International Journal of Software Engineering and Knowledge Engineering
 World Scientific Publishing Company

Reuse-based Agile Development Process for Drone Software Systems

Mahmoud Hussein[1] and Réda Nouacer[2]
[1]*Faculty of Computers and Information, Menofia University, Egypt*
[2]*Université Paris-Saclay, CEA, LIST, Software and System Engineering Department (DILS), F-91120, Palaiseau, France*
mahmoud.hussein@ci.menofia.edu.eg and reda.nouacer@cea.fr

Drones are able to perform air operations that are difficult to execute using manned aircrafts. The use of drones in different domains brings significant environmental benefits and economic savings while reducing risks to human life. Recently, a number of approaches have been introduced to support the development of drone software systems. However, developing a customized drone software system based on end-user needs is still a time consuming process. Such delay in software production does not match end-users expectations. Therefore, in the COMP4DRONES project, we propose an agile-development process that is based on reuse to shorten the drone software development. In this process, based on the user requirements, a set of reusable components are selected from a repository that matches the requirements. These components are then integrated to have a fully functioning drone system. This repository will be filled by reusable components that are going to be developed during the COMP4DRONES project as key enabling technologies for drones.

Keywords: Drones, Software Development, Agile Methodology, DO-178C, Software Reuse

1 Introduction

Drones are able to perform air operations that cannot to be executed using manned aircrafts. The use of drones in different domains (e.g. transport, constructions, logistics, etc.) brings significant environmental benefits, and economic savings while reducing risks to human life [1]. In recent years, drones have reached millions of sales in the leisure market. Therefore, evolution on enabling technologies, regulation rules, and society acceptance need an accelerated development of drone systems for professional applications [2] [3].

The development of drone avionics is driven by certification requirements as indicated in the DO 178C standard [4]. Thus, a development lifecycle following this standard has a clear separation between phases with dedicated reviews and Stages of Involvement (SOI) to be acceptable by certification authorities. These reviews and SOIs ensure that the transition criteria for next phase are achieved (i.e. quality gates). However, the key problem of the DO-178C is that software errors that are detected late during the development lifecycle leads to much re-work which is costly and leads to many delays in the software delivery. In addition, a software project that needs to comply with the DO-178C could see a cost increase from 25 to 40 percent compared to a project that does not require such compliance [5].

The solution to the above problems can be provided through a careful deployment of agile methodologies [6]. Within agile process, the size of software increments is reduced, so that they can be implemented through one sprint. It also promotes close collaboration between all stakeholders, while ensuring that the certification objectives are met. Following this solution, a number of approaches in avionics domain have been introduced [7] [8] [9] [10]. These approaches provide a good step towards speeding up the development process and reducing the time to market while considering the drone safety.

In the COMP4DRONES (C4D, for short) project, the aim is to provide a framework of key enabling technologies for safe and autonomous drones [11]. In particular, the project is leveraging modularity and composability concepts to develop customizable and trusted autonomous drones for different civilian services. The project is taking into account recent regulation development [12]. To support the proposed project framework, an engineering methodology is provided. The methodology is based on reusability and agility to go a step ahead of existing approaches for speeding the system development and its qualification.

In our proposed development process, after the planning and requirements identification phases, a repository that contains software components is checked to identify COTS (Commercial off-the-shell) components that exist and can be used to satisfy the end user requirements (this can be done by the use of assume-guarantee contracts approach [13]). In case of such a COTS component exists, the development process starts from the integration phase. Otherwise, the full development cycle needs to be followed from design to delivery. The main idea of this process is to speed up the development process through reusing existing components that support the end user requirements. This reuse-based strategy is also adequate to smooth the qualification process when used with the concept of a dependability certificate that contains all information on the dependability attached to the reusable components [14] or the concept of Safety Element out of Context (SEooC) development [15].

In the C4D project, we have identified three technologies groups: system functions, payloads, and tools [16]. There are also technologies needed to support the U-space services that enable secure, safe, and efficient access to airspace by drones. These technological groups in a form of reusable components will be developed during the C4D project to feed the repository of COTS components and support the proposed process.

The remaining of the paper is organized into five sections. Section 2 reviews existing approaches for the drone software development. In Section 3, we present our proposed reuse-based agile process to shorten the development time. The reusable components that are going to be developed within the C4D project are listed in Section 4. The expected gain from the proposed C4D process is presented in Section 5. Finally, conclusions and future work are put in Section 6.

2 Drone Software System Engineering

To develop a drone software and enable its qualification, a process that can be followed is the development lifecycle described in the DO-178C standard (see Figure 1) [4]. This process recommends the use of the waterfall model. In this model, each phase of the development is finished completely before starting the next phase. In addition, there are four stages of involvement (SOI) for the certification liaison process. Further to these four stages, there is a software development review after each lifecycle phase.

The purpose of DO-178C is to guide the development of airborne software while ensuring that it performs the intended functions with a level of confidence to meet airworthiness requirement. DO-178C is objective-driven and companies can use different ways to achieve compliance as long as the objective(s) in question are met. To comply with the DO-178C standard, companies have to provide multiple documents and records about their development processes.

