is shown that a retry is permissible only under controlled step semantics. In works devoted to deterministic replay of execution [14, 22], it is specifically emphasized that fast recovery must rely on reproducible input data, service markers, and execution order. That is why, in this study, the rule for local process recovery is formulated as a formal decision-making problem between micro-recovery and micro-restart, while simultaneously accounting for local state corruption, resource-time constraints, the semantics of re-execution, and the reproducibility of the data and service markers required to resume process execution.

For the process p_i at the time step t, the following set of local actions is considered:

$$U_i(t) = \{\rho_i, \kappa_i\} \quad (13)$$

where ρ_i is the micro-restoration of a local state fragment without restarting the step; κ_i is the micro-restart of the process with a return to a previously valid state and the re-execution of a portion of the steps.

The expected process inconsistency vector after applying the action $u \in U_i(t)$ is given as:

$$\vec{\delta}_i^{\,u}(t+1) = \bar{T}_i(u,t)\,\vec{\delta}_i(t) + \vec{\varepsilon}_i(u,t) , \quad (14)$$

where $\bar{T}_i(u,t)$ is the transformation matrix of the current misalignment vector under the action of u; $\vec{\varepsilon}_i(u,t)$ is an additional vector that accounts for the portion of misalignment that remains after the action is performed or arises due to the incomplete accuracy of the process state reconstruction.

Equation (14) can be interpreted as follows. First, the current inconsistency vector $\vec{\delta}_i(t)$ is transformed by the matrix $\bar{T}_i(u,t)$, which shows exactly how the selected action changes each component of the inconsistency. After that, the vector $\vec{\varepsilon}_i(u,t)$ is added, which represents the residual inconsistency not

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

eliminated by this action. Thus, $\vec{\delta}_i^{\,u}(t+1)$ is an estimate of what each component of the inconsistency will become after applying action u.

For the micro-restoration ρ_i, the matrix $\bar{T}_i(\rho_i,t)$ takes the form:

$$\bar{T}_i(\rho_i,t)=\operatorname{diag}\left(\alpha_{i,1}(t),\alpha_{i,2}(t),\alpha_{i,3}(t),\alpha_{i,4}(t)\right),\quad 0\le\alpha_{i,j}(t)<1. \tag{15}$$

The coefficient $\alpha_{i,j}$ indicates what portion of the j-th component of misalignment remains after micro-restoration. If $\alpha_{i,j}=0$, then the corresponding component is completely eliminated. If $\alpha_{i,j}$ is close to one, then micro-recovery hardly reduces this component. Thus, the diagonal matrix (15) provides a separate estimate of the micro-recovery result for each component of the vector $\vec{\delta}_i(t)$.

For micro-restart κ_i, similarly:

$$\bar{T}_i(\kappa_i,t)=\operatorname{diag}\left(\beta_{i,1}(t),\beta_{i,2}(t),\beta_{i,3}(t),\beta_{i,4}(t)\right),\quad 0\le\beta_{i,j}(t)<1. \tag{16}$$

The coefficient $\beta_{i,j}(t)$ (16) indicates what portion of the jth inconsistency component remains after the process returns to a previously valid state and the relevant portion of the steps is re-executed. Unlike micro-recovery, a micro-restart does not change the process state locally at the current point, but rather by returning to a previous state in which the process execution was still valid.

For components related to the input data of the current step, the internal data of the process, and the step number, the following inequality typically holds:

$$\beta_{i,1}(t)\le\alpha_{i,1}(t),\quad \beta_{i,2}(t)\le\alpha_{i,2}(t),\quad \beta_{i,3}(t)\le\alpha_{i,3}(t),\quad \beta_{i,4}(t)\le\alpha_{i,4}(t). \tag{17}$$

Equations (17) imply that, for the first three components, a micro-restart typically leaves no greater inconsistency than a micro-recovery. This is because returning to a previously valid state eliminates the discrepancy between the actual and confirmed step numbers and allows the input data and internal process data for the current step to be regenerated. For the fourth component, which involves re-recording the result outside the process, this advantage does not hold, so its evaluation is performed separately.

The expected scalar estimate of the process state after applying action u is determined as follows:

$$\Theta_i(u,t)=\vec{\omega}_i^{\mathsf{T}}\tilde{\vec{\delta}}_i^{\,u}(t+1), \tag{18}$$

The quantity $\Theta_i(u,t)$ (18) is the weighted sum of the components of the vector $\tilde{\vec{\delta}}_i^{\,u}(t+1)$ and indicates how consistent the state of the process is expected to be after applying action u. The smaller the value of $\Theta_i(u,t)$, the smaller the expected damage to the local state of the process after executing this action. That is why the quantity $\Theta_i(u,t)$ is used to compare microrestore and microreset based on the result expected after their application.

To account for the limitations of the mobile platform, a vector of action requirements for time and resources is introduced:

$$\vec{c}_i(u,t)=\left(c_{i,1}(u,t),c_{i,2}(u,t),c_{i,3}(u,t),c_{i,4}(u,t)\right)^{\mathsf{T}}, \tag{19}$$

where $c_{i,1}(u,t)$ – the time required to perform a local recovery operation; $c_{i,2}(u,t)$ – the computational resource requirement (type 1, e.g., processor power); $c_{i,3}(u,t)$ – the computational resource requirement (type 2, e.g., amount of RAM); $c_{i,4}(u,t)$ – the amount of service data that must be read, transmitted, or written during state correction or reproduction.

Let the vector of resources available at time *t*, whose components are consistent with $\vec{c}_i(u,t)$ (19), be:

$$\vec{o}_i(t)=\left(o_{i,1}(t),o_{i,2}(t),o_{i,3}(t),o_{i,4}(t)\right)^{\mathsf{T}}. \tag{20}$$

Then the relative load indicator of the operation is determined as follows:

$$\psi_i(u,t)=\sum_{j=1}^{4}\varphi_j\frac{c_{i,j}(u,t)}{o_{i,j}(t)},\quad \sum_{j=1}^{4}\varphi_j=1,\quad \varphi_j\ge 0. \tag{21}$$

The value of $\psi_i(u,t)$ (21) allows us to express the ratio between the requirements of action u and the time and resources available at time t as a single number. For each component separately, we compare exactly how much is required to perform the action and how much is actually available, after which these ratios are summed with weights φ_j. The smaller the value

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

of $\psi_i(u,t)$, the easier it is to perform the corresponding action in the current state of the mobile platform. Therefore, $\psi_i(u,t)$ is used to compare micro-restarts and micro-reboots based on their resource requirements.

A separate check is performed to determine whether repeating the current step would result in a duplicate of a result already recorded outside the process. For micro-restoration ρ_i, the current step is not repeated, therefore:

$$\lambda_i(\rho_i,t)=0\ . \tag{22}$$

For micro-restart κ_i, the value of $\lambda_i(\rho_i,t)$ is determined by the membership of the current step $a_{i,r_i(t)}$ in the sets $\Pi_i^{(1)}$, $\Pi_i^{(2)}$, $\Pi_i^{(3)}$ (10)–(12):

$$\lambda_i(\kappa_i,t)=\begin{cases}0, & a_{i,r_i}(t)\in\Pi_i^{(2)},\\ 1-\varsigma_i(\tau), & a_{i,r_i}(t)\in\Pi_i^{(2)},\\ 1, & a_{i,r_i}(t)\in\Pi_i^{(2)},\end{cases} \tag{23}$$

where $\varsigma_i(t)\in[0,1]$ is a coefficient indicating what proportion of the duplicated result can be eliminated locally " " (if $\varsigma_i(t)=1$, the duplicated result can be eliminated entirely; if $\varsigma_i(t)=0$, local elimination is not possible).

Thus, if the current step belongs to the set $\Pi_i^{(1)}$, a micro-restart does not create a duplicate of an already recorded result. If the step belongs to $\Pi_i^{(2)}$, the consequences of re-execution depend on the possibility of locally eliminating the duplicate result. If the step belongs to $\Pi_i^{(3)}$, a micro-restart leads to an invalid repetition of an already recorded action.

