NASA/CR–2017-219359

Comprehensive Lifecycle for Assuring System Safety

John C. Knight and Jonathan C. Rowanhill
Depending Computing LLC, Charlottesville, Virginia

April 2017

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/CR–2017-219359

Comprehensive Lifecycle for Assuring System Safety

John C. Knight and Jonathan C. Rowanhill
Dependable Computing LLC, Charlottesville, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NNL13AA08C

April 2017

Acknowledgments

Dependable Computing thanks their colleagues at Barron Associates, at Ferrell and Associates Consulting, and at the University of Virginia for their technical contributions to the research reported here.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction 1

2 Safety Lifecycle 2
- 2.1 Lifecycle Concept 2
- 2.2 Lifecycle Scope 2

3 CLASS Origins And Terminology 3

4 CLASS Phase 1 4
- 4.1 Fundamental Principles 5
- 4.2 The CLASS Meta Process 6
- 4.3 The CLASS Instance Process 9
- 4.4 CLASS Resource Repository 12
- 4.5 Safety Information Repository 14

5 CLASS Phase 2 14
- 5.1 Process Concepts and Structure 15
- 5.2 CLASS Safety Case Development Process 17
- 5.3 CLASS Resource Repository 18
- 5.4 Expert Judgment 19
- 5.5 Integration With Existing Standards 20
- 5.6 Safety Case Condition Monitoring 20

6 CLASS Phase 3 20
- 6.1 Process Model 22
- 6.2 Expert Judgment 23
- 6.3 Analysis Framework 23
- 6.4 Development Process Monitoring 25
- 6.5 Certification 25

7 Empirical Studies 27

8 Bibliography of Published Papers 27

References 31

Introduction

This is the final report for Dependable Computing's contract #NNL13AA08C "Argument-Based Safety Assurance" in fulfillment of Items 17-20 of Exhibit B "Contract Deliverables Requirements". The participants in this research project were:

- Dependable Computing LLC, Charlottesville, VA.
- Barron Associates, Charlottesville, VA.
- Ferrell And Associates Consulting, Inc., Charlottesville, VA.
- The Department of Computer Science, University of Virginia, Charlottesville, VA.

The website that details the majority of the scientific results of this project is: http://www.dependablecomputing.com/class

CLASS is a novel approach to the enhancement of system safety in which the system safety case becomes the focus of safety engineering [1]. CLASS also expands the role of the safety case across all phases of the system's lifetime, from concept formation to decommissioning. As CLASS was developed, the concept was enhanced from a process and document focus to an information focus. The concept was also generalized to a more comprehensive notion of *assurance* becoming the driving goal, where safety is an important special case.

The definition of a safety case given by the UK Ministry of Defence Standard 00–56, Safety Management Requirements for Defence Systems, is [2]:

> "The Safety Case shall consist of a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment."

The structured argument referred to in this definition is a rigorous argument about properties of the subject system. The argument provides the explicit rationale for belief in a claim about the system. The claims made about the systems of interest to this research are *safety* claims. Nevertheless, the conclusions and concepts are influenced by and support the more general notion of assurance.

Placing the safety case at the core of system development and analysis provides a number of benefits. Nevertheless, this concept only has value if the safety case is well maintained and always consistent with the system – a property we refer to as *synchrony*. Maintaining synchrony requires that a system and its safety case be regarded as a single, composite entity, always linked and always representing one another correctly. The need for system development to maintain synchrony in a precise, controlled manner is a fundamental principle upon which CLASS is based.

CLASS was developed as a series of prototypes in an evolutionary manner. As CLASS evolved, various technologies were developed and added to CLASS with each new prototype. An important aspect of this project was the conduct of *empirical studies* of the concepts and ideas as they evolved. These empirical studies led to major changes to the various CLASS prototypes over the duration of the project. The major differences that arose during the development of CLASS are characterized as three phases. Phase 1 was the original concept. Phase 2 was an expanded and enhanced version of the original concept in which the notion of process dominated. Phase 3 was a further enhancement in which information became the central focus of CLASS.

Specific technical topics are only summarized in this report, because the details are documented in papers that have either been published or are under review. A bibliography of these papers is included at the end of the report.

The reader of this report is assumed to be familiar with the essential background technologies upon which this research is based. These technologies are:

- Modern system safety cases.
- Rigorous argument.
- The Goal Structuring Notation (GSN).

- Safety engineering.
- Safety-critical software engineering.
- Aerospace systems.
- Relevant regulatory processes.

Safety Lifecycle

Lifecycle Concept

For a safety-critical system, many system elements have to function correctly and their functionalities have to compose in an appropriate way if a system of interest is to:

- Be adequately safe.
- Be considered adequately safe.
- Remain adequately safe.

Fundamentally, to achieve these properties a great deal of analysis is required across three critical dimensions:

- Details of the system.
- Contexts within which the system will be used.
- Timeframes of system use.

The focus of this research project was to address the system safety issue for complex aerospace systems where the basic mechanism of assurance is the safety case, in particular to examine the way in which a safety case can support safety assurance across the three critical dimensions listed above.

Safety is an emergent property. An emergent property is one that is present at the system level as a result of various other properties held by system components and subsystems. In general, an adequate level of safety at the system level cannot be achieved or maintained if either: (a) analysis is limited to establishing properties of elements in isolation, or (b) analysis is undertaken for just part of the lifecycle, and, as a result, not managed properly as the subject system evolves over time. A system-wide, lifecycle view of safety is required if the safety requirements are to be met.

Without a lifecycle view, two major difficulties can arise:

- The assumptions made in development or operation of one system element might not be respected in other elements or subsystems.
- The dependability requirements stated, achieved, or maintained by one system element might not be sufficient for other elements or subsystems.

CLASS is a system-wide lifecycle based on safety cases that is designed to avoid these difficulties.

Lifecycle Scope

The engineering and assurance of complex, safety-critical systems for domains such as aerospace involve many different technologies and disciplines. The innovation brought to these domains by the introduction of safety cases displaces some technologies, enhances other technologies, and introduces yet other technologies.

Some existing technologies are well established and successful, and are not influenced directly by the introduction of safety cases (although these technologies might and probably will provide evidence that becomes part of the safety case). Nevertheless, these technologies are influenced by the introduction of the safety-case lifecycle. CLASS addresses these technologies only to the extent that the technologies have to be adapted to work with the basic CLASS concepts, i.e., these technologies are considered to be out of scope except to the extent that they have to be enhanced to support CLASS.

CLASS Origins and Terminology

The inspiration for CLASS was the concept of Assurance Based Development (ABD) [3, 4]. The fundamental concept of ABD is to drive the development of a software system from an assurance case for the software. All development decisions, including process steps and sequencing, originate from the creation of the assurance case.

ABD was developed for software and has been demonstrated in that domain. CLASS was an effort to extent the concept in three dimensions:

- The *temporal* dimension, i.e., to extent the concept to the complete system lifecycle, not just development.
- The *assurance* dimension, i.e., to extend the concept to other dependability properties, not just the assurance of functional correctness of software.
- The *application* dimension, i.e., to extend the concept to the subsystem and system levels, not just the software component.

Over the period of performance of this project, CLASS was developed as a series of prototypes, each building on the previous one, that can be thought of as three different *phases* of CLASS. These phases and their major characteristics are illustrated in Fig. 1.

Throughout the development of CLASS, aerospace domain experts were consulted regularly to provide assessment of the concepts. These consultations led to major enhancements to CLASS as the phases evolved.

In the following three sections of this report, each phase of CLASS is discussed in order to show how CLASS evolved. Phase 1 built on Assurance Based Development. Phase 2 built on phase 1 and both enhanced the techniques developed in phase 1 and added new techniques. Phase 3 built on phase 2 and made a significant revision in which various underlying principles were revised and the associated technologies enhanced appropriately.

Three important concepts that support CLASS and its evolution are: (a) the CLASS Resource Repository (CRR), (b) the Safety Information Repository (SIR), and (c) assurance analysis:

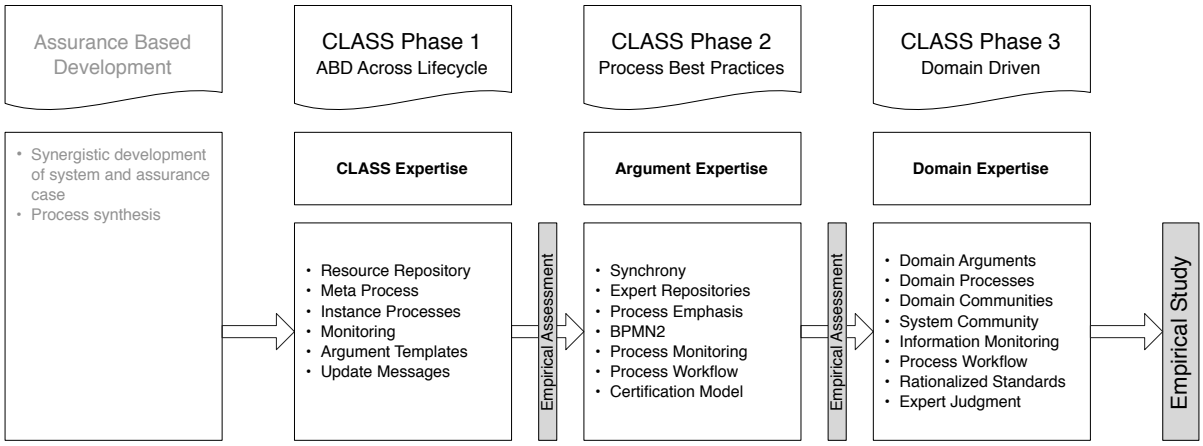

Fig. 1. The three phases of CLASS that were developed over the period of performance.

- **CLASS Resource Repository**. The CRR is a general repository that holds various resources likely to be of value to CLASS users.
- **Safety Information Repository**. The SIR is a system-specific repository designed to hold all of the safety-related assurance information for a specific subject system over its lifetime.

- **Assurance Analysis**. Assurance analysis is integral to CLASS. The approach used by CLASS is based on the Filter Model introduced by Steele and Knight [5] and motivated a number of crucial elements of CLASS, in particular the monitoring system.

These concepts are referred to frequently and discussed in depth at various points in this report.

Four additional concepts, *lifecycle*, *process*, *guidance* and *architecture*, are summarized here so as to provide the reference framework for subsequent CLASS explanatory sections.

- **Lifecycle**. Lifecycle represents the stages of a system in which the primary purpose or activity of the system differs. For example, an engineered system will have a stage during which it is created, and another stage during which it is operated. The primary goal of creation differs from operation. Each stage of a lifecycle is dominated by a purpose that differs from other stages. These purposes are goals for the system. They are divisible, so that a lifecycle stage might have sub-stages. A large system, such as a nuclear reactor, might have an online sub-stage and an offline maintenance sub-stage within its operation stage. Stages can cycle. The online and offline sub-stages, for example, might run many cycles during the operational stage of a nuclear power plant.
- **Processes**. Processes represent activities to support a system in its various lifecycle stages. A process is a program (in an abstract sense) that exists to serve a lifecycle stage. It performs work in support of the lifecycle stage. In cybernetic systems (involving people and machines) its primary purpose is to coordinate human and automated actions, as well as track users and their tools, to assure the rules of the process are met.
- **Guidance**. Guidance is a collection of qualitative statements and more detailed presentations designed to help users make decisions. At many points in a process, the documented content of a process must be created, or flow-chart decisions must be made. In both cases, capturing such creative input in process form is generally a poor idea. Instead, users must be informed through guidance mechanisms such as documents, expert systems, training, practice, and examples.
- **Architecture**. Architecture is a means to implement lifecycle, process, and guidance. Architecture is generally concerned with operating these features but also with achieving non-functional properties such as scalability and dependability. For example, architecture in support of building CLASS for large-scale systems is concerned with supporting large, inter-institutional hierarchies of vendors, designers, suppliers, implementers, and regulators, in applying processes within lifecycles using guidance. The CLASS architecture is also interested in information and knowledge sharing, amongst other functional features to create a Safety Engineering Ecosystem.

CLASS Phase 1

CLASS phase 1 addressed the major issues that arose from the expansion of Assurance Based Development into the temporal, assurance, and application dimensions discussed in section *CLASS Origins And Terminology* above.

CLASS phase 1 introduced the basic notion of a meta or generic overall lifecycle approach from which specific instances are instantiated for specific projects. To support this process picture, CLASS phase 1 introduced the general notion of a resource repository, the CLASS Resource Repository, to provide a source of materials through the associated project lifecycle. Separately, the Safety Information Repository was introduced that provides a well-defined structure for holding all of the project materials related to safety. The Safety Information Repository holds the safety case and all related materials along with operational data.