The main problem of the DO-178C is that errors that are detected late during the development process leads to much re-work that is costly and introduces many delays in drone software delivery. In addition, projects that comply with the DO-178C standards may see cost increases from twenty-five to forty percent compared to projects that do not require such type of compliance. The sources of these additional costs include: developer productivity is reduced due to the development process complexity; overall traceability is needed during product development; manual documentation and reporting processes; qualification activities that are involved in compliance; and running manual SOI to communicate to authorities.

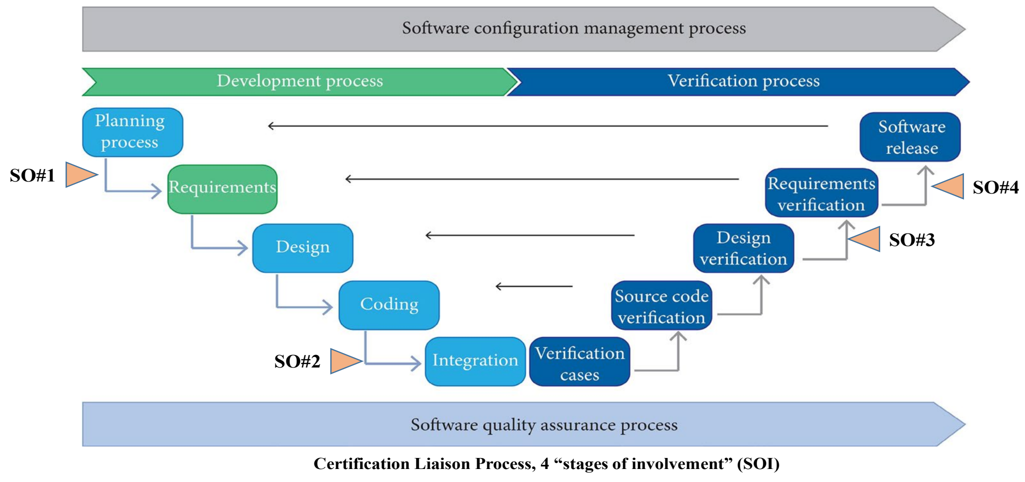

Fig. 1. DO-178C software development lifecycle [4]

As discussed above, a drone software system (which can be used by multiple customers) needs to be developed according to a process that complies with the DO-178C standards. It also needs to reduce risks, qualification cost, and the time to market. This raises a very important question: how can this be performed knowing that the system development process follows the waterfall model.

Recently, a number of approaches have been proposed to shorten the drone software development lifecycle. In the following, we describe some of them. First, within Airbus, the software development follows the "Requirements-Based Engineering" paradigm. In such paradigm, every software function is justified with a user requirement [8]. The requirements should be validated for ensuring the right system is built, and verified so that the system is built in a right way. The early requirements verification and validation can avoid high cost of late detection of software errors. Airbus has also introduced the use of agile development process for shortening the software development time. In this process, the development is done rapidly in an incremental way using several iterations.

Second, a new development process that combines advantages of agile and model-based development has been proposed to introduce a Model-Based Agile (MBA) process. This process is capable of satisfying Federal Aviation Administration (FAA) objectives for software of different Design Assurance Levels [7]. A key element of this proposed process is the use of incremental and iterative approach to requirements' collection and verification. In addition, in case of an appropriate modeling tool is used that enables the specification of executable models. A software engineer can start by creating the use cases and then iteratively refine them to have executable models. These models bring the advantages of agile requirements' collection before the detailed system design, implementation, and testing begin.

Third, Scrum framework has been applied to the development of avionics software and its verification processes described in DO-178C [9]. During the preparation phase of Scrum, the system requirements (or a subset of it) are taken to produce high-level requirements (HLRs) in the form of features. These features are further split into user stories. A software architecture is then defined which together with the prioritized HLRs are provided to the development phase. The development phase is composed of a sequence of sprints with same fixed duration (from one to four weeks). The result of each sprint is a number of implemented and validated user stories which are integrated with the software under development. The sprint also produces information required by the assessors. To start the closure phase, a sufficient set of features should be completed to grantee the software release. During the closure phase, all data items required for compliance with DO-178C standard that already exists are updated. In addition, a number of remaining data items are created by processes such software configuration and certification liaison. As required by DO-178C, all processes' outputs are verified through review or analysis methods. Furthermore, the software planning process that is responsible for creating and updating all plans is kept outside the Scrum framework.

Forth, an agile DO-178 development process has been introduced [10]. In this process, the development plans are developed, analyzed, and approved in the same way as traditional methods. After the approval of the process document, the development process starts and differences become apparent from the traditional methods. In the agile DO-178 process, all artifacts that are required for a specific feature are created in same iteration. After a pre-defined number of iterations, the FAA through the Designated Engineering Representative (DER) performs an SOI audit that covers all the objective tables for features that are developed since the last performed audit. After the audits and all necessary features are developed, the software becomes ready for the final certification review by the DER.

The above discussed approaches represent a good approach to reduce the development time while taking into account the qualification requirements to ensure drones safety. Thus, in the COMP4DRONES project, we go a step ahead by introducing a reusability aspect with agile development process (described in the next section) to shorten the time to market as required by the end-users.