To determine the extent to which the current step number, input data, and service flags match the set required for a deterministic restart, the following vector is entered

$$\vec{z}_i(t)=\left(r_i(t),\vec{u}_i(t),\vec{v}_i(t)\right)^{\mathsf{T}}, \tag{24}$$

and the reference vector

$$\hat{\vec{z}}_{i,r_i(t)}=\left(r_i(t),\hat{\vec{u}}_{i,r_i(t)},\hat{\vec{v}}_{i,r_i(t)}\right)^{\mathsf{T}}, \tag{25}$$

where $\hat{\vec{v}}_{i,r_i(t)}$ is the reference vector of service marks for step $a_{i,r_i(t)}$.

The vector $\vec{z}_i(t)$ (24) contains those components of the current process state that are relevant specifically to the restart of the current step: the step number, the input data for this step, and service flags that record the status of its execution. The vector $\hat{\vec{z}}_{i,r_i(t)}$ (25) specifies the reference set of these same values under which the restart of step $a_{i,r_i(t)}$ is permissible.

Then, the indicator of the current state's compliance with the conditions for restarting is determined as follows:

$$\zeta_i(t)=\exp\left(-\left\|\vec{z}_i(t)-\hat{\vec{z}}_{i,r_i(t)}\right\|_1\right)\ . \tag{26}$$

The value of $\zeta_i(t)$ (26) lies in the interval $(0,1]$, equals one when the current and reference values required for restarting are exactly the same, and decreases as the deviation between them increases. Therefore, the closer the current state of the process is to the reference startup state, the larger the value of $\zeta_i(t)$ is.

To use $\zeta_i(t)$ in the selection rule, the following indicator is introduced:

$$\iota_i(u,t)=\begin{cases}0, & u=\rho_i,\\ 1-\zeta_i(t), & u=\kappa_i.\end{cases} \tag{27}$$

The value $\iota_i(u,t)$ (27) is taken into account only for those actions that require the step to be restarted. For micro-reset ρ_i, a restart is not performed, so $\iota_i(u,t)=0$. For micro-restart κ_i, the value of $\iota_i(u,t)$ increases as the deviation between the current state and the reference state for the restart increases. This means that as the current state becomes less consistent with the restart conditions, the micro-restart becomes less suitable for use.

To verify whether a local action can be applied to the process in its current state, a barrier function is used, consistent with the approaches of admissible state sets and recovery estimation based on barrier functions [8, 22]. After the action is applied, the process must remain within a locally admissible state or return to it. In this regard, the following quantity is introduced:

$$\Gamma(u,t)=1-\frac{\Theta_i(u,t)}{\bar{\theta}_i}\ ,\quad \bar{\theta}_i>0\ , \tag{28}$$

where $\vec{\theta}_i$ is the maximum permissible value of the scalar process state estimate after the action is executed.

The value $\Gamma(u,t)$ (28) indicates the ratio between the expected state of the process after applying the action u and the maximum permissible state. If $\Gamma(u,t)\ge 0$, then after executing the action, the process

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

does not exceed the bounds of the locally permissible state. If $\Gamma(u,t)<0$, then the action does not ensure that the process returns to a state from which its execution can be continued. Thus, $\Gamma(u,t)$ characterizes whether the expected state of the process after the action satisfies the constraint on the permissible level of damage.

Then the admissibility criterion for action u for process p_i is defined as follows:

$$\chi_i(u,t)=1,\quad \psi_i(u,t)\le 1,\quad \Gamma(u,t)\ge 1,\quad \lambda_i(u,t)\le \bar{\lambda},\quad \iota_i(u,t)=\bar{\iota}\ . \tag{29}$$

If at least one of the conditions in (29) is not satisfied, we assume that $\chi_i(u,t)=0$. Therefore, action u is considered admissible for application only if it can be executed within the available time and resources, does not take the process beyond the bounds of a locally admissible state, does not exceed the maximum permissible level of repeated fixation of the result outside the process, and does not require a restart in the event of an excessive deviation of the current state from the reference startup state.

Among the admissible actions, the one for which the value of the objective function is the smallest is selected:

$$\wp_i(u,t)=\varpi_1\Theta_i(u,t)+\varpi_2\psi_i(u,t)+\varpi_3\lambda_i(u,t)+\varpi_4\nu_i(u,t)-\varpi_5\Gamma_i(u,t), \tag{30}$$

where $\varpi_1,\varpi_2,\varpi_3,\varpi_4,\varpi_5\ge 0$ – importance coefficients of the objective function components, which are specified during method configuration in accordance with the requirements of a specific ISMP, in particular: ϖ_1 corresponds to a scalar assessment of the process state after an action is performed; ϖ_2 – the action's requirements regarding time and resources; ϖ_3 – the re-recording of the result outside the process; ϖ_4 – the deviation of the current state from the reference state of a restart; ϖ_5 – maintaining the process within an acceptable state.

In the function $\wp_i(u,t)$ (30), the first four terms increase its value as the expected damage to the state after an action increases, along with time and resource requirements, the re-recording of the result outside the process, and the deviation of the current state from the reference state at restart. The last term with a minus sign reduces the function's value for actions after which the process remains within the permissible state with a greater distance from the maximum permissible damage level.

The optimal local action is determined by the rule

$$u_i^*(t)=\arg\min_{\substack{u\in U_i(t),\\ \chi_i(u,t)=1}}\wp_i(u,t) \tag{31}$$

Given the two-element set $U_i(t)$ (13), rule (31) can be expressed in terms of the difference in the values of the objective function:

$$\Delta_i(t)=\wp_i(\rho_i,t)-\wp_i(\kappa_i,t). \tag{32}$$

Then, if both actions are locally feasible, the selection rule takes the form:

$$u_i^*(t)=\begin{cases}\rho_i, & \Delta_i(t)\le 0,\\ \kappa_i, & \Delta_i(t)\le 0.\end{cases} \tag{33}$$

Thus, micro-restoration ρ_i is selected when the local correction of the state fragment based on the set of considered indicators yields a value of the objective function no greater than that of a micro-restart. A micro-restart κ_i is selected when returning to a previously feasible state with the re-execution of some steps yields a smaller value of the objective function.

If neither of the actions ρ_i nor κ_i satisfies condition (29), a set of actions admissible for the process is introduced:

$$U_i^{\dagger}(t)=\left\{u\in U_i(t)\,\middle|\,\chi_i(u,t)=1\right\}. \tag{34}$$

If

$$U_i^{\dagger}(t)=\varnothing, \tag{35}$$

then local recovery of the process at time t is considered impossible, and the process p_i is transferred to higher-level recovery mechanisms:

$$p_i\to\sum(t)\ , \tag{36}$$

where $\sum(t)$ is the set of processes requiring service-level recovery, profile switching, or other ISMP system reconfiguration [15].

Thus, relations (13)–(36) define a rule for selecting a local recovery action as the sole means of distinguishing between micro-recovery and micro-restart. Unlike selection based on a single damage criterion, the proposed rule simultaneously takes into account the scalar estimate of the process state after the action is performed $\Theta_i(u,t)$ (18), the ratio between

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

the action's requirements and the available time and resources $\psi_i(u,t)$ (21), the level of result re-fixation outside the process $\lambda_i(u,t)$ (22), (23) and the deviation of the current state from the reference state of the restart $\iota_i(u,t)$ (27). It is precisely this structure that makes it possible to formally determine which of the two local recovery actions is appropriate for the current state of the process.

5.3. Method for Deterministically Reconstructing the Minimally Sufficient State of a Process

Since rules (31)–(33) determine whether it is appropriate to apply micro-restoration ρ_i or micro-restart κ_i to the process p_i, then if for the moment t the following holds

$$u_i^*(t) = \kappa_i \ , \qquad (37)$$

then, to continue execution, it is necessary to roll back the process to one of the previously confirmed checkpoints and restore a local state that is sufficient for an unambiguous restart and, at the same time, minimal in terms of service data. This is precisely the essence of the method for deterministically restoring the minimally sufficient state of the process.

