SUNSET - A Sensor-fUsioN based semantic SegmEnTation exemplar for ROS-based self-adaptation

Andreas Wiedholz[1,*], Rafael Paintner[2], Alwin Hoffmann[1], Carlos Hernandez[3] and Tobias Huber[1]

[1]XITASO GmbH, Augsburg, Germany
[2]German Aerospace Center (DLR) Institute of Flight Systems
[3]TU Delft, Department of Mechanical Engineering

Abstract

The fact that robots are getting deployed more often in dynamic environments, together with the increasing complexity of their software systems, raises the need for self-adaptive approaches. In these environments robotic software systems increasingly encounter (1) failures whose symptoms are easy to observe but root causes might be ambiguous or (2) multiple failures appearing concurrently. We present SUNSET, a ROS2-based exemplar that enables rigorous, repeatable evaluation of architecture-based self-adaptation in such conditions. It implements a sensor fusion semantic-segmentation pipeline driven by a trained Machine Learning (ML) model whose input preprocessing can be perturbed to induce realistic performance degradations. The exemplar exposes five observable failures, each of which can be caused by different faults and supports concurrent failures spanning self-healing and self-optimisation. SUNSET includes the segmentation pipeline, a trained ML model, fault-injection scripts, a baseline controller for further comparisons, and step-by-step integration and evaluation documentation to facilitate reproducible studies. The code is available at https://github.com/XITASO/sunset.

Keywords

Self-adaptation, ROS, exemplar, sensor fusion

1. Introduction

Robotic systems increasingly operate in dynamic, unmapped environments (e.g., Unmanned Aerial Vehicle (UAV)), which raises the need for adaptation. The research area of Robotics Software Architecture-based Self-adaptive Systems (RSASS) aims to deal with the challenges accompanying this development by adapting the software during runtime [1].

It is often much easier to detect failures of a system than their causes, which might be be an internal fault or an external perturbation (both of which we will refer to as "fault" in the remaining paper for simplicity). However, since the true root cause of the failure is often unknown, this raises uncertainty in the overall system. We focus on two challenging cases that are particularly relevant for robotic systems: (1) symptoms of failures that do not map cleanly to a single root cause, and (2) concurrent failures that appear together — either raised independently by multiple faults or as ripple effects. While general self-adaptive systems have studied concurrent failures[3], RSASS lacks exemplars that evaluate self-adaptivity under concurrent, cause-ambiguous failures.

To fill this gap, this paper introduces SUNSET — an exemplar which implements a Sensor fUsioN based Semantic sEgmentaTion pipeline (see Figure 1). To make SUNSET suitable for evaluating self-adaptive approaches in robotics, we chose a common and realistic robotic task (semantic segmentation) [4] and implement it in the Robot Operating System (ROS) the main technology used for RSASS[1]. SUNSET supports external self-adaptation with a clearly separated managed and managing system [5]. This separation simplifies the integration of different managing systems on the same managed system and follows common practice in robotic and ROS-based self-adaptivity exemplars [6, 7, 8].

RoSE'26: International Workshop on Robotics Software Engineering
*Corresponding author.
andreas.wiedholz@xitaso.com (A. Wiedholz)
0000-0003-0118-8765 (A. Wiedholz)