Finally, CLASS phase 1 introduced an initial monitoring mechanism that linked assumptions made in safety arguments with experience of the system during operation.

Fundamental Principles

At the highest level, the CLASS phase 1 concept is shown in Fig. 2. A subject system begins as a concept and passes through a series of world states over the system's lifetime, beginning with design and development, and moving through operation and maintenance to decommissioning/retirement.

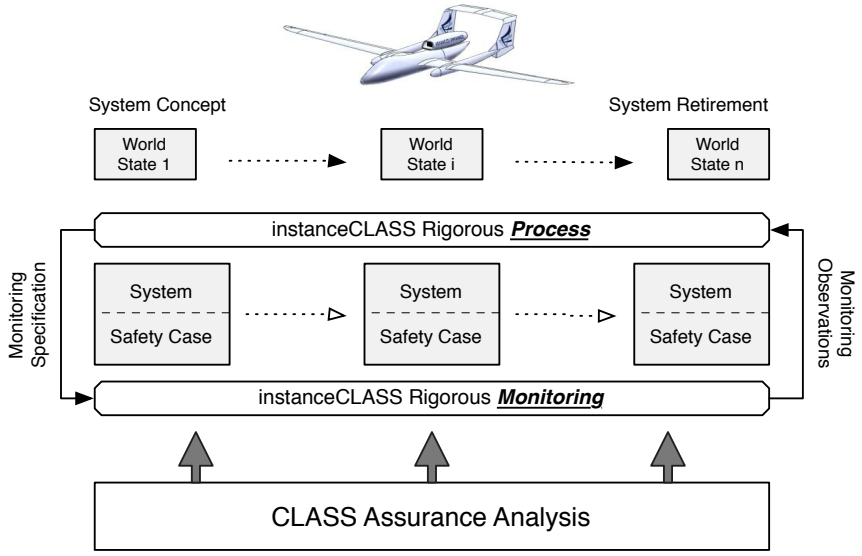

Fig. 2. The CLASS concept.

CLASS phase 1 was based on a set of fundamental principles that were used to guide its development. The principles evolved from: (a) the analysis of the sequence of CLASS prototypes, and (b) a basic goal of being able to analyze CLASS so as to try to predict its effectiveness.

Many software development methodologies are defined by artifacts in addition to the system implementation. For example, tests guide development of an implementation in *test driven development*. Similarly, *requirements driven development* refines requirements to system implementation. Finally, in an *enterprise architecture* approach, architectural patterns demonstrate useful system properties and the implementation must in turn observe the architecture.

Each of these approaches involves synchrony between a non-functional, analytic product and the functional system. This synchrony is a co-informing relationship, as depicted in Fig. 2. The functional system can inform the analytical product, and/or the analytical product can inform the functional system. Furthermore, the two cannot contradict each other, but are instead extrapolations of one another. Finally, just as the pieces of the functional system must be coherent, so must the analytic products.

These concepts were adopted as an analytic principle in CLASS phase 1. In CLASS phase 1, the non-functional, analytic product is the *safety case*, and the notion of synchrony that is inherent in these advanced software development methodologies is a fundamental principle of CLASS.

The exact semantics of synchrony evolved over the three CLASS phases. We define the term here:

> **Assurance Synchrony**. Assurance synchrony is the property that an assurance case is in synchrony with the system about which it argues. For example, a safety case is in

synchrony with its system when all of its safety claims and evidence are true for the current implementation or furthest-so-far refinement of the system model. This further requires that all analytic products are coherent. For example, evidence and hazard analysis cannot be allowed to ignore or contradict one another.

Assurance synchrony is the key property of the CLASS creation process. Failure of assurance can result from a failure of synchrony, or assurance synchrony fault. In this case, there is an underlying error in the assurance case, implementation, or both, such that a lack of synchrony exists between the two in the main image of the system.

The principles upon which CLASS phase 1 rested are illustrated in Fig. 2. The principles are:

- **Composite Entity**. The subject system and the associated safety case are treated as a single, composite entity. We refer to this entity as the *System and Safety Case Pair* (SSCP). This principle is a central, explicit element of CLASS.
- **Synchrony**. Synchrony between the subject system and the associated safety case is maintained whenever they should be synchronized. Timing of synchronization is determined by analysis. A useful result of the introduction of the SSCP is that many elements of the basic CLASS process and overall CLASS analysis are put on a much more rigorous footing.
- **Assured Properties**. The CLASS lifecycle process structure precludes the introduction of safety defects to the extent possible. Assurance of this process property is by analysis.
- **Monitoring**. CLASS phase 1 was designed to undertake continuous monitoring during operation of assumptions made in the system safety case. Necessary sensors[1] and monitoring algorithms are determined by analysis. This initial form of monitoring was motivated by the need to ensure that safety-case assumptions were respected during execution. In phases 2 and 3 the concept of monitoring was extended to include all aspect of the CLASS process over all time. With comprehensive monitoring in this way, properties that follow from the definition of the process are assured.

The CLASS Meta Process

CLASS needs to handle all of the world states that arise in system development, and, to accommodate variation in system types, CLASS is defined as a meta process that is instantiated for the specific configuration of a subject systems. This structure is shown in Fig. 3.

The CLASS meta process, *metaCLASS*, provides lifecycle support for the SSCP. Fig. 4 illustrates the basic metaCLASS instantiation mechanism and the major outputs of the associated instanceCLASS. Also shown in Fig. 4 is the Safety Information Repository (SIR).

The SIR is a structure that holds the safety case and all related assets that result from the creation and use of an instanceCLASS for a subject system. The SIR is created as part of the instantiation process and is a lifetime entity permanently associated with the subject system, i.e., the SIR accompanies the system throughout the system's lifetime. The SIR provides all of the artifacts needed to support the instanceCLASS as changes become necessary during the system lifetime.

[1] The term "sensor" in CLASS refers to any mechanism that provides CLASS with either a value or details of an event that are relevant to the monitoring process. For example, if a safety argument relies upon a component probability of failures being less than some specific value, then a sensor would be an element of a maintenance system that supplied CLASS with details of failure events and associated repairs.

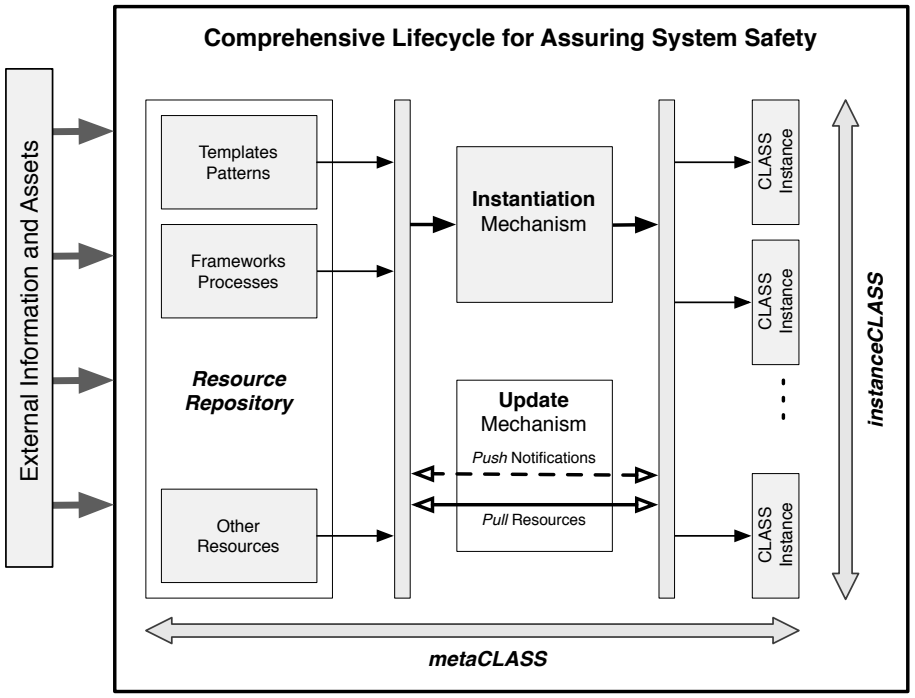

Fig. 3. CLASS instantiation process deriving CLASS instances from metaCLASS.

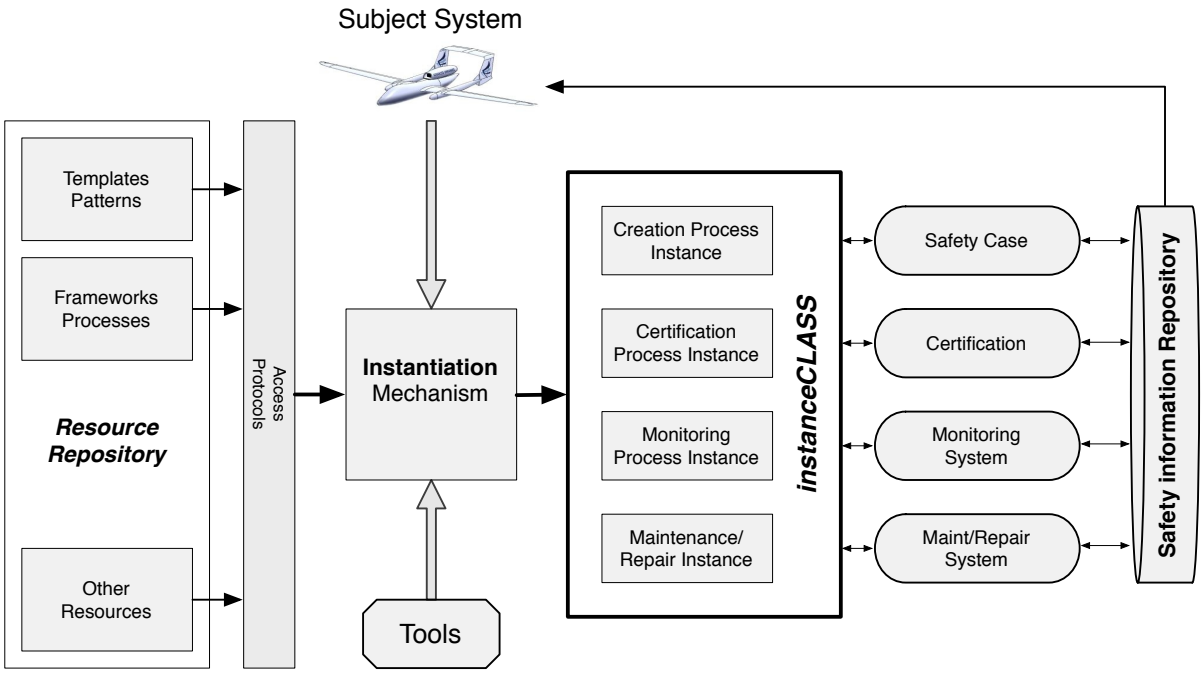

Fig. 4. The primary elements of the CLASS instantiation process including the Safety Information Repository

metaCLASS is composed of two basic components:

- **Development Component**. The development component of metaCLASS along with the CLASS Resource Repository (CRR) supports development of instanceCLASS processes, both initially and over the system lifecycle.
- **Update Component**. The update component of metaCLASS along with the CRR supports updates of instanceCLASS processes over the system lifecycle.

Development Component

An instanceCLASS is needed for any project using CLASS, and each instanceCLASS is created by metaCLASS using resources from the CRR. Once the instanceCLASS is created and the necessary activities needed for the system safety case initiated, metaCLASS remains in operation and available to provide lifecycle support. Additional resources can be obtained from the CRR at any point.

The CLASS meta process provides access to the CLASS Resource Repository (CRR) using a set of access protocols designed to facilitate:

- Location of useful resources.
- Appropriate use of those resources.

As an example of an access protocol, consider the resources needed for certification. In an aviation context, the resources needed for certification will be influenced by whether the subject system is for ground or airborne use, the criticality level of the subject system, the overall system requirements, and so on. The role of the access protocols in this case is to provide a means of locating the right versions of standards, process support entities, tools, regulatory mechanisms and so on.

Update Component

The update component of metaCLASS is responsible for ensuring that all instanceCLASSs remain properly synchronized with all available resource information. The resource update mechanism in CLASS is shown in Fig. 5.

The issues that force interaction between instanceCLASS and metaCLASS arise from the temporal dimension of SSCP support: The problem that the update component is designed to address is best illustrated by the following examples:

- **Meta Resource Updates**. Updates to resources within the CRR and the availability of new resources within the CRR need to be communicated to all instanceCLASSs that might have an interest in those resources, e.g., an instanceCLASS that was derived from those resources. This communication is especially important for defects that are detected and repaired in the CRR. As an example, consider the possibility of a defect in an argument pattern being detected. All safety cases relying on that pattern would need to be checked and possibly updated.
- **Instance Resource Updates**. Necessary updates to CRR resources might be detected by an instanceCLASS that was instantiated using those resources. The CRR needs to be informed of such updates and then instanceCLASSs derived from those resources need to be informed. As an example, again consider the possibility of a defect in an argument pattern being detected in the instantiation being used by an instanceCLASS. Correcting that defect in the CRR and all derived instanceCLASSs is crucial.

