Ensuring survivability of complex super-critical systems based on hierarchical abstraction model and adaptive reconfiguration in post-critical state

Dmyrto Huminnyi[1]

[1] Kyiv National University of Construction and Architecture,
Povitryanykh Sil avenue, 31, Kyiv, 03037
[1] apollo.d.g@gmail.com, https://orcid.org/0000-0001-6736-0543

Received: 04.11.2025, accepted: 08.12.2025
https://doi.org/10.32347/st.2025.4.1209

Abstract. This paper presents a novel concept for ensuring survivability of complex super-critical systems in post-critical state based on a four-level abstraction hierarchy and adaptive reconfiguration mechanism. A mathematical model of the survivability function S(t) is developed, integrating resource states, function activities, and their weight coefficients. An algorithm for optimal system configuration selection during component degradation is proposed, maximizing survivability under minimum functionality constraints. Practical implementation of rapid reconfiguration mechanisms based on Post-Build Configuration architecture for automotive ECUs in accordance with ISO 26262 is described.

Keywords: system survivability, post-critical state, reconfiguration, graceful degradation, abstraction hierarchy, ISO 26262, ASIL, Lyapunov function, super-critical systems.

INTRODUCTION

Modern cyber-physical systems, particularly automotive electronic control systems, are characterized by high levels of complexity and criticalityas well as strict functional safety requirements imposed on their design and operation [1, 4]. The problem of ensuring system operation after partial failures — in the so-called post-critical state, when some components have failed or operate with faults, but the system must continue to perform critical functions — is of particular relevance [5, 8, 10].

The traditional approach to functional safety, based on ISO 26262, provides for system transition to a Safe State upon failure de tection [1]. However, for many applications, particularly autonomous driving systems,

Dmytro Humennyi
Ph.D., Engineering Manager at N-iX, Associate Professor of the Department of Cybersecurity and Computer Engineering

complete system shutdown may be more dangerous than continued operation with limited functionality. This requires new approaches to designing systems capable of Graceful Degradation and rapid reconfiguration.

The purpose of this research is to develop theoretical foundations and practical methods for ensuring survivability of complex systems in post-critical state based on a hierarchical abstraction model that enables rapid system reconfiguration
while maintaining maximum possible functionality.

MATHEMATICAL MODEL OF SYSTEM SURVIVABILITY

2.1. Resource State Model

Let the system consist of a set of resources $R = \{r_1, r_2, ..., r_□\}$, where each resource can be in one of three states: H (Healthy), D (Degraded), F (Failed). The state of the i-th resource at time t is denoted as $x_i(t) \in \{H, D, F\}$.

For each resource, an operability coefficient $\rho_i(t) \in [0, 1]$ is introduced, characterizing its ability to perform functions:

$$\rho_i(t) = \rho_{0i} \cdot \exp(-\lambda_i \cdot t\_degraded), \quad (1)$$

where $\rho□_i$ is the initial operability coefficient (1.0 for H, 0.3-0.7 for D, 0 for F), λ_i is the degradation intensity, t_degraded is the time spent in degraded state. The dynamics of transitions between states is described by a Markov process with intensity matrix Q [5, 10].

2.2. System Survivability Function

The system implements a set of functions F = {F□, F□, ..., F□}, each having a weight w_j □ [0, 1] reflecting its importance for mission execution. The system survivability function is defined as:

$$S(t) = \Sigma_j\, w_j \cdot f_j(\rho□, ..., \rho□) \cdot \delta_j(t), \quad (2)$$

where $\delta_j(t)$ □ {0, 1} is the activation indicator for function F_j (determined by current configuration), $f_j(\rho)$ is the dependency function of the j-th function on resource operability coefficients:

$$f_j(\rho) = \prod_{i}□R_j\, \rho_i \cdot \prod□\,□O_j\, \max(\rho□, \rho\_backup) \quad (3)$$

where R_j is the set of required resources for function F_j, O_j is the set of optional (redundant) resources, ρ_backup is the backup channel operability coefficient.

2.3. Mission Objective Function

The level of mission execution by the system is determined by the objective function:

$$\Phi(t) = \Sigma_i\, w_i \cdot \varphi_i(t), \text{ where } \varphi_i \in \{0, 1\}, \quad (4)$$

The system transitions between operating modes (NOMINAL → DEGRADED → LIMP_HOME → SAFE_STOP) depending on the ratio of Φ(t) to threshold values Φ_threshold(mode). The survivability management task consists of maximizing S(t) subject to constraints Φ(t) ≥ Φ_min(mode).