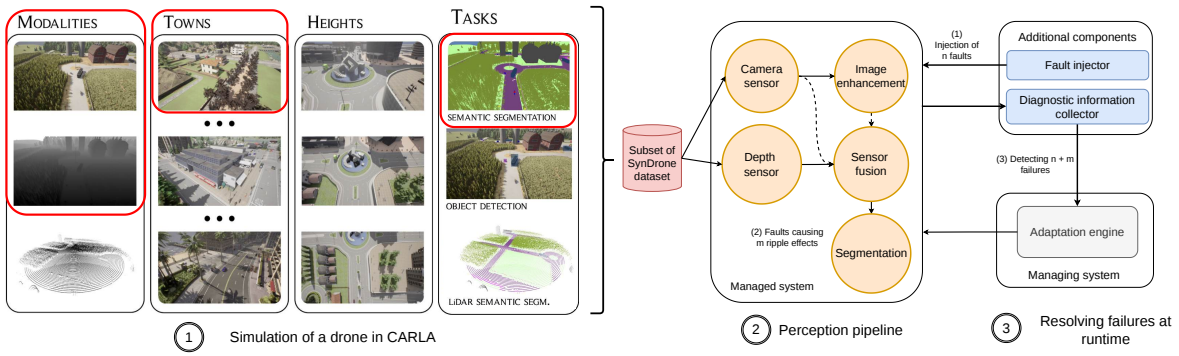


Figure 1: SUNSET - A sensor fusion based semantic segmentation pipeline based on the SynDrone dataset[2] with a fault injector for testing self-adaptive ROS-based systems. During experiments multiple faults will be injected which may cause ripple effects. The managing system detects multiple failures at the same time and has to adapt the managed system accordingly.

For the semantic segmentation, SUNSET implements a sensor-fusion-based approach that uses a real machine-learning model whose inputs can be perturbed to produce realistic degradations in performance. Moreover, the exemplar supports the four main types of architectural adaptations common in RSASS[1] which are aimed at either self-optimization or self-healing, i.e., more critical failures, capabilities.

This combination of realistic ROS2-based robotics software integration, rich adaptation capabilities, and concurrent failures with unknown sources makes SUNSET a novel and challenging exemplar for evaluating RSASS approaches. To summarize, our work offers the following contributions:

- An exemplar in which multiple faults can lead to the same failure requiring root cause analysis.
- The first ROS-based exemplar that allows all common adaptations for self-adaptive robotic applications to be performed by a managing system to resolve concurrent failures.
- A ML model offering realistic data for degraded performance.
- An exemplar with detailed documentation how to evaluate new managing systems with an exemplary baseline.

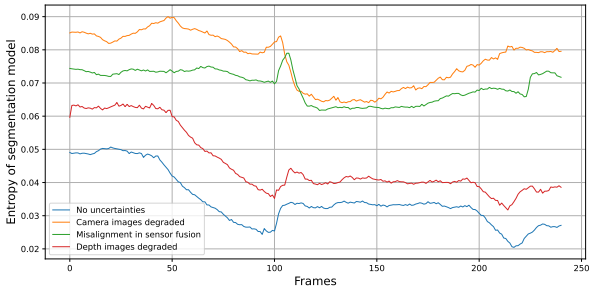
2. Related work

RSASS research has been present for at least 15 years with ROS being the main common ground in this field. Many of these approaches are focused on their respective use case, causing many evaluation scenarios to be tailored to the proposed managing system itself [1]. This makes it hard to prove that one method can be implemented for multiple use cases. Therefore, more exemplars allowing ROS-based self-adaptive approaches to be evaluated on are required.

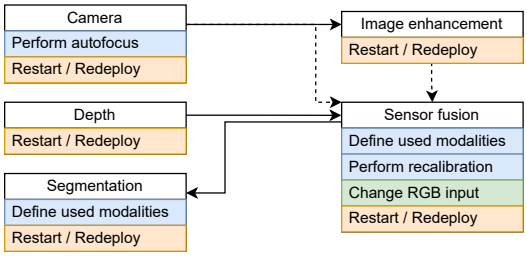
A detailed overview of self-adaptive robotic exemplars is given in Robomax [9] which contains 12 exemplars of different robotic applications and details of the robotic mission and the environment. Apart from the Body Sensor Network [6] — a patient health status monitoring application implemented in ROS1 — the exemplars focus on theoretical scenarios instead of empirical exemplars.

Silva et al. [7] introduce SUAVE which simulates a ROS2-based underwater robot with the mission to detect and inspect a pipeline. In the extended version, SUAVE contains 3 failures, namely environmental factors, the need to recharge a battery or the breakdown of one of the robot's thrusters. All of the failures in SUAVE occur sequentially and have a clearly defined adaptation spanning self-healing and self-optimization concerns that resolves it. The main difference to SUNSET is that in SUAVE the adaptation to resolve a detected failure is always the same.

An exemplar for drone control implemented in ROS1 has been implemented by Imrie et al. [8]. The use of Gazebo allows a physically accurate simulation and contains physical concerns as well as time-related concerns raising the need for self-adaptation. This can be related to failing components in the system or environmental factors necessitating changes in the plans.