Let the log of service marks and the stored internal data for the process p_i at time t contain a set of restart reference points:

$$B_i(t) = \left\{ b_{i,\zeta} \,\middle|\, \zeta = 1,\dots M_i(t),\ b_{i,\zeta} < r_i(t) \right\} \ , \qquad (38)$$

where $b_{i,\zeta}$ is the step number after which a confirmed local state exists for the process, to which it can return for re-execution; $M_i(t)$ is the number of such points, determined by the service marks $\vec{v}_i(t)$ and the confirmed order of step execution.

Not every point $b_{i,\zeta} \in B_i(t)$ is valid for replay. For a point to be valid, replaying the segment starting from that point must be semantically valid and must not exceed the time and resource limits. Therefore, a set of valid reference points is introduced:

$$B_i^{\dagger}(t) = \left\{ b_{i,\zeta} \in B_i(t) \,\middle|\, \lambda_i(b_{i,\zeta},t) \le \bar{\lambda},\ \upsilon_i(b_{i,\zeta},t) \le \bar{\upsilon},\ \chi_i(\kappa_i,t) = 1 \right\}. \qquad (39)$$

In formula (39), $\lambda_i(b_{i,\zeta},t)$ is a scalar measure of the re-fixation of the result outside the process during the re-execution of the fragment from the reference point $b_{i,\zeta}$; $\upsilon_i(b_{i,\zeta},t)$ is a scalar measure of the ratio between the resource requirements for such re-execution and the values of these quantities available at time t; $\bar{\lambda}$ and $\bar{\upsilon}$ are the maximum permissible values of these indicators; $\chi_i(\kappa_i,t)$ – a micro-restart feasibility indicator, defined in formula (29).

From the set $B_i^{\dagger}(t)$ (39), it is necessary to select the point for which reconstruction requires the lowest total cost. To do this, the objective function (criterion) for selecting the reference point is defined as follows:

$$\Xi_i(b,t) = \tau_1\left(r_i(t) - b\right) + \tau_2\upsilon_i(b,t) + \tau_3\lambda_i(b,t) + \tau_3\vartheta_i(b,t) \ , \qquad (40)$$

where $\left(r_i(t) - b\right)$ – the number of steps that must be repeated after returning to point b; $\upsilon_i(b,t)$ – a scalar measure of the ratio between the time and resource requirements of this repetition and their available values; $\lambda_i(b,t)$ – a scalar estimate of the repeated saving of the result outside the process during the re-execution from point b; $\vartheta_i(b,t)$ – the volume of internal data and service markers that need to be recreated to start from point b; $\tau_1,\tau_2,\tau_3,\tau_4 \ge 0$ – coefficients that specify the contribution according to the number of re-execution steps, time and resource constraints, re-capturing of results outside the process, and the volume of data that needs to be restored (their values are set during the configuration of a specific ISMP).

In formula (40), the first term accounts for how far back the process needs to be rolled back in terms of the number of steps. The second term accounts for whether re-execution from the selected point is excessive in terms of resources and time for the current ISMP state. The third term accounts for whether re-execution from this point could lead to the result being recorded outside the process again. The fourth term accounts for the amount of data and service markers that need to be restored to start the process from the selected point. Thus, the $\Xi_i(b,t)$ function allows selecting not the closest point automatically, but the one that is most appropriate based on the combination of the number of steps to be re-executed, resource and time requirements, re-capturing the

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

result, and the amount of data required to reproduce the process.

The optimal support point is determined as follows:

$$b_i^*(t) = \arg\min_{b \in B_i^\vee(t)} \Xi_i(b,t) \quad (41)$$

Equation (41) allows one to move to the nearest previous point for which the total cost of restarting is the lowest, taking into account the number of re-execution steps, resource and time requirements, the need to re-capture the result outside the process, and the volume of data that needs to be reproduced.

After selecting the point $b_i^*(t)$ (41), a subset of the coordinates of the process's complete local state is formed, which are necessary for restarting from this point. Let the complete local state of the process at the point $b_i^*(t)$ be given by $\vec{s}_i\left(b_i^*(t)\right) \in \mathbb{R}^{d_i}$. Then, the binary coordinate selection matrix is defined as:

$$\bar{G}_i(t) \in \{0,1\}^{d_i^{\min}(t) \times d_i}, \quad d_i^{\min}(t) \le d_i \quad (42)$$

where $d_i^{\min}(t)$ is the number of coordinates of the complete local state stored for restart.

Each row of the matrix $\bar{G}_i(t)$ contains exactly one element and determines which specific coordinate of the complete local state is included in the vector of coordinates to be reproduced.

The coordinate vector of the local state to be reconstructed is defined as follows:

$$\vec{h}_i(t) = \bar{G}_i(t)\vec{s}_i\left(b_i^*(t)\right). \quad \vec{h}_i(t) = \bar{G}_i(t)\vec{s}_i\left(b_i^*(t)\right) \quad (43)$$

The quantity $\vec{h}_i(t)$ (43) contains only those coordinates of the complete local state without which restarting the process from the point $b_i^*(t)$ is impossible. Therefore, $\vec{h}_i(t)$ includes only those state data that are directly used when starting the process from the selected point.

To generate the complete set of data required for reproduction, the value of $\vec{h}_i(t)$ alone is insufficient. It is also necessary to record the input data and service marks throughout the entire replay interval. For this reason, a sequence of input vectors is introduced:

$$\vec{X}_i(t) = \left(\vec{x}_i\left(b_i^*(t)\right), \vec{x}_i\left(b_i^*(t)+1\right), \ldots, x_i\left(r_i(t)\right)\right), \quad (44)$$

and the sequence of service marks:

$$\vec{V}_i(t) = \left(\vec{v}_i\left(b_i^*(t)\right), \vec{v}_i\left(b_i^*(t)+1\right), \ldots, \vec{v}_i\left(r_i(t)\right)\right). \quad (45)$$

Then, the minimally sufficient state of the process for reproduction is formulated as follows:

$$\vec{s}_i^{\min}(t) = \left(\vec{h}_i(t), \vec{X}_i(t), \vec{V}_i(t)\right)^{\mathsf{T}}. \quad (46)$$

The vector $\vec{s}_i^{\min}(t)$ (46) is used as the input dataset for the reproduction procedure. Its sufficiency means that, based on $\vec{s}_i^{\min}(t)$, it is possible to restart the process from the point $b_i^*(t)$ and bring the process to the same local state that corresponds to the correct initial execution. Its minimality means that removing at least one essential component from (46) results in the loss of the ability to ensure such a restart and such a reproduction of the process state.

The conditions under which the vector $\vec{s}_i^{\min}(t)$ is sufficient and at the same time minimal for restarting the process are defined as follows:

$$\pi_i\left(\Re_i\left(\vec{s}_i^{\min}(t)\right)\right) = \pi_i\left(\vec{s}_i^*(t)\right), \quad (47)$$

$$\forall \tilde{\vec{s}}_i^{\min}(t) \prec \vec{s}_i^{\min}(t) : \pi_i\left(\Re_i\left(\tilde{\vec{s}}_i^{\min}(t)\right)\right) \ne \pi_i\left(\vec{s}_i^*(t)\right), \quad (48)$$

where $\Re_i$ is the operator that generates the local state of the process after a restart; $\vec{s}_i^*$ is the reference local state of the process after the restart is complete; the relation $\prec$ denotes the removal of at least one component from the vector $\vec{s}_i^{\min}(t)$. Equation (47) implies that the vector $\vec{s}_i^{\min}(t)$ contains all the data without which it is impossible to reach the reference local state $\vec{s}_i^*$. Equation (48) means that removing any component from $\vec{s}_i^{\min}(t)$ violates this property. Thus, equation (47) means that state (46) ensures the attainment of the reference local state of the process, while equation (48) shows that removing any of its components violates this possibility.

Determinism in recovery means that for identical vectors $\vec{s}_i^{\min}(t)$ and identical restart rules, the result of process recovery is the same. This requirement is stated as follows:

$$\left(\vec{s}_i''(t), \vec{\sigma}_i''(t)\right) = \left(\vec{s}_i'''(t), \vec{\sigma}_i'''(t)\right) \Rightarrow \Re_i\left(\vec{s}_i''(t), \vec{\sigma}_i''(t)\right) = \Re_i\left(\vec{s}_i'''(t), \vec{\sigma}_i'''(t)\right), \quad (49)$$

where $\vec{\sigma}_i(t)$ is a vector of restart service rules that specifies the order of input data presentation during the restart interval, the order of service mark usage, restrictions on re-storing the result outside the process, and the order of eliminating redundant result storage.