3 C4D Reuse-based Agile Development

In the current economic environment, it is important for companies to minimize the additional costs related to DO -178C development. In Table 1, we re-list the reasons for the additional costs presented above, and our suggestions to reduce them. This leads us, in the COMP4DRONES project, to propose a reuse-based agile process for cost minimization.

Table 1. DO-178C problems and suggested solutions

Problem	Proposed Solution
Reduced developer productivity	Introduce more automation through increased usage of qualified tools as a support (e.g. code generator, certified compiler, automated testing, etc.)
Establishing overall traceability during product development	Use of model-based approaches (e.g. SysML, Simulink, AADL, etc.) and tools interoperability
Manual reporting and documentation	Model-based document generation
Qualification activities involved in compliance	Incremental certification through increasing the reusability of qualified components (or sub-systems)
Running manual SOI to communicate to authorities	Having a workflow engines to enable standard compliant engineering and generation of certification reports

Our reuse-based agile development process is shown in Figure 2. In this process, after the planning and requirements identification phases, a repository that contains software components is checked to identify COTS (Commercial off-the-shell) components that exist and can be used to satisfy the requirements by the use of assume-guarantee contracts approach [13] [17]. If such a COTS component exists, the development process starts from the integration phase. Otherwise, the full development cycle needs to be followed from design to delivery (see Figure 2). The main idea of this process is to speed the development process through reusable components that satisfies the users' requirements. This reuse-based strategy is also adequate to smooth the certification process if it is used with the concept of dependability certificate (such certificate contains all information on the dependability attached to the reusable components [14]) or the concept of Safety Element out of Context (SEooC) development [15].

Following this reuse-based agile process, the workflow for developing a drone software system can be divided into two main phases: development and integration as shown in Figure 3. In this workflow, the drone system is decomposed into sub-systems which are later divided into components. These components are either reused or fully developed from scratch. After having all required components developed or made ready for integration, the integration phase starts where the components are integrated together to form a sub-system. These sub-systems are then integrated to have a fully functioning system.

The proposed reuse-based agile process will be constructed following a model-driven engineering approach. Indeed, interest in using model-based system engineering/design (MBSE/D) has been steadily increasing in the system engineering community [18]. The INCOSE defines MBSE as “the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later lifecycle phases” [19]. MBSE relies upon system level models and offers convenient frameworks to integrate different dedicated analysis views within a global modeling environment. Hence, to be successfully implemented, an MBSE approach necessitates the following tree elements: a) a modeling language, b) a modeling methodology and c) a modeling framework that implements the modeling languages, preferably customized to support the development methodology.

