Dynamic Aided-Design of UAV Navigation Systems

V.M. Sineglazov
Aviation Computer-Integrated Complexes Department,
Educational & Research Institute of Information and Diagnostic Systems,
National Aviation University
Kyiv, Ukraine
svm@nau.edu.ua

A.P. Godny
Aviation Computer-Integrated Complexes Department,
Educational & Research Institute of Information and Diagnostic Systems,
National Aviation University
Kyiv, Ukraine
andrewgodny@gmail.com

***Abstract*—Presented navigation system for unmanned aerial vehicles based on microservices with an integrated environment introduces a new approach to managing the design process. Used in the proposed medium design scenario can greatly simplify the work of the designer. Available in medium monitor provides the flexibility of design processes with a flexible structure description of design procedures in the design scenario.**

Keywords—unmanned aerial vehicles; dynamic integration; microservices; integrated environment; scheduling mechanism; design

I. Introduction

The general trend in the market development of navigational systems of mobile objects is such that developers are moving towards deepening integration between inertial, satellite and other systems under the impact of increasingly stringent requirements. At the same time, the International Civil Aviation Organization (ICAO) on Future Air Navigation System (FANS) recommends using on-board SNS with mandatory combination with the inertial navigation system as the central link of the navigation complex.

Currently, the design objectives "improving the accuracy and reliability of navigation parameters estimation" are achieved, as a rule, due to the use of autonomous navigation systems in the INS structure with higher resolution and accuracy. However, the equipment cost of autonomous navigation systems with improved characteristics and its manufacturing in mass-produced conditions of UAVs and, accordingly, INCs is sufficiently large.

Therefore, for INS, the most acceptable from the standpoint of realizing the design objectives of "improving the accuracy and reliability of navigation parameters estimation" and the "efficiency-cost" criterion is the use of data integration systems.

II. Problem Statement

Today to improve the accuracy and reliability of UAV's navigation system you need to create more robust navigation complex. New complex will be heavier and more expensive. And what more importantly, it will affect all other UAV's characteristics. On the other hand, you can combine couple UAV into one information system. This approach will keep both price and weight of this UAVs down.

So, it is necessary to create a dynamic informational system based on Microservices where each UAV will act as one micro service connected with others to form one dynamic navigation system so they can share data to solve navigation issues.

III. Microservices-Based Architectures

Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures a system as a collection of loosely coupled services.[6] In a microservices architecture, services should be fine-grained and the protocols should be lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity and makes the system easier to understand, develop and test. It also parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently. It also allows the architecture of an individual service to emerge through continuous refactoring. Microservices-based architectures enable continuous delivery and deployment. [7]

There is no industry consensus yet regarding the properties of microservices, and an official definition is missing as well. Some of the defining characteristics that are frequently cited include:

- Services in a microservice architecture (MSA) are often processes that communicate with each other over a network in order to fulfill a goal using technology-agnostic protocols such as HTTP. However, services might also use other kinds of inter-process communication mechanisms such as shared memory. Services might also run within the same process;
- Services in a microservice architecture should be independently deployable;
- The services are easy to replace;
- Services are organized around capabilities, e.g., user interface front-end, recommendation, logistics, billing, etc.;
- Services can be implemented using different programming languages, databases, hardware and software environment, depending on what fits best;
- Services are small in size, messaging enabled, bounded by contexts, autonomously developed,

978-1-5386-1816-5/17/$31.00 ©

independently deployable, decentralized and built and released with automated processes.

- A microservices-based architecture [6]:
- Naturally enforces a modular structure;
- Lends itself to a continuous delivery software development process. A change to a small part of the application only requires one or a small number of services to be rebuilt and redeployed;
- Adheres to principles such as fine-grained interfaces (to independently deployable services), business-driven development (e.g. domain-driven design), ideal cloud application architectures, polyglot programming and persistence, lightweight container deployment, decentralized continuous delivery, and DevOps with holistic service monitoring;
- Provides characteristics that are beneficial to scalability.
- The microservices approach is subject to criticism for a number of issues:
- Services form information barriers;
- Inter-service calls over a network have a higher cost in terms of network latency and message processing time than in-process calls within a monolithic service process;
- Testing and deployment are more complicated;
- Moving responsibilities between services is more difficult. It may involve communication between different teams, rewriting the functionality in another language or fitting it into a different infrastructure;
- Viewing the size of services as the primary structuring mechanism can lead to too many services when the alternative of internal modularization may lead to a simpler design.

Data integrity is a main problem with such approach that needs to be solved. With each microservice responsible for its own data persistence. As a result, data consistency can be a challenge. Embrace eventual consistency where possible. To solve this issue we developed new approach – dynamic data integration.

IV. Dynamic Data Integration

This approach implies the existence of a link between all modules. It manages all available components of the system, is responsible for communication between modules, data conversion (if necessary), quality control.

This approach allows to minimize the cost of adding new modules and upgrading of current, reduces maintenance costs of the whole system, and simplifies the management of data flow in the system. The advantages of this approach are obvious, but it requires the establishment of common rules for the interaction of all integrated components and create a unified reporting format to simplify the processes of interaction between different UAVs [4]. To solve these problems, you can apply the method of dynamic data integration.

The method of dynamic data integration developed to link different types of UAVs in a single information process. In addition, all considered UAVs have equal rights. The order is determined by the interaction of conditions and information processing requirements [1]. With dynamic data integration, system operates with commands forming the object parameters, and implements communication objects directly between the commands, while providing a more flexible way of combining and "understanding" of different types of data. In the system parameter of an object should not be separated from the command. The parameter is only a formal representation of data in the system.

Considered an integrated environment CAD has the following properties:

- Completeness and integrity of the design object descriptions, provides integrated transformations;
- Simplicity and convenience of operation of writing, design procedures, it builds descriptions of many alternatives – integrated project operations;
- Flexibility of design processes, provides a flexible framework of descriptions of design procedures that allows you to manage transitions scenario design and modify the contents of responses to possible events;
- A variety of classes of design operations in accordance with the level of complexity of project tasks and qualifications assure the combination of design operations and their use as a whole;
- Simplicity and convenience of operations for the joint processing of graphics, text and spreadsheet object descriptions;
- Support and combine object-oriented and subject-oriented descriptions of design processes with the ability to connect descriptions of the processes;
- Evolutionary development, provide feedback based on the logging of user actions used in conjunction with data and their further structuring;
- The accumulation of knowledge acquired for the subsequent synthesis of executable elements that allows developing evolutional system and configuring it to various classes of design objects;
- Simplicity and convenience of management conversational interaction provides a unified operations dialog interaction kernel environment.

V. Data Standardization

For the improvement of CADS functioning it is necessary to supply data standardization that includes such positions:

- As a standard message, body accepts standard XML.

- As a standard interface, description system accepts standard XSD.
- As a standard, modified standard XML messages accept XSLT.
- As a first access to the message body language, take XPATH.

These standards are well established and have wide support. [4]

In practice, not all systems support XML-interface for data exchange. In this case, it is necessary to use ASB transformation services message format in XML-messages.

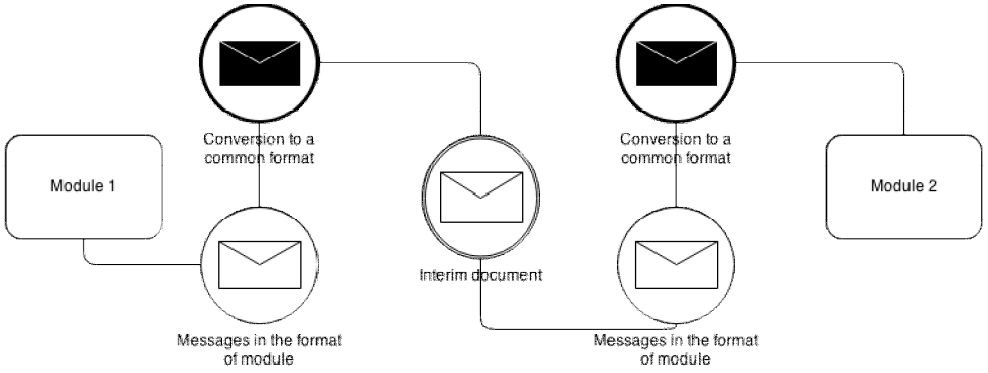

Fig. 1. Format conversion mechanism.

If C1 and C2 formats – not XML, converters may be a specific software product. Nevertheless, even in this case, the conversion from XML in a simple straightforward format (e.g., comma delimited) can be performed using XSLT-transformation. Steps to change formats you can easily make at the ASB level. Received by ASB data will be converted to the intermediate XML format, and the output will be converted to the format of the recipient. Application systems themselves do not need to be reworked in this regard, because all conversions are performed at the level of ASB.

If the format is XML, the conversion can be performed by means of XSLT transformations.

The most common mode of transmission was and is still is working through the files. Some systems for interaction do the following: perform a data dump of C1, file transports by mail or through the carrier and loads into C2, and all operations are performed manually. This process does not meet the requirements of modern business.

VII. Priority Inheritance Protocol

Since the navigation system has to guarantee the optimal use of computing resources and to ensure a minimum command processing time, system that organizes the task scheduler is required, which will be responsible for compliance with the required criteria of the navigation system. We propose to use Asymmetric preventive inheritance priority protocol (APIPP).

Operation of synchronizing mechanism, implemented in APIPP, characterized by the following provisions [8]:

Each resource is assigned with two threshold priorities: a threshold priority of readers and threshold priority of writers.

Ceil priority for readers used as meeting the objectives of readers' requests and is numerically equal to the priority of the task with the highest priority of those tasks that can capture this resource for writing:

$$ceil\_read_r = \max_{\{i|r \in modified\_by(i)\}} pri_i.$$

Ceil priority for writers is used while satisfying the query of task-writer and is numerically equal to the priority of the task with the highest priority of those tasks that can that can capture this resource for reading:

$$ceil\_write_r = \max_{\{i|r \in used\_by(i)\}} pri_i.$$

Task τ, which has the highest priority among all the active tasks, takes control. Before entering the critical section in relation to the resource r, task τ must capture the resource for reading, if it does not modify data, or for writing, if it would modify the data.

Task τ is performed with a base priority only if it has no shared resources. Otherwise, its priority is the greatest ceil priority among all ceil priorities captured its shared resources:

$$effective\_pri_i = \max_{\{r,t|(r,t) \in got\_by(i)\}} ceil_{r,t}.$$

When shared resources are released task τ gets base priority back.

Task τ_1 can supplant task τ_2 only if τ_1 priority strictly greater than the effective priority τ_2.

The task can't be completed or voluntarily suspend execution until the release of all occupied resources.

Critical sections are nested, i.e., shared resources are released in reverse order to their capture (stack).

A. The properties of the protocol

The properties Adirondack Park Invasive Plant Program's (APIPP) coincide with those of the original PIP and APIP. Therefore, we confine ourselves to the following list of APIPP advantages:

1) APIPP eliminates the possibility of deadlocks.
2) APIPP eliminates multiple blocks.
3) APIPP eliminates composite blocks.
4) APIPP reduces the number of task switches.
5) APIPP allows tasks to be performed in one stack mode.
6) APIPP suitable for synchronizing with interrupt handlers.
7) APIPP more effective than PIP and APIP.

For these reasons, in practice it is preferred to use APIPP.

B. *Example of using the protocol*

There are four tasks in the system: τ_1 with the highest priority, τ_2 medium priority, τ_3 low and τ_4 the lowest priority; and a shared resource *r*. Task τ_2 tries to get resource *r* for writing, other tasks are trying to get it for reading. Behavior of the system in the case of APIPP is shown on Fig. 2.

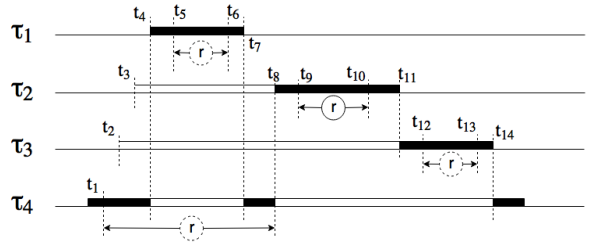

Fig. 2. Example of APIPP.

According APIPP, ceil priorities of resource are set as follows: ceil priority for readers *r* established at the level of the priority of task τ_2 (writer with the highest priority), ceil priority for writers *r* established at the level of the priority of task τ_1 (reader with the highest priority).

Task τ_4 gets resource *r* for reading (t_1). Then more priority-reader task is generated τ_3 (t_2). However, the task switching does not occur, because at this time priority of the task τ_4 equal to a ceil priority of the readers who captured resource, i.e. to the priority of tasks τ_2. Such a preventive blocking of tasks-readers, whose priority is less than the ceil priority of readers avoids composite blocks. The fact that the tasks which are using resource that is captured by other task, do not get control before its release (rather than blocking) reduces the number of task switches and allows to perform all tasks via single stack. Then higher priority task-writer is generated τ_2 (t_3). However, the task switch does not occur again due to the same circumstances. This way of organizing mutual exclusion mode prevents multiple blocking of tasks and even more so - eliminates the possibility of a deadlock. Next, task τ_4 is superseded by task τ_1 (t_4), which has been successfully performed (t_7) using the resource *r* (from t_5 to t_6) for reading, as the priority τ_1 is strictly greater than the ceil priority of *r* readers. As a result of this overlapping of critical sections of two tasks-readers we accomplished an increase of the efficiency of planning. Next, task τ_4 releases *r* (t_8). At this point, the most priority task among proactively blocked (τ_2) is unlocked and completed successfully (t_{11}) using the resource *r* for writing (from t_9 to t_{10}). Thereafter, control is passed to the task τ_3, which also completed successfully (τ_{14}) using the resource *r* (from τ_{12} to τ_{13}). Control again is passed to the low priority task τ_4.

VII. Conclusions

Presented design for UAV navigation system based on microservices with an integrated environment introduces a new approach to managing the navigation process. Used in the proposed medium design scenario can greatly simplify the work of the designer. Available monitor provides the flexibility of design processes with a flexible structure description of design procedures in the scenario design. Properties listed above for such system, coupled with the ability to integrate data of various aspects of presentation in a single information process, this system is isolated in a special class of software employed to integrate heterogeneous data.

References

[1] K. Lee, CAD Basics (CAD/CMA/CAE), Peter Press, 2004.
[2] V.V. Kupriyanov, O.Y. Pechenkin, M.L. Suslov, CAD and artificial intelligence systems, ROSNY and IT UP: DBMS, 1995. (in Russian)
[3] G. Berezhnoj, “Problems building large IT systems”, PCworld, 1998. (in Russian)
[4] V.M. Sineglazov and A.P. Godny, “Dynamic data integration in the design of complex computer-aided design systems,” Electronics and Control Systems 2014, no. 2(40), ISSN 1990-5548, pp. 51-58.
[5] Kristi Morton. Dynamic Workload Driven Data Integration U. of Washington, 2012.
[6] Richardson, Chris. Microservice architecture pattern. 2017.
[7] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, Microservice Architecture: Aligning Principles, Practices, and Culture, O’Reilly 2016.
[8] V.M. Sineglazov and A.P. Godny, Integrated Computer-Aided Design System Software of Navigation Complex International conference “Computer Algebra and Information Technology,” Kyiv, October 2016, pp. 59-62.