(a) The effect of different faults on the entropy of our segmentation model. This refers to failure F4.



(b) Potential adaptations in SUNSET. Blue := Reparametrization, Green := Change of communication, Orange := Software/Hardware restart

Figure 2: Overview of the characteristics of SUNSET

To the best of our knowledge, SUNSET is the first ROS-based exemplar that supports concurrent failures with unknown causes necessitating a managing system to prioritise between detected failures. It is implemented in ROS2 instead of the majority of the ROS1-based related work [9, 8, 6] and enables managing systems to use all adaptations that are common in RSASS [1]. This enables robotics research to evaluate managing systems with enhanced capabilities on an independent exemplar.

3. SUNSET

3.1. Use-Case Description

SUNSET models the perception of an UAV that must semantically understand its surroundings during a mission — a core capability in many modern robotic systems [4]. As data basis for our exemplar, we utilise the SynDrone Dataset (Town 01)[2], which simulates a UAV in a CARLA simulation containing both urban and rural scenes (see fig. 1). The UAV is equipped with RGB and depth cameras, and the resulting data is continuously fused and interpreted by a semantic segmentation model to support tasks such as safe navigation and environment mapping.

3.2. System Architecture

The ROS architecture of the semantic segmentation pipeline in SUNSET consists of standard components for sensor fusion (see Figure 1). A **camera sensor** and **depth sensor** node read RGB and depth images from a ROS bag that contains pre-recorded flight data and provide them to the rest of the pipeline. Using pre-recorded data rather than live simulations improves the replicability of SUNSET experiments, as the same sensor inputs can be replayed across different runs and configurations. SUNSET also entails an **image enhancement** node that is able to reverse a degradation of RGB images. The two data modalities are aligned by a **sensor fusion** node that matches RGB–D pairs based on the timestamps of the received images. The fused (or single-modality) images are then processed by a **segmentation** node that performs semantic segmentation. For each configuration (RGB, Depth, RGB+Depth), we train a separate ML model from scratch on the SynDrone dataset[2], while keeping the rest of the mission scenario unchanged. For the model architecture, we use the ResNet50 based Deeplab-V3 model with the late fusion strategy tested by Rizzoli et al. [2].

Alongside the segmentation pipeline, a **fault injector** introduces faults (see Section 3.3) directly into the ROS nodes. Additionally, we provide a ROS node that collects information needed by the managing system to detect failures and therefore the necessity of an adaptation of the managed system, e.g., degraded performance of the segmentation model. For the managing system we propose a baseline (see Section 4), which can be substituted with any other ROS2-based managing system.

3.3. Faults

Our fault injector can introduce 11 faults with every fault being the reason for a failure. The faults in SUNSET result in 5 different failures (*F1-5*) that require self-healing (*F1-3*) and self-optimising (*F4-5*) capabilities. Each failure can be solved with at least one of the possible adaptations. Resolving all injected faults requires all of the adaptations our system is capable of (see Section 3.4). Categorised by their failure, the introduced faults are:

F1-3: Communication channel outage These failures can be produced by the RGB camera (*F1*), sensor fusion (*F2*), or the segmentation node (*F3*) and can be detected by monitoring the frequency of their respective topic. For these failures, SUNSET simulates 6 faults: hardware disconnect of the camera, issues in the driver, memory leakage or synchronization issue in the sensor fusion or GPU failure and CUDA out of memory exception in the segmentation node. All of these faults require different adaptations — lifecycle restart or node redeploy — but result in the same kind of failure in the system.

F4: Reduced segmentation performance Since we use a trained ML model for segmentation, we are able to measure realistic effects on the uncertainty of the model in its predictions during runtime when degrading the input. We achieve this by calculating the mean entropy of the models' segmentation logits. If the entropy increases, i.e., the model gets uncertain about its predictions, the quality of the segmentation decreases. As the segmentation is dependent on multiple nodes, SUNSET simulates four faults that can lead to this symptom (see Figure 2a). First, a degraded image quality is simulated by shifting the colour space of the images provided to the camera node. Second, if the image enhancement is applied on non-degraded images, it leads to degraded images and has the same effect on the entropy of the segmentation model. Third, we simulate misalignment during fusion of the RGB and depth images by shifting one of the images, which results in a spatial misalignment between RGB and depth. Lastly, the depth images can be degraded with Gaussian noise.