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

The result of applying the method is the restored local state:

$$\vec{s}_i(t+1) = \Re_i\left(\vec{s}_i^{\min}(t), \vec{\sigma}_i(t)\right) \quad (50)$$

for which the condition for the process to transition to a locally stable state must be satisfied:

$$\vec{\sigma}_i\left(\vec{s}_i(t+1)\right) = \vec{0}, \quad \mu_i\left(\vec{s}_i(t+1)\right) \le 1. \quad (51)$$

where $\ell_1, \ell_2, \ell_3, \ell_4 \ge 0$ are coefficients that determine the contribution based on the number of steps to be repeated, the size of the local state elements that must be saved for a subsequent run, the total size of the input data, and the total size of the service markers.

In formula (52), the term $\left(r_i(t) - b_i^*(t)\right)$ determines the number of steps that must be repeated after returning to the reference point $b_i^*(t)$. The value $\left\|\vec{h}_i(t)\right\|_0$ determines the number of local state elements that must be saved and passed for the restart. The sum $\sum_{\zeta=b_i^*(t)}^{r_i(t)} \left\|\vec{x}_i(\zeta)\right\|_1$ characterizes the total volume of input data over the restart interval, and the sum $\sum_{\zeta=b_i^*(t)}^{r_i(t)} \left\|\vec{v}_i(\zeta)\right\|_1$ characterizes the total volume of service marks over the same interval. The smaller the value of $\Im_i(t)$, the lower the cost of restarting the process at the selected checkpoint. Therefore, the $\Im_i(t)$ function is used as a quantitative measure of recovery costs to compare different options for returning the process to its functional (previous) state.

Thus, the sequence of steps for implementing the method is as follows.

Step 1. For the process $p_i(t)$, for which the condition $u_i^*(t) = \kappa_i$ (37) holds, a set of reference points $B_i(t)$ (38) is formed based on the service marks $\vec{v}_i(t)$.

Step 2. From the set $B_i(t)$, points are selected that satisfy the constraints regarding re-execution feasibility and resource-time characteristics. As a result, a set of admissible reference points $B_i^{\dagger}(t)$ (39) is formed.

Equations (50) and (51) imply that after a restart and recovery, the process must transition to a state without local inconsistency (5), and the result of the current step cannot be recorded outside the process more than once.

The total cost of restarting the process can be estimated by the following functional:

$$\Im_i(t) = \ell_1\left(r_i(t) - b_i^*(t)\right) + \ell_2\left\|\vec{h}_i(t)\right\|_0 + \ell_3 \sum_{\zeta=b_i^*(t)}^{r_i(t)} \left\|\vec{x}_i(\zeta)\right\|_1 + \ell_4 \sum_{\zeta=b_i^*(t)}^{r_i(t)} \left\|\vec{v}_i(\zeta)\right\|_1, \quad (52)$$

Step 3. Based on criterion (40), the reference point $b_i^*(t)$ is selected (41).

Step 4. For the point $b_i^*(t)$, a local state coordinate vector is generated, which must be saved for the subsequent execution of $\vec{h}_i(t)$ (43); thereafter, the input data sequence $\vec{X}_i(t)$ (44) and the control tag sequence $\vec{V}_i(t)$ (45) are appended to it. As a result, a minimally sufficient state vector $\vec{s}_i^{\min}(t)$ (46) is generated.

Step 5. It is checked whether the vector $\vec{s}_i^{\min}(t)$ ensures the achievement of the reference local state of the process (47) and whether it contains redundant components (48). If condition (47) is not satisfied, the set of coordinates in $\vec{h}_i(t)$ is expanded by re-tuning the matrix $\vec{G}_i(t)$ (42), after which an updated vector $\vec{s}_i^{\min}(t)$ is formed.

Step 6. For the constructed vector $\vec{s}_i^{\min}(t)$, the local state of the process $\Re_i$ (50) is reconstructed while satisfying the determinism condition (49).

Step 7. Once the process recovery is complete, the condition for the process to transition to a locally valid state is checked (51). If this condition is met, the process proceeds to further execution. If not, the results of the restart are discarded, and the process is handed off to service-level recovery mechanisms or ISMP reconfiguration (36).

Thus, relations (38)–(52) and the given sequence of steps define a method for local recovery of ISMP processes with idempotent restart and deterministic state reproduction. Its difference from a full process restart lies in the fact that only that set of data and service flags is used for the restart, without which it is impossible to ensure that the reference local state of the process is achieved. And the difference from service-level recovery lies in the fact that recovery is performed for a single

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

process and does not require switching between service profiles or a general reconfiguration of the ISMP.

5.4. Evaluation of the Proposed Method's Effectiveness

The effectiveness of the proposed method was evaluated using a series of model scenarios of local disruptions to the internal state of ISMP processes. A comparison was conducted with two alternative recovery modes: a full process restart and deferred service-level recovery. The simulation code used to generate the experimental results is presented in [23], and the FAIR data set with the original numerical arrays of the experiment is presented in [24].

Within the experiment, the same scenario set was used for all three modes, including identical sequences of input events, identical times and types of local process state violations, identical time series of available unoccupied mobile platform resources, and identical parameters of connectivity windows with variable duration and bandwidth.

The proposed local recovery method implements a process-level response to internal state violations and, for each failure episode, selects between micro-recovery and micro-restart according to rules (31)–(36), followed by deterministic reconstruction of the minimally sufficient local state according to relations (38)–(52). The full process restart mode is considered the basic recovery method, in which the process, upon detecting a violation, returns to its initial execution state without selecting a reference point, without selecting a minimally sufficient set of local data, and without considering the division of steps based on the admissibility of re-execution. The deferred service recovery mode is considered a higher-level recovery method, in which no local action is performed at the process level, and recovery is deferred until it becomes possible to apply service procedures such as reconfiguration, profile switching, or other recovery actions at the service or ISMP node level.

The comparison is performed based on six metrics: the dynamics of local process state corruption, cumulative recovery deficit, probability of successful local recovery, average time for the process to return to correct execution, the amount of state required for recovery, and the risk of re-generating a side effect. This set of metrics allows us to evaluate the proposed method simultaneously across three groups of properties: the speed of the process's return to correct execution, the data and resource costs of recovery, and the ability to avoid undesirable consequences of re-executing process steps. First, it is advisable to examine the dynamics of the local process state damage metric. Fig. 1 shows that the proposed method ensures the fastest reduction of this metric. The maximum permissible level of 0.18 for the proposed method is reached at the 7th recovery step, whereas for deferred service recovery it is reached at the 19th step, and for a full process restart at the 25th step. Thus, based on this indicator, the proposed method reduces the time required to return the process to an acceptable state by approximately 2.7 times compared to deferred service recovery and by 3.6 times compared to a full process restart. Furthermore, after just 10–12 recovery steps, the value of the indicator for the proposed method decreases to a level close to zero and remains low thereafter. This indicates the rapid elimination of local inconsistencies in the process state without their recurrence.

