Comprehensive Framework for the V&V of Decision Support System Design

Imen Bourguiba

DRDC Valcartier
Comprehensive Framework for the V&V of Decision Support System Design

Imen Bourguiba
NSERC Visiting Fellow

The scientific or technical validity of this contract report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of the Department of National Defence of Canada.

Defence Research and Development Canada – Valcartier
Contract Report
DRDC Valcartier CR 2013-372
December 2013
IMPORTANT INFORMATIVE STATEMENTS

The scientific or technical validity of this contract report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of the Department of National Defence of Canada.
Abstract

The objective of this report is to introduce a framework supporting the verification and validation (V&V) of decision support systems (DSS) design. In this report, we propose a V&V framework with multi-dimensional facets to assist the decision maker in choosing the most appropriate approach/approaches for the V&V of DSS design. The framework serves as practice guidance for documentation of the V&V of architectural and detailed design. Also, our framework intends to facilitate the communication, and the validation of concepts before system development. Besides, it will be the basis for choosing and assessing models, methods and techniques for V&V in the next steps of a software life cycle.
Résumé

L'objectif de ce rapport est de présenter un cadre supportant la vérification et la validation (V&V) de la conception des systèmes d'aide à la décision (SIAD). Dans ce rapport, nous proposons un cadre avec des facettes multidimensionnelles pour aider le décideur dans le choix de l'approche/approches les plus appropriées pour la V&V de la conception des SIAD. Le cadre sert de guide pratique pour la documentation de la V&V de la conception architecturale et détaillée. En outre, notre cadre vise à faciliter la communication et la validation de concepts avant le développement du système. En plus, il sera la base pour le choix et l'évaluation de modèles, méthodes et techniques de V&V dans les prochaines étapes du cycle de vie du logiciel.
Executive summary

Comprehensive Framework for the V&V of Decision Support Systems Design


The objective of this report is to introduce a framework supporting the verification and validation (V&V) of decision support systems (DSS) design. In this report, we propose a V&V framework with multi-dimensional facets to assist the decision maker in choosing the most appropriate approach/approaches for the V&V of DSS design. The framework serves as practice guidance for documentation of the V&V of architectural and detailed design. Also, our framework intends to facilitate the communication, and the validation of concepts before system development. Besides, it will be the basis for choosing and assessing models, methods and techniques for V&V in the next steps of a software life cycle.

Sometimes money, time, and resources are invested to develop Decision Support Systems (DSS) that do not meet users’ needs. An assessment of DSS users’ benefit before the implementation of the DSS would avoid an unnecessary time, effort or resources, given that it is usually very expensive to fix errors have been developed.

The framework that we propose for the verification and validation of DSS design offers guidance to architects, designers, developers, testers, decision makers who will be using the system, and the V&V team. The framework has four phases, which are the V&V plan, the V&V selection, the V&V execution, and the V&V documentation. The V&V plan outlines the sequence of actions that need to be performed, including an estimate about the time and resources. The V&V plan also includes the candidate methods, techniques, and tools that are available for the V&V activities. For the V&V selection, a multi-dimensional approach is used to select the most appropriate approach/approaches for the verification and validation of DSS design. For the execution phase, the selected V&V methods/techniques are executed at the design phase. The documentation is an essential part of the V&V process. In this phase, the methods/techniques adopted for the execution of the V&V, the results of V&V activities, the domain and the conditions under which the V&V were elaborated, and the errors found during V&V activities should be documented. The documentation is fundamental for future decisions, and also for the knowledge reuse. It assists users in making an informed decision and gives the rationale behind the decisions already made.

Incremental V&V should be conducted thorough the software lifecycle to ensure confidence that the software fits its purposes. In this work, we proposed a framework that supports the V&V of DSS design. Our framework aims to ease the validation of concepts before system deployment, and to enhance the decision making processes. It will be the basis for choosing and assessing models, methods and techniques for V&V in the next steps of the software life cycle such as detailed design, and implementation.

For future work, we aim at applying our generic framework for decision support systems for large distributed systems, which are relevant in the context of military command and control applications.
Sommaire

Comprehensive Framework for the V&V of Decision Support Systems Design


L'objectif de ce rapport est de mettre en place un cadre de soutien à la vérification et à la validation (V&V) de la de conception des systèmes d'aide à la décision. Dans ce rapport, nous proposons un cadre avec des facettes multidimensionnelles pour aider le décideur dans le choix de l'approche/approches les plus appropriées pour la V&V de la conception des systèmes d'aide à la décision. Le cadre sert de guide pratique pour la documentation de la V&V de la conception architecturale et détaillée. En outre, notre cadre vise à faciliter la communication et la validation de concepts avant le développement du système. En outre, il sera la base pour le choix et l'évaluation de modèles, méthodes et techniques de V&V dans les prochaines étapes du cycle de vie du logiciel.

Parfois plusieurs ressources sont investies pour développer des systèmes d'aide à la décision qui ne répondent pas aux besoins des utilisateurs. Une évaluation des besoins des utilisateurs des systèmes d'aide à la décision avant la mise en œuvre du système permettrait d'éviter la perte du temps, efforts, et ressources inutiles, étant donné qu'il est généralement très coûteux de corriger les erreurs des systèmes qui ont été déjà développés.

Le cadre que nous proposons pour la vérification et la validation de la conception des systèmes d'aide à la décision offre des conseils aux architectes, concepteurs, développeurs, les décideurs qui utiliseront le système, et l'équipe de V&V. Ce cadre comporte quatre phases, qui sont le plan, la sélection, l'exécution, et la documentation de la V&V. Le plan de V&V décrit la séquence d'actions qui doivent être effectuées, y compris une estimation du temps et des ressources. Le plan comprend également les méthodes candidates, les techniques et les outils qui sont disponibles pour les activités de V&V. Pour la sélection de la V&V, une approche multidimensionnelle est utilisée pour sélectionner l’approche/ les approches les plus appropriées pour la vérification et la validation de la conception des systèmes d'aide à la décision. A la phase d'exécution, les méthodes/techniques de V&V qui ont été sélectionnées sont exécutées au niveau conceptuel. La documentation est une partie essentielle du processus de V&V. Dans cette phase, les méthodes/techniques adoptées pendant la phase d'exécution sont documentées. Les résultats des activités de V&V, le domaine et les conditions dans lesquelles la V&V ont été élaborées, et les erreurs constatées lors des activités de la V&V doivent être documentés. La documentation est fondamentale pour les décisions futures, ainsi que pour la réutilisation des connaissances. Elle aide les utilisateurs à prendre une décision éclairée et explique le choix derrière les décisions. Déjà la V&V devrait être menée en profondeur durant le cycle de vie du logiciel afin de garantir que le logiciel s'adapte aux fins de l'utilisateur. Dans ce travail, nous avons proposé un cadre de soutien à la vérification et à la validation de la de conception des systèmes d'aide à la décision. Notre cadre vise à faciliter la validation des concepts avant le développement du système, afin d'améliorer le processus de prise de décision. Ce cadre sera la base pour le choix et l'évaluation de modèles, méthodes et techniques de V&V dans les prochaines étapes du cycle de vie du logiciel.

Pour les travaux futurs, nous visons à appliquer notre cadre générique pour les systèmes d'aide à la décision pour les grands systèmes distribués qui sont pertinents dans le contexte des applications de commandement et de contrôle.
# Table of contents

1 Introduction ............................................................................................................................... 1
2 Background ............................................................................................................................... 3
  2.1 The Architectural Design ............................................................................................... 3
  2.2 The Detailed Design ...................................................................................................... 4
  2.3 Decision Support Systems Design ................................................................................. 5
3 Motivation & Problem Statement: .......................................................................................... 11
4 Verification and Validation Framework .................................................................................. 13
  4.1 Plan .............................................................................................................................. 13
  4.2 V&V selection ............................................................................................................. 13
  4.3 V&V Execution ........................................................................................................... 24
  4.4 V&V documentation .................................................................................................... 25
5 Case study ............................................................................................................................... 27
6 Conclusion and future work .................................................................................................... 35
7 References ............................................................................................................................... 37
List of figures

Figure 1: The 4+1 view model ......................................................................................................... 4
Figure 2: Design process of a DSS .................................................................................................. 7
Figure 3: DSS design methodology ................................................................................................. 8
Figure 4: V&V Framework ............................................................................................................ 13
Figure 5: The spectrum of situational methods .............................................................................. 19
Figure 6: V&V of DSS design ....................................................................................................... 25
Figure 7: Example of a multiple criteria (decision matrix) ............................................................ 30
Figure 8: Multiple Criteria Performance ........................................................................................ 32
Figure 9: Process of the V&V of the DSS design .......................................................................... 33
List of tables

Table 1: Criteria Evaluation........................................................................................................... 29
Acknowledgements

The author is grateful to Micheline Belanger for her constant support and fruitful discussions. The author would like to thank the Dockyard Laboratory Pacific, CFB Esquimalt for the kind and warm hospitality where part of this work was completed. This research was conducted thought an NSERC-VF fellowship program.
1 Introduction

Decision Support Systems (DSS) support the complex decision making and problem solving [Shim et al.]. They also help improving the decision quality, shortening the decision time cycle, and minimizing risk components [Guitouni et al; 2008]. DSS adopt tools to support directly or indirectly the decision making. These tools allow the decision maker to manage data, volume and complexity. DSS performance and preference are inherently related to the selected analytical methods, because the failure to select the best method(s) often leads to software that will not effectively support the decision-making process. According to Adelman, there are two groups of persons using and requiring the results of decision support systems. The first is the development team including decision makers who will be using the DSS, designers, developers, and evaluators. The second group is the sponsoring team [Adelman, 1992].