HIERARCHICAL ABSTRACTION MODEL FOR RECONFIGURATION

To enable rapid system reconfiguration while maintaining its integrity, a four-level abstraction hierarchy is proposed (Fig. 1), where each level encapsulates implementation details of lower levels and provides interfaces for management by higher levels.

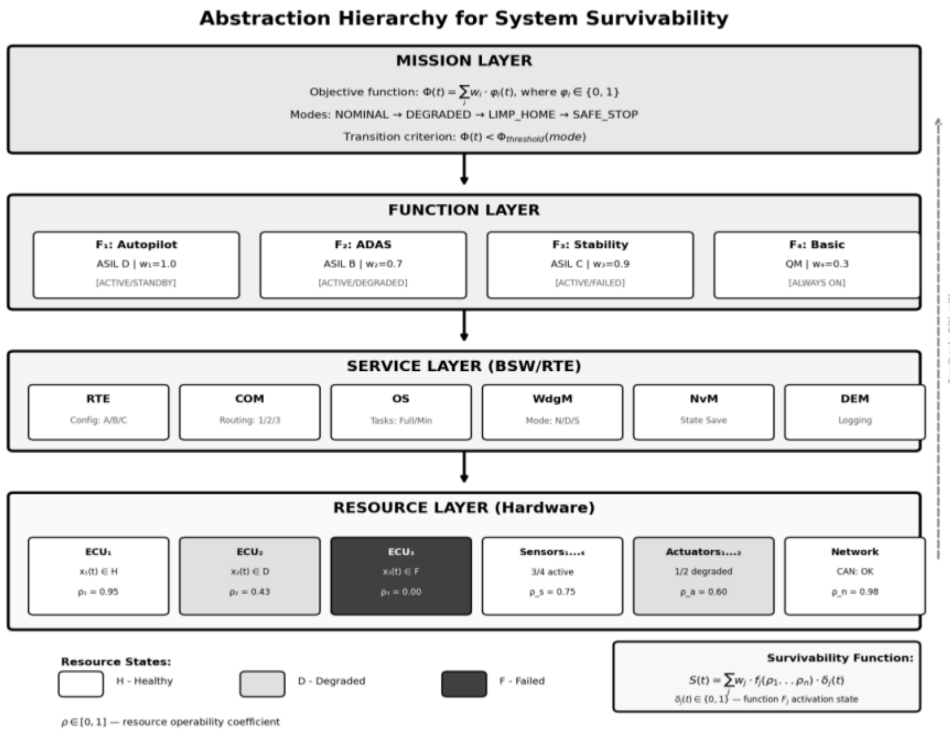


Fig. 1. Abstraction hierarchy for ensuring system survivability

3.1. Mission Layer

The highest abstraction level defines global system objectives and permissible degradation modes. At this level, decisions are made about transitions between NOMINAL, DEGRADED, LIMP_HOME, and SAFE_STOP modes based on the objective function value Φ(t). Transition criterion: $\Phi(t) < \Phi\_threshold(current\_mode) \rightarrow$ transition to lower mode.

3.2. Function Layer

This level defines the set of system functions and their states (ACTIVE, STANDBY, DEGRADED, DISABLED). Each function is characterized by weight w_j, ASIL level, and resource dependencies. Reconfiguration at this level consists of changing the set of active functions $\delta_j(t)$ to maximize S(t) for a given mission mode.

3.3. Service Layer

The Basic Software (BSW) and Runtime Environment (RTE) level with multiple configuration support. Implemented through the Post-Build Configuration mechanism, allowing switching of COM routing tables, OS settings, WdgM parameters without recompilation. Each service supports several predefined configurations (Config A/B/C).

3.4. Resource Layer

The lowest abstraction level, responsible for monitoring the state of physical components (ECU, sensors, actuators, networks) and providing information about operability coefficients $\rho_i(t)$ to higher levels. Feedback from this level is the basis for reconfiguration decisions.

ADAPTIVE RECONFIGURATION MECHANISM

4.1. Degradation State Machine

The system operates as a finite state machine with four main states and a nested reconfiguration process in the DEGRADED state (Fig. 2) [6, 7]. Transitions between states are determined by the survivability function value S(t) and thresholds S_nom, S_deg, S_min.