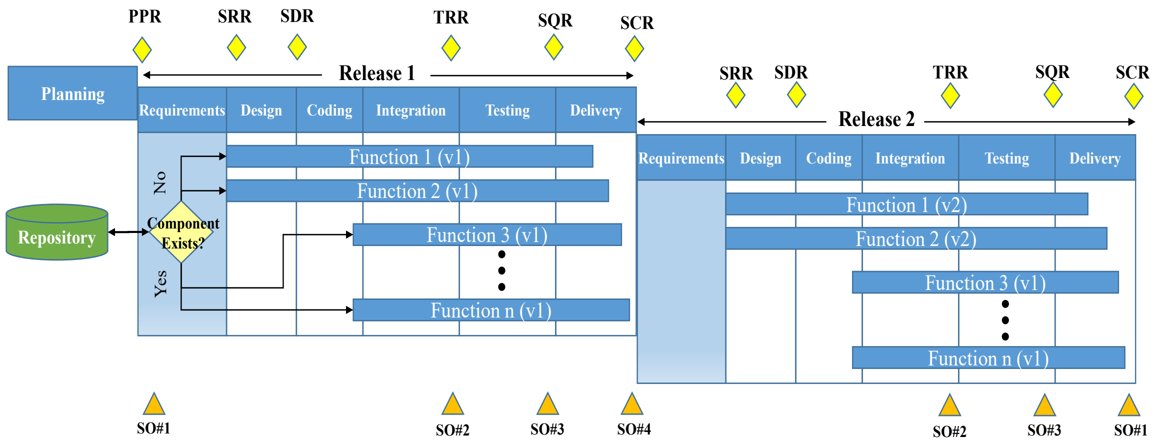

Fig. 2. Reuse-based agile development process

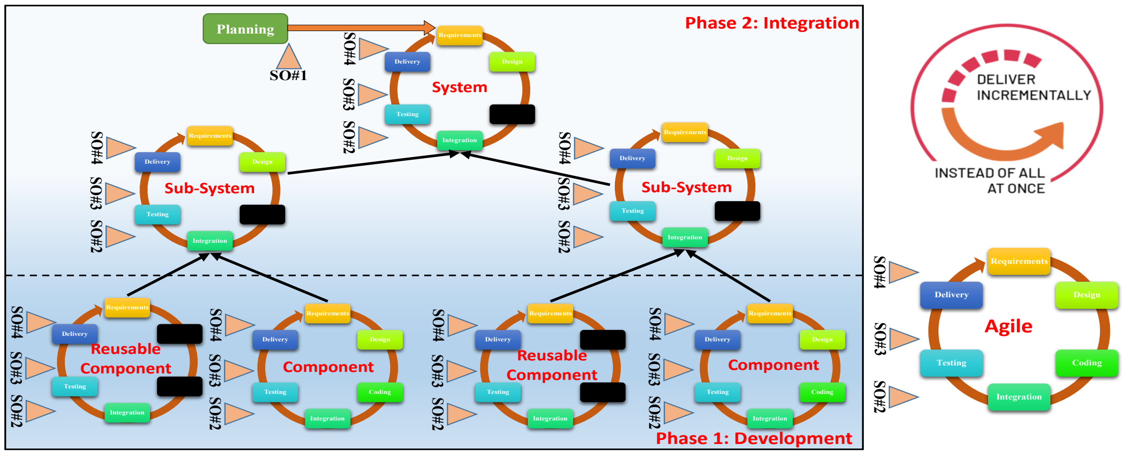

Fig. 3. Reuse-based agile development process workflow

The modeling language defines the notation (i.e. visual representation) and the semantics (meaning) used to construct the model. Recent studies place the Unified Modeling Language (UML) as the most used, tool supported and disseminated modeling language [20]. UML provides a standard extension mechanism called profile, which allows enriching the language with domain specific concepts. SysML [21] is such an UML profile that specialized UML concepts for system engineering. SysML is designed to provide simple but powerful constructs for modeling a wide range of system engineering problems. It specifically addresses the areas of requirements, structure, behavior and allocations/constraints to support various engineering analyses. SysML can be itself further specialized using profiles to cope with a specific methodology depending of the domain or application.

In C4D, the modeling methodology will be realized by an agreed meta-model. We are aware that agreeing on common modeling structures is a long process, and then the project will be built on existing modeling methods, such as RobMoSys project [22] (to enable composition and separation of concerns in robotics) and AADL (for timing and performance concerns) [23]. For instance, our proposed approach will follow the RobMoSys composition approach. RobMoSys has developed a System Engineering Approach to enable the design of safe and efficient robots based composability of qualified components. In RobMoSys, system composition requires a structure. This structure has requirements coming from three different perspectives: composability, workflow, and tools. Composability is the ability to combine (re-combine) components into different compositions by a workflow that involves all steps with the support of a proper tooling.

System composition is enabled through composability feature of system parts (i.e. making them building blocks) [24]. This enables the putting of different existing parts together in a meaningful way to create a new whole (system). Composability is not just about making the individual parts work together; it is the capability for selecting and assembling components in different combinations into valid simulation systems to meet specific user requirements. The composability should also be addressed between different components (i.e. syntax), alternatives of same component (i.e. semantics), and components with the application requirements (i.e. application and technical level perspectives).

The composition workflow is the activity to bring together all participants and building blocks to create and use a structure for system composition. It has to address each participant needs for the system composition. Stakeholders provide components and consume them for composition. To do so, a prior alignment of what is provided and what is needed (e.g. interfaces, functional boundaries, etc.) need to be considered. Within the workflow, these artifacts (components) are exchanged between stakeholders and the workflow steps while maintaining, managing, and ensuring composability during the whole process. The two main stakeholders are suppliers that provide building parts and system builders that use the parts to compose new systems. While there is a connection between both stakeholders, they do not necessarily work in the same team or even know each other. This is done by having a stable structure and all suppliers and system builders can rely on it. The suppliers can work within clear boundaries and the blocks can be connected through well-defined interfaces.

Structure support can have multiple shapes. One of the adequate supports is tools for the participants in a system composition. Tools assure that the developed system complies with the architecture defined at the model level. Tools realize the underlying structure and are utilized to prevent errors and provide automation (i.e. speeding the development lifecycle). Without adequate tools, the stakeholders of the ecosystem will have a hard time "accessing" the methods and concepts. The concepts remain unused or are going to be used in a wrong way, causing fewer acceptances and decreasing consistency. As a tool support, Papyrus for Robotics is an open-source [25], Eclipse-based modeling tool for robotic systems. It is a customization of the Papyrus UML modeling tool. Papyrus for Robotics conforms to the RobMoSys modeling approach. This implies that the tool supports different views for different stakeholders (i.e. service designers, component developers, safety engineers, and system architects).

A user of Papyrus for Robotics can specify a robotic (or drone) system as an assembly of component instances. The components in turn provide or require services via ports. These services are ideally not project specific but standardized by domain experts in order to facilitate the reuse of component definitions. In case of Papyrus for Robotics, the service definitions are part of a set of model libraries that have been obtained from ROS message and service definitions via reverse engineering. This means that a Papyrus for Robotics user has access to a wide range of service definitions that have been established by the community of several years (e.g. geometry-related messages or messages that are needed for navigation). A component instance can be configured, i.e. the configuration parameters of the component definition can be specified for a particular instance.

4 Reusable Components to be Developed as a support for Reuse-based Agile Method

To support the reuse-based agile methodology, a set of reusable components need to be developed and made ready in a repository. To identify and develop such reusable components for drones, a workflow has been proposed in COMP4DRONES project as shown in

Figure 4. First, the different project demonstrators are specified and analyzed to get a unified list of requirements (WP1). Second, the unified list of requirements is used to identify the key enabling technologies that are going to be developed during the project (WP2). Third, the identified key technologies are characterized and decomposed into the technical work packages: the architecture and its generic components (WP3), technologies for safe autonomous decision (WP4), trusted communication technologies (WP5), and tools for design, verification, performance analysis, etc. (WP6). Finally, the key technologies are integrated and evaluated by using them for the development of different demonstrators (WP1).