An assessment of DSS users’ benefit before the implementation of the DSS would avoid an unnecessary time, effort or resources, given that it is usually very expensive to fix errors that have been developed. In [Boehm, 1984b], Boehm estimated that the cost of fixing errors significantly increases as the project progresses in its phases. Therefore, if errors are detected in early phases, they can be corrected easier, and at less expense. Accordingly, in this work we propose a framework for the Verification and validation (V&V) of DSS design.

Verification and validation are intended to establish confidence that the software meets its objectives. In the literature there are many definitions of V&V. The most popular definition is the one of Bohem. For verification one should ask: “Am I building the product right”, and for validation: “Am I building the right product” [Bohem, 1984a]. One of the definitions of NASA is: “verification is the confirmation, through objective evidence, that the specified requirements have been fulfilled”, and “validation is the confirmation, through objective evidence, that the system will perform its intended functions. The intended functions, and how well the system performs those functions, are determined by the customer” [Plastow et al., 2009]. According to the IEEE STANDARD - 1012-200, the software verification and validation (V&V) processes consists in checking if the development products of an activity is conform to its requirements, and if the software meets its intended use and user needs [IEEE STANDARD - 1012-200, 2004].

The task of V&V of software products is very substantial [Cukic et al. 1998]. It has been and continues to be an active research area. However, V&V is time and money consuming. For instance, the annual budget for V&V for the NASA Space Shuttle Program for three years from 1999 to 2001 is approximately $3-3.5 million. Perhaps surprisingly that within this huge amount, the V&V is only performed on critical phases and not on all the entire system due some constraints [Zelkowitz and Rus, 2001]. Therefore, the efforts to execute the V&V should not be underestimated. Ideally, it should be applied at each stage of the software life cycle [Sommerville, 2004]. However, due to its costs in time and effort, it is not something regularly done.

In Section 2, we introduce the background of our work. In Section 3, we present the motivation and problem statement. Our V&V framework is depicted in Section 4. In Section 5, we discuss a case study illustrating our work. Finally, in Section 6, we outline the conclusion and provide some ideas for future work.
2 Background

The design is a very crucial task in an engineering product development cycle. The software design provides the structure to any artifact. It decomposes the system into parts, and ensures that these parts fit together to achieve a global goal. The software design phase translates the requirements into detailed design. It encompasses the architectural phase and the detailed design phase. The software design phase has for input the requirements specification, and for outputs the product design specification describing the architecture of the software product and its components; and the detailed design specification defining the inputs, outputs, algorithms, and procedures [Bohem, 1984a]. The software architecture (architectural design) and the detailed design (non-architectural design) are discussed in the following.

2.1 The Architectural Design

The architectural design is also known in the literature as preliminary design, general design, or software architecture. However, there is not a universal definition for such terms [Clements et al, 2002]. It depicts the main components of the system, the relationship between them, the rationale for the system decomposition into components, and the constraints that must be respected by any design of the components. The architectural design phase consists in forming the system structure by decomposing its requirement specifications. The computation of the architectural design from the requirements plays a vital role in the development of critical software.

Software architecture deals with abstraction, decomposition and composition. Software architecture descriptions are commonly organized in multiple views or perspectives. For instance, the 4+1 architectural view model enables to address large and complex architectures [Kruchten, 1995]. It is composed of five main views (see Figure 1).

- The logical view, describes the structure of the system through the main components and the relationship between them. For instance within UML diagrams, class diagrams, communication diagrams, and sequence diagrams are used to represent the logical view.
- The process view captures the dynamic aspects of the system. It describes the system processes, and their communications. Also, this view addresses concurrency and synchronization aspects of the design. Activity diagrams are used to represent the process view of UML diagrams.
- The physical view depicts the mappings of the software components onto the hardware and also the physical connections between these components. The deployment diagram is used to represent the physical view UML Diagrams.
- The development view illustrates the static organization of the software in its development environment. For example, UML component diagram can be used to describe system components, and package diagram can be used to represent the development view.
These views permit to describe an architecture. This description is illustrated by use cases or scenarios which is the fifth view. Use cases or scenarios describe sequences of interactions between objects and processes. Also, they are used to identify architectural elements and to illustrate and validate the architecture design.

The Architecture Description Language (ADL) is used to describe a software architecture. In the literature, we find ADLs that have been developed since the 1990s. The most commonly used ones are: AADL (SAE standard), Wright (developed by Carnegie Mellon), ACME (developed by Carnegie Mellon), xADL (developed by UCI), Darwin (developed by Imperial College London), DAOP-ADL (developed by University of Malaga), and ByADL (University of L’Aquila, Italy) [Taylor et al, 2010].

2.2 The Detailed Design

Detailed design is the process of defining the lower level of the components, modules\(^1\), inputs and outputs of each software component, and interfaces. It consists in transforming the system structure into procedures and algorithms. The detailed design extends the architectural design to the bottom level components. At this phase, the details of the software components, and design decisions of each module are defined. They are documented in the detailed design document. The detailed design document is an output of the detailed design phase. Its purpose is to describe the detailed solution to the problem stated in the software requirements specification document. At the design level, there are some principles that should be considered such as separation of concerns, modularization and abstraction [Parnas and Clements 1986]. The separation of concerns divides the problems into smaller parts. It deals with different aspects of a problem with the focus on each separately. For instance, in software architecture the separation of concerns consists in separating the system’s structure into components and connectors. The modularization divides the software system into smaller pieces called modules. The modules can be grouped into subsystems. Abstraction selects some concepts to represent more general and complex ones. It ignores irrelevant details [Taylors et al, 2010].

In [Feather et al., 2004], the verification and validation at the architectural design consists in checking the architectural design for consistency, performing behavior checking and reachability analysis via simulation.

---

\(^{1}\) A software module is a cohesive collection of data and procedures that provides a set of services to other modules.
For the verification and validation of the detailed design, some design verification tools are required. Such tools should include the following properties:

- Verification of safety and liveness properties
- Verification of user correctness requirements
- Validation of protocol systems (e.g., communication)
- Validation of the design model during simulated execution
- Limited performance verification support
- Traces of logical design errors in distributed systems design
- Reachability analysis at a more refined level
- Detection of race conditions and deadlocks
- Detection of timing violations
- Detection of concurrency errors

### 2.3 Decision Support Systems Design

In [More and Chang, 1980], More and Chang emphasized that it is flawed to apply Management Information Support Systems (MIS) development life-cycle approach for the design of DSS. Indeed, the design of DSS is different, and much more complex than the design of MIS. In their work, they also depicted DSS design difficulties dictated by the organizational and technological constraints. Such difficulties emerge since traditional MIS development life-cycle approach cannot be applied within DSS design.

In this work Moore and Chang introduced four concepts to be used for the design of DSS. The concepts are discussed in the following.

1. **System/problem migration**

   System/problem migration considers the dynamic nature of DSS implementation efforts. In the development life-cycle approach, usually the decision making problem is static. However, for a DSS, the decision process is continuously modified. It is continuously migrating in the mind of the DSS designer and the decision maker.

2. **Subset evolution**

   In a DSS, the decision process is continuously changing in a decision space. The decision space includes the problem setting, the user's preferences, the technology, and the environment. This decision space complexity is augmented by the design space complexity where the designer adds his/her preferences from the hardware and software technology. Therefore, the designer has to control the migration of the decision space and the design space. Hence, the proposed approach consists in evolving system capabilities. Namely, the DSS generates a subset of future capabilities including earlier capabilities and existent requirements in order to take into consideration the decision maker's preferences related to the design approach. The decision maker is the end-user who will be using the DSS.
3. “Hard” and “soft” capabilities

The "hard" and "soft" capabilities are closely related to the "subset evolution" concept. They ensure to have a bug-free conceptual design, and take into consideration the user’s preferences. The “soft” capability suggested considers the development of generalized capabilities, which can be subsequently refined in order to have a bug-free conceptual design. The "soft" capabilities can be replaced by “off the shelf” packages.

4. “Weak-strong” process

The "weak-strong" design makes the distinction between DSS designs considering the user's preferences and others including refinement of the user's decision making process.