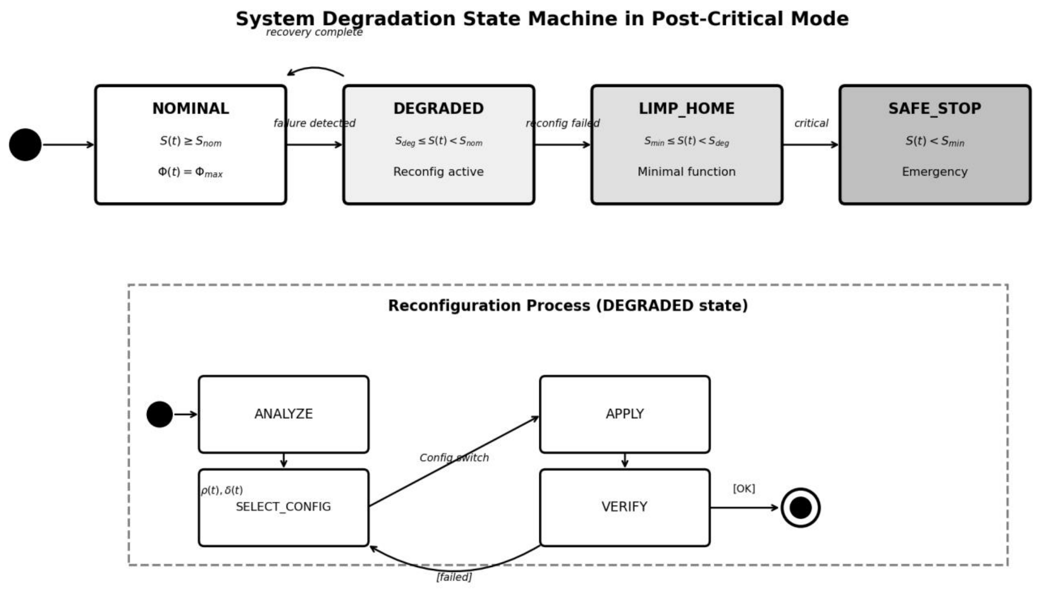


Fig. 2. System degradation state machine in post-critical mode

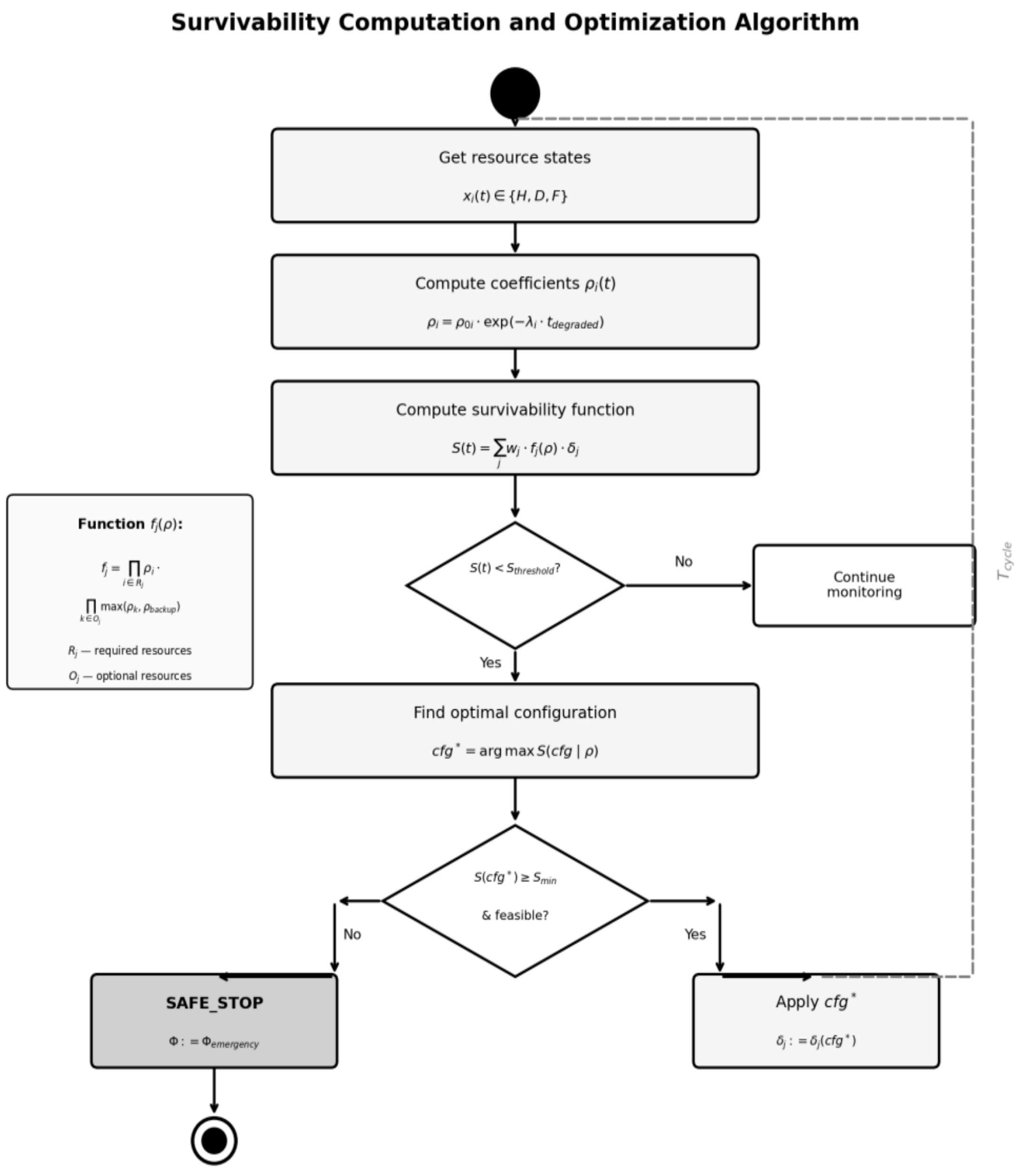
A key feature of the model is the presence of reverse transitions (recovery), allowing the system to return to a higher functionality level after successful reconfiguration or resource recovery. The reconfiguration process in the DEGRADED state includes stages: ANALYZE → SELECT_CONFIG → APPLY → VERIFY. Fig. 2. System degradation state machine in post-critical mode.