Updates must be coordinated. Effecting an update to either the CRR or the resources being used by an instanceCLASS in an unsynchronized manner could disrupt ongoing activities. Conversely, the CRR, metaCLASS and all instanceCLASSs need to be informed promptly of an update.

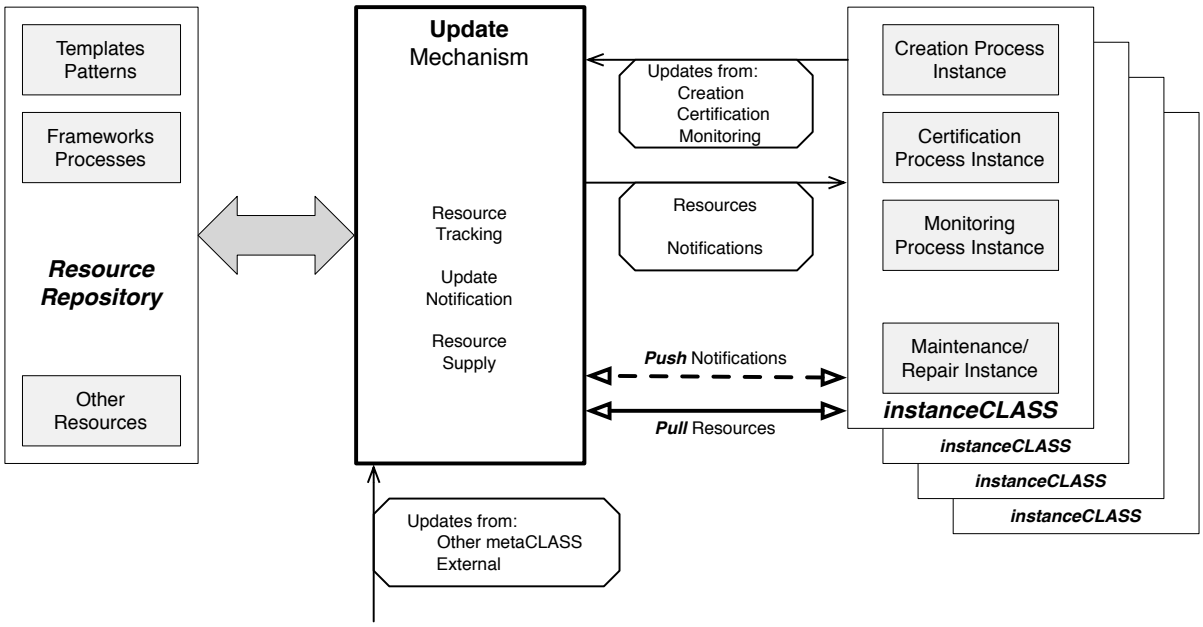

Fig. 5. CLASS resource update mechanism

The CLASS update mechanism is divided into a *Notifications* component and a *Resources* component. A notification includes the details of an update but is, as their name indicates, merely to notify all interested processes of an update. Processes can then decide on the relevance and importance of the update. Notifications are “pushed” to all interested processes. instanceCLASSs notify metaCLASS of necessary updates, and metaCLASS manages the subsequent notifications to interested processes.

The Resources element of the update mechanism in CLASS provides the mechanism for interested processes to acquire updated resources. Resources are “pulled” by the interested processes and so the installation of an update in under their control.

The CLASS Instance Process

The anticipated lifetime of systems supported by safety cases developed with CLASS is considerable, and so the associated SSCPs must be kept synchronized with the subject systems, i.e., the property of synchrony must be maintained.

Fig. 6 illustrates this essential temporal component of the problem. As time passes, the world state changes. Areas in which changes might occur that could affect system safety include the system design, development environment, operating environment, and the mission requirements.

The subject system has to operate with adequate safety from initial system deployment to system retirement. All of the development artifacts feed into safety assessment, as do all changes during operation.

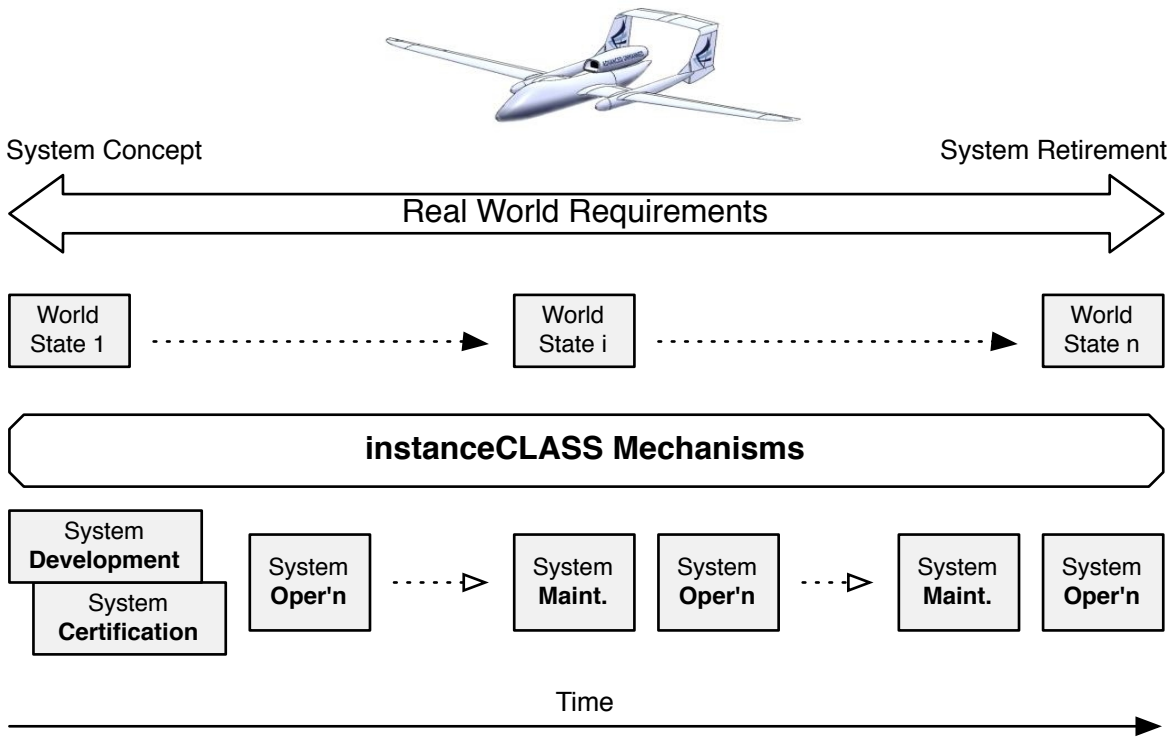

Fig. 6. CLASS instance long-term operation

The activities necessary to maintain SSCP synchrony are primarily the responsibility of instanceCLASS. Thus, an instanceCLASS has to have access to the necessary resources and has to be activated appropriately.

A wide variety of types of information are used within a CLASS instance, and Fig. 7 is organized around these items of information and their flows through the structure. The key elements of the information flow structure are:

- The safety case flow from the developers of the safety case to the regulating agency. Development of suitable regulatory certification processes is a key component of CLASS.
- The safety case design flows to and from a variety of sources and sinks. The concept underlying the design flow is to ensure that the developers of the subject system and the associated safety case are properly apprised of the interests of other stakeholders.
- The safety case data flow from various sources to various sinks. The concept underlying the safety-case data flow is to document precisely and rigorously the sources, sinks and content of data that is needed to support operational monitoring.

An instanceCLASS in CLASS has the structure shown in Fig. 8. Important aspects of instanceCLASS are:

- The explicit identification of the safety-case repair function and the merging of this function with the safety-case creation function.
- The explicit introduction of a path for receipt and development of requirements for changes to the system and associated changes to the safety case.