F5: Blurred camera image An incorrect camera focus is simulated by blurring the image.

3.4. Possible adaptations

In order to resolve failures in SUNSET, i.e., identifying the fault and recovering from it, we can perform the four main adaptations in RSASS[1] which are described in the following. Figure 2b gives an overview of all the possible adaptations per ROS node in SUNSET that can be used to resolve runtime failures. In the following we detail the behaviour of the ROS nodes in SUNSET during an adaptation.

Reparametrisation In order to reparametrise a ROS node, we use the standard ROS parameters. Additionally to setting a new value, these parameters can be used in some nodes, e.g., the camera or sensor fusion, to perform an action aimed for self-optimisation when set to True.

Change of communication The ROS node can change any of its subscribed topics at runtime in the following way: After the managing subsystem provides the name of the new topic, the adapted ROS node destroys its current subscription and creates a new one with the same message type.

Addition / removal of components Since our common base class used for all ROS nodes in SUNSET inherits from the Lifecycle node class, all nodes can be within four predefined lifecycle states during runtime. We consider a node added to the system if it is in the *ACTIVE* state and removed if it is in any other state. Depending on the injected fault, *F1-3* can be resolved by restarting the respective node, i.e., removing and adding it to the system.

Redeploy If a node is redeployed, the ROS node with the OS process is completely terminated simulating a hardware restart. To simulate this, an artificial delay is introduced when redeploying the respective nodes, e.g. camera. In general, redeploying a node typically resolves all ROS node internal fault; however failures with external causes, e.g., image degradation remain persistent.

Table 1
Baseline performance of SUNSET. Directional arrows specify metric desirability.

Managing system	$\frac{f_{resolved}}{s_{executed}} \uparrow$	$t_{react}[s] \downarrow$	#redeploy$_u \downarrow$	$t_{down}[s] \downarrow$	IoU ↑
None w/o faults	N/A	N/A	N/A	N/A	0.47 ± 0.02
None w/ faults (F4 and F5)	N/A	N/A	N/A	N/A	0.28 ± 0.08
Baseline	0.89 ± 0.16	1.44 ± 1.32	2.02 ± 1.04	6.52 ± 4.67	0.30 ± 0.09

3.5. Expanding SUNSET

Adding new image domains to SUNSET is straight forward. For example, for the SynDrone dataset we already provide a script that can add additional video sequences from different towns and heights (see Figure 1). SUNSET can also be extended to other common robotic tasks like object detection, LiDAR-based segmentation or even control with the original SUNSET embedded as perception part of a complete robotic system. To facilitate the implementation of new nodes, we provide a base class that extends the ROS lifecycle node class and implements the behaviour of all described adaptations.

4. Evaluation

To measure the performance of the managing system and the impact on the managed system, we provide several metrics which we calculate based on log files. The performance of the managed system can be measured by the quality of the segmentation model's predictions (Intersection over Union (IoU)) over all frames and the availability of the overall managed system. For the availability of the managed system, we introduce the system downtime that accumulates the time that the segmentation algorithm didn't send new data in the expected frequency. This indicates that one component had an outage. For context, one experiment runs for 20 seconds, i.e., a downtime of 5 seconds would result in a system availability of 75%. Other metrics we provide solely focus on the performance of the managing system:

- Ratio of resolved failures and executed adaptations $\frac{f_{resolved}}{a_{executed}}$: How many failures were resolved per executed system adaptation? This metric measures how many concurrent failures a single system adaptation solves.
- Reaction time t_{react}: Since for each detected failure, there are multiple faults potentially being responsible, this metric measures the time it took to detect a symptom and select the correct adaptation of the system.
- Number of unnecessary redeploys #redeploy$_{\text{u}}$: Number of redeploys that were executed even though a cheaper adaptation would have sufficed.

To provide first results for our metrics, we implement a managing system based on the MAPE-K feedback loop [5] with straightforward adaptations to the system. This can be used by other researchers as baseline to compare their managing system with. First, the managing system redeploys every node that stops sending data, i.e., $f_{\text{node}} == 0$ (*F1-3*). Second, it performs a recalibration in the sensor fusion node as soon as the entropy of the segmentation model is higher than 0.06 (*F4*, see Figure 2a).