4.2. Optimal Configuration Selection Algorithm

Upon detecting system degradation, an optimal configuration search algorithm is launched (Fig. 3), solving the optimization problem:

$$cfg^* = argmax\ S(cfg \mid \rho□, ..., \rho□), \quad (5)$$

subject to: Φ(cfg) ≥ Φ_min(current_mode)



Survivability Computation and Optimization Algorithm
Get resource states
$x_i(t) \in \{H, D, F\}$
Compute coefficients $\rho_i(t)$
$\rho_i = \rho_{0i} \cdot \exp(-\lambda_i \cdot t_{degraded})$
Compute survivability function
$S(t) = \sum_j w_j \cdot f_j(\rho) \cdot \delta_j$
Function $f_j(\rho)$:
$f_j = \prod_{i \in R_j} \rho_i \cdot$
$\prod_{k \in O_j} \max(\rho_k, \rho_{backup})$
R_j — required resources
O_j — optional resources
$S(t) < S_{threshold}$?
No
Continue monitoring
T_{cycle}
Yes
Find optimal configuration
$cfg^* = \arg\max S(cfg \mid \rho)$
$S(cfg^*) \geq S_{min}$
& feasible?
No
Yes
SAFE_STOP
$\Phi := \Phi_{emergency}$
Apply cfg^*
$\delta_j := \delta_j(cfg^*)$

Fig. 3. Survivability computation and optimization algorithm

4.3. Reconfiguration Sequence

The reconfiguration process upon failure detection includes interaction of WdgM, SafetyManager, ConfigManager, RTE, BSW, and DEM components (Fig. 4). The sequence consists of five phases: Detection (failure detection), Analysis (computation of S(t) and ρ_i), Selection (optimal configuration selection), Apply (applying new configuration), Verify (result verification).