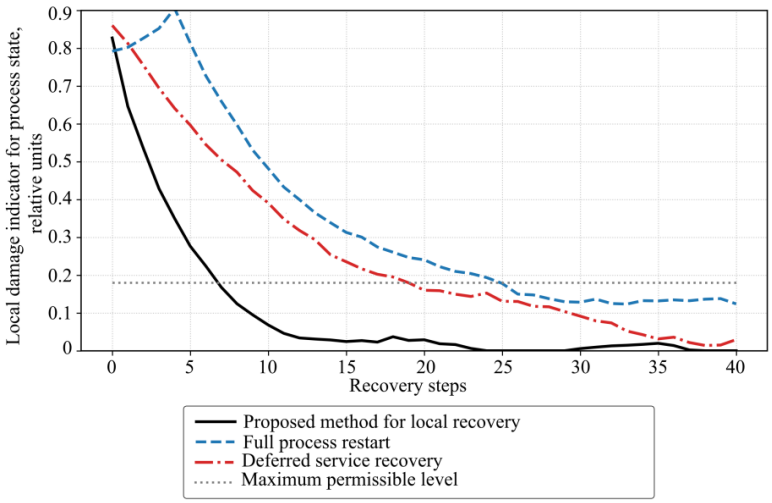

Fig. 1. Dynamics of the local process damage indicator

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

The cumulative effect of this reduction is confirmed by the values of the cumulative recovery deviation index (Fig. 2). The final value of this indicator for the proposed method is 2.03 relative units, whereas for a full process restart it is 7.64 relative units, and for a deferred service recovery it is 5.52 relative units. Accordingly, the reduction in the cumulative recovery deviation index is 73.5% compared to a full process restart and 63.3% compared to a deferred service recovery. Based on the average value across the entire simulation interval, the advantage also remains: 1.95 relative units for the proposed method versus 6.46 and 4.86 relative units for the two comparison modes. This means that the proposed method ensures a smaller cumulative recovery deviation across the entire interval of the process's return to correct execution.

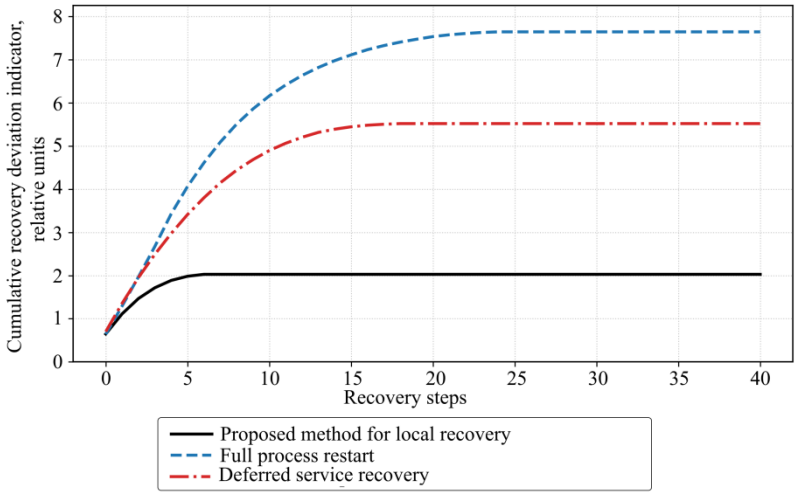

Fig. 2. Cumulative recovery deviation index

The next indicator is the probability of successful process recovery depending on the severity of local damage to its state. Figure 3 shows that, across the entire range of severities, the proposed method yields the highest probability of successful recovery completion. At the maximum intensity value, corresponding to the right boundary of the range, the probability of success for the proposed method is 0.706, for a full process restart – 0.542, and for a deferred service recovery – 0.435. Thus, even in the most challenging scenario, the advantage is 30.3% compared to a full process restart and 62.3% compared to a deferred service recovery. The advantage also holds true for the average value across the entire interval: 0.863 versus 0.775 and 0.699. This confirms that the combination of local state correction, the elimination of invalid re-executions, and the deterministic reproduction of a minimally sufficient local state ensures higher recovery performance as the intensity of failures increases.

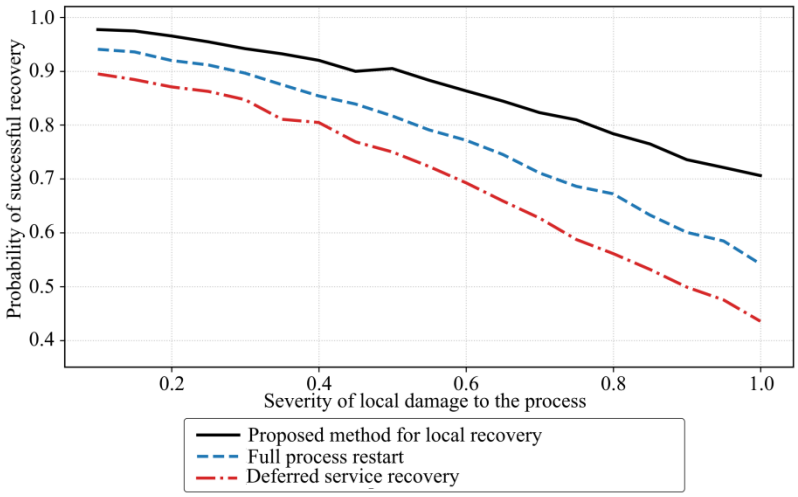

Fig. 3. Probability of successful process recovery depending on the severity of local damage to the state

ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

A key performance indicator of the proposed method is the average time required for the process to return to correct execution. Figure 4 shows that, for the proposed method, this time increases at the slowest rate as the intensity of local state corruption increases. At the maximum intensity value, the average time for the process to return to correct execution is 6.13 for the proposed method, 9.96 for a full process restart, and 12.39 for deferred service recovery. Thus, the reduction in this metric is 38.4% compared to a full process restart and 50.5% compared to deferred service recovery. Over the entire range of intensity variation, its average value is 4.05 for the proposed method, versus 6.55 and 8.22 for the two comparison modes. This is consistent with the formalism of the subdivisions of the introduced rule and the method based on it within the framework of this study, since the choice between micro-recovery and micro-restart reduces the length of the re-executed fragment, and the reconstruction of a minimally sufficient local state eliminates the need for a full process restart or a transition to service-level recovery.

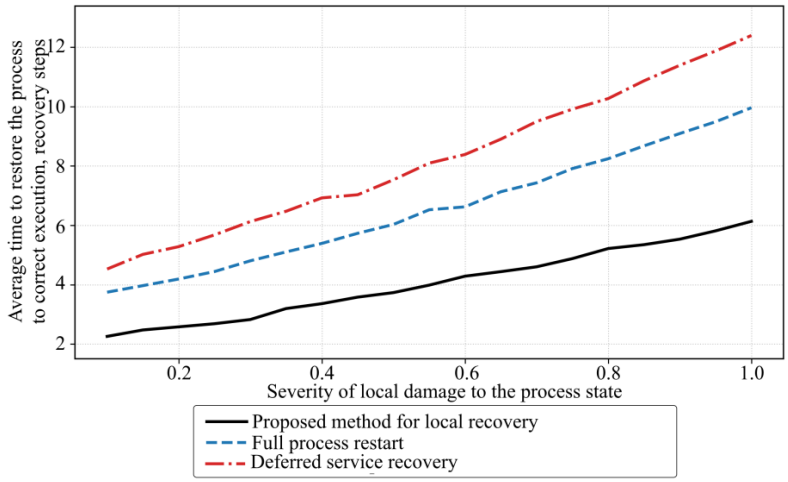

Fig. 4. Average time for the process to resume correct execution

Special attention should be paid to the amount of data required for process reconstruction. This metric directly characterizes the method's requirements regarding the volume of data used during reconstruction. Based on the results of modeling the method's operation [24], it was found that the proposed method requires the smallest amount of data needed for process recovery across the entire range of local damage intensity values of the process state (Fig. 5). At the maximum intensity value, this volume is 37.19 MB for the proposed method, whereas for a full process restart it is 96.60 MB, and for deferred service recovery it is 74.33 MB. Thus, the amount of data required to restore the process is reduced by 61.5% compared to a full process restart and by 50% compared to deferred service recovery. On average, across the entire range of changes in the intensity of local damage to the process state, these values are 25.02, 69.26, and 51.04 MB, respectively. This result is due to the fact that in the proposed method, only the set of process coordinates, input data, and service markers necessary to achieve the reference local state of the process is generated for process restart, in accordance with equations (42)–(48).