In this work Moore and Chang did not provide a step by step procedure for the design of DSS. They simply introduced the previously discussed concepts to be used for the design of DSS.

In [Breton et al.], Breton et al. introduced a human-centric approach to the design of DSS for command and control. They proposed a design process for DSS with three major phases which consist in evaluating, designing, and testing the system. The process is presented in Figure 2. The evaluation phase encompasses the task analysis process and identification of human needs. The design phase includes the identification of design requirements from the identified human needs, and prototyping of the design requirements. Ultimately, the testing phase considers testing and validation of the prototypes and identification of the human performance. The paper discusses different cognitive approaches which are: task analysis, cognitive task analysis, and cognitive work analysis that can be used at the evaluation phase. However, the design and test phase are to be considered in their future work.
In [Raghunathan, 1996], Raghunathan introduced a DSS design methodology. It is composed of phases supported by tools and guidelines presented in Figure 3. The first entry to the DSS design process is the analysis and modeling of the problem domain. At this level the focus is on the domain rather than a specific problem or a specific situation since any DSS implementation satisfying the design, and satisfying different situations occurring in that domain should be considered. At this phase, a structured modeling tool is adopted to develop the structured model representation of the domain. This latter is then transformed into database design and model base design. The transformation is performed via the database and the model base design procedures. After that an integration procedure is applied to have a model containing the relevant data from the database design and the model base design. In the next phase, the database, model base, the integration procedure, the problem and decision making characteristics are used in order to identify the relevant database and model base to support the decision making in the given contexts. Ultimately the specific DSS design is developed, and then implemented. The methodology uses the diagrammatic representation of the model which is quite similar to the entity relationship (ER) modeling, and the Geoffrion’s structured modeling language as a tool. The Geoffrion’s representation of a structured model enables to represent uncertain data with probability distribution in the model.
In [Blair et al., 1997], there are some methodologies such as data/control flow diagrams, decision trees, relational diagrams, inference flows, rule propagation, module function, object diagrams, and Object oriented (OO) analysis and design methodologies that have been used for designing Intelligent Decision Support Systems (IDSS). In the literature, IDSS are also known as intelligent decision systems, expert-
based systems, knowledge-based DSS, and expert support systems. IDSS support the decision making process by adopting artificial intelligence techniques.

In [Farahbod et al., 2008], Farahbod et al. proposed a formal approach to design situation analysis and decision support systems. They adopted two mathematical modeling paradigms which are Abstract State Machines (ASM) and interpreted systems for the design and development of decision support systems. ASM techniques have been successfully used in many industrial applications to model complex distributed systems. In their work, they were combined with CoreASM to implement design specifications through symbolic execution (model checking), simulation and testing. CoreASM is an extensible and executable specification language. It enables the execution of ASM models, and is very useful for rapid prototyping. However, ASMs do not support well the reasoning about knowledge and uncertainty. For that reason, they were combined with interpreted systems in their work. Interpreted systems enable the formal reasoning about knowledge and uncertainty in multiagent systems. Therefore, the combination of ASM and interpreted systems enables to have a comprehensive framework with a precise semantics, and knowledge. The authors presented a case study to illustrate their approach. However, to better assess how this combined approach works in practice, and that the developed software systems design is sound, more real-life complicated case studies need to be considered.
3 Motivation & Problem Statement:

The literature on the V&V of DSS is unfortunately scarce, possibly reflecting the difficulty of the issue [Boreinstein, 1998]. In [Paris et al.; 2004], Paris et al., claimed that the V&V techniques found in the literature cannot support sophisticated functions with an advanced level of intelligence, and where real time decisions are required to detect failures. Therefore, new techniques must be proposed to maintain crew safety and to reduce the cost of operations. In [Schwer, 2006], the aim of the V&V computation mechanics community was to have a V&V standard. However, the ASME guide proposed was definitely neither a standard nor a step by step V&V process that should be followed by users. In fact, the author claimed that such a standard is a project for many years in the future.

In [Maley, 2004], within the C4ISR Architecture Framework, only a set of guidelines, recommendations, and a process for development of architectural descriptions are provided without specifying the methodology and tools.

The successor of C4ISR is DoDAF (Department of Defence Architecture Framework Working Group 2004). It is a standard for documenting system architecture [Clements et al]. It is a multi-viewpoints approach ensuring a common understanding, comparison, and integration of different architectures within organizations, enterprises, and defense agencies. It is organized around data, models, views, and viewpoints. It is mainly a data-centric approach focusing on the storage, collection, and data maintenance.

DNDAF (Department of National Defence Architecture Framework) is an enterprise architecture (EA) framework supporting the development of Integrated Command and Control (C2) capability. When DNDAF version 1.5 was released in March 2008, it was the most advanced military EA frameworks compared to other such as DoDAF (Department of Defense Architecture Framework), MODAF (Ministry of Defence Architecture Framework), TOGAF (The open Group Architecture Framework), NAF (NATO Architecture Framework) and others. In particular, DNDAF represents the DND/CF (Department of National Defence and the Canadian Forces) enterprise. It depicts its components, their relationships, and their roles. It is also describes in terms of views and sub-views. In [Farahbood et al., 2011], a set of guidelines, recommendations, methods, and tools that should be developed in order to improve DNDAF and to come up with a DND/CF EA methodology and framework. Namely, the set of fundamental guiding principles should support the enterprise business and process by applying the EA. Also, a process or methodology should be applied to guide the development and maintenance of architectural artifacts.

In [Masys, 2006], a High Level Architecture (HLA) was developed by the Defence Modeling and Simulation Office (DMSO) of the Department of Defence (DoD) to satisfy the needs of defence projects with distributed simulations. However, results of the paper did not provide systematic steps that can be applied to accomplish Verification Validation and Accreditation (VV&A) activities within an HLA environment. They just introduced few engineering practices to facilitate the decision making process. From the literature we notice that V&V technology remains a challenging open problem due to the absence of universal methods [Hao, 2011].

In [Kim and Xiouchakis, 2008], Kim and Xiouchakis reported that it is very important to focus on the design level since most of the product can be predetermined by the end of the conceptual design phase. At the design level, experts from different backgrounds need to participate and collaborate together because of the presence of a big amount, and different kinds of knowledge.
To handle the V&V of DSS at the design level, we consider a multi-dimensional problem with its five facets. The first facet represents the class of the DSS. The second facet depicts the methods/techniques used for the V&V of DSS design. The third facet introduces the contextual methods. The fourth facet presents the user preferences. Finally, the fifth facet outlines the knowledge. Within these facets, we take into consideration the user’s preferences and knowledge, the existing V&V methods and techniques, and the type of the DSS to be able to select the most suitable approach/approaches for the V&V at the design level. The facets allow us to select the appropriate method or methods. Obviously it is more costly to have more than one method, but it is something possible if the resources and budget enable that. Namely, in this work we propose a framework that supports the verification and validation of DSS design. The framework that we suggest is a generic process that intends to facilitate the validation of concepts before system development. Moreover, it serves as practice guidance for documentation of the V&V process. Besides, it will be the basis for choosing and assessing models, methods and techniques for verification and validation in the current next steps of the software life cycle. Our V&V framework is discussed in the following.
4 Verification and Validation Framework

The framework that we propose for the verification and validation of DSS design offers guidance to architects, designers, developers, testers, decision makers who will be using the system, and the V&V team. We assume that the requirements used to derive the architectural design are consistent and complete. For more details, we refer the reader to the work of Khedri et al, [Khedrietal, 2004] which proposed a systematic and rigorous approach that consists in the derivation of a set of architectural designs satisfying the functional requirements. Our framework is illustrated in Figure 4. It has four phases, which are the V&V plan, the V&V selection, the V&V execution, and the V&V documentation. The framework is discussed in the following.

![Figure 4: V&V Framework](image)

4.1 Plan

The V&V plan outlines the sequence of actions that need to be performed, including an estimate about the time and resources. The V&V plan encompasses the candidate methods, techniques, and tools that are available for the V&V activities. Also it presents an estimate about the resources available for the V&V activities. Besides, it describes the V&V activities, and their corresponding teams.

4.2 V&V selection

The selection of verification and validation activities must be performed judiciously. In the following, we discuss each one of the five facets that will support the selection of the appropriate method/technique for the V&V of the DSS design.
4.2.1 Facet 1: DSS Classification

In the literature there are different classifications and taxonomies for DSS. In the following we discuss the DSS components and some DSS taxonomies.

The DSS architecture encompasses three fundamental components which are:

- User interface
- Model base
- Database or knowledge base

The user interface enables the decision maker to assign numerical values to the criteria via the model interface. The model base encompasses all the methods that are candidate such as a Multiple Criteria Decision Aid (MCDA), Multiple Criteria Decision Making (MCDM), etc. The database or knowledge base contains all the data related to the domain.

Decision support systems can be classified into: communication driven DSS, data driven DSS, document driven DSS, knowledge driven DSS, and model driven DSS [SIGDSS]:

- A communication driven DSS supports more than one person collaborating and working together on a common task.
- A data driven DSS or data oriented DSS enables access and manipulation of internal and external data as well.
- A document driven DSS supports managing, retrieving, and manipulating unstructured information in a variety of document formats.
- A knowledge driven DSS improves problem-solving expertise found in rules, procedures, or in similar structures. The expertise mainly comes from the knowledge about a particular problem domain.
- A model-driven DSS supports the access and manipulation of the models (e.g. statistical, financial, optimization, or simulation model). Model-driven DSS assist decision makers in analyzing a situation by taking into consideration their input (e.g. data and parameters).

In [Schneider, 1994], DSS are classified into:

- Information systems and information analysis (e.g. data analysis, documentation systems, data bases, and some expert systems).
- Systems aids to decision making (e.g. expert systems, software supporting the choice).
- Communication and cooperation systems (e.g. negotiation system, cooperative system).

In [Rundong et al., 1993] Rundong et al. presented and developed the knowledge-based design support system (KBDSS) for radar system scheme design. The KBDSS consists of five parts:

- user interface
- knowledge base
- expert system inference engine
- design case base
- model base for radar system analysis.

The knowledge representation in KBDSS is based on object-oriented paradigm. The radar scheme design synthesis and decision-making greatly depend on radar engineer's previous experience and domain knowledge.
Since the architecture of a DSS can be the architecture of any information system, plus the reasoning component. Therefore, in this facet we provide two classes. The first one is related to information systems, and the second include of DSS, IDSS, KBS, and ES.

We claim that the candidate methods/techniques for V&V can be adopted from the ones related to information system or DSS, IDSS, KBS, or ES. Such methods and techniques are discussed in the following.

4.2.2 Facet 2: V&V Methods and Techniques

“There are relatively few decision support design situations that permit objective evaluations” [Adelman, 1992]. Usually, the selection of V&V methods and techniques and the evaluation are biased by the evaluators’ expertise. In the following we survey the literature to discuss the V&V methods and techniques for information systems, and for DSS.

4.2.2.1 V&V Methods and Techniques for Information Systems

According to Bohem [Bohem, 1984a], the most important software requirements and design V&V criteria are completeness, consistency, feasibility, and testability. The main purpose of the V&V of software requirements and design specifications is to identify and correct software problems and high-risk issues at an early stage of the software life cycle.

V&V methods/techniques range from informal to formal ones. In the literature there are different classifications of V&V techniques [Wallace 1996, Balci 1997]. In [Wallace, 1996], Wallace depicted a classification of V&V techniques according to the software development process phase. In the following we elicit V&V techniques that can be used at the requirements and design level [Wallace, 1996].

At the design level, the V&V techniques that can be used enclose algorithm analysis, analytic modeling, control flow analysis, critical timing/flow analysis, database analysis, data flow analysis, desk checking, event tree analysis, inspections, interface analysis, petri nets, prototyping, regression analysis and testing, reviews, simulation, sizing and timing analysis, software fault tree analysis, walkthrough. There are other design methods and techniques that have been widely recognized such as Petri Nets [Yau and Caglayan, 1983], Finite State Machines (FSM), Lotos, Z, Estelle, Abstract Syntax Notation (ASN) [Gunawan et al., 1993], the Specification and Description Language (SDL) used for the specification and design of distributed real-time systems [Glasser et al., 2003].

According to Yau and Tsai [Yau and Tsai, 1986] the techniques used in the architectural design phase can be categorized into process-oriented and data-oriented approaches. The process-oriented approach focuses on the decomposition of the software architecture. It encompasses approaches such as modular programming, functional decomposition, data flow design methods, data structure design methods, HIPO (Hierarchy, plus Input, Process, Output), and module interconnection languages. Data-oriented approaches focus on data design components, and the techniques for deriving them. It includes object-oriented design technique and conceptual database design methodology.

The detailed design techniques can be categorized into graphical representation techniques and language representation techniques. The graphical representation techniques include flowcharts and hierarchical graphs. In the literature, we also find techniques such as stepwise refinement, structured programming, pseudo coding, Jackson Structured Programming (JSP), the Program Design Language (PDL) a language design tool based on English-like statements that are commonly used at the detailed design phase.
In [Balci, 1997], Balci categorized the V&V techniques into four classes: informal, static, dynamic, and formal techniques. Most of the techniques are adopted from the software engineering discipline. The taxonomy is presented in Figure 3.

a. The informal techniques do not have restriction (e.g., natural language). They are not built upon mathematical formalism. This class encompasses techniques such as audit, desk checking, documentation checking, face validation, inspections, reviews, Turing test, and walkthroughs.

b. Static V&V techniques present the structure of the model. They determine its accuracy without executing it. They can be supported by additional automated tools to enhance the V&V process. The static analysis class includes techniques such as cause-effect graphing, control analysis, data analysis, fault/failure analysis, interface analysis, semantic analysis, structural analysis, syntax analysis, symbolic evaluation, and traceability assessment.

c. By opposition to static V&V techniques, dynamic V&V techniques require model execution. They are supposed to evaluate the model with regard to its execution behavior. Most of the time, there are three stages for applying dynamic V&V techniques. The first one consists in instrumenting the executable model, then executing it, and finally evaluating it. The dynamic analysis class encloses V&V techniques such as assertion checking, black-box testing, bottom-up testing, debugging, execution monitoring, execution profiling, execution tracing, field testing, graphical comparisons, predictive validation, regression testing, sensitivity analysis, statistical techniques, stress testing, submodal testing, symbolic execution and debugging, top-down testing, visualization, white-box testing.

d. Formal methods are based on mathematical and logical formalism including set theory, automata theory, and formal logic to produce unambiguously requirements specification, requirements document, and design document. They enable the production of proofs of correctness of the system which are a thorough means of model V&V. By adopting formal methods for the V&V, it is easy to check for properties such as consistency, correctness, completeness, etc. The formal class comprises finite state verification, induction, inference, Lambda calculus, logical deduction, model checking, predicate calculus, predicate transformations, and proof of correctness. These techniques are briefly discussed in the following.

- The inferences enable to derive logical conclusions from the premises that are assumed to be true. Rules of inferences (e.g., Modus Ponens) are applied in order to derive the conclusion from the given premises.

- The induction is a method of proof based on a recursively defined structure. More precisely the recursively defined structure and the proof method are specified by an induction principle. In the literature, the terms “recursion” and “induction” are often used interchangeably.

- Inductive assertions are used to evaluate model correctness. However, in induction, assertions do not necessarily lead to true conclusions.

- Lambda calculus (also written as λ-calculus) is a formal system in mathematical logic that was first formulated by Alonzo Church. It encompasses a language of lambda terms based on a formal syntax, with a set of transformation rules. The Lambda calculus enables transforming the model into formal expressions model so that mathematical proof of correctness techniques can be applied.

- Logical deduction or deductive reasoning consists in reasoning from one or more premises to attain a logical conclusion. It links premises with the conclusions.
Predicate calculus known also predicate logic or First-order logic is the study of statements about individuals using functions, predicates, and quantification which extends propositional logic. Quantification is allowed only over individuals not over functions and predicates. It is the basis of many formal specification languages [Backhouse 1986].

Predicate transformation enables to verify model correctness based on its semantics [Dijkstra 1975].

Formal proof of correctness consists in formally proving the correctness of the model, and satisfying its requirements specification [Backhouse 1986].

In [Balci, 1997], Balci presented 75 techniques; however this list is far from being exhaustive. In the following we discuss other formal methods and techniques from the literature. We decided to discuss formal methods since nowadays there is a trend to adopt them for complex systems [D. M. Berry, 1999].

In the following we survey the literature regarding the V&V methods and techniques for DSS. This class encompasses IDSS, KBS, and ES.

### 4.2.2.2 V&V Methods and Techniques for Decision Support Systems

In [Wagner and Bertacco, 2008], Dichotomic Finite State Machines (DFSM) are used to verify memory coherence protocols in large multi-core and multi-processor designs. To achieve that, they adopt a distributed network of cooperating agents, where each agent performs its own verification goals, and then helps other agents on theirs whenever there is a need for that. Each agent has its own DFSM.

In [Csertan et al. 2002], Csertan et al. developed the VIsual Automated model TRAnsformations (VIATRA) framework. The VIATRA framework aims at adopting UML, and other mathematical models like Petri nets, dataflow networks, hierarchical automata in order to have an open tool independent architecture.

In [Jeyaraman et al., 2005], Jeyaraman et al used Kripke models which play a role similar to the Lindenbaum–Tarski algebra in algebraic semantics. Kripke models are used for formal modelling of reactive systems. They are represented as directed, unlabeled graphs that can be unwound into an infinite computation tree with the possible transition paths. Kripke models can also be represented by an automaton. Then, temporal logic is adopted to present the desired properties of the system.

In [Hao et al., 2011], Integrated Health Management (IHM) systems have been successfully used in many military and industrial applications. The aim of adopting V&V technology for IHM before system deployment is mainly to support operational and maintenance decisions, and to avoid costly errors. The author discussed metrics which are very important to enhance the design of IHM system. There are three classes of metrics. Metrics for diagnostic algorithms, metrics for prognostic algorithms, and metrics for cost-benefit assessment which are intended to measure the cost and benefit provided by IHM systems. The authors discussed also some V&V techniques related to IHM field found in the literature such as parametric testing, and Monte Carlo method employed to the V&V of algorithms.

Cehlot and Sloane [Cehlot and Sloane, 2006] proposed a formal approach to verify and validate heterogeneous, wireless, patient-care-device networks. More specifically, they propose a V&V toolkit to ensure safety and reliability of wireless medical device networks. The V&V techniques used are the black-box, white-box, and grey-box testing.

They also gave an example of a patient monitoring system that was modeled with Coloured Petri Nets (CPN). As a future work, the authors suggested to improve the V&V toolkit methodology, by adding the verification of interoperability of embedded medical device decision support systems and intelligent alarms with CPN or other formal method.
In [Vanthienen et al., 1998], Vanthienen et al. adopted decision tables to verify knowledge based systems, and developed a table-based development tool called Prologa. Classical decision tables date back to early years in computer science. In the literature, there are many kinds of decision tables (DT), and they have been used for many years in software engineering [Hurlay, 1983]. For instance, Hoover and Chen [Hoover and Chen, 1995] proposed decision tables to define functions and relations. The tables contain only variable names and values. No logical symbols are used since the logical connectives are expressed by the table structure itself. However, for decision tables, the check for consistency and completeness is elaborated at a primitive level [Bourguiba and Janicki, 2009].

In [Farahbod et al., 2008], Farahbod et al. proposed a formal approach to design situation analysis and decision support systems. They adopted two mathematical modeling paradigms which are Abstract State Machines (ASM) and interpreted systems for the design and development of decision support systems. ASM techniques have been successfully used in many industrial applications to model complex distributed systems. In their work, ASMs were combined with CoreASM to implement design specifications through symbolic execution (model checking), simulation and testing.

In [Borenstein, 1998], the validation of DSS is discussed in order to avoid costly errors. The main objective of his work is to propose a practical and efficient method to validate DSS. Therefore, the proposed approach combines methods developed to validate Operations Research/Management Science (OR/MS) models and expert systems validation frameworks. In this approach, it is proposed that a DSS is evaluated through laboratory testing encompassing face validation, subsystems verification and validation, predictive validation, and field testing.

In [Sojda, 2004], Sojda emphasized that evaluation is critical, and should be based on empirical testing whenever it is possible. Evaluation encompasses both verification and validation. Sojda presented an example of evaluation of a DSS for trumpeter swan management. He adopted a multiagent system of experts systems, interacting intelligent agents, and a queuing model. For the verification of the components and the whole system, he used graphical representations and flowcharts that were discussed with experts for modification and refinement. Also, static checks were performed to detect incomplete rules. The validation of the expert systems components involves meetings and telephone conversations where the involved parties run some scenarios, and then provide their comments and feedback.

In [Papamichail et al. 2004], Papamichail and French introduced the V&V of the Evaluation subSYstem (ESY) which is an intelligent DSS. In this paper, they use expert systems technology to improve the decision making process for nuclear emergencies. To verify the ESY code, they use static testing methods to determine the accuracy of the system’s design without executing its code. Also, they adopt dynamic testing to execute the system’s code using different sets of data. For performance validation, they check for consistency, completeness and accuracy of the knowledge base of the ESY.

Within this facet we surveyed the literature regarding the V&V methods and techniques that can be used at the design level. We classified them into V&V methods and techniques for IS, and for DSS. The DSS class encompass the IDSS, KBS, and ES. This facet combination enables to guide the selection of the most appropriate V&V methods/techniques at the design level, with regard to the type of the DSS to build, and the domain application.
4.2.2.3 Contextual methods facet

Many studies have shown that usually the adopted methods do not support well users’ needs and preferences. Also, some studies aimed to have a universal method; however this was impossible since methods are confronted to rapid technological evolution and new paradigms. The diversity of the existing methods, their rapid evolution, and the absence of a universal method that can be adapted for any system induced that the existing methods do not allow user guidance, sharing and reusing users’ expertise in a systematic way. Therefore situational methods emerged in order to better address users and designers’ needs and preferences, to take into consideration different situations characterizing the system to build, and to be able to change the method whenever the situation changes [Hiddings, 1994].

In the literature, terms such as “situational method engineering”, “method adaptation”, “method tailoring” are used interchangeably. They are defined as: “A process or capability in which human agents through responsive changes, and dynamic interplays between contexts, intentions, and method fragments determine a system development approach for a specific project situation”.

The situational approach shows that a method is built according to a given situation. The advantage of adopting situational methods is to improve the decision making process.

Situational methods take into consideration the knowledge used, they enhance the decision making process. Their aim is to provide efficient solutions to the development of methods by focusing on their construction, design, and evaluation. Also, it facilitates the selection of a method between others, and eases the selection, and integration of method chunks.

Situational methods consider modeling the system and the domain knowledge on which the system should provide information, and the system environment.

In [Seligmann, 1989], Seligmann and Wijers delineated that a development method of information system encompasses the “way of thinking”, the “way of modeling”, and the “way of supporting”, and the “way of organizing” composed of the “way of working”, and the “way of control”.

The spectrum of situational methods proposed by Harmseen 1994 [Harmseen 1994] ranges from the use of rigid methods, to the modular construction of methods (see Figure 1).

![Figure 5: The spectrum of situational methods](image-url)
He presented six classes of methods which are discussed in the following.

- **Rigid methods**: are not adapted to any situation, and focus on only one type of project.
- **Selection of rigid methods**: mainly focus on the selection of a rigid method based on the situation of a project.
- **Approach toolkit/multiview**: consists in including several methods, where each one of them addresses a particular aspect, until reaching the target method. Depending on the situation, a set of tools are used for the given project. The disadvantage of such methods is that they do provide neither guidance nor concrete suggestions on how to select the tools. Also this approach does not offer validation of the chosen tools. Moreover, for such approach the compatibility of the selected tools is very challenging.
- **Selection of a path in a method**: this approach allows choosing one path from different ones of the same method.
- **Selection and adaptation of a method**: this approach offers the selection of an approach between different ones, and also the selection of one approach according to a given situation. Such approaches should be supported by adaptation tools to support the selection and adaptation of the method.
- **Modular construction of a method**: this method is mainly based on the modular construction of situational methods which are usually built from methods chunks. Methods chunks are usually assembled via rules. The integration process will lead to a consistent situational method.
- **The spectrum** showed that the methods evolved from a low level of flexibility departing from rigid methods to a high level of flexibility attaining situational methods.
- **This classification of methods** has also evolved another one classifying methods into rigid methods, semi-rigid methods, and situational methods. They are discussed in the following:
  - **Rigid methods**: they do not offer guidance to adapt a method for other projects. The same method is used for any kind of project without any flexibility.
  - **Semi-rigid methods**: this class of methods allows the selection from a panel of methods or rigid paths. The selection is based on the situation of a project. The most suitable method to the given situation is selected. However, the method is used without any adaption to the situation of the project.
  - **Situational methods**: they are methods adapted to project situations. They also take into consideration the specificities of a situation. The adaptation of situational methods with the project situation consists in proposing some specific activities that will be performed.

According to Dowson [Dowson, 1988], the models used to describe the methods are categorized into three classes, which are:

- Model-oriented activities
- Model-oriented product
- Model-oriented decision

In such models, the concept of decision depends on the state of the product.
In practice, the designers adapt and modify the methods according to the situation, and also according to their personal preferences. In some cases, they develop a method from scratch. In other cases, they modify an existing method or reuse some parts from others methods, then integrate them in order to come up with a new method.

Usually, it is difficult to consider the whole project situation. Therefore, refined situations can be considered. These refined situations encompass the situations encountered during the development process. This will lead to a dynamic development of the method.

In [Olle, 1991], Olle identified three classes of situational criteria, which are:
- project life cycle,
- contextual criteria related to project requirements,
- technical factors related to the design

In [Slooten, 1993], Slooten identified four classes of situational criteria, which are:
- domain characteristics
- external factors
- technical factors
- knowledge of the persons involved in the project

According to Offenbeek [Offenbeek, 1994], the situational criteria are categorized into:
- functional uncertainty
- potential conflict
- technical uncertainty
- potential resistance
- uncertainty related to the resources

In [Sweede, 1994], Van Sweede identified five classes of situational criteria, and which related to:
- results
- environment
- system development
- human resources
- project management

In [Hoef, 1995], Hoef identified four classes of situational criteria, and which related to:
- environment
- project
- project organization
- constraints
Also in the literature we can find another classification of the situational criteria into two classes which are environment, and the project organization.

For the first class, the situational criteria are related to:
- domain problem
- project importance
- organization impact
- resistance degree
- feasibility in terms of human resources and budget
- degree of formality of the information
- knowledge and experience user

For the second class, the situational criteria are related to:
- Expertise of the project team
- Team size which is determined by the number of people involved in the project
- Relationship with regard to other projects

The designer is confronted to a huge number of situational criteria. From these situational criteria and the requirements, the designer will consider the past situations and then select the most appropriate methods.

Usually the adopted methods/techniques are based on the experience of the methods engineers, developers and designers.

### 4.2.3 Facet 3: User Preferences

The primary success of the developed DSS is that it meets users’ needs and preferences. Indeed, building a system that does not address its users’ needs and preferences may and will often lead to its failure. An assessment of DSS users’ benefit before the implementation of the DSS would avoid an unnecessary time, effort or resources, given that it is usually very expensive to fix errors once the system has been developed. In [Software Engineering Economics], Boehm estimated that the cost of fixing errors significantly increases as the project progresses in its phases [Boehm 1981]. Therefore, addressing users’ preferences and evolving their needs earlier in systems enables to reduce the larger costs associated with error correction later in systems development [Davis, 1982]. Once the errors are detected in early phases, they can be corrected easier, and at less expense. However, addressing users’ needs and preferences has been always difficult to achieve for many reasons that are discussed in the following.

The different persons involved in using and requiring the results of decision support systems such as decision makers, designers, developers, domain experts, and evaluators may have different backgrounds, knowledge, skills, perceptions, concerns, and expression means [Lamsweerde, 2000]. Also, the DSS users may not be sure about what the new DSS will provide. Therefore, they cannot express appropriately their needs and preferences. There are also other situations, where DSS users have difficulties to articulate their needs and preferences. Besides, some stakeholders are not aware of the costs of their needs, so they may give unrealistic demands within the time frame, budget, and resources dedicated to the DSS development. In addition, users may have difficulties to adopt the method given their background, and also the lack of familiarity with the method to be used. For instance, a study on the application of Software Cost Reduction (SCR) method on a Space Station Biological Research Project at the NASA Ames Research Center (ARC) [Pressburger, 2006] showed that the developers were not aware of the concepts, constructs of the general SCR model, and this makes the task of building the specification difficult. As a matter of fact, building the SCR specification took a lot of time. Hence, the SCR project team had to intervene, and use their expertise about mathematical models and state machine models to help building the initial SCR specification, and
then give it to the ARC developers for modification and extension.

Therefore, within this facet, the purpose is to implicate the user in order to accelerate the decision making process, and to take into consideration the preferences and cognition of the decision maker. This aspect enables to support the choice of the most suitable approach maximizing the user’s satisfaction with regards to the user needs, preferences, way of thinking, and the situation at hand.

When DSS’ users including analysts, architects, designers, developers, testers, decision makers who will be using the DSS, and the V&V team members share experiences in the domain problem, it is easy to address their needs and preferences. However, when it is not the case, addressing users’ needs and preferences becomes very challenging. Therefore, we propose to adopt a process of mutual learning or co-discovery where the users can benefit from it, and that can result in design innovation [Moores, 2004]. The purpose is to determine users' needs, perceptions and judgments in the context of their daily tasks. These daily tasks are mainly based on the users’ knowledge. For instance, Davis designed a decision support system allowing the user to be involved in exploring the online analytical processing (OLAP) functionality, and to adjust the data and analysis to their tasks. For that purpose, he adopted RepGrid a cognitive approach supporting eliciting requirements in situations where there is a need to have a degree of co-discovery [Davis, 2008].

The aim of this facet is to elicit and analyse user’s preferences. Therefore, we propose to present a preference ordering rule adopted by the decision maker to evaluate different alternatives. To refine the user opinions we use the typological tree or design decision trees. Decision trees are good candidates to represent, and refine the user opinions. They offer a simple and clear representation of decisions even for people who are not very knowledgeable of such concepts. In addition, new requirements and situations can be easily added to decision trees.

There are some organizations or companies that adopt an independent V&V (IV&V) group. In practice, IV&V is recommended in order to improve the software product quality. The IV&V group is responsible to ensure the correctness and consistency of the system [Zelkowitzet al; 2001]. In such a case, at this phase, both of the IV&V and the development group have to collaborate together to select the methodologies and tools that will be applied for the execution of the V&V.

### 4.2.4 Facet 4: Knowledge facet

In this section we discuss the benefits of considering the decision makers’ knowledge in order to enhance the quality of decision making at the design level. The decision making process enables a better understanding of the problem and the process, and the generation of new knowledge.

In [Bolloju et al. 2002], Bolloju et al. defined the decision makers as: “individuals responsible for solving problems for the purpose of attaining a goal or goals”. In the decision making process in a design project, the decision makers use tacit knowledge and explicit knowledge in order to select an acceptable alternative among others. Organizations and companies should preserve the so called “crucial knowledge” related to the decision making process at the design level. This knowledge can be reused in similar design problems.

The design project has been always a complex task producing one or more outputs, and encompassing the creation and update of knowledge [Perrin, 1999]. This knowledge is the result of the interaction between the designer and users, engineers, decision makers, developers, etc. However, one part of this knowledge stays in the actor’s mind. This kind of knowledge is referred to tacit knowledge by Polanyi [Polanyi, 1966].

In [Saad et al, 2005], Saad et al. proposed an approach identifying the crucial knowledge inherited in the design. The approach consists of three phases which are discussed in the following:
The first phase of the approach consists in selecting a set of potential crucial knowledge. The second phase of the approach consists in determining a model qualifying the knowledge presented. At this phase, a family of criteria is constructed and adopted in order to assess the knowledge with regard to the users’ preferences. The third phase of the approach consists in selecting the best methods and tools for the development of the knowledge map. However, this phase is still under development, and only a class diagram was used to represent the interactions of the tacit and explicit knowledge in the design between individuals.

Also, they proposed also a model qualifying the knowledge. Their model is inspired by Le Moigne systematic approach [Le Moigne, 2006] to build three families of criteria. The three families of criteria are discussed in the following:

- The first family of criteria is built with regard to the functional point of view. The main purpose of this family of criteria is to compute the level of contribution of the knowledge to the objectives of the project.
- The second family of criteria is built with regard to the ontological point of view. The main purpose of this family of criteria is to assess the knowledge vulnerability.
- The third family of criteria consists in constructing a criterion according to the genetic point of view which is defined as the period of use of the knowledge utility in the organization with the users having a vision of the company’s objectives.

Then, the model was applied to qualify the potential set of crucial knowledge in a pilot project. The families of criteria have been ordered according to the users’ preferences.

For the architectural and detailed design phase, the most appropriate V&V technique(s) are identified according to the different dimensions of the different facets. In the following we discuss the V&V execution phase.

### 4.3 V&V Execution

At this phase the selected V&V methods/techniques are executed at the design phase.

The software design decomposes the system into parts, ensuring that these parts fit together to achieve a global goal. The software design phase includes the architectural phase and the detailed design phase. The software design phase has for input the requirements specification. The outputs of the design phase are:

- The product design specification describing the architecture of the software product and its components;
- The detailed design specification defining the inputs, outputs, algorithms, and procedures [Bohem, 1984a].

The DSS design (see Figure 4) encompasses the software architecture, the detailed design, the database containing all the data related to the domain, and the model base encompassing all the candidate methods.

For the V&V at the design level, the first phase consists in the verification and validation of the DSS architectural design. More precisely, the verification consists in confirming that the components specifications meet the requirements at the design stage. We define a software component as an encapsulated software entity with an explicit interface to its environment [Khedri et al; 2004]. At the validation stage, we show whether the architectural design is conform to the customer’s real needs. The second phase is the verification and validation of the DSS detailed design. Namely, it consists in verifying
the interfaces, algorithms, and procedures of the DSS, and checking whether the software design meets its system requirements [Wallace, 1996]. According to Bohem, the most important software design V&V criteria are completeness, consistency, feasibility, and testability [Bohem, 1984a]. However, depending on the system, its design can be checked for properties such as: liveness, performance, reachability, deadlock-freedom, etc.

Figure 6: V&V of DSS design

4.4 V&V documentation

The documentation phase is very important. It is an essential part of the V&V process. In this phase, the methods/techniques adopted for the execution of the V&V, the results of V&V activities, the domain and the conditions under which the V&V were elaborated, and the errors found during V&V activities should be documented. Eventually, the documentation is fundamental for future decisions, and also for the knowledge reuse. It aims at assisting the user in making an informed decision and gives the rationale behind the decisions already made. In the following, we present a case study illustrating our approach.
This page left intentionally blank
5 Case study

In this work we will be focusing on the Course of Actions (COAs) for our case study. They are discussed in the following.

The Collaborative Operations Planning System (COPlanS) is a computer-based system that has been developed at DRDC Valcartier to support the Canadian Forces Operational Planning Process (CFOPP).

The CFOPP includes the following main stages:

- “The Initiation stage results in the activation of the planning staff and the commander’s guidelines about the kind of planning process to achieve;
- The Orientation stage results in the development of the commander’s planning guidance. At this stage, the commander orients his/her staff towards the determination of the nature of the problem and the confirmation of the results to be achieved;
- The Course of Action (COA) development stage results in the production of the CONOPS (CONcept of OPerationS), that identifies the commander’s line of actions, in order to accomplish his/her mission. It presents the COA that will be implemented;
- The Plan Development stage results in a set of orders, based on the commander’s decision, to provide subordinate and supporting units with all necessary information to initiate the planning or the execution of operations; and
- The Plan Review stage results in a regular review of the plan to evaluate its viability. The review period of the plan depends on the evolution of the situation, on the type of operation and on the environment” [Porter et al, 2009].

In [Abi-zeid], a COA is constructed in response to an event, and is defined by a set of actions. They describe an action as the following:

- What action will be performed?
- What equipment will be used?
- How much of each type of equipment is required?
- Where does it come from?
- Where is it going to?
- When should it depart from its present location?
- When should it arrive at its desired location?
- Will the resource arrive in time?

The Course of Actions (COAs) is the set of possible scenarios to be executed, in order to accomplish a mission. Usually there are 3 to 5 CoAs. One of the CoAs will be selected to become the plan. Once the COAs are defined, their viability must be verified. If a COA fails to respect all the viability criteria, it should be revised to satisfy the viability criteria. Each COA should satisfy the following criteria in order to be viable [Porter et al, 2009]:

- Suitability: the staff planners should revise and test each COA in order to check whether it is able to perform the military mission and to attain the desired end state.
- Feasibility: it consists in checking whether the available resources are sufficient to control and assist the operation.
- Acceptability: checks if the probable results assert the estimated costs related to the potential losses in time, material and military personnel.
Compliance: verifies if the COA adheres with the approved CF doctrine, applicable policy, regulations, legislation and guidelines.

Exclusivity: ensures that the COA is exclusively different from the others ones that are developed.

Completeness: verifies whether the COA identifies the force requirements, timings, and objectives unambiguously.

At the design level we have to consider all the possible inputs. The CoAs differ according to the situation, and there is infinity of situations. Therefore, our input will be a generic set of CoA encompassing attributes such as the actions to be performed, the equipment to be used, the amounts of each type of equipment, the resources available, alternate resources, etc. Then, the CoAs should be evaluated according to a set of evaluation criteria.

In [Abi-Zeid et al.], 14 evaluation criteria were constructed for the counterdrug scenario in a Multicriteria Decision Aid (MCDA) framework. Each criterion is either qualitative or quantitative, and belongs to one the following factors: flexibility, complexity, sustainability, cost of resources, and risk. Table I presents the different criteria depicted. The first column presents the factors. The second column depicts the criterion. The third column identifies the optimization of the criterion whether it should be maximized or minimized. The fourth column gives the measurement scale of the criterion. The fifth column presents the global evaluation of the criterion whether it is deterministic or non-deterministic, crisp or a distribution (depicted by a vector evaluation or a by a fuzzy number), and continuous or discrete variable.

In the literature we find other evaluation criteria. For instance in [Papamich et al; 2005], the evaluation criteria are: perceived utility, relevance, completeness, format of output, volume of output, ease of use, ease of learning, timeliness, flexibility or adaptability, performance, and usefulness. They structured into hierarchies (see Figure 6).

The definitions of these criteria are given below.

- The perceived utility depicts the user’s opinion regarding the trade-off between the cost and the DSS’s usefulness.
- The relevance outlines the accordance between the user’s needs and the DSS’s functionalities.
- The understanding of the system determines the degree of the user’s comprehension about the system and its functionalities.
- The completeness delineates the comprehensiveness of the output information.
- The format of output presents the outline of the design and layout of the output information.
- The volume of the output determines the amount of the information provided to the user.
- The ease of use is presented by the amount of effort required by the user to use the tools offered by the system.
- The ease of learning is depicted by the capability of the system to demand minimal effort in learning how to use it.
- The timeliness is given by the availability of the output information at the appropriate time.
- The flexibility or adaptability of the system determines the system capabilities to adapt to new circumstances, conditions, and requirements.
- The performance is given by the capability of a DSS to support the decision maker in performing and elaborating tasks more efficiently.
- The usefulness depicts the extent to which an application’s capability to improve the user’s performance.
Table 1: Criteria Evaluation

<table>
<thead>
<tr>
<th>Factors</th>
<th>Criterion</th>
<th>Optimization</th>
<th>Scale</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>Covering operational tasks, C₁</td>
<td>Maximize</td>
<td>Cardinal on [0, 1]</td>
<td>Crisp, deterministic, continuous</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Covering Mission's locations, C₂</td>
<td>Maximize</td>
<td>Cardinal on [0, 1]</td>
<td>Crisp, probability, continuous</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Covering Enemy's COA, C₃</td>
<td>Maximize</td>
<td>Cardinal on [0, 1]</td>
<td>Crisp, probability, continuous</td>
</tr>
<tr>
<td>Complexity</td>
<td>Operations Complexity, C₄</td>
<td>Minimize</td>
<td>Ordinal, 5 echelons</td>
<td>Crisp, deterministic, discrete</td>
</tr>
<tr>
<td>Complexity</td>
<td>Logistics Complexity, C₅</td>
<td>Minimize</td>
<td>Ordinal, 5 echelons</td>
<td>Crisp, deterministic, discrete</td>
</tr>
<tr>
<td>Complexity</td>
<td>Command and Control Complexity, C₆</td>
<td>Minimize</td>
<td>Ordinal, 5 echelons</td>
<td>Distributional, deterministic, discrete</td>
</tr>
<tr>
<td>Sustainability</td>
<td>Sustainability, C₇</td>
<td>Maximize</td>
<td>Cardinal, R⁺</td>
<td>Crisp, deterministic, continuous</td>
</tr>
<tr>
<td>Cost of resources</td>
<td>Cost of resources, C₈</td>
<td>Minimize</td>
<td>Cardinal, R⁺</td>
<td>Crisp, deterministic, continuous</td>
</tr>
<tr>
<td>Risk</td>
<td>Impact of sensors coverage gap, C₉</td>
<td>Minimize</td>
<td>Ordinal, 3 echelons</td>
<td>Distributional, deterministic, discrete</td>
</tr>
<tr>
<td>Risk</td>
<td>Military personnel loss, C₁₀</td>
<td>Minimize</td>
<td>Ordinal, 7 echelons</td>
<td>Crisp, deterministic, discrete</td>
</tr>
<tr>
<td>Risk</td>
<td>Collateral damage, C₁₁</td>
<td>Minimize</td>
<td>Ordinal, 7 echelons</td>
<td>Crisp, deterministic, discrete</td>
</tr>
<tr>
<td>Risk</td>
<td>Confrontation risk, C₁₂</td>
<td>Minimize</td>
<td>Ordinal, 5 echelons</td>
<td>Crisp, deterministic, discrete</td>
</tr>
<tr>
<td>Risk</td>
<td>Equipment reliability, C₁₃</td>
<td>Minimize</td>
<td>Cardinal on [0, 1]</td>
<td>Crisp, deterministic, continuous</td>
</tr>
<tr>
<td>Risk</td>
<td>Personnel effectiveness, C₁₄</td>
<td>Maximize</td>
<td>Ordinal, 5 echelons</td>
<td>Fuzzy, deterministic, continuous</td>
</tr>
</tbody>
</table>
The evaluation of CoAs on the criteria range from ordinal to cardinal, crisp to distribution, and from deterministic to non-deterministic.

The criteria and their priorities are usually assigned by the DM, and are also subject to some adjustments. In [Guitouni et al; 2008], the criteria were informally validated through discussions with the DM and different planning staff. The validation of the criteria ensures that they form a coherent family of criteria for the DM. The coherence property implies that the criteria used are exhaustive, cohesive, and non-redundant. Also, this family of criteria may have other desirable properties such as readability and operational. Three CoAs were presented and evaluated. The CoAs are not real, and are simply used to illustrate the concepts. The three CoAs named CoA A, CoA B, CoA C, and their evaluation are depicted in Figure 7.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Criterion</th>
<th>CoA A</th>
<th>CoA B</th>
<th>CoA C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>C1: Covering operational tasks</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>C2: Covering Mission’s Locations</td>
<td>90%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>C3: Covering Enemy’s COA</td>
<td>83.7%</td>
<td>100%</td>
<td>95.3%</td>
</tr>
<tr>
<td>Complexity</td>
<td>C4: Operations Complexity</td>
<td>Low</td>
<td>Medium</td>
<td>Very low</td>
</tr>
<tr>
<td></td>
<td>C5: Logistics Complexity</td>
<td>Low</td>
<td>Very low</td>
<td>Very low</td>
</tr>
<tr>
<td></td>
<td>C6: Command and Control Complexity</td>
<td>(0, 0, 0.5, 0.5, 0)</td>
<td>(0, 0, 1, 0, 0)</td>
<td>(1, 0, 0, 0, 0)</td>
</tr>
<tr>
<td>Sustainability</td>
<td>C7: Sustainability</td>
<td>0.5</td>
<td>1.083</td>
<td>1.0</td>
</tr>
<tr>
<td>Cost of resources</td>
<td>C8: Cost of resources</td>
<td>$316K</td>
<td>$684.7K</td>
<td>$373.7K</td>
</tr>
<tr>
<td>Risk</td>
<td>C9: Impact of sensors coverage gap</td>
<td>(0.33, 0.67, 0)</td>
<td>(1, 0, 0)</td>
<td>(1, 0, 0)</td>
</tr>
<tr>
<td></td>
<td>C10: Military personnel loss</td>
<td>Low</td>
<td>Very very low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>C11: Collateral damage</td>
<td>Low</td>
<td>Very low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>C12: Confrontation risk</td>
<td>Low</td>
<td>Low</td>
<td>Very high</td>
</tr>
<tr>
<td></td>
<td>C13: Equipment reliability</td>
<td>67%</td>
<td>87.6%</td>
<td>66.3%</td>
</tr>
<tr>
<td></td>
<td>C14: Personnel effectiveness</td>
<td>Very high</td>
<td>Very high</td>
<td>Very high</td>
</tr>
</tbody>
</table>

Figure 7: Example of a multiple criteria (decision matrix)
In most of the situations it is not evident to draw a conclusion from the decision matrix alone. Indeed, the decision matrix may lead to conflicting conclusions. Therefore, it is recommended to aggregate the alternative evaluations in order to make recommendations. Namely, the decision maker’s preferences and the information contained in the decision matrix should be aggregated in order to determine the “best decision”. To this end, in the literature MCDA methods and procedures have been proposed. MCDA approaches consist in structuring the decision problem, articulating and modelling the preferences, aggregating the alternative evaluations (preferences), and making recommendations. Within MCDA, the concept of optimality is substituted with the concept of best compromise [Guitouni et al; 2008].

In the next step, the COAs are compared using a MultiCriterion Decision Aid (MCDA) approach to provide a ranking of the COAs with their explanations as well.

In this context, an MCDA supporting the Air Operations Center (AOC) should [Guitouni et al. 2009]:

- consider the evaluations ranging from ordinal to cardinal, crisp to distribution, and from deterministic to non-deterministic,
- enable the decision maker to prioritize the CoAs [Roy, 1985],
- enable the decision maker to assign weights to different criteria or to provide a ranking for them,
- monitor the compensation between criteria, since a good performance on one criterion may counterbalance poor performance on another criterion,
- handle properly the preferences and the hesitations of the decision maker according to the doctrine and the rules of operations,
- consider the uncertainties related to the domain problem.

In [Bélanger et al., 2009], in order to provide more flexibility to the user, two approaches were adopted for the comparison of COAs. The first approach is based on a qualitative analysis (descriptive comparison), and the second is based on a quantitative analysis (numerical and ordinal comparison). In the qualitative analysis grid, the planner has the possibility to document the advantages and disadvantages of the COAs according to their criteria. In the quantitative analysis, the planner has the opportunity to determine the evaluation of each COA according to each criterion. The evaluation can be either cardinal or ordinal, depending on the definition of the criteria. In [Guitouni et al; 2008], CASAP was implemented. It is a prototype encompassing the description of the information related to the development of the important CoA, their evaluation, and their ranking in order to determine the best CoA. The “Procedure d’Aggregation Multicritères de Surclassement de Synthèse pour Evaluations Mixtes” (PAMSSEM) was adopted. PAMSSEM has been implemented as an MCDA outranking procedure to prioritize CoAs. PAMSSEM generates either a partial or a total order for the alternatives. Besides it takes into consideration the decision maker (DM) preferences. It consists of two phases: an aggregation phase to produce an outranking relation, and an exploitation phase to recommend a sound alternative. The authors reported that in order to assess the benefits of the system, an emphasize on validation should be performed. First of all the criteria and their settings must be validated. Next, these criteria must also form a coherent family of criteria for the DM. Finally, the analysis approaches developed should be validated to ensure that they satisfy the stakeholders’ needs.

Let \( A = \{a_1, a_2, \ldots, a_l, \ldots, a_m\} \) be the set of CoAs, and \( F = \{C_1, C_2, \ldots, C_j, \ldots, C_n\} \) be a consistent family of criteria. A set of criteria is said to be a consistent family of criteria if it satisfies conditions of cohesion, exhaustiveness, and non-redundancy [Roy, 1985]. The evaluation of the \( i^{th} \) CoA according to the \( j^{th} \) criterion is denoted by \( e_{ij} \). The evaluation of all CoAs according to the set of criteria results in the multiple criteria performance table \( E \), also known as the decision matrix shown in Figure 5.
In our work, we adopted the V&V framework to assist in the selection of the appropriate approach/approaches (e.g. qualitative analysis, quantitative analysis) to support the comparison of CoA before the implementation. Namely, we verify and validate that the selected approach/approaches correspond to the best decision, and that is before implementing the DSS. The V&V framework that we proposed assists the decision maker in choosing the most appropriate approach/approaches for the V&V of DSS design according to the situation at hand. Our multi-dimensional facets enable to select the most appropriate approach with regard to the user’s preferences, knowledge, the existing V&V methods and techniques that can be used at the design level, and the type of the developed DSS. In our approach, we also consider the use of contextual methods supporting the selection of the most suitable situational methods that can be used at the design level.

In Figure 6, we present an example of an application of our multi-dimensional facet within this case study. It encompasses 4 phases. The first phase consists in the verification and validation of the DSS architectural design. More precisely, the verification is to confirm that the CoA meet the requirements at the design stage. The second phase is the verification and validation of the DSS detailed design. Namely, it consists in verifying the interfaces, algorithms, and procedures of the CoA. In both phases we take into consideration the DM preferences. The third phase consists in the selection, verification and validation of the evaluation criteria. The evaluation criteria were previously discussed. In a DSS the process of decision making involves selecting and evaluating criteria and alternatives followed by selecting the most suitable solution. Usually DMs interact with the user interface to select the criteria to consider, and to assign their respective weights. The criteria are assigned a weight or are rated depending on the decision analysis technique used. The model base encompasses all the possible criteria.

The fourth phase consists in the selection and verification and validation of multi-criteria decision analysis approach. Multi-criteria decision analysis approaches are used to assess DSSs. Their aim is to provide the DM with different analysis tools to support the decision problem. Within MCDA, the concept of optimality is substituted with the concept of best compromise [Guitouni et al; 2008]. The MCDA methodology can be seen as a recursive process composed of four steps which are the following:

- structuring the decision problem,
- articulating and modelling the preferences,
- aggregating the alternative evaluations (preferences)
- making recommendations.
Figure 9: Process of the V&V of the DSS design
6 Conclusion and future work

Incremental V&V should be conducted thorough the software lifecycle to ensure confidence that the software fits its purposes. In this work, we proposed a framework that supports the V&V of DSS design. The V&V framework that we proposed with its multi-dimensional facets assists the DM in choosing the most appropriate approach/approaches for the V&V of DSS design. Our framework aims to ease the validation of concepts before system deployment, and to enhance the decision making processes. Besides, it will be the basis for choosing and assessing models, methods and techniques for V&V in the next steps of the software life cycle such as detailed design, and implementation.

For future work, we aim at applying our generic framework for decision support systems for large distributed systems which are relevant in the context of military command and control applications. Also, our architectural framework can be integrated into DND/CF Architecture Framework (DNDAF). In fact, DNDAF [add footnote] is missing a rigorous approach that enables the verification and validation of its architectural artifacts [DNDAFASM, 2010].
References

I. Abi-Zeid, M. Bélanger, A. Guitouni, J.-M. Martel, K. Jabeur, A Multicriteria method for evaluating courses of action in Canadian airspace violation situations, 4th International Command and Control Research and Technology Symposium, National Defence College (Sweden) & CCRP (USA), Nasby Park (Stockholm), Sweden, 14-16 September 1998, pp. 521-533


Bélanger, M., Guitouni,A., Pageau, N., Decision support tools for the operational planning process, 14th International Command and Control Research and Technology Symposium, CCRP, Washington, 15-17 June 2009.


J.A. Lane and B. Boehm, “System of systems lead system integrators: Where do they spend their time and what makes them more or less efficient?”, Systems Engineering Vol. 11, no. 1, pp 81-91, 2008.


## List of symbols/abbreviations/acronyms INITIALISMS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN</td>
<td>Abstract Syntax Notation</td>
</tr>
<tr>
<td></td>
<td>Specification and Description