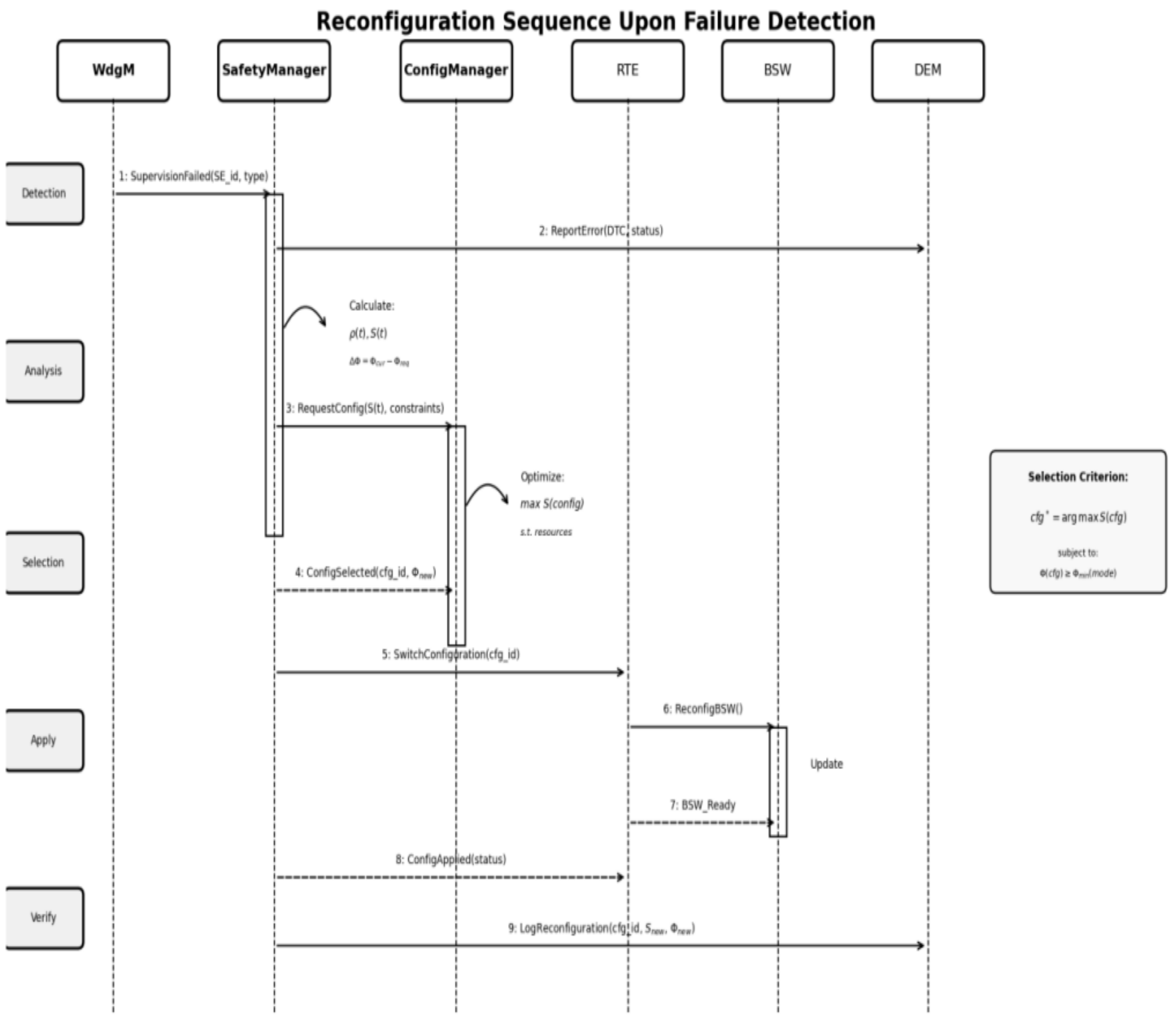

Reconfiguration Sequence Upon Failure Detection
WdgM
SafetyManager
ConfigManager
RTE
BSW
DEM
Detection
Analysis
Selection
Apply
Verify
1: SupervisionFailed(SE_id, type)
2: ReportError(DTC, status)
Calculate:
$\rho(t), S(t)$
$\Delta\Phi = \Phi_{cur} - \Phi_{req}$
3: RequestConfig(S(t), constraints)
Optimize:
max S(config)
s.t. resources
Selection Criterion:
$cfg^* = \arg\max S(cfg)$
subject to:
$\Phi(cfg) \geq \Phi_{min}(mode)$
4: ConfigSelected(cfg_id, Φ_{new})
5: SwitchConfiguration(cfg_id)
6: ReconfigBSW()
Update
7: BSW_Ready
8: ConfigApplied(status)
9: LogReconfiguration(cfg_id, S_{new}, Φ_{new})

Fig. 4. Reconfiguration sequence upon failure detection

PRACTICAL IMPLEMENTATION IN AUTOMOTIVE SYSTEMS

5.1. Reconfiguration-Enabled Architecture

Practical implementation of the hierarchical survivability model is based on Post-Build Configuration architecture, allowing storage of multiple configurations in NVM and switching between them at runtime without recompilation (Fig. 5) [1, 2].

The SurvivabilityManager component implements upper levels of the abstraction hierarchy and makes reconfiguration decisions based on WdgM (monitoring) data and resource state from HAL. ConfigurationManager is responsible for applying the selected