Next, we analyze the probability of duplicating an external result depending on the proportion of steps involving non-idempotent re-execution in the process. Duplication of the external result is understood here as a repeated write to an external log, repeated transmission of a message, repeated update of a shared store, or repeated sending of a command to another node component or ISMP. Fig. 6 shows that as the proportion of such steps increases, the value of this metric rises for all three comparison modes; however, for the proposed method, it remains the lowest across the entire range of the argument. In the most complex scenario, where the proportion of steps with non-idempotent retries approaches one, the final value of the metric is 0.0449 for the proposed method, 0.2034 for a full process restart, and 0.1566 for deferred service recovery. Accordingly, the reduction in this metric compared to a full process restart is 77.9%, and compared to deferred service recovery, it is 71.3%. On average, across the entire range of changes in the proportion of steps with non-idempotent re-execution, the advantage

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

also persists: 0.0228 for the proposed method versus 0.1140 and 0.0868 for the two comparison modes. This is consistent with formulas (22) and (23), since for micro-recovery, the current step is not re-executed, and for micro-restart, the consequences of re-execution are determined by the step's membership in sets (10)–(12) and the possibility of locally eliminating the duplicated result.

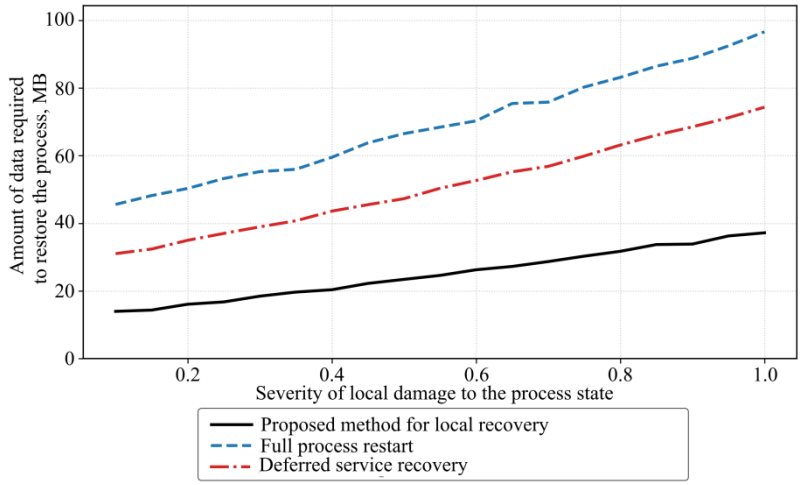

Fig. 5. Amount of data required to restore the process, depending on the severity of local damage to the process state

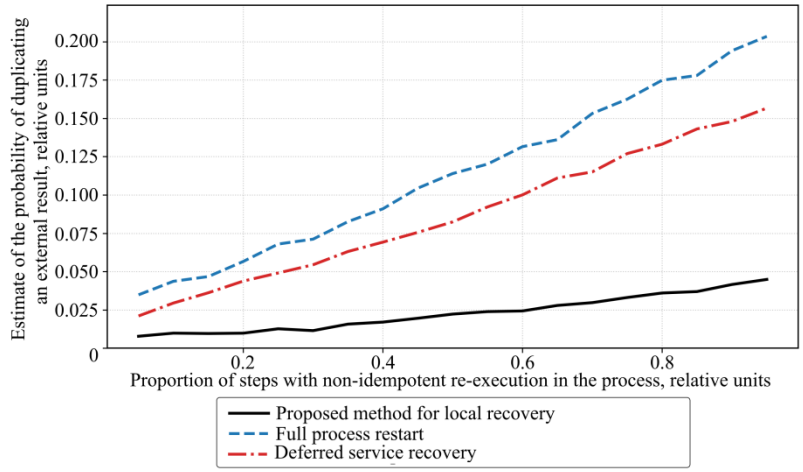

Fig. 6. Dependence of the estimated probability of duplicating an external result on the proportion of steps involving non-idempotent repetition in the process

The results of the evaluation of the proposed method's effectiveness show that it offers a consistent advantage over the two comparison methods across all studied metrics. This advantage is manifested in the fact that the local damage indicator of the process state decreases more rapidly, the cumulative recovery deviation indicator is significantly lower, the probability of successfully returning the process to correct execution is higher, the average recovery time is shorter, the amount of data required for process recovery, and the probability of duplicating the external result is reduced. This provides grounds for asserting that local recovery with idempotent re-execution and deterministic reproduction of the minimally sufficient local state is appropriate for ISMP in those operating modes where a full process restart is excessive in terms of time and the volume of re-executed actions, and deferred service recovery does not ensure the timely return of the process to correct execution.

5.5. Discussion of Research Results

The results of the model study indicate that the proposed method provides a consistent advantage across all investigated metrics [24]. Within the framework of the adopted formulation, this means that local process recovery should be considered as an independent level of recovery intervention in ISMP. Its place is defined between a full process restart

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

and service-level recovery. This conclusion follows from the fact that the method simultaneously reduces the time required for the process to return to correct execution, decreases the cumulative recovery deviation metric, increases the probability of successful recovery, reduces the amount of data required for restart, and limits the probability of duplicating the external result. Thus, the method's advantage is systemic in nature and reflects the properties of the local recovery procedure itself, rather than a random improvement in a single metric.

The content of the obtained result is determined by a combination of two formalized decisions. The first decision is related to the selection of a recovery action based on relations (30)–(33). In this procedure, a local disruption of the process state is not equated with an automatic return to the start of execution. For each disruption episode, the following are evaluated: the local admissibility of the action, the relationship between its requirements and the available time and resources, the assessment of the replication of the external result, and the compliance of the current state with the conditions for restarting. The second solution involves constructing a minimally sufficient local state for restarting the process according to relations (41)–(49). As a result, not the entire volume of local data is involved in the recovery, but only that part of it which is necessary to achieve the reference local state of the process.

It is precisely this construction that explains why the achieved improvement manifests itself simultaneously in temporal, functional, and volumetric metrics.

Modern approaches related to this subject area confirm individual components of this conclusion but do not encompass them within a single process mechanism. The direction of checkpoint orchestration and accelerated restart of virtual functions [25] confirms the importance of reducing recovery latency. Here, the state of the execution environment serves as the control object. Consequently, the following remain outside the scope of consideration: classes of process steps based on re-execution feasibility, the procedure for choosing between micro-resumption and micro-restart, and the minimally sufficient local process state for continuing execution. This is fundamental for this article, since the time savings achieved are linked to the localization of the recovery effect itself at the level of an individual process.

A technologically related area is local recovery and partial state preservation in stream processing systems [26]. Its significance for this study lies in confirming that reducing the scope of recovery improves the effectiveness of recovery procedures. At the same time, the object of recovery is a subset of stream graph operators. For ISMP, this formulation is insufficient, since the process level requires consideration of the semantics of individual steps, the recording of the external result, and the choice between two modes of local recovery. In this sense, the proposed method extends the mechanism of recovery localization to the process-step level, where the volume of data to be recovered and the admissibility of process re-execution are essential.

To justify deterministic reproduction, the results obtained for transactional systems with mutable state [27] and the results of behavioral consistency verification for applications with mutable state [28] are essential. These lines of research confirm that correct recovery is impossible without reproducibility of execution and without verifying that the obtained result matches the expected state. However, in [27], determinism is achieved through the repeated presentation of the input stream and the architecture of transactional execution, while in [28], the subject of study is the verification of the identity of results for different execution variants. In this work, the problem is formulated in a different context: it is necessary to construct a local data set sufficient for the correct resumption of a single process after a violation of its internal state. That is why the selection of a reference point, the construction of a recovery vector, and the verification of reaching the reference local state of the process occupy a central place in the formalism of the method.

Related areas include approaches to the porting of functions and data in edge environments [29] and state management systems for serverless cloud applications [30]. Their significance for this study lies in the fact that they emphasize the importance of explicit state management in environments with variable resource availability.

At the same time, these approaches focus on process migration between nodes, external state management, or infrastructure-level fault tolerance. For ISMP, this is insufficient in cases where the recovery action must be completed within an available local time window and with an available idle resource, without waiting for migration, profile switching, or other higher-level reconfiguration. It is here that the proposed method occupies a distinct application niche.

Thus, the results of the discussion provide grounds to believe that the task of local process recovery in ISMP constitutes a distinct subclass of tasks within the system for ensuring the survivability of information

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

systems. Its specificity is determined by the simultaneous consideration of four groups of factors: local damage to the internal state of the process, the admissibility of re-executing process steps, resource- and time-constraints for recovery, and the need for deterministic reconstruction of a minimally sufficient local state of the process. The comprehensive consideration of precisely these factors is absent in the contemporary approaches discussed [25–30]. Therefore, the obtained results define an independent process-based recovery mechanism for ISMP.