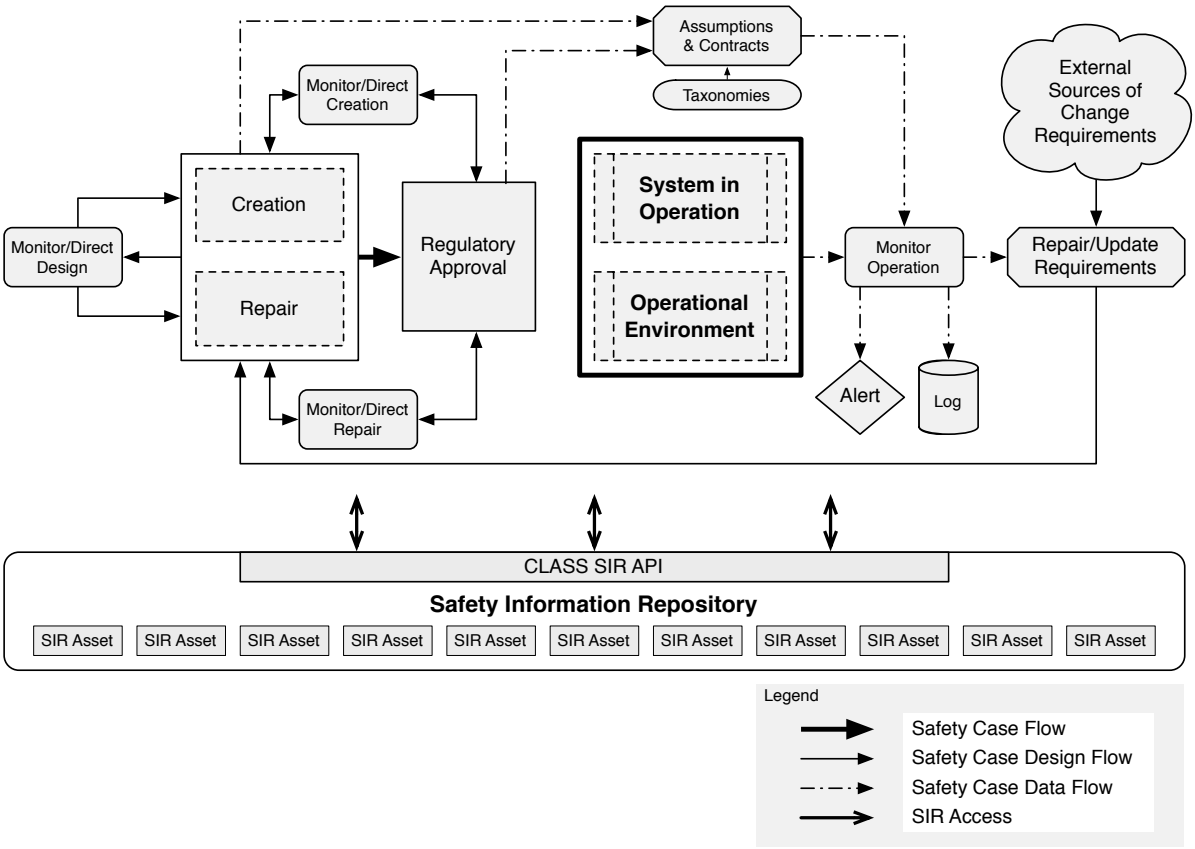

Fig. 7. instanceCLASS information flow

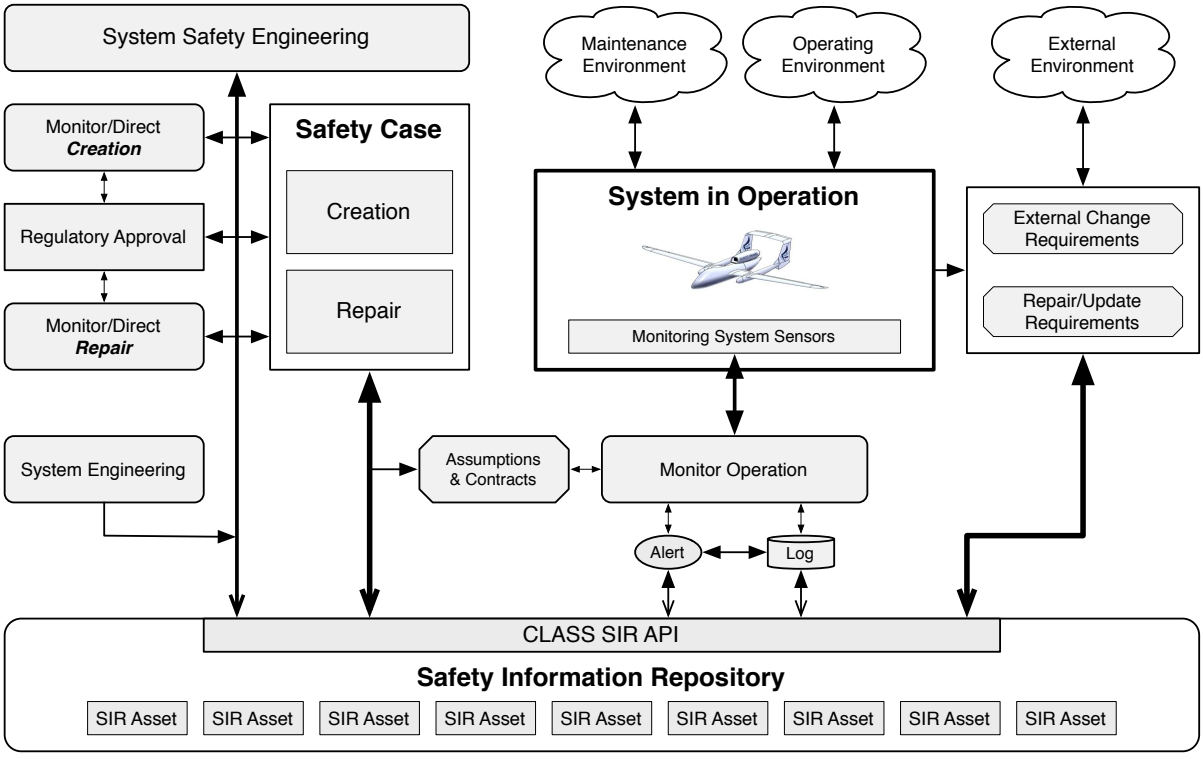

Fig. 8. instanceCLASS structure

- The explicit introduction of monitoring during system operation.
- The explicit introduction of the system operating environment into the system elements to be considered during operation. Clearly, monitoring a system in operation requires monitoring of the relevant details of the associated operating environment.
- The central role of the Safety Information Repository (SIR). The SIR is present to maintain all the system safety information (see section *Safety Information Repository* below).

CLASS Resource Repository

The CLASS Resource Repository (CRR) supplies resources that metaCLASS uses to build an instanceCLASS. The expectation is that the CRR will provide resources of many types and permit a great deal of reuse across projects.

Access to the CRR is through a set of *Access Protocols*. These protocols are designed to provide guidance to the user when accessing the CRR. The guidance is designed to take the user's requirements and attempt to show the user the CRR material that is relevant to their goals and how it should be used. For example, consider an instantiation of CLASS that will be used for a large UAS in the NAS. This application will require a lot of material relevant to topics such as UAS hazard analysis, FAA guidance on operating in the NAS, FAA-approved standards for supporting development of the various UAS components, and so on. The Access Protocols are intended to guide the user in locating and selecting these relevant resources.

The CLASS Resource Repository cannot be unique, i.e., there cannot be a single CRR that provides resources to all projects. Inevitably, any organization that is using CLASS will require organization-specific elements in the CRR. All instantiations of the CRR should be (though need not be) derived from a central facility so as take advantage of the common base of assets. Nevertheless, a tailoring mechanism is required to permit a generic CRR to be adapted to the needs of a specific organization.

To accommodate the various modifications necessary within the CRR and to permit flexible use of those modifications, CLASS defines a hierarchy of CRRs. The CLASS CRR hierarchy is illustrated in Fig. 9.

The hierarchy begins with a generic CRR used by metaCLASS. Along with the specific content designed to support CLASS, this generic CRR includes domain specific resource repositories that are provided by other sources. Examples of this type of repository are:

- Standards and documentation repositories maintained by organizations such as the FAA and RTCA.
- GSN pattern libraries maintained by organizations such as NASA and the University of York.
- Software libraries that provide common services such as those available from AdaCore and Green Hills.

At the next layer of the hierarchy are a set of local CRRs, each designed to support the metaCLASS operated by a specific organization, such as an aerospace system supplier. Each such CRR is adapted from the generic CRR to support the particular organization.

Within a given organization, each instanceCLASS is created from the local CRR. As shown in Fig. 9, a set of instanceCLASSs within an organization is derived from the organization's local CRR.

The CLASS Resource Repository is not static. As shown in Fig. 3, over time updates or additional materials derived from external sources might be entered into the CRR. Other changes to the CRR might be motivated by either resource demands or determination of defects in resources as a result of their use. System instanceCLASSs might benefit from these updates or additional materials. In summary, the CRR is updated as needed from both external and internal sources.

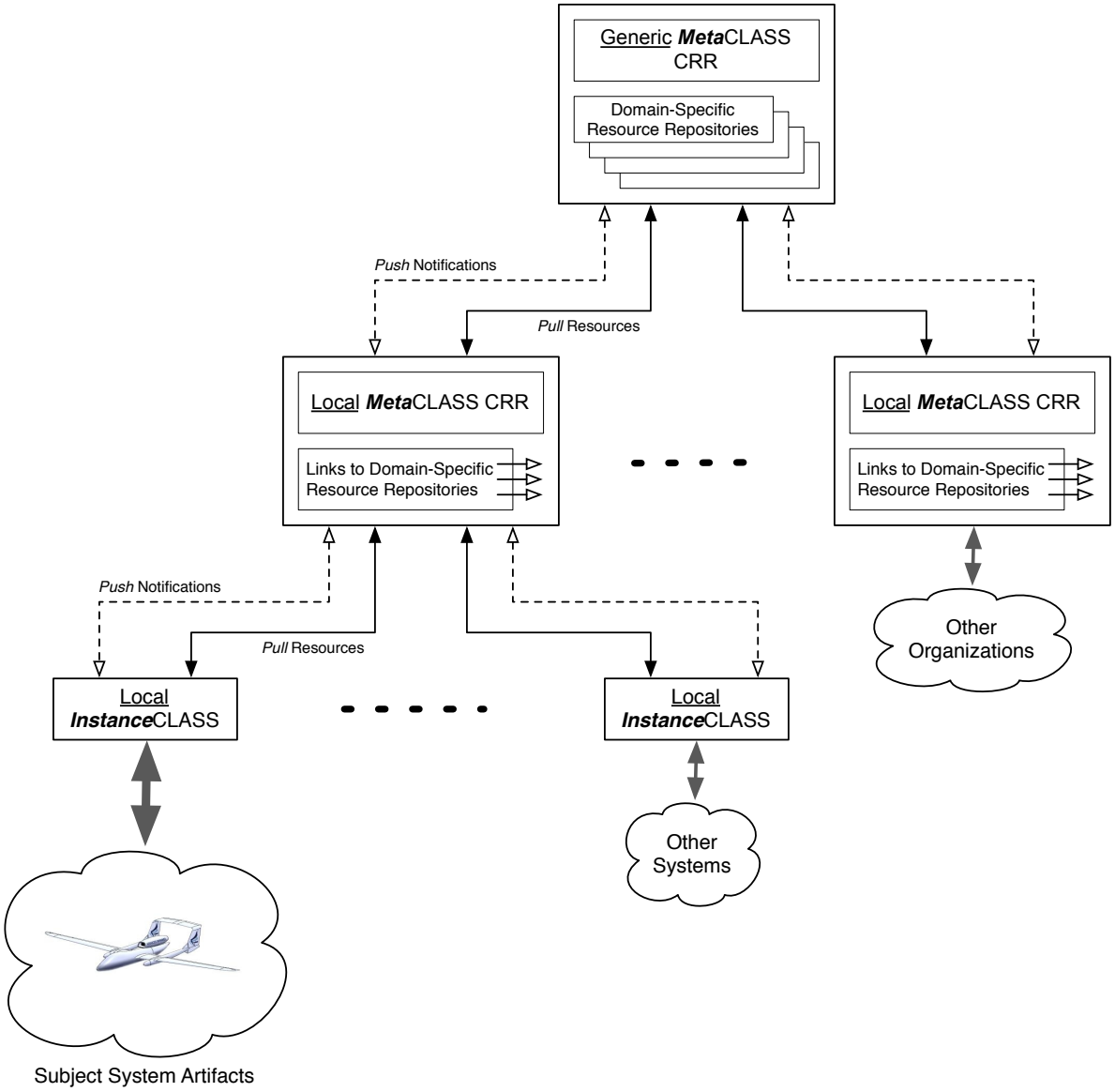

Fig. 9. CLASS Resource Repository hierarchy

No matter what the cause, changes in the CRR must be handled carefully, because changes to the CRR might disrupt an ongoing development. Enhanced resources might not be of immediate value to existing instanceCLASSs, and the decision about their adoption must be the responsibility of the individual instanceCLASS. Declining to enter a new or enhanced resource within the CRR might be preferable. Similarly, defects identified in resources by an instanceCLASS must be made available to other instanceCLASSs in order to protect the latter from the effects of the identified defect. Nevertheless, each instanceCLASS must be able to exercise explicit individual control over the action taken when resources are updated.

The CRR hierarchy implements the metaCLASS update model for the CRR with the following characteristics (see Fig. 9):

- Resource *transfers* are “pull” connections meaning that resources are transferred on demand.
- Update *notifications* are “push” connections meaning that notifications are transferred when they are generated.

Safety Information Repository

Each CLASS instance maintains a Safety Information Repository (SIR). The SIR is a sink for all of the information that flows in a CLASS instance and a source for that information also. The SIR expands the concept of a safety case to include all relevant information that might have value across the lifecycle.

The role of the SIR in the overall CLASS architecture is shown in Fig. 4. The information flow of the subject system, the associated instanceCLASS, and the SIR is shown in more detail in Fig. 7. Fig. 8 shows the architecture of instanceCLASS and the interaction between the SIR and the rest of instanceCLASS.

As examples of the role of the SIR, consider: (a) the safety case itself and (b) the results of operational monitoring of the subject system:

- The safety case has to be accessible and available for update and repair at any point during the subject system's lifecycle. The SIR provides this storage and access facility for the safety case and all associated information.
- Operational monitoring is designed to check on the conformance of the subject system to the constraints upon which the safety case was built. Any deviations raise issues about system and safety case consistency and validity. Details of the monitoring record need to be available in an accessible and predefined form. The SIR provides this storage and access facility for the monitoring data and all associated information.

The (preliminary) expected content of the SIR as defined in CLASS either in the form of the actual artifacts or links to the actual artifacts is:

- A catalog of the artifacts contained within the SIR along with the dates of their creation, modification, etc.
- The system's safety case.
- All source artifacts used to create the system's safety case including:
  - All items of evidence.
  - The tools needed to reproduce the evidence.
  - The safety arguments.
- All artifacts associated with the certification of the system via the safety case.
- Logs of system maintenance.
- Logs of system monitoring.

As shown in Fig. 7 and Fig. 8, the SIR has an API that supports access and management of the SIR. For any lifecycle activity that involves safety-related information, processes that use the SIR through the API would be available.

CLASS Phase 2

CLASS phase 2 enhanced the basic structures established in phase 1 with a process-centric approach. The more abstract process concepts were established in phase 1 with metaCLASS and instanceCLASS. These concepts were made far more explicit in phase 2.

The increased level of detail at the process level led to numerous other enhancements:

- Detail process descriptions for the various activities in safety-critical system development that were driven by assurance goals.
- An enhancement to the notion of monitoring (and hence to the requirements for the CLASS monitoring system) driven by the goal of assuring conformance during development to the process definitions.
- A realization that process activities and conformance to the correct execution of these activities relate closely to the goal of developing systems with as few faults as possible. In CLASS phase 3, this realization then led to the elimination of indirect evidence from the types of evidence that need to be considered in CLASS.

- The development of a mechanism for allowing CLASS processes to be defined so that they produce direct rather than indirect evidence.

In summary, CLASS phase 2 was a highly process-centric lifecycle. All of the elements of CLASS phase 1 were updated to accommodate the process-centric focus. Processes were defined using the Business Process Model and Notation 2 (BPMN2) defined by the Object Management Group. This notation provided a convenient and effective graphic representation of process ideas. Since the notation is graphic, process definitions can be edited electronically. In addition, the formal nature of BPMN2 allows machine processing of process descriptions thereby enabling input to be generated for workflow management systems.

Process Concepts and Structure

Details of the process concepts evolved for CLASS phase 2 are documented in a technical report (paper number 13 in section *Bibliography of Published Papers* below). For completeness, a summary of the material is provided here.

CLASS phase 2 explored the role of process in creating an assured system. The key issues investigated were:

- The mapping of software development methodology to the properties of synchronization, quality, and completeness.
- The encoding of these mappings into concrete, automatable workflow processes written in the BMPN2 workflow language.
- The demonstration of the application of these processes to the simulated creation of aircraft collision avoidance software.
- Description and analysis of the lessons learned from this exercise.