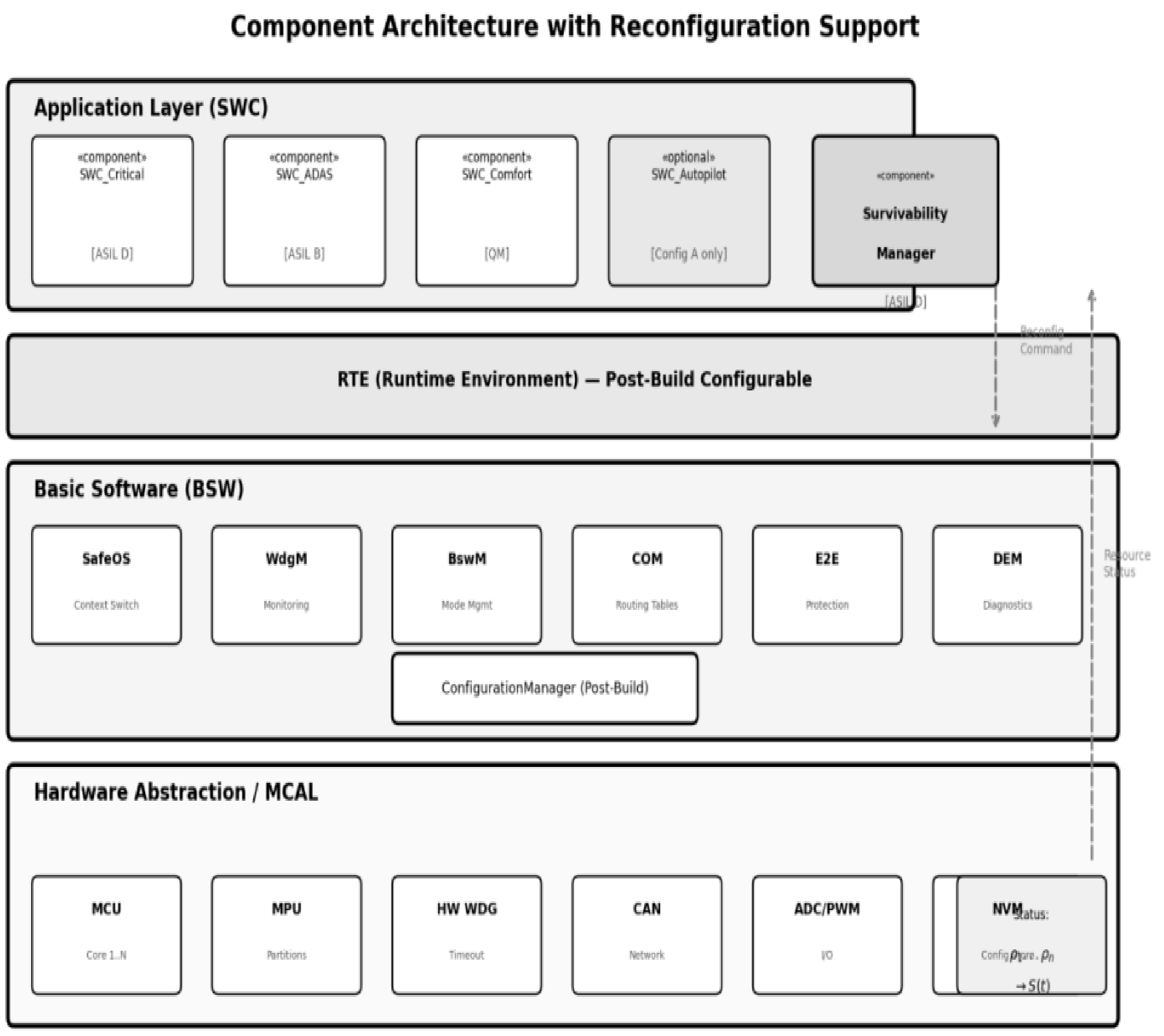

Component Architecture with Reconfiguration Support
Application Layer (SWC)
«component» SWC_Critical [ASIL D]
«component» SWC_ADAS [ASIL B]
«component» SWC_Comfort [QM]
«optional» SWC_Autopilot [Config A only]
«component» Survivability Manager [ASIL D]
Reconfig Command
RTE (Runtime Environment) — Post-Build Configurable
Basic Software (BSW)
SafeOS Context Switch
WdgM Monitoring
BswM Mode Mgmt
COM Routing Tables
E2E Protection
DEM Diagnostics
ConfigurationManager (Post-Build)
Resource Status
Hardware Abstraction / MCAL
MCU Core 1..N
MPU Partitions
HW WDG Timeout
CAN Network
ADC/PWM I/O
NVM Config Store
Status: $\rho_1 \ldots \rho_n$ → $S(t)$

Fig. 5. Component architecture with reconfiguration support

configuration through BswM, which controls BSW module operating modes.

5.2. Survivability System Class Model

The software model of the survivability system (Fig. 6) includes classes: SurvivabilityManager (survivability management), Resource (resource model with ρ coefficient), SystemFunction (function with weight and dependencies), Configuration (system configuration), ConfigSelector (optimal configuration selection), and LyapunovAnalyzer (stability analysis).