The results obtained should be interpreted taking into account the conditions adopted in the model study, which covers scenarios of local disruption of the state of a single process and a comparison of three recovery modes based on a fixed set of indicators [24]. Under the conditions adopted in this model study, the effectiveness of the local process recovery method and its advantage over the two comparison modes have been confirmed. Further study is required for scenarios involving simultaneous disruption of multiple interdependent processes, competition modes for an unoccupied mobile platform resource under higher load, and the impact of architectural heterogeneity of nodes on the selection of a recovery reference point. Such an expansion will enable the transition to studying the multiprocessing mode of ISMP operation.

The scientific novelty of the obtained results lies in the fact that, for the first time, a method for local recovery of information system processes on a mobile platform has been developed which, unlike known approaches, formalizes the choice between micro-recovery and micro-restart based on an assessment of local damage to the internal state of the process, the feasibility of re-executing steps, and the trade-off between recovery requirements and available time and resources, and defines a procedure for deterministically reproducing the minimally sufficient local state of the process, which allows for the correct resumption of process execution without resorting to a full restart or service recovery, reduces the duration of local operational disruptions, and thereby enhances the resilience of the information system on a mobile platform.

6. Conclusions and Prospects or Further Research

This article solves the problem of developing a method for the local recovery of information system processes on a mobile platform under conditions of local disruption of their internal state, limited resources, and intermittent connectivity. To this end, the local state of the process was formalized, a vector of inconsistency and a scalar measure of local damage to the process state were introduced, which made it possible to proceed to the process-level analysis of local recovery.

On this basis, a rule was developed for choosing between micro-recovery and micro-restart, taking into account the expected state of the process after the recovery action, the relationship between the requirements of the action and the available time and resources, the assessment of the replication of the external result, and the compliance of the current state with the conditions for restart. A method has been developed for the deterministic reproduction of the minimally sufficient local state of the process, which determines the selection of the restart reference point and the formation of the set of local state coordinates, input data, and service markers required for correct continuation of execution.

The results of the model study confirmed the effectiveness of the proposed method compared to a full process restart and deferred service recovery. It has been established that its application ensures a faster reduction in the local process state damage metric, a lower cumulative recovery deviation metric, a higher probability of successful process return to correct execution, a shorter average recovery time, a smaller volume of data required for restart, and a lower estimate of the probability of duplicating the external result.

The results obtained confirm that local process recovery with idempotent re-execution and deterministic reproduction of the minimally sufficient local state is appropriate as a separate process mechanism in the system for ensuring the survivability of an information system on a mobile platform, as it allows reducing the duration of local process execution failures without resorting to a full restart or service-level recovery in cases where local recovery is sufficient.

Prospects for further research should focus on extending the proposed method to cases of simultaneous state violations in several interdependent processes, accounting for inter-process dependencies when constructing a minimally sufficient local state for recovery, and integrating local process recovery with service recovery methods, dynamic resource reservation, and prediction of cascading transitions within a unified survivability assurance system.

Conflict of Interest

The author declares that there is no conflict of interest, particularly of a financial, personal, authorial, or any other nature, that could influence the research or the results published in this article.

Data Availability

The data supporting the results of this study are available in open access in the Zenodo repository [24]. The modeling code used to obtain the presented results is provided in [23].

Funding

The research was conducted without financial support.

Use of Artificial Intelligence

The author confirms that he did not use artificial intelligence technologies to write this paper.

References

1. Ergenç, D., Memedi, A., Fischer, M., Dressler, F. (2025), "Resilience in Edge Computing: Challenges and Concepts", *Foundations and Trends® in Networking*, Vol. 14, No. 4, pp. 254–340. DOI: https://doi.org/10.1561/1300000074
2. Shaikh, S., Jammal, M. (2024), "Survey of fault management techniques for edge-enabled distributed metaverse applications", *Computer Networks*, Vol. 254, Article No. 110803. DOI: https://doi.org/10.1016/j.comnet.2024.110803
3. Zhang, Y., Zhao, K., Yang, Y., Zhou, Z. (2025), "Real-Time Service Migration in Edge Networks: A Survey", *Journal of Sensor and Actuator Networks*, Vol. 14, No. 4, 79. DOI: https://doi.org/10.3390/jsan14040079
4. Trung, K.N., Tran, M.-N., Kim, Y. (2026), "Enabling Service Continuity for Stateful Service Segmentation in Mobile Edge Computing Toward 6G", *IEEE Access*, Vol. 14, pp. 20589–20604. DOI: https://doi.org/10.1109/ACCESS.2026.3661972
5. Додонов, О.Г., Кузнєцова, М.Г., Горбачик, О.С. (2025), "Моделювання і оцінювання функціональної стійкості інформаційних систем", *Реєстрація, зберігання і обробка даних*, Т. 27, № 1, С. 76–88. DOI: https://doi.org/10.35681/1560-9189.2025.27.1.335752
6. Додонов, О.Г., Горбачик, О.С., Кузнєцова, М.Г. (2007), "Живучість інформаційно-аналітичних систем: понятійний апарат, моделі аналізу та оцінки", *Реєстрація, зберігання і обробка даних*, Т. 9, № 3, С. 61–72.
7. Kharchenko, V., Sachenko, A., Kochan, V., Fesenko, H. (2016), "Reliability and Survivability Models of Integrated Drone-Based Systems for Post Emergency Monitoring of NPPs", *Proceedings of The International Conference on Information and Digital Technologies*, pp. 127–132. DOI: https://doi.org/10.1109/DT.2016.7557161
8. Bindschaedler, L. (2026), "Rebooting Microreboot: Architectural Support for Safe, Parallel Recovery in Microservice Systems", *ARCS 2026: 39th GI/ITG International Conference on Architecture of Computing Systems*, URLt: https://binds.ch/papers/microreboot2026.pdf
9. Jia, Z., Witchel, E. (2024), "Boki: Towards Data Consistency and Fault Tolerance with Shared Logs in Stateful Serverless Computing", *ACM Transactions on Computer Systems*, Vol. 42, No. 3–4, pp. 1–35. DOI: https://doi.org/10.1145/3653072
10. Qi, S., Feng, H., Liu, X., Jin, X. (2025), "Efficient Fault Tolerance for Stateful Serverless Computing with Asymmetric Logging", *ACM Transactions on Computer Systems*, Vol. 43, No. 1–2, pp. 1–43. DOI: https://doi.org/10.1145/3725985
11. You, J., Chen, K., Zhao, L., Li, Y., Chen, Y., Du, Y., Wang, Y., Wen, L., Hu, K., Li, K. (2025), "AlloyStack: A Library Operating System for Serverless Workflow Applications", *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 921–937. DOI: https://doi.org/10.1145/3689031.3717490
12. Ding, H., Wang, Z., Shen, Z., Chen, R., Chen, H. (2023), "Automated Verification of Idempotence for Stateful Serverless Applications", *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pp. 887–910, available at: https://www.usenix.org/conference/osdi23/presentation/ding
13. Galstyan, N. (2024), "Application-Integrated Record-Replay of Distributed Systems", *Technical Report No. UCB/EECS-2024-4*, University of California, Berkeley, URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-4.pdf
14. Zhong, X., Zhao, X., Zhang, B., Li, J., Wang, Y., Li, Y. (2025), "A Record–Replay-Based State Recovery Approach for Variants in an MVX System", *Information*, Vol. 16, No. 10, 826. DOI: https://doi.org/10.3390/info16100826
15. Ткачов, В. (2026), "Метод відновлення критичних сервісів інформаційної системи на мобільній платформі в умовах керованої деградації", *Автоматизовані системи управління та прилади автоматики*, (188), с. 78–97. DOI: https://doi.org/10.30837/0135-1710.2026.188.078
16. Tkachov, V., Ruban, I. (2025), "Integral survivability metric of an information system on a mobile platform under functional cascading and secondary failures", *Innovative Technologies and Scientific Solutions for Industries*, No. 4(34), pp. 78–100. DOI: https://doi.org/10.30837/2522-9818.2025.4.078