The results of our baseline are presented in Table 1. In the first two rows, we present the results for running SUNSET without any faults and with faults present the whole time (except outages, therefore no t_{down}) to show the best and worst possible IoU. The other metrics are not applicable here since there is no downtime and no managing system present that can be evaluated. In every run one fault for self-healing (F1-3) and two faults for self-optimization (F4-5) are injected simultaneously. To ensure reproducibility, 54 scenarios were created containing every possible combination of these faults. The last row shows the baseline that ran every scenario three times. By design the baseline is not able to detect three out of four faults for *F4* which results in a high standard deviation for the IoU. As the baseline redeploys every node in which *F1-3* is detected, $\#redeploy_u$ has a high value.

5. Conclusion & Future work

In this work, we present SUNSET an exemplar for evaluating managing systems for self-adaptive robotic systems. Our exemplar contains a segmentation pipeline with a real ML model and different failures causing degraded performance of the model or entire system outages. In SUNSET, multiple faults can (1) cause the same failure and (2) occur simultaneously, necessitating a deeper analysis of the system. Therefore, SUNSET presents a novel and challenging benchmark for evaluating RSASS.

Acknowledgments

The authors would like to thank the Federal Ministry of Economic Affairs and Climate Action of Germany for funding the described activities through the LuFo VI-3 program, funding code 20F2201D.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] E. Alberts, I. Gerostathopoulos, I. Malavolta, C. Hernández Corbato, P. Lago, Software architecture-based self-adaptation in robotics, Journal of Systems and Software 219 (2025) 112258. doi:10.1016/j.jss.2024.112258.

[2] G. Rizzoli, F. Barbato, M. Caligiuri, P. Zanuttigh, SynDrone – multi-modal UAV dataset for urban scenarios, in: 2023 IEEE/CVF international conference on computer vision workshops (ICCVW), Los Alamitos, CA, USA, 2023, pp. 2202–2212. doi:10.1109/ICCVW60793.2023.00235.

[3] T. Vogel, mRUBiS: an exemplar for model-based architectural self-healing and self-optimization, in: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 101–107. doi:10.1145/3194133.3194161.

[4] M. Tzelepi, A. Tefas, Semantic Scene Segmentation for Robotics Applications, in: 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), 2021, pp. 1–4. doi:10.1109/IISA52424.2021.9555526.

[5] J. Kephart, D. Chess, The vision of autonomic computing, Computer 36 (2003) 41–50. URL: https://ieeexplore.ieee.org/document/1160055. doi:10.1109/MC.2003.1160055.

[6] E. B. Gil, R. Caldas, A. Rodrigues, G. L. G. da Silva, G. N. Rodrigues, P. Pelliccione, Body Sensor Network: A Self-Adaptive System Exemplar in the Healthcare Domain, in: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2021, pp. 224–230. doi:10.1109/SEAMS51251.2021.00037, iSSN: 2157-2321.

[7] G. R. Silva, J. Päßler, J. Zwanepol, E. Alberts, S. L. T. Tarifa, I. Gerostathopoulos, E. B. Johnsen, C. H. Corbato, SUAVE: An Exemplar for Self-Adaptive Underwater Vehicles, in: 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2023, pp. 181–187. doi:10.1109/SEAMS59076.2023.00031, iSSN: 2157-2321.

[8] C. Imrie, R. Howard, D. Thuremella, N. M. Proma, T. Pandey, P. Lewinska, R. Cannizzaro, R. Hawkins, C. Paterson, L. Kunze, V. Hodge, Aloft: Self-Adaptive Drone Controller Testbed, in: 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '24, New York, NY, USA, 2024, pp. 70–76. doi:10.1145/3643915.3644107.

[9] M. Askarpour, C. Tsigkanos, C. Menghi, R. Calinescu, P. Pelliccione, S. García, R. Caldas, T. J. von Oertzen, M. Wimmer, L. Berardinelli, M. Rossi, M. M. Bersani, G. S. Rodrigues, RoboMAX: Robotic Mission Adaptation eXemplars, in: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2021, pp. 245–251. doi:10.1109/SEAMS51251.2021.00040, iSSN: 2157-2321.