Following the C4D workflow, we have identified the key enabling technologies for drones (see Figure 5) [11]. First, to gain access to the U-Space, the drone's systems must be enhanced with the required U-space capabilities. Second, we have identified a set of technologies on the bases of the different usages of drones, the state of the art drone systems, and the best practices.

These technologies are grouped into system functions, payloads, and tools. In the following sub-sections, we list and describe all identified technologies.

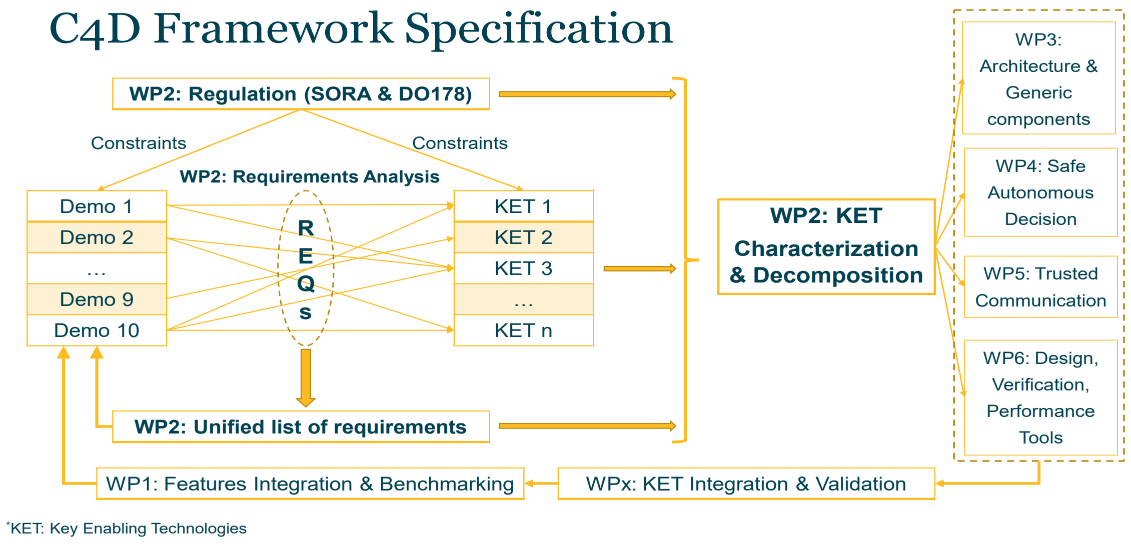
C4D Framework Specification
WP2: Regulation (SORA & DO178)
Constraints
Constraints
WP2: Requirements Analysis
Demo 1
Demo 2
…
Demo 9
Demo 10
R E Q s
KET 1
KET 2
KET 3
…
KET n
WP2: Unified list of requirements
WP2: KET Characterization & Decomposition
WP3: Architecture & Generic components
WP4: Safe Autonomous Decision
WP5: Trusted Communication
WP6: Design, Verification, Performance Tools
WP1: Features Integration & Benchmarking
WPx: KET Integration & Validation
*KET: Key Enabling Technologies

Fig. 4. The overall work flow of the COMP4DRONES project

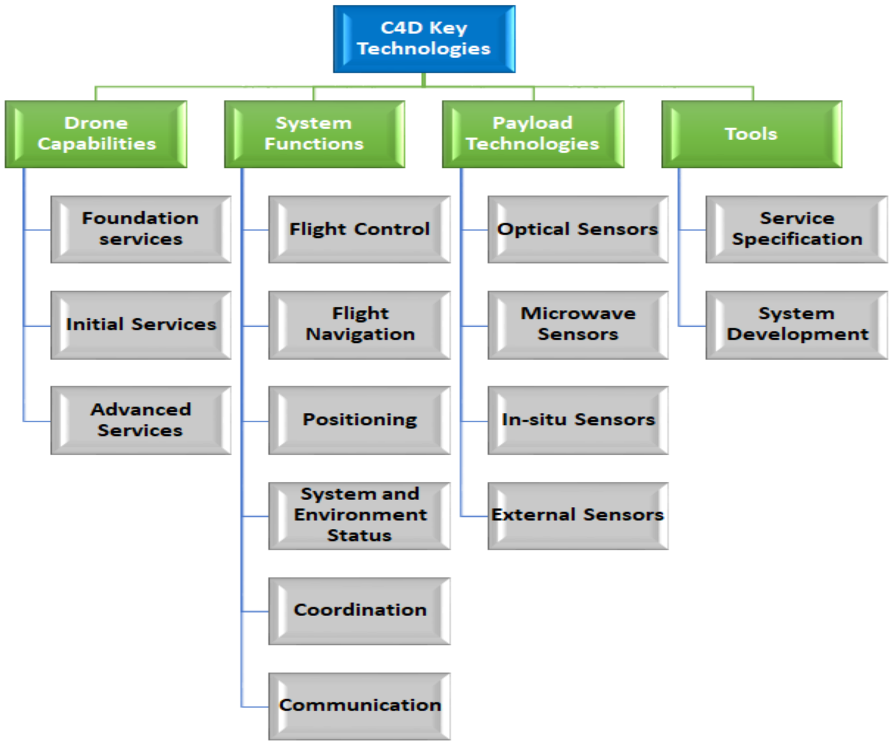

Fig. 5. C4D Key Technologies Framework

4.1 Drone Capabilities for U-space

The capabilities expected to enable U-space services are divided into three groups to support different types of service: *foundation, initial, and advanced* [26]. First, the capabilities for the foundation services include e-identification, geofencing, security, telemetry, operation management, command and control, surveillance, communication, and navigation. Second, the initial services capabilities include tracking and emergency recovery. Third, the capabilities for the advanced services are: detect and avoid, vehicle to infrastructure communication, and vehicle-to-vehicle communication.