The critical issue in the mapping of software development methodology to CLASS is to ensure that the requirements of synchronization are met. As shown in Fig. 10, modern parallel software development processes can easily fail to synchronize correctly.

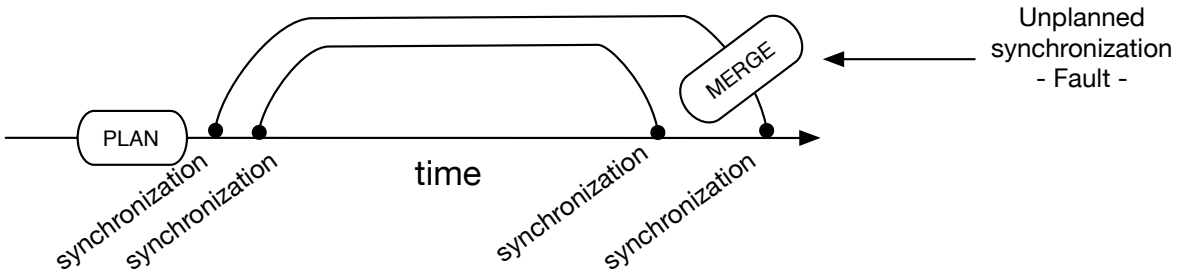

Fig. 10. Parallel development and unplanned synchronization

What is required is support for parallel development using a Spiral process model [6] with proper attention to synchronization of development and assurance activities. The CLASS overall process structure is shown in Fig. 11.

In the CLASS process structure, the System Safety Case Pair (SSCP) are developed by teams operating with whatever degree of independence is considered appropriate for the overall project. The SSCP is shown at the top of the figure. Different development groups operate separate Spiral processes in parallel using a conventional methodology. These parallel spiral processes are shown in the center of the figure.

Synchronization between development groups is achieved at the Synchronization Boundary shown at the top of the set of Spiral processes in the figure. This boundary allows independent developments to be combined and for the SSCP to once more be in synchrony. Issues and

necessary improvements together with assurance failures (such as would arise from the development of evidence) are stored for processing in a central tracking repository.

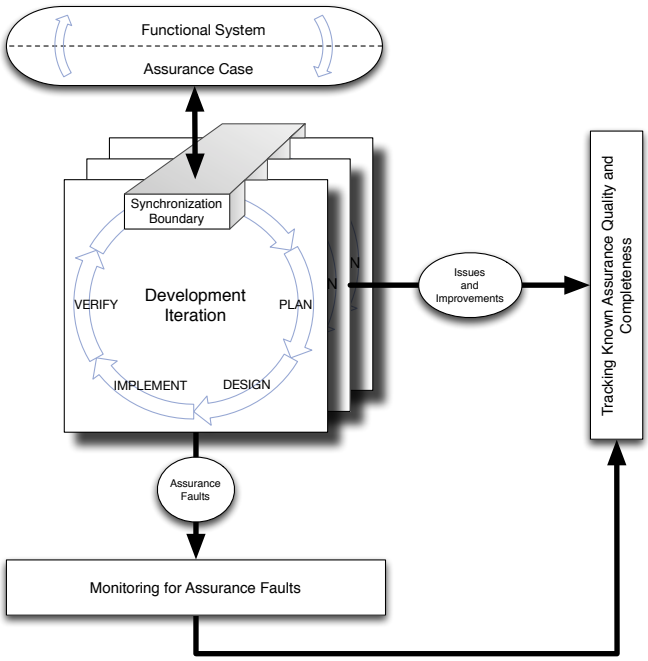

Fig. 11. General CLASS process structure

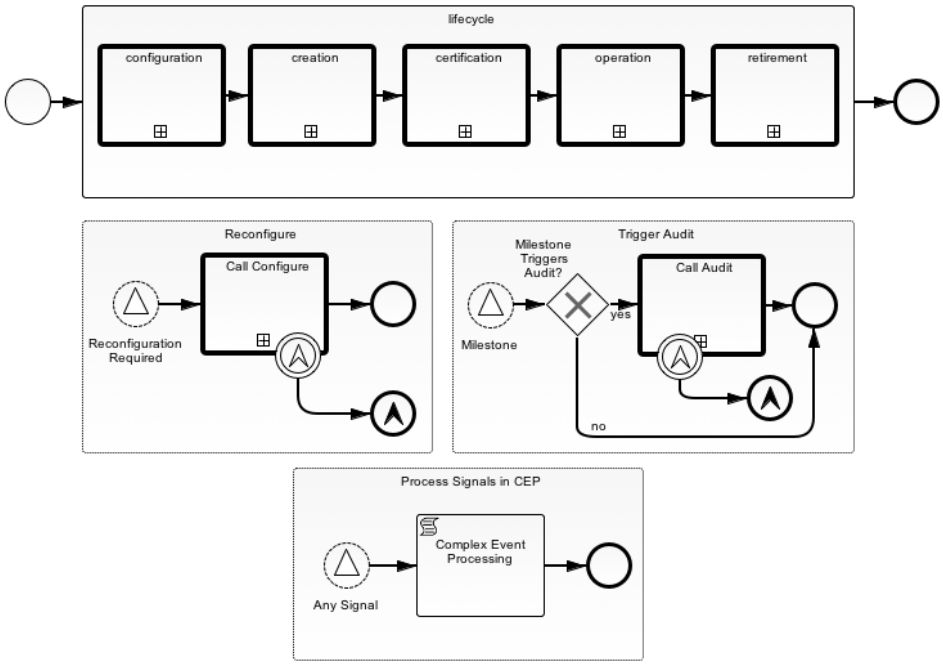

Fig. 12. CLASS process lifecycle as a BPMN2 process. A key to the graphics used in BPMN2 is available at: http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf. A tutorial on BPMN2 is available at: https://camunda.org/bpmn/tutorial/

The use of a formal language to describe process structures brings many benefits. To explore this idea, many process ideas were defined in CLASS phase 2 using BPMN2. An example of the use of BPMN2 is shown in Fig. 12.

The empirical studies carried out during CLASS phase 2 were designed to assess the process ideas that were developed. Many of the details of these process ideas, defined in BPMN2, are documented in the empirical studies report supplied to the sponsor under separate cover.

CLASS Safety Case Development Process

A specialized element of CLAS phase 2 was the development of the safety case development process. This process was developed to provide a framework for handling the interactions between all of the various stakeholders in the development of large safety cases.

Inevitably, modern aviation systems will require large safety cases, and CLASS is a complex entity designed to support the development of large safety cases. The safety case component of system development requires a substantial team of engineers to interact in complex ways over extended periods of time with a wide variety of dependencies between their various activities. Just as with any large development, uncertainties arise over what actions are needed, the effort level required, the approach to be followed, and so on.

The CLASS safety-case development process (a sub-process within an instanceCLASS) is based on a Spiral model [6]. The Spiral model captures and expands on both the practical and conceptual aspects of the established *Early and Often* development process [1]. The CLASS Spiral model also deals with the process elements that arise because of the anticipated scale of the safety cases of interest.

A CLASS instance has to support the four major phases of the safety-case lifecycle:

- Creation.
- Certification.
- Operation
- Maintenance.

In providing this support, the instanceCLASS development sub-process has to accommodate multiple inputs including:

- Artifacts used in developing the safety case including required content and formats necessary for certification.
- Questions and argument challenges resulting from the deliberations of the certifying authority.
- Data observed during operation that relates to the execution-time checking of safety claims.
- Maintenance requirements.

In addition, the instanceCLASS development sub-process must be able to inform engineers in a variety of ways about the exact state of the lifecycle. As part of this requirement, instanceCLASS must be able to provide alerts to conditions defined by engineers as requiring attention.

To address these lifecycle requirements, a major component of the instanceCLASS design is the system's workflow infrastructure. This structure is the basis upon with the Spiral model is constructed. The workflow infrastructure does not control or define the workflow. Rather the workflow infrastructure provides status information about:

- The safety case and all associated artifacts. This status information allows engineers throughout the lifecycle to examine the state of the safety case in order to support assessment.
- The work flows in the lifecycle. This status information identifies the crucial parameters of the workflow, the steps required to complete it, and the state of each workflow.

In order to represent the relationships between components (workflow items) tracked in the workflow fully, a fairly complex data model is required.

Arguments are decomposed into related components that are described in the workflow as argument tasks. These tasks are typically smaller than a single argument, but may include sub-graphs from multiple, related arguments. For example, tasks might include:

- Top-level argument structure.
- Functional hazards.
- Hardware boundaries.
- Software functions.

Some of these — such as software functions — will require direct evidence whereas others — like the top-level argument structure — might not. Each argument task has a well-defined set of activities that are required to complete it:

1. **Drafting**: Development of initial argument structure, in consultation with domain experts.
2. **Assignment**: Identification and association of ownership teams (consisting of a regulator and a subject-matter expert) who will be responsible for completing and overseeing the development and approval of the sub-graph associated with the argument task.
3. **Pre-validation**: Acceptance of ownership and initial vetting of the draft argument structure.
4. **Validation**: Elaboration and review of the argument structure, to ensure that the logic is complete and correct.
5. **Evidence Negotiation**: Development of evidence requirements for terminal or leaf claims in the argument structure.
6. **Approval**: Acceptance of evidence and, therefore, of the top-most claims in each sub-graph associated with the argument task.

During the assignment activity, several ownership teams might be associated with smaller sub-graphs of the argument task's GSN; each argument task may therefore have many instances of activities 3 through 6.

One of the tools in the CLASS SCT, the Workflow Management System, allows details of the workflow to be determined.

CLASS Resource Repository

As part of CLASS phase 2, a number of new resource concepts were developed, and these were added to the CRR. The enhancements to the CRR are illustrated on the left of Fig. 13.

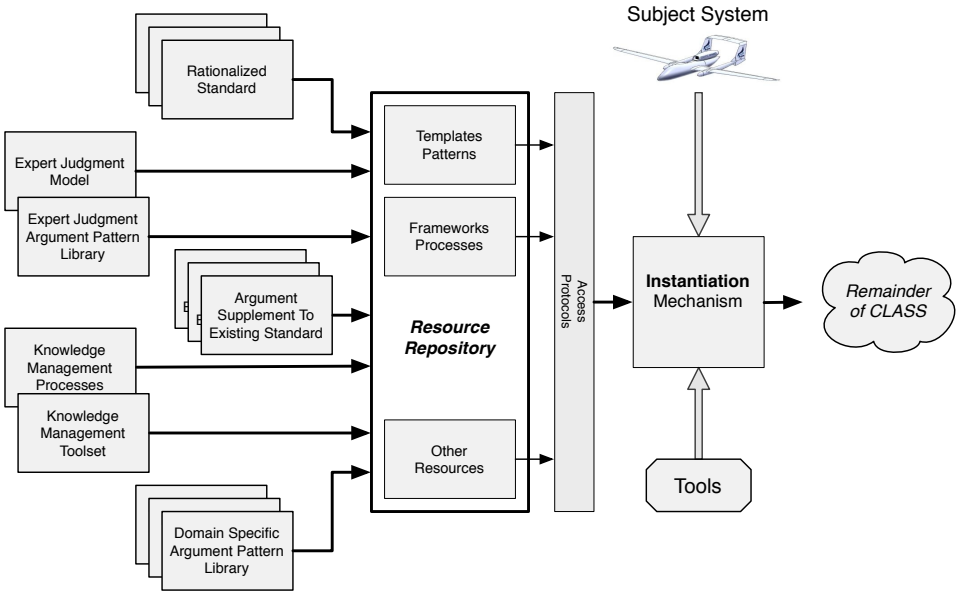

Fig. 13. Enhancements to the CLASS Resource Repository.

The resource concepts added to the CRR in phase 2 were:

- **An initial model of *expert judgment*.** The model is based upon the expert-judgment processes used by: (a) the medical profession in diagnosis, and (b) the FAA in software approval. The model is explained in depth in paper number 6 in section *Bibliography of Published Papers* below.
- **Illustrative samples of a rigorous argument added to the RTCA standard DO-178B.** This material was developed to improve the access to assurance and safety arguments by practicing engineers. The concept is explained in depth in paper number 9 in section *Bibliography of Published Papers* below.
- **The concept of the *rationalized standard*.** A rationalized standard is designed to provide a stronger basis for defining safety standards. The concept is explained in depth in paper number 2 in section *Bibliography of Published Papers* below.
- **An initial *knowledge management* process concept.** The concept was expanded and enhanced in CLASS phase 3 as was a preliminary prototype support toolset. The phase 3 versions are explained in depth in paper number 10 and paper number 3 in section *Bibliography of Published Papers* below.
- **The concept of *domain-specific arguments*.** In a production setting, the CRR would contain sample domain-specific arguments to support various domains of interest. The concept of domain-specific arguments was developed extensively in CLASS phase 3, and small examples developed to assess and illustrate the concept.