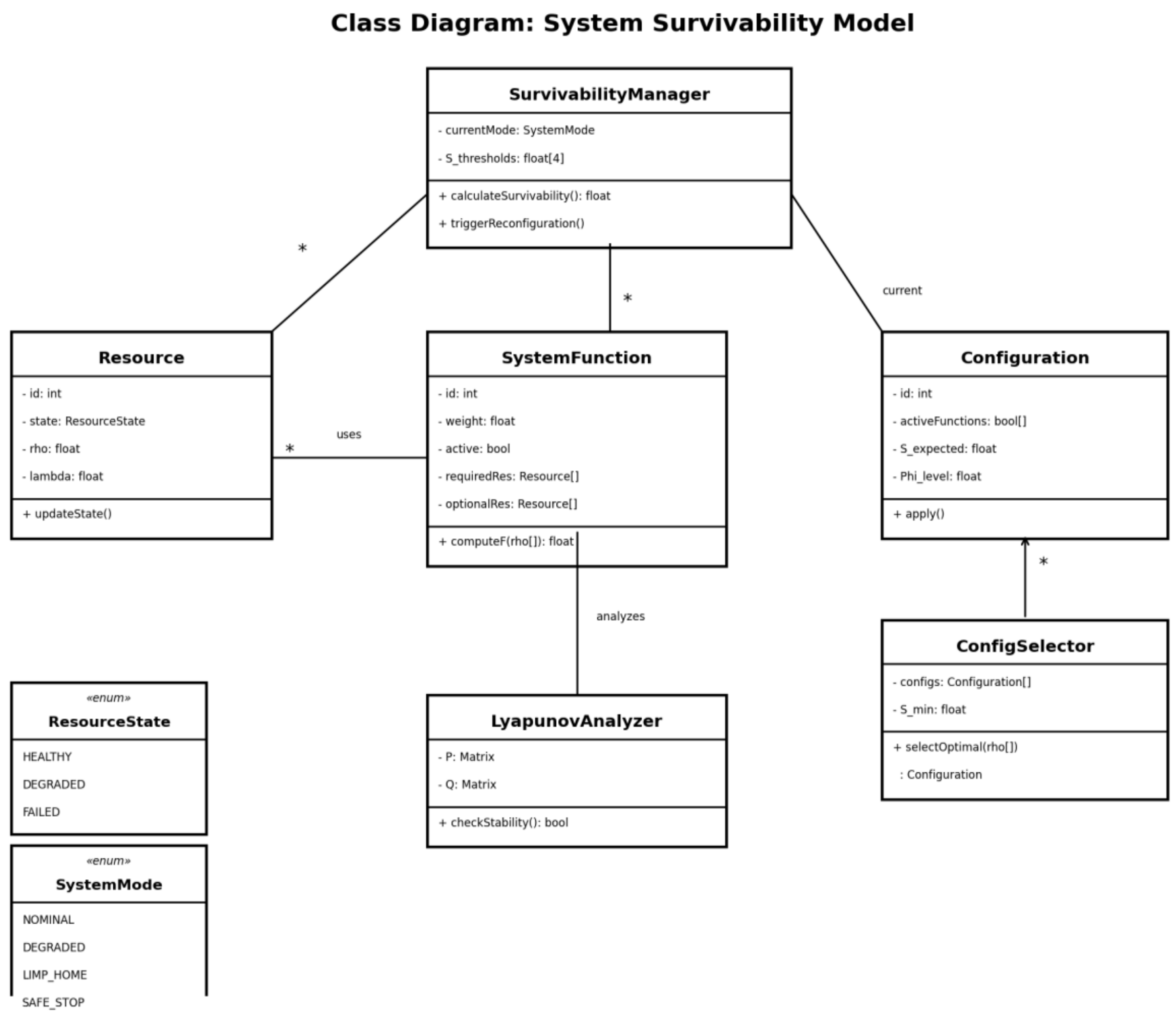


Fig. 6. Survivability system class diagram

5.3. Rapid Reconfiguration Mechanisms

To ensure minimum reconfiguration time, the following mechanisms are used:

– Post-Build Selectable Configuration — pre-compiled configurations stored in NVM and activated through BswM;

– Hot Standby Functions — backup functions maintained in ready state for instant activation;

– Partition Restart — ability to restart individual OS-Applications without affecting critical functions;

– Pre-computed Configurations — table of optimal configurations for typical degradation scenarios computed offline.