4.2 System Functions

The drone system functions are the core functions (see Figure 5) required for the drone to perform its flying stages in safe and efficient manner. First, the *flight control system* includes (a) Intelligent Mission Management (IMM) that is onboard and/or ground technologies for providing a desired mixture of human-directed and autonomous drone operations; (b) Intelligent Outer-Loop Control (IOLC) which is an on-board capability to enable autonomous and semi-autonomous operations.

Second, the *flight navigation system* contains (a) Flight planning and scheduling that are general technologies that take higher-level goals, constraints, and objectives and then turns these into detailed plans and schedules; (b) Fail-safe Mission which is the ability of a drone system to adapt to software or hardware failures for having an acceptable level of safety; (c) Contingency Management that is an on-board capability for reacting to unforeseen events to minimize the likelihood of property damage and human casualties; (d) Deconfliction which is a function that is used to resolve potential conflicts that can occur between drones' trajectories in the phase of planning of trajectories; (e) Detect and Avoid (DAA) which involves capturing the surrounding environment, assessing potential of colliding with hazards that are detected, and taking corrective actions to avoid the hazards when a collision is coming.

Third, the *positioning system* includes (a) an Indoor Positioning System (IPS) that is a set of devices used for locating people or objects when GPS (Global Positioning System) and satellite technologies fail or lack precision; (b) Geofencing which is a virtual barrier that can be created by combining GPS network and LRFID (Local Radio Frequency Identifier) connections (e.g. Wi-Fi, Bluetooth, etc); (c) Georeferencing that is the task of assigning real-world coordinates to the pixels of a raster; (d) Simultaneous Localization and Mapping (SLAM) which is the process of recording environment and location awareness in a map of an autonomous vehicle.

Forth, *system and environment status* contains (a) Data Fusion which is the task of integrating many data sources to produce consistent, useful, and accurate information that cannot be provided by an individual source; (b) Intelligent System Health Management (ISHM) that is a technology designed for assessing the "health" of a system and performing actions that ensure the vehicle will remain healthy in future.

Fifth, the *coordination system* includes (a) the necessity of coordination between unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) is particularly evident to do missions in remote areas where human may be exposed to dangerous situations; (b) Swarm Formation and Cooperation that is the reasoning and making decision entity that is responsible for using mission requirements, observations, and system constraints to have a specific organization of the UAVs.

Finally, the *communication system* contains (a) Net-centric communications which is the concept of operations that uses advanced technology for shifting to a data-centric paradigm from an application-centric paradigm; (b) Over the Horizon Communications (OTH) that is a basic function required for UAVs to operate in the global airspace.

4.3 Payload Technologies

The drone system includes a set of payload technologies to support the drone mission specific operations. The payload technologies include optical sensors, microwave sensors, in-situ sensors, and external sensors (see Figure 5). First, the *optical sensors* incudes (a) Active Optical sensors (i.e. lidar) that use optical source such as a laser for sensing either hard or the atmosphere; (b) Passive Optical sensors which capture infrared radiation emissions, and direct or reflected solar energy to photosensitive detectors through an imaging optics system.

Second, *microwave sensors* include (a) Active Microwave sensor that is an imaging device for emitting microwave radiation and then records the echoes returned from the scene to be observed; (b) Passive Microwave sensors, which are used for both surface imaging and atmospheric measurements through gathering data through detecting light, vibrations, radiation, etc.

Third, *in-situ sensors* contain (a) Chemical Sensor Arrays technology that allows measurements for a range of chemical species in the UAV's surrounding environment; (b) Meteorological Data such as air density, temperature, and wind affecting UAV operations need to be measured; (c) Difference Frequency Generation (DFG) Lasers that are used as advanced in-situ detectors for tracing gases and their composition in the mid-infrared; (d) the CO2 Detection sensor that measures CO2 using a quantum cascade laser spectrometry in the flight configuration.

Finally, *external sensors* include (a) Dropsonde which is a weather reconnaissance device that is designed to be dropped from an aircraft to measure temperature, pressure, winds, and humidity; (b) Seismic (Geophysical) Sensors that are deployed on the ground to record sound wave velocities coming from an activated seismic source (like vibrator truck); (c) a Weather Station that is very useful to monitor the changes in real time for allowing the UAV system to detect and act when important changes in the forecast happen; (d) Perimeter Sensors that use light-detecting, passive infrared (PIR), and infrared (IR) sensors to monitor surroundings.