The intent throughout this research project was to enhance the CRR concept and content as new aspects of CLASS were developed. In phase 2, the major enhancements outlined about moved the CRR from a document-oriented repository to a process-oriented repository in which documents were associated with useful processes and process fragments.

Expert Judgment

The notion of expert judgment is central to the application of inductive logic, and CLASS relies upon expert judgment. During CLASS phase 2, the idea of examining and possibly refining the concept of expert judgment was introduced. Expert judgment is a critical component of virtually all safety and assurance cases. Engineers appeal to expert judgment whenever an assessment is required but there is no quantitative basis for the assessment.

During CLASS phase 2, several existing techniques for expert judgment were analyzed and the core elements of two were used to build an initial model of expert judgment for CLASS. The initial model was expanded in CLASS phase 3.

The concept is explained in depth in paper number 6 in section *Bibliography of Published Papers* below.

Integration with Existing Standards

CLASS introduces numerous new concepts and technologies. An important question that became prominent in CLASS phase 2 was how CLASS in general and safety cases in particular could be integrated into existing practice.

An initial effort was made to investigate this question by seeing how a safety case could be integrated into an existing software assurance standard. Given the domain of interest for CLASS, the standard chosen was RTCA DO-178B [7].

A mechanism was developed to provide a bridge between existing practices that are based on ad hoc processes associated with DO-178B and the use of a safety case as a fundamental engineering asset. The mechanism was developed with extensive consultation with the FAA DERs who are part of the research team. The major conclusion of the work is that the poorly defined objectives in DO-178B/C and the associated ad hoc audit mechanisms can be placed on a rigorous basis by requiring that developers prepare rigorous arguments to justify belief in their system meeting the standard's objectives.

The results of the exercise were quite positive. The concept is explained in depth in paper number 9 in section *Bibliography of Published Papers* below.

Safety Case Condition Monitoring

The CLASS monitoring concept was expanded considerably in CLASS phase 2. The emphasis in CLASS phase 2 was on process, and the importance of following processes completely and correctly is well understood.

Given that CLASS employs a monitoring system, the idea of expanding monitoring to cover conformance to prescribed processes was introduced in CLASS phase 2. The change required very little modification to the design of the monitoring system, because its original design was completely general. The only significant enhancement that was required was the development of new sensors.

Despite this change, the potential of process monitoring was not fully understood in CLASS phase 2. In phase 3, the process-monitoring concept was enhanced by driving the monitoring off of the basic CLASS analysis mechanism. This concept is explained in the next section.

CLASS Phase 3

During the development of CLASS phase 2 extensive interactions with practicing engineers were undertaken in empirical studies of the process-related ideas. This interaction led to development of various argument and process artifacts that were used to evaluate the principles of CLASS phase 2. Based on the results of the empirical studies conducted during CLASS phase 2 (and partially from those in CLASS phase 1), we concluded that: (a) the focus of CLASS on process was too restrictive, and (b) the notion of basing assurance on a safety case as usually practiced was also too restrictive.

The focus of CLASS on process was found to have two fundamental limitations:

- The only "variable" available in the design of CLASS in phase 2 was the process. In other words, in order to effect some desirable outcome, the primary mechanism needed to be a previously defined process.
- Domain experts did not find that extremely detailed process concepts were either easy to understand or easy to follow. More importantly, it was difficult for them to understand

why detailed prescriptive processes were necessary and hence what value these processes had.

The notion of basing assurance on a safety case as usually practiced was found to have two fundamental limitations also:

- The value of safety cases to domain experts was not as high as we expected. The acceptance of safety cases within the community depends, in part, on their perceived value. We identified assurance arguments that were being created and used by domain experts entirely separately from safety cases.
- As intended, the emphasis in CLASS phases 1 and 2 was on the safety case. This emphasis we found to be too rigid in that all of the tools and technologies were limited to a single piece of information, the safety case. We identified many opportunities for generalizing the concepts being developed to be a broader approach to the knowledge of domain experts.

With these observations, CLASS phase 3 was developed based on a *domain-centric* model. This model easily encompassed everything that had been developed in CLASS phases 1 and 2. The safety case is merely one form of information involving several domains. Similarly, processes and process models are merely a different form of information, again involving several domains.

Once the notion of a domain-centric model was established, then the idea of making the model an "open" model immediately emerged. By open we mean that:

- Access to domain information is not and need not be restricted except to protect proprietary interests. Good ideas, examples, better ideas, and various artifacts can be added to the available information.
- Domains and domain information is at the heart of approval. Standards and their interpretation are merely examples of domain information. A standard, viewed as information can be approved by a regulating agency, consulted by engineers, and discussed by engineers based on experience. Interfaces and controls to: (a) facilitate information access, (b) provide repositories of information related to information, and (c) facilitate social processes of discussion and debate are possible.

CLASS phase 3 pursued these ideas in order to explore their potential. Clearly, the CLASS Resource Repository had a significant role to play in the enhancements undertaken in CLASS phase 3. The CRR is the domain information repository, and so the idea of a domain-centric model for CLASS makes the CRR the focus.

In the original CLASS concept, the CLASS Resource Repository (CRR) was designed to provide resources that users could exploit in the development of safety cases. Subsequently, that idea was broadened slightly to include materials that were viewed as central to the aerospace application domain but still limited to resources relevant to safety cases. In CLASS phase 3, the CRR:

- Contains a broad range of resources.
- Is a central repository of artifacts of all types including examples of argument templates, processes, tools, references, standards, and questions.
- Provides a central facility for comment and discussion about the various items in the CRR. Such discussions might lead to updates to other domain resources, and thereby provides a mechanism for keeping domain resources accurate and up to date.

The major advantages that these changes provide are:

- Users of the CRR will be able to access much, perhaps all, of the resources needed to undertake a system development.
- All of the resources can be kept up to date and updated as necessary, rather than remaining available but either not updated or only updated only after extensive delay.
- Useful variants of resources can be developed over time and made available to the community.

Process Model

To support the enhanced CRR and the concept of generalized domain information resources, the high-level CLASS process was generalized. The generalized form is shown in Fig. 14.

The process concept breaks down into two major components: Domain Engineering shown in the lower half of Fig. 14, and Product Engineers shown in the upper half of Fig. 14. On the left of the figure is the CLASS Resource Repository. The CRR has now become the holder of a wide variety of resources designed to support both Domain Engineering and Product Engineering. On the right of the figure is the new Process Support Toolset. The toolset is also designed to support both Domain Engineering and Product Engineering.

An important characteristic of the Domain Engineering model is that process and assurance are treated equally. This characteristic is illustrated in the figure by the notion that properties are spread across the combination of assurance and process. The original goal with CLASS was to focus on safety by making the system safety case the focus on process activities. This goal has been generalized by taking all the CLASS concepts and generalizing their focus.

A second important characteristic of the Domain Engineering model is the notion of constant assessment and improvement. This characteristic is illustrated in the figure as the iteration denoted by the arrow loop.

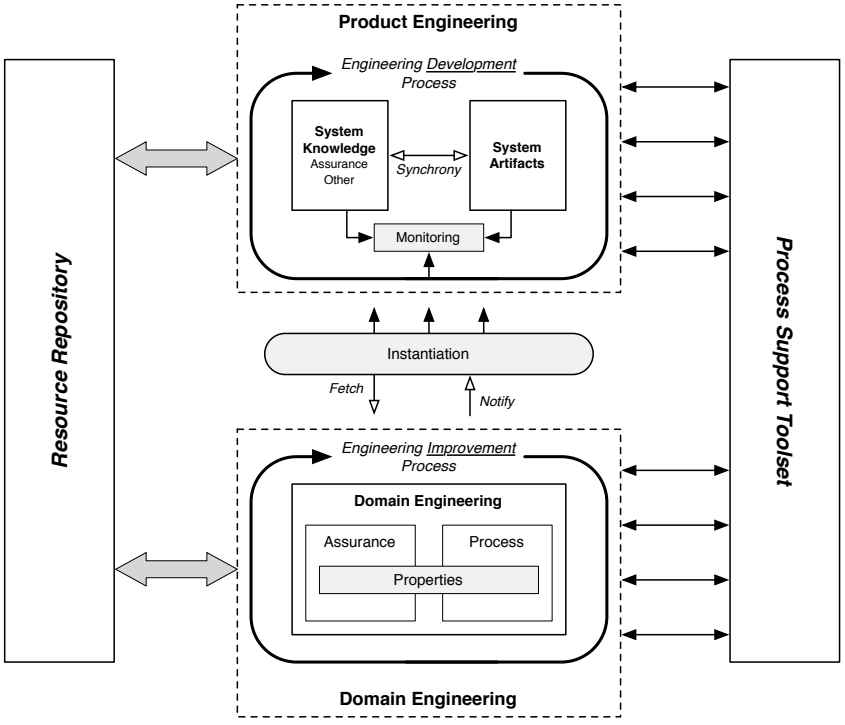

Fig. 14. Enhanced CLASS process concept with tool support and interface to CRR.

An important characteristic of the Product Engineering model is that engineering development and assurance are treated equally. The original goal with CLASS was to focus on safety by making the system safety case the focus on process activities. Taking all the CLASS concepts and expanding their focus have generalized this goal.

Expert Judgment

In CLASS phase 2, the role of expert judgment was introduced as a fundamental element of CLASS. This was done, because of the crucial role that expert judgment plays in the assurance of safety-critical systems.

In CLASS phase 2, expert judgment was characterized with a model that had been derived from two other domains. When applying that model, a weakness was found that limited the model's value. The problem was that the model was entirely prescriptive and provided no mapping from those prescriptive elements to the desired judgment accuracy.

This problem is identical to the problem with most existing, prescriptive safety standards: no rationale is provided for the guidance. This problem with existing safety standards was the motivation for introducing the concept of rationalized standards in CLASS phase 2.

The same solution is being pursued to complete the model of expert judgment. That work is incomplete.

Analysis Framework

CLASS is designed to support the development of safety-critical systems. Clearly, an important goal of the development process is to maximize the desired dependability of the subject system. To do so, CLASS has to be designed to minimize the defects in the subject system. This goal requires the introduction into CLASS of mechanisms to achieve this minimization.

A question that arises immediately is: what techniques should be used and how would developers know of their efficacy? The CLASS Analysis Framework answers this question.

The CLASS Analysis Framework is based upon the Filter Model introduced by Steele and Knight [5]. The framework is designed to analyze CLASS itself so as to provide explicit support for rigorous statements about properties of the subject system. The framework principle is to enable examination of CLASS based on a list of desired properties of the subject system and to determine the extent to which class precludes the introduction of defects that could make a property false. If the framework can show that CLASS avoids the introduction of or enables the complete elimination of a class of defects, then that result translates immediately into a property of the desired system.

The principle upon which the framework is based is to treat CLASS itself as a safety critical system. In other words, allowing a defect to be introduced into or allowing a defect to remain in a subject system is considered an accident. With that concept in place, the framework can apply all of the techniques used in safety engineering to reduce the residual risk associated with CLASS. Techniques that can be applied include hazard identification, hazard analysis, fault-tree analysis, failure-modes-effects-and-criticality analysis, hazard-and-operability analysis, and so on. The concept upon which the framework rests is shown in Fig. 15.

CLASS has eliminated the need for indirect evidence by introducing the framework. By analyzing CLASS in the manner outlined above, everything undertaken in CLASS leads to rigorous statements about the subjects system and the necessary evidence to support the associated claim. Thus, the framework feeds directly and immediately into the safety case for the subject system.