STABILITY AND CONVERGENCE ANALYSIS

To prove system stability in post-critical state, the Lyapunov function $V(x) = x^T P x$ is used, where x is the deviation vector from the target state [4]. The system is asymptotically stable if there exists a configuration cfg* for which the derivative $dV/dt < 0$. The condition for existence of such a configuration:

$$\exists cfg: S(cfg \mid \rho) \geq S\_min \wedge \Phi(cfg) \geq \Phi\_min(mode), \tag{6}$$

If the condition is not satisfied for any configuration, the system transitions to SAFE_STOP state with functionality Φ_emergency, guaranteeing minimum safety.

CONCLUSIONS

This paper presents a comprehensive approach to ensuring survivability of complex super-critical systems in post-critical state. Main results:

1. A mathematical model of system survivability $S(t) = \Sigma_j w_j \cdot f_j(\rho) \cdot \delta_j$ has been developed, integrating resource states, functions, and their weight coefficients, allowing quantitative assessment of system ability to execute its mission.

2. A four-level abstraction hierarchy (Mission → Function → Service → Resource) is proposed, enabling decomposition of the reconfiguration task and encapsulation of implementation details.

3. An algorithm for optimal configuration selection $cfg^* = \text{argmax } S(cfg)$ subject to minimum functionality constraints $\Phi(cfg) \geq \Phi\_min$ has been developed.

4. Practical implementation based on Post-Build Configuration architecture for automotive ECUs with rapid configuration switching support is described.

5. System stability conditions in post-critical state based on Lyapunov functions are proven, guaranteeing convergence to a stable mode or safe transition to SAFE_STOP.

REFERENCES

1. **ISO 26262:2018**. Road vehicles — Functional safety. International Organization for Standardization.
2. **AUTOSAR**. Specification of Basic Software Mode Manager. Release R22-11.
3. **Humennyi D., Humennyi O.** (2023). Established Definitions of Super-Critical Operational Modes // Pidvodni Tehnologii. — 2023. — Vol. 13(1).
4. **Leveson, N.G.** (2016). Engineering a Safer World: Systems Thinking Applied to Safety; MIT Press: Cambridge, MA, USA, 2016.
5. **A. Avizienis, J. . -C. Laprie, B. Randell and C. Landwehr**. (2004). Basic concepts and taxonomy of dependable and secure computing," in IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, Jan.-March 2004, doi: 10.1109/TDSC.2004.2.
6. **Knight, John C.** (2002). Safety critical systems: challenges and directions." Proceedings of the 24th international conference on software engineering. 2002.
7. **Shelton, Charles P., and Philip Koopman**. (2004). Improving system dependability with functional alternatives." International Conference on Dependable Systems and Networks, 2004. IEEE.
8. **Ellison, R. J., Fisher, D. A., Linger, R. C., Lipson, H. F., & Longstaff, T.** (1997). Survivable network systems: An emerging discipline (No. CMUSEI97TR013).
9. **W. Heimerdinger, and C. Weinstock.** (1992). A Conceptual Framework for System Fault Tolerance," Carnegie Mellon University, Software Engineering Institute's Digital Library. Software Engineering Institute, Technical Report CMU/SEI-92-TR-033, 1-Feb-1992 [Online]. Available: https://www.sei.cmu.edu/library/a-conceptual-framework-for-system-fault-tolerance/. [Accessed: 22-Dec-2025].
10. **Laprie, Jean-Claude.** (1992). Dependability: Basic concepts and terminology." Dependability: Basic Concepts and Terminology: In English, French, German, Italian and Japanese, Vienna: Springer Vienna, 3-245.

Забезпечення живучості складних супер-критичних систем на основі моделі ієрархічної абстракції та адаптивної реконфігурації у післякритичному стані

Дмитро Гуменний[1]

[1]*Київський національний університет будівництва і архітектури*

Анотація. У даній роботі представлено новий концепт забезпечення живучості складних суперкритичних систем у посткритичному стані на основі чотирирівневої ієрархії абстракції та механізму адаптивної реконфігурації. Розроблено математичну модель функції живучості S(t), що інтегрує стани ресурсів, активності функцій та їх вагові коефіцієнти. Запропоновано алгоритм вибору оптимальної конфігурації системи під час деградації компонентів, що максимізує живучість за умов мінімальних функціональних обмежень. Описано практичну реалізацію механізмів швидкої реконфігурації на основі архітектури Post-Build Configuration для автомобільних ЕБК відповідно до ISO 26262.

Ключові слова: живучість системи, посткритичний стан, реконфігурація, плавна деградація, ієрархія абстракції, ISO 26262, ASIL, функція Ляпунова, суперкритичні системи.