4.4 Tools

To support the development of drone systems, a number of tools need to be developed (see Figure 5). These tools are divided into two groups: tools for service specification, and tools for system development. First, the *service specification* contains tools for (a) User Requirements that referred to as needs that specify what the user wants from the system (i.e. what activities the system enables the users to do); (b) Acceptance Testing which is a level of testing, where a system is tested for user acceptability; (c) Data Analytics which is a process of cleaning, inspecting, transforming, and modeling data with the aim to discover useful information and conclusions to support decision-making; (d) Mission Planning that is the process to produce a flight plan that describes a proposed drone flight.

Second, *HW/SW system development cycle* include tools for (a) System Requirements that are the main blocks that developers use for building the system; (b) System Design process that provides the sufficient detailed information about the system and its sub-system to enable the system implementation; (c) System Implementation which follows the structure created during the system design, and the system analysis results to construct system components; (d) System Integration which is the process to bring together the different component/sub-systems into one system; (e) Verification and Validation (V&V) that is checking that a system meets its specifications and fulfills its intended purpose (goals).

5 Expect Gains from Reuse-based Agile Method

Traditional software development is complex as the effort required to add a functionality increases as the process progresses [27]. In the case of developing certification-driven software, at a project beginning (i.e. software phase), the development follows the curve of the traditional development. A divergence is seen when the software is ready to get tested in the field (see Figure 6). The process is slower due to hardware dependencies and semi-automation of acceptance testing (i.e. embedded phase). The significant slowdown is faced when the certification phase starts. In this phase, the software is supposed to be bug-free, however much manual testing and documentation are needed to provide artifacts that are required for the certification [28]. To reduce this complexity, agile development aims to have an ideal, flattened curve that allows a constant development step (as shown in Figure 6 - a modified version of the Figure presented in [28])

In the following, we discuss agile opportunities for every lifecycle phase, which define two new curves for the agile DO-178C and the reuse-based agile DO-178C processes. First, in the software phase, the software is developed independently and not affected by any constraint. Thus, all reuse-based agile practices can be used, and changes in requirements and hardware are not a problem. In addition, the user and acceptance testing of software are fully automated if needed.

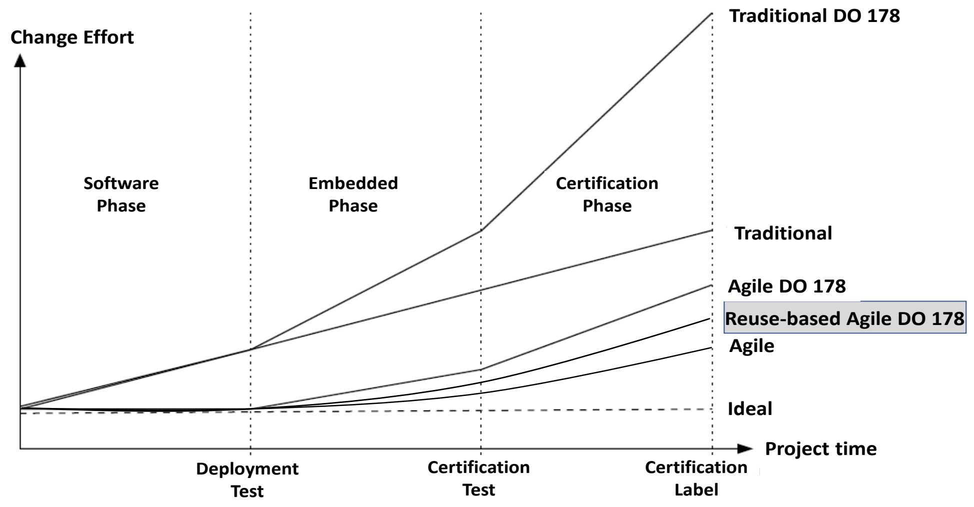

Fig. 6. Software processes compared

Second, during the embedded phase, the software is deployed and tested on the selected hardware configuration(s). In this phase, no fundamental reasons hinder the use of agility and reusability. The opportunities in this phase to speed the development include the automation of software installation, environmental tests, hardware retesting, and documentation generation.

Finally, the certification phase brings many additional tasks to be executed with each software change. These tasks include code coverage analysis, establishing traceability, reviewing, and manual testing. Evidences of these tasks need to be collected and documented. Therefore, there is a need to limit changes amount. This can be done through minimize requirements' changes, and the use of model-based traceability capabilities to identify the effect of a change on the already completed activities and artefacts. Furthermore, time spent for managing, creating, and reviewing documents can be reduced through automatic documents generation, having all documents under version control system, and manage documents dependencies.

Based on the above-suggested strategies to apply reuse-based agile methodology to drone software development, we expect a reduction in efforts as shown in Figure 6. The C4D project just stated its second year, and during this second year, we are going to use our methodology to develop the key enabling technologies described in Section 4. This will enable us to quantify the amount of effort reduction gained from the proposed methodology.

6 Conclusion

The development of drone avionics software is driven by the certification requirements as indicated in the DO 178C standard. However, the key problem of the DO-178C is that software errors that are detected late during the development lifecycle lead to much re-work which is costly and leads to many delays in the software delivery. Therefore, in the COMP4DRONES project methodology, we introduce a reuse-based agile development process. The main idea of this process is to speed the development process through integrating reusable components that supports the user requirements and incremental development. These reusable components are going to be developed during the C4D project. As the project just completed its first year, in the second year of the project, we are going to apply this proposed process to assess its applicability and the gain from using the reuse-based agile methodology.