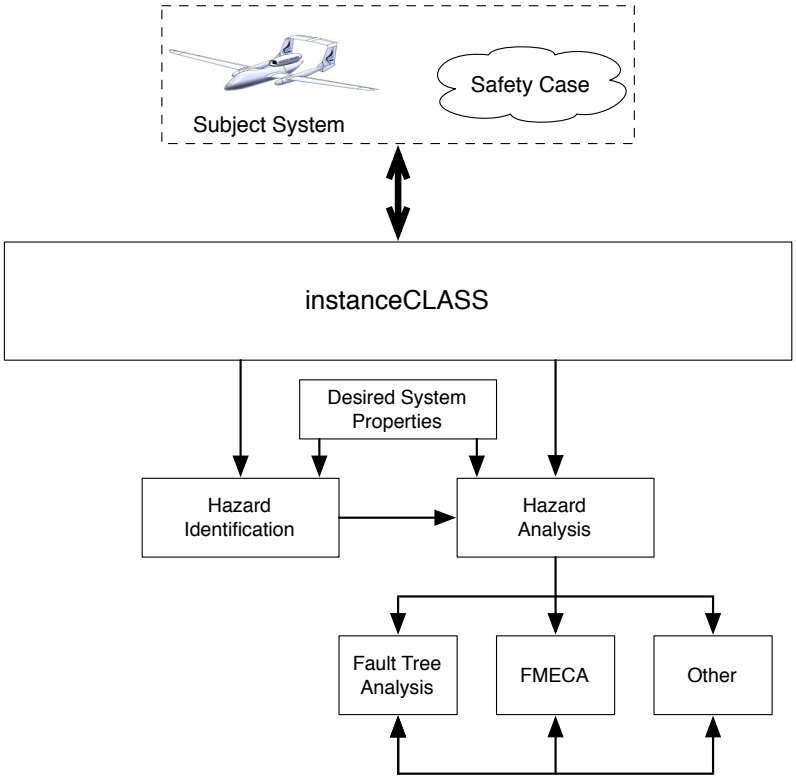

Fig. 15. CLASS Analysis Framework principles

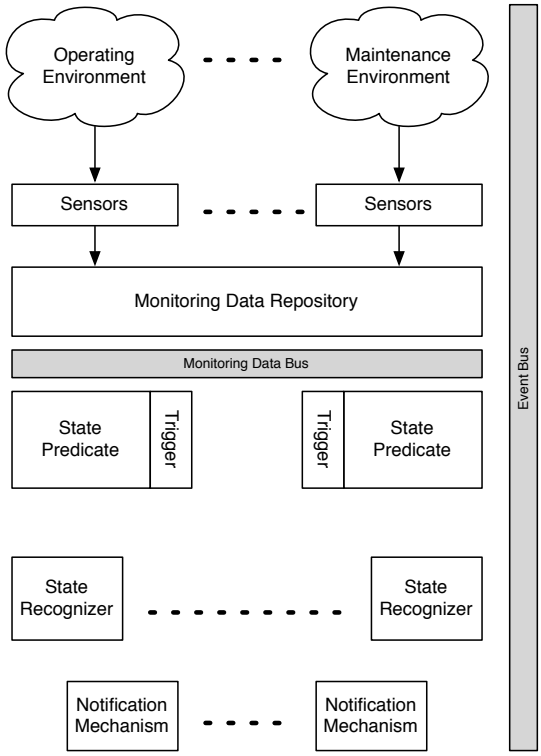

Fig. 16. The CLASS monitoring system architecture.

Development Process Monitoring

A high level of assurance is required that all process steps been followed correctly and in the correct order if the process evidence of an instanceCLASS is to be treated as direct evidence. This assurance requires monitoring, and this indicated the need to expand the CLASS monitoring system comprehensively to include process monitoring.

The architecture of the system is illustrated in Fig. 16. All relevant environments (development, operation, maintenance, etc.) are instrumented with sensors that are read according to a prescribed schedule. Sensor data is placed in a data repository that is referenced as necessary by the state predicates. These predicates define states of interest to the monitoring system, and are enabled/disabled by a triggering mechanism.

State predicates define events if they recognize a state of interest, and these events are supplied to a set of state recognizers by the event bus. State recognizers are finite-state machines that pass through states upon receipt of predefined events. When they enter a final/terminating state, they emit one of more events that cause prescribed notification mechanism(s) to act.

The CLASS monitoring system is explained in depth in paper number 8 in section *Bibliography of Published Papers* below.

Certification

CLASS addresses certification as a fundamental part of the entire lifecycle from system concept formation to system decommissioning and bases certification on the system safety case. This explicit attention to certification across all of the lifecycle phases is manifested in the following considerations:

- The development phase of the lifecycle must ensure that artifacts developed support certification throughout the remainder of the lifecycle. Changes in either the system or the operating environment will require re-certification.
- The operation and maintenance phases of the lifecycle must be prepared to support certification of revised versions of the subject system irrespective of the cause of the revision.
- The primary focus of certification will be with the lifecycle phase following development and preceding deployment. The focus must be supported by both artifact structures and processes that maximize the value and minimize the cost of certification to all of the system stakeholders and the regulating agency.

Although the safety case is the basis of certification, the entire certification process is informed by the likely need to meet the requirements of a regulating agency. In the case of aviation systems (the primary application domain of interest in CLASS), the regulating agency involved is the Federal Aviation Administration (FAA). CLASS provides a framework within which adequacy of protection of the public interest as defined by the FAA can be determined. CLASS is designed specifically and deliberately to ensure that elements of CLASS can be adjusted as necessary to meet the needs of the FAA.

In CLASS, the safety case combines and structures many items that would normally be identified and elicited separately as part of existing certification mechanisms. By definition, the safety case documents the rationale for belief in the adequacy of the safety of the subject system, and maintenance of the safety case across the complete lifecycle facilitates the requisite certification activities.

Since the CLASS Analysis Framework allows indirect evidence about the subject system to be eliminated, the doubts about the properties of a system that are engendered by indirect evidence are avoided. The CLASS monitoring mechanism provides a high level of confidence that the expected properties established by the Analysis Framework will be true. Thus, the starting point for certification is a safety case for the subject system with properties established by direct evidence in which stakeholders and certifiers can have high confidence.

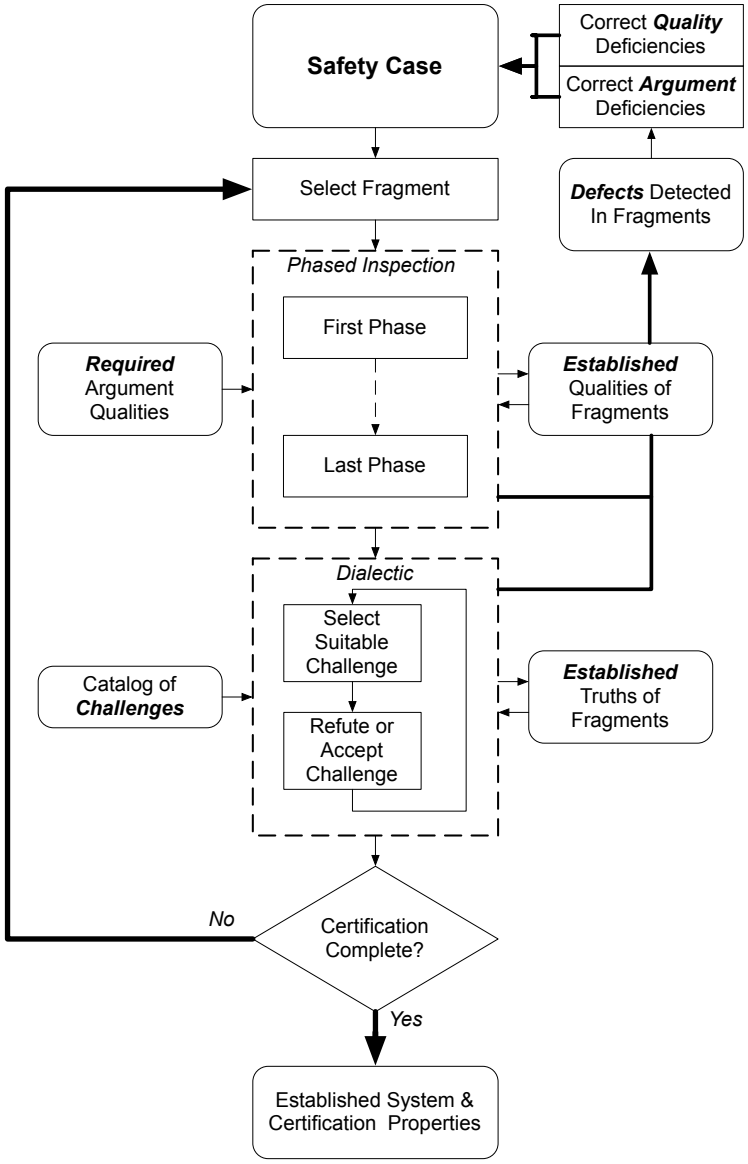

Fig. 17. The safety case audit process.

The CLASS certification approach is based on the work of Graydon, Knight, and Green [8]. The approach is composed logically of two significant audit phases that are expected to be accomplished by experts from the regulating agency. For the FAA, these experts would be licensed Designated Engineering Representatives (DERs). The audit phases are:

- **Process Audit**. By design, CLASS precludes faults in a wide variety of categories. In other words, provided CLASS is used properly, the resulting system should be demonstrably free of faults in certain categories. The details of the categories of faults that are precluded together with all the associated evidence for the subject system are contained in the safety case. Despite the role of both the Analysis Framework and the Monitoring System, an audit is required to confirm to the extent possible that the claimed properties of the subject system are present.
- **Safety Case Audit**. By definition, the safety case describes the rationale for belief in the safety of the subject system. Thus the second audit need not examine the product. Rather, the safety of the subject system should be argued in a compelling manner by the safety case. The second audit is of the safety case. Many aspects are examined in the

audit including a variety of simple yet important properties. The main emphasis of the product audit is the assessment of the degree to which the safety argument has the essential properties of being comprehensive, valid, and compelling. In CLASS these properties are established by a rigorous inspection process based on the technology of Phased Inspections originally design for software inspection.

The overall organization of the safety case audit is illustrated in Fig. 17. Full details of the audit process are available elsewhere [8].

Empirical Studies

As noted throughout this report, various empirical studies have been conducted at various times and with various goals throughout the research project's duration. The results of these studies have affected CLASS profoundly at a variety of stages. Many specimen artifacts were created by these empirical studies.

A detailed report of the most significant empirical studies has been supplied to the sponsor under separate cover. Some of the key results in the field of domain arguments are presented in paper number 1 in section *Bibliography of Published Papers* below.

Bibliography of Published Papers

The majority of the scientific results from the project reported here are documented in conference papers and technical reports. This section provides a list of those publications together with the abstract for each one. In some cases more than one paper discussed the same topic. This apparent duplication is either because the later paper is a refined version of an earlier paper, or because the papers present a wider study of the topic. All of the papers are cited here for completeness and are available through the Dependable Computing website: http://www.dependablecomputing.com/class

In brief, paper 1, presents and analyzes the notion of domain arguments. Paper 2 introduces a new concept for safety standards called rationalized standards. The design and implementation of the CLASS server toolset are presented in paper 3. A rigorous approach to expert judgment is discussed in paper 4, and paper 5 is an annotated bibliography on expert judgment. Paper 6 presents a comprehensive model of expert judgment, and paper 7 is a summary of CLASS. The CLASS monitoring system is discussed in paper 8, and paper 9 presents an approach to integrating an assurance case into the structure and use of the RTCA DO-178B software assurance standard. Paper 10 introduces the notion of an assurance knowledge ecology upon which CLASS phase 3 is built. The mechanism of certification as it might be instantiated for an agency such as the FAA is presented in paper 11. Paper 12 is a survey of the literature on lifecycle technology. Paper 13 presents an early version of the CLASS monitoring system, and paper 14 discusses the mechanism by which assurance is analyzed in CLASS. Finally, paper 15 describes the SCT safety case toolkit.

Details of the papers follow.

1. Jonathan Rowanhill and John C. Knight, *Domain Arguments in Safety Critical Software Development*, submitted to 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, Canada (October 2016)

   Abstract. This paper explores domain arguments—arguments about why techniques, processes, and designs possess properties as believed by their domain experts. An elicitation technique for their recovery from domain documents is presented. This is followed by demonstrated application of the technique to several domain artifacts from aviation engineering. The elicited arguments are presented and analyzed for their properties. The inherent importance of such arguments is discussed as well as their potential contribution to system assurance arguments such as the safety case.

2. John C. Knight and Jonathan Rowanhill, *The Indispensable Role Of Rationale In Safety Standards,* International Conference on Computer Safety, Reliability and Security SAFECOMP, Trondheim, Norway (September 2016)

   Abstract. In this paper, we argue that standards, especially those intended to support critical applications, should define explicitly both the properties expected to accrue from use of the standard and an explicit rationale that justifies the contents of the standard. Current standards do not include an explicit, comprehensive rationale. Without a rationale, the use, maintenance, and revision of standards is unnecessarily difficult. We introduce a new concept for standards, the rationalized standard. A rationalized standard combines: (a) an explicit goal defining a property desired for conformant systems, (b) guidance that, if followed correctly, should yield an entity with the property stated in the goal, and (c) the rationale showing the reasoning why there is assurance with reasonable confidence that a conformant entity will have the property defined by the goal. We illustrate the utility of an explicit rationale using an existing safety standard, ISO 26262.

3. Jonathan Rowanhill, *CLASS Server Toolset: Design and Implementation*, Dependable Computing Technical Report 2016-01 (May 2016)

   Abstract. The goal of the Comprehensive Lifecycle for Assurance of System Safety (CLASS) is to obtain assurance for relevant elements of a domain over their lifecycle. The Server Toolset provides an infrastructure that supports all of the information in a domain, i.e., an area of knowledge. The notion of “support” in this case means supporting: (a) all aspects of information management, (b) all phases of the system lifecycle, and (c) all elements of assurance, including the CLASS concept of synchrony. The toolset combines state-of-the-art software project management and knowledge capture and management technologies with custom software and powerful tools developed elsewhere.

4. Patrick McGee and John Knight, *A Rigorous Approach To Expert Judgment In The Engineering Of Safety-Critical Systems*, pending.
5. Patrick McGee and John Knight, *An Annotated Bibliography of Expert Judgment*, University of Virginia, Department of Computer Science, Technical Report CS-2016-02 (May 2016)

   Abstract. Expert judgment is an important element of decision making in many areas of engineering. Although quantitative methods are usually preferred in assurance, necessary probabilities frequently cannot be determined and then engineers have to resort to the judgment of experts. Obviously, this situation leads to concern about how accurate the judgments of experts are. Accurate judgments are particularly important in systems for which the consequences of failure are high, such as security- and safety-critical systems. This report is an annotated bibliography on expert judgment.

6. Patrick McGee and John C. Knight, *Expert Judgment in Assurance Cases*, 10th IET System Safety and Cyber Security Conference, Bristol UK (October 2015)

   Abstract. Quantitative methods are usually preferred in safety assessment, but, frequently, necessary probabilities cannot be determined accurately. As a result, expert judgment plays a significant role in safety arguments and system deployment decisions. Trust in expert judgment is based on the assumption that the expert’s experience, knowledge, and training will enable the expert to make a suitable decision. The record indicates, however, that this assumption does not always hold and that decisions tend to be ad hoc and unprincipled. In this paper we present an analysis of expert judgment and a rigorous process for utilizing expert judgment in safety arguments.

7. John Knight, Jonathan Rowanhill, Anthony Aiello, and Kimberly Wasson, *A Comprehensive Safety Lifecycle*, ASSURE 2015: The 3rd International Workshop on Assurance Cases for Software-Intensive Systems, Delft, The Netherlands (September 2015)

   Abstract. CLASS is a novel approach to the safety engineering and management of safety-critical systems in which the system safety case becomes the focus of safety engineering throughout the system lifecycle. CLASS expands the role of the safety case across all phases of the system's lifetime, from concept formation and problem definition to decommissioning. Having the system safety case as the focus of safety engineering and management only has value if the safety case is properly engineered and appropriately consistent with the system. To achieve these properties, CLASS requires that a system and its safety case be regarded as a single composite entity, always linked and always correctly representing one another. CLASS introduces new techniques for the creation, approval and maintenance of safety cases, a rigorous analysis mechanism that allows determination of properties that relate to defect detection in subject systems, and a set of software support tools.

8. John Knight, Jonathan Rowanhill, and Jian Xiang, *A Safety Condition Monitoring System*, ASSURE 2015: The 3rd International Workshop on Assurance Cases for Software-Intensive Systems, Delft, The Netherlands (September 2015)

   Abstract. In any safety argument, belief in the top-level goal depends upon a variety of assumptions that derive from the system development process, the operating context, and the system itself. If an assumption is false or becomes false at any point during the lifecycle, the rationale for belief in the safety goal might be invalidated and the safety of the associated system compromised. Assurance that assumptions actually hold when they are supposed to is not guaranteed, and so monitoring of assumptions might be required. In this paper, we describe the Safety Condition Monitoring System, a system that permits comprehensive yet flexible monitoring of assumptions throughout the entire lifecycle together with an alert infrastructure that allows tailored responses to violations of assumptions. An emphasis of the paper is the approach used to run-time monitoring of assumptions derived from software where the software cannot be easily changed.

9. John Knight, Jonathan Rowanhill, Uma Ferrell, Alec Bateman, Neha Gandhi, *Integrating An Assurance Case Into DO-178B Compliant Software Development*, 34th IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), Prague, Czech Republic (September 2015)

   Abstract. Software assurance in the aviation industry is closely tied to compliance with RTCA DO-178B. In this paper, we introduce an approach to enhancing the value of DO-178B using a software assurance case. The heart of the approach is to use an assurance case to link the detailed development techniques chosen by developers to the objectives defined in the guidance in a rigorous way. The assurance case documents precisely and explicitly the developers' rationale for stating that the objectives have been satisfied. The use of an assurance case has three major advantages: (1) the contributions to assurance of the technologies used in development are documented and can be examined to determine weaknesses and omissions, (2) the system's stakeholders can scrutinize the assurance case to assess completeness of software assurance, and (3) auditors can guide their analyses of the subject system using the assurance case. We present details of the approach and illustrate the approach by applying it to the planning phase of DO-178B. We show how savings elsewhere in development yielded by the enhancement approach can accommodate the cost of the assurance case itself.
   The approach can be adapted easily to DO-178C.

10. Jonathan Rowanhill and John Knight, *CLASS Assurance Knowledge Ecology*, Dependable Computing Technical Report TR2015-1 (May 2015)

    Abstract. We propose Assurance Knowledge Ecology. By inverting ownership of arguments away from generalist experts and towards domain knowledge experts, ownership and competitive fitness of arguments break down the ad-hoc, expensive, and static nature of system assurance. This model recognizes that domains of engineering already have assurance arguments. They are merely implicit. By encouraging domains to make their knowledge explicit, it can be continuously improved and rendered available to system arguments. Arguments exist and are owned in subject matter experts' domains first, rather than from the needs of individual systems. System arguments, in turn, are derived therefrom and are test cases against domain knowledge. A model of this ecology is presented, and an experiment conducted in explicating domain arguments.

11. Jonathan Rowanhill, John Knight, and Uma Ferrell, *CLASS System Certification*, Dependable Computing Technical Report TR2014-4 (December 2014)

    Abstract. CLASS addresses certification as a fundamental part of the entire lifecycle from system concept formation to system decommissioning, and bases certification on the system safety case. The safety case combines and structures many items that would normally be identified and elicited separately as part of existing certification mechanisms. By definition, the safety case documents the rationale for belief in the adequacy of the safety of the subject system, and maintenance of the safety case across the complete lifecycle facilitates the requisite certification activities. CLASS includes an Analysis Framework that allows indirect evidence about the subject system to be eliminated, and so the doubts in the properties of a system that are engendered by indirect evidence are avoided. CLASS also includes a monitoring mechanism which provides a high level of confidence that the expected properties established by the Analysis Framework will be true. In order to conform to the requirements of regulating agencies, CLASS certification includes two audit phases that are expected to be accomplished by experts from the regulating agency. For the FAA, these experts would be licensed Designated Engineering Representatives (DERs).

12. Jonathan Rowanhill, *CLASS Lifecycle Technology Survey*, Dependable Computing Technical Report TR2014-3 (December 2014)

    Abstract. In this paper, we review the state of the art in lifecycle technology and use the results to analyze early CLASS prototypes. The result of the analysis led to substantial modifications to CLASS. An engineered system can be modeled with a lifecycle describing the temporal phases of its existence. Engineering systems with safety properties must consider safety of the system in each and every phase. Thus, a lifecycle model of a system is an excellent representation through which to prescribe safety processes and concerns. CLASS is designed to "enhance system safety in which the safety case becomes the focus of safety throughout the system lifecycle". CLASS defines lifecycles for systems with strict safety requirements. The central role of the safety case supports the analytical rigor required to both develop a safe system and reasonably assure safety. With safety cases as the focus, CLASS lifecycles highlight the relationship between the system's safety, safety arguments, and supporting evidence. These arguments and evidence are:

    - Documents: inherently language-based communication between people and tool.
    - Rigorous: structured and well-defined in order to reduce ambiguity and maintain precision. About systems: made to support systems.
    - Safety focused: focused on system safety.
    - Managed: living documents that must be accessed and modified over time.

There are lifecycle models defined separately for each of these attributes. These models have been used to enhance CLASS.

13. Jonathan Rowanhill and John Knight, *CLASS Safety Condition Monitoring System*, Dependable Computing Technical Report TR2014-2 (December 2014)

    Abstract. CLASS safety analysis depends on assumptions about the correct implementation of the development process for the subject system and the associated safety case. In the event that an assumption is false or becomes false during system development, the rationale for belief in a claim within a safety argument might be invalidated and the safety of the associated system compromised. Assumptions are also made within safety arguments, and these assumptions made during development of the safety argument must remain true throughout the lifecycle, especially during operation. Assurance that both types of assumptions actually hold when they are supposed to is not guaranteed, and so monitoring of assumptions is essential. Monitoring the development of a system and monitoring the system itself during operation to check for violations of assumptions made in the system's safety argument is an important aspect of the system safety lifecycle. The CLASS Safety Condition Monitoring System provides comprehensive monitoring of the assumptions made in both the system development process and the system's safety argument together with an alert infrastructure that allows flexible responses to violations of assumptions.

14. Jonathan Rowanhill and John Knight, *CLASS Analysis Framework*, Dependable Computing Technical Report TR2014-1 (December 2014)

    Abstract. CLASS is designed to support the safety assurance of a system over the system's entire lifecycle. Major goals of CLASS are: (a) to provide sufficient structure to the process that faults arising in the target system from lifecycle process defects are reduced to the extent possible, and (b) to include within the lifecycle process techniques and methods that avoid or eliminate classes of defects. In this paper, we present the CLASS Analysis Framework, a framework designed to determine the extent to which CLASS achieves these goals. The theory upon which the Analysis Framework is based is a technique known as the Filter Model that was developed in separate research. A preliminary application of the technique to a hypothetical CLASS instantiation is described.

15. M. Anthony Aiello, Ashlie B. Hocking, John Knight, Jonathan Rowanhill, *SCT: A Safety Case Toolkit*, International Workshop on Assurance Cases for Software-intensive Systems (ASSURE 2014), Naples IT (November 2014)

    Abstract. SCT is a safety case toolkit designed to support the development and maintenance of safety cases for large, safety-critical systems. SCT supports safety case development by providing facilities to manage the file structure associated with the safety case, editors for various notations including GSN, and a build system that creates a custom web site to store the safety case. The web-based representation of the safety case includes a variety of features for safety case examination including comprehensive hyperlinking of elements, a GSN viewer, an argument index, and various custom reports.

References

1. Timothy Kelly, "Arguing safety — a systematic approach to safety case management," Ph.D. dissertation, University of York (1999).
2. UK Ministry of Defence, "Defence standard 00-56-1 issue 6. Safety management requirements for defence systems: Part 1 requirements," UK Ministry of Defence (April, 2015).

3. Patrick Graydon, John Knight and Elisabeth Strunk, “Assurance Based Development of Critical Systems”, Proceedings: 37th IEEE/IFIP International Conference on Dependable Systems and Networks, Edinburgh, Scotland (June 2007)
4. Patrick Graydon and John Knight, “Process Synthesis in Assurance Based Development of Dependable Systems”, Proceedings: 8th European Dependable Computing Conference Valencia, Spain (May 2010)
5. Panayotis Steele and John Knight, “Analysis of Critical System Certification”, Proceedings: 15th IEEE International Symposium on High Assurance Systems Engineering, Miami, FL (January 2014)
6. Barry Boehm, “A spiral model of software development and enhancement”, IEEE Computer, Volume: 21, Issue: 5, pp: 61 – 72 (1988)
7. RTCA Inc, “Software Considerations in Airborne Systems and Equipment Certification”, RTCA Inc., Washington DC (2011)
8. Patrick Graydon, John Knight, and Mitchell Green, “Certification and Safety Cases”, Proceedings: 28th International System Safety Conference, Minneapolis, MN (September 2010)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 04-2017	Contractor Report	

4. TITLE AND SUBTITLE

Comprehensive Lifecycle for Assuring System Safety

5a. CONTRACT NUMBER

NNL13AA08C

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Knight, John C.; Rowanhill, Jonathan C.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

999182.02.50.07.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA-CR-2017-219359

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 62
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

Langley Technical Monitor: C. Michael Holloway

14. ABSTRACT

CLASS is a novel approach to the enhancement of system safety in which the system safety case becomes the focus of safety engineering throughout the system lifecycle. CLASS also expands the role of the safety case across all phases of the system's lifetime, from concept formation to decommissioning. As CLASS has been developed, the concept has been generalized to a more comprehensive notion of assurance becoming the driving goal, where safety is an important special case. This report summarizes major aspects of CLASS and contains a bibliography of papers that provide additional details.

15. SUBJECT TERMS

Argument; Assurance; Lifecycle; Safety

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
U	U	U	UU	39	19b. TELEPHONE NUMBER *(Include area code)* (757) 864-9658