17. Whitaker, M., Volckaert, B., Al-Naday, M. (2025), "Idempotency in Service Mesh: For Resiliency of Fog-Native Applications in Multi-Domain Edge-to-Cloud Ecosystems", *Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER 2025)*, pp. 182–189. DOI: https://doi.org/10.5220/0013293900003950
18. Qi, S., Liu, X., Jin, X. (2023), "Halfmoon: Log-Optimal Fault-Tolerant Stateful Serverless Computing", *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, pp. 314–330. DOI: https://doi.org/10.1145/3600006.3613154
19. Tošić, A. (2024), "Run-Time Application Migration Using Checkpoint/Restore In Userspace", *Journal of Web Engineering*, Vol. 23, No. 05, pp. 735–748. DOI: https://doi.org/10.13052/jwe1540-9589.2357
20. Calagna, A., Yu, Y., Giaccone, P., Chiasserini, C. F. (2023), "Design, Modeling, and Implementation of Robust Migration of Stateful Edge Microservices", *IEEE Transactions on Network and Service Management*, Vol. 21, No. 2, pp. 1877–1893. DOI: https://doi.org/10.1109/TNSM.2023.3331750
21. Rong, C., Wang, J.H., Wang, J., Zhou, Y., Zhang, J. (2024), "Live Migration of Video Analytics Applications in Edge Computing", *IEEE Transactions on Mobile Computing*, Vol. 23, No. 3, pp. 2078–2092. DOI: https://doi.org/10.1109/TMC.2023.3246539
22. Liu, Z., Unal, M., Parkinson, M.J., Kogias, M. (2025), "DORADD: Deterministic Parallel Execution in the Era of Microsecond-Scale Computing", *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '25)*, pp. 282–296. DOI: https://doi.org/10.1145/3710848.3710872
23. Tkachov, V. (2026), "Local-process-recovery-figures", *GitHub repository*, URL: https://github.com/tikey/local-process-recovery-figures
24. Tkachov, V. (2026), "Source data and generated figures for: A Method for Local Recovery of Processes in Information System on Mobile Platform Using Idempotent Re-Execution and Deterministic State Reconstruction", *Zenodo*. DOI: https://doi.org/10.5281/zenodo.19565291
25. Kohli, S., Kharbanda, S., Bruno, R., Carreira, J., Fonseca, P. (2024), "Pronghorn: Effective Checkpoint Orchestration for Serverless Hot-Starts", *Proceedings of the 19th European Conference on Computer Systems (EuroSys '24),* pp. 298–316. DOI: https://doi.org/10.1145/3627703.3629556
26. Takdir, Kitagawa, H., Amagasa, T. (2025), "Local recovery and partial snapshot in distributed stateful stream processing", *Knowledge and Information Systems*, Vol. 67, pp. 9407–9435. DOI: https://doi.org/10.1007/s10115-025-02509-z
27. Psarakis, K., Christodoulou, G.C., Fragkoulis, M., Katsifodimos, A. (2025), "Transactional Cloud Applications Go with the (Data)Flow", *Proceedings of the 15th Annual Conference on Innovative Data Systems Research (CIDR '25)*, URL: https://vldb.org/cidrdb/papers/2025/p25-psarakis.pdf
28. Zhu, K., Whittaker, M., Petrovic, S., Grandl, R., Ghemawat, S. (2025), "Vive la Différence: Practical Diff Testing of Stateful Applications", *Proceedings of the VLDB Endowment*, Vol. 18, No. 7, pp. 2018–2030. DOI: https://doi.org/10.14778/3734839.3734841
29. Nardelli, M., Russo Russo, G. (2024), "Function Offloading and Data Migration for Stateful Serverless Edge Computing", *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24)*, pp. 247–257. DOI: https://doi.org/10.1145/3629526.3649293
30. Li, T., Chandramouli, B., Burckhardt, S., Madden, S. (2024), "Serverless State Management Systems", *Proceedings of the 14th Conference on Innovative Data Systems Research (CIDR 2024)*, URL: https://vldb.org/cidrdb/papers/2024/p16-li.pdf

Received (Надійшла) 15.03.2026
Accepted for publication (Прийнята до друку) 10.04.2026
Publication date (Дата публікації) 29.05.2026

Відомості про авторів / About the Authors

Ткачов Віталій Миколайович – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, докторант кафедри електронних обчислювальних машин, Харків, Україна;

Vitalii Tkachov – Candidate of Technical Sciences, Associate Professor, Kharkiv National University of Radio Electronics, Doctoral candidate at the Department of Electronic Computers, Kharkiv, Ukraine;

e-mail: vitalii.tkachov@nure.ua

ORCID ID: https://orcid.org/0000-0002-6524-9937

ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

МЕТОД ЛОКАЛЬНОГО ВІДНОВЛЕННЯ ПРОЦЕСІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ НА МОБІЛЬНІЙ ПЛАТФОРМІ

Предметом дослідження в статті є локальні порушення внутрішнього стану окремих процесів інформаційної системи на мобільній платформі, які унеможливлюють коректне продовження виконання без спеціального відновлювального впливу. **Мета роботи** – розроблення методу локального відновлення процесів інформаційної системи на мобільній платформі, який у разі локального порушення внутрішнього стану процесу забезпечує продовження його виконання за допомогою ідемпотентного повторного виконання й детермінованого відтворення мінімально достатнього локального стану в умовах обмежених ресурсів і переривчастої зв'язності. У статті необхідно виконати такі **завдання**: формалізувати локальний стан процесу й ознаки його порушення; встановити класи кроків процесу за допустимістю повторного виконання; розробити правило вибору між мікровідновленням і мікроперезапуском; запропонувати метод детермінованого відтворення мінімально достатнього локального стану процесу; виконати модельне дослідження ефективності запропонованого методу порівняно з повним перезапуском процесу й відкладеним сервісним відновленням. **Досягнуті результати**. Розроблено формалізацію локального стану процесу у вигляді вектора, що містить номер поточного кроку, внутрішні дані процесу, вхідні дані кроку, службові познаки виконання та кратність фіксації результату поза межами процесу. Локальне порушення стану визначено через вектор неузгодженості та скалярний показник локального пошкодження стану процесу. Запропоновано правило локального відновлення, яке формалізує вибір між мікровідновленням і мікроперезапуском з огляду на очікуваний стан процесу після відновлювальної дії, співвідношення між вимогами дії та доступними часом і ресурсами, оцінки дублювання зовнішнього результату, відповідності поточного стану умовам повторного запуску. Розроблено метод детермінованого відтворення мінімально достатнього локального стану процесу, який задає вибір опорної точки повторного запуску й формування сукупності координат локального стану, вхідних даних і службових познак, необхідних для коректного продовження виконання. Результати модельного дослідження показали, що запропонований метод забезпечує швидше зменшення показника локального пошкодження стану процесу, менше значення накопичувального показника відхилення відновлення, вищу ймовірність успішного повернення процесу до коректного виконання, менший середній час відновлення, менший обсяг даних, потрібних для повторного запуску, і меншу оцінку ймовірності дублювання зовнішнього результату. **Висновки:** запропонований метод є доцільним як окремий процесний механізм у системі забезпечення живучості інформаційної системи на мобільній платформі, оскільки дає змогу скоротити тривалість локальних порушень виконання процесів і підвищити живучість системи без переходу до повного перезапуску або сервісного відновлення в разі, коли локальне відновлення є достатнім.

Ключові слова: інформаційна система на мобільній платформі; процес; ідемпотентність; детерміноване відтворення стану; живучість.

Бібліографічні описи / Bibliographic descriptions

Ткачов В. М. Метод локального відновлення процесів інформаційної системи на мобільній платформі. *Автоматизовані системи управління та прилади автоматики*. 2026. № 2 (189). С. 72–94. DOI: https://doi.org/10.30837/0135-1710.2026.189.072

Tkachov, V. (2026), "A method for local recovery of information system processes on a mobile platform", *Management Information System and Devices*, No. 2 (189), P. 72–94. DOI: https://doi.org/10.30837/0135-1710.2026.189.072