Acknowledgments

We cordially thank all our colleagues from the COMP4DRONES project for their contributions, inspiring discussions, and fruitful collaborations. The partners cover important geographical areas in Europe and ensure that project outcomes will have lasting impact on European Drones Technology Industry.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826610. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands. The total project budget is 28,590,748.75 EUR (excluding ESIF partners), while the requested grant is 7,983,731.61 EUR to ECSEL JU, and 8,874,523.84 EUR of National and ESIF Funding. The project has been started on 1st October 2019.

References

[1] SESAR JU, "Report: European Drones Outlook Study," 2016. [Online]. Available: https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf.

[2] SESAR JU, "European ATM Master Plan: Roadmap for the safe integration of drones into all classes of airspace," 2017. [Online]. Available: https://www.sesarju.eu/sites/default/files/documents/reports/European%20ATM%20Master%20Plan%20Drone%20roadmap.pdf.

[3] McKinsey analysis; Teal Group, "World Civil Unmanned Aerial Systems Market Profile & Forecast," 2017.

[4] Hilderman, V. and Baghi, T., Avionics certification: a complete guide to DO-178 (software), DO-254 (hardware), Avionics Communications, 2007.

[5] IBM, DO-1 78C compliance: turn an overhead expense into a competitive advantage, Aerospace and Defense, 2014.

[6] J. Shore, The Art of Agile Development: Pragmatic guide to agile software development, O'Reilly Media, Inc., 2007.

[7] Coe, David J., and Jeffrey H. Kulick, "A model-based agile process for DO-178C certification," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2013.

[8] John Marsden, André Windisch, Rob Mayo, Jürgen Grossi, Julien Villermin, et al., "ED-12C/DO-178C vs. Agile Manifesto – A Solution to Agile Development of Certifiable Avionics Systems," in *ERTS 2018*, Toulouse, France, 2018.

[9] Hanssen G.K., Wedzinga G., Stuip M., "An Assessment of Avionics Software Development Practice: Justifications for an Agile Development Process," in *International Conference on Agile Software Development*, 2017.

[10] S. H. VanderLeest and A. Buter, "Escape the waterfall: Agile for aerospace,"," in *IEEE/AIAA 28th Digital Avionics Systems Conference*, Orlando, FL, 2009.

[11] M. Hussein et al., "Key Enabling Technologies for Drones," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, Kranj, Slovenia, 2020.

[12] J. SESAR, A proposal for the future architecture of the European airspace, 2019.

[13] Oh, Chanwook, et al., " Optimizing Assume-Guarantee Contracts for Cyber-Physical System Design."," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[14] Ardagna, Claudio A., Ravi Jhawar, and Vincenzo Piuri, "Dependability certification of services: a model-based approach," *Computing ,* vol. 97, no. 1, pp. 51-78, 2015.

[15] Schneider, Rolf, et al., "Safety element out of context-a practical approach," *SAE Technical Paper,* Vols. No. 2012-01-0033, 2012.

[16] M. Hussein and R. Nouacer, "Towards an Architecture for Customizable Drones," in *IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 2020.

[17] Alves, Carina, and Jaelson Castro, "CRE: A systematic method for COTS components selection," in *XV Brazilian Symposium on Software Engineering (SBES)*, Rio de Janeiro, Brazi, 2001.

[18] Schmidt, Douglas C., " Model-driven engineering," *Computer-IEEE Computer Society,* vol. 39, no. 2, 2006.

[19] Incose, Incose Systems Engineering Handbook V4, T. M. Shortell, Ed., John Wiley and Sons, 2015.

[20] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione and A. Tang, "What industry needs from architectural languages: A survey," *TSE Journal,* vol. 39, no. 6, p. 869–891, 2012.

[21] Friedenthal, Sanford, Alan Moore, and Rick Steiner, A practical guide to SysML: the systems modeling language, M. Kaufmann, Ed., 2014.

[22] Inglés-Romero, Juan F., et al., "A Component-Based and Model-Driven Approach to Deal with Non-Functional Properties through Global QoS Metrics," in *MODELS Workshops* , 2018.

[23] Feiler, Peter H., and David P. Gluch., Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language, Addison-Wesley, 2012.

[24] Mikel D. Petty and Eric W. Weisel, "A Composability Lexicon," in *Simulation Interoperability Workshop*, Orlando, USA, 2003.

[25] November 2020. [Online]. Available: https://www.eclipse.org/papyrus/components/robotics/.

[26] Barrado, Cristina, et al. , "U-Space Concept of Operations: A Key Enabler for Opening Airspace to Emerging Low-Altitude Operations," *Aerospace ,* vol. 7, no. 3, 2020.

[27] Boehm, B.W., Software Engineering Economics, Prentice-Hall Advances in Computing Science & Technology Series, 1981.

[28] Wils A., Van Baelen S., Holvoet T., De Vlaminck K., " Agility in the Avionics Software World," in *Extreme Programming and Agile Processes in Software Engineering. XP 2006*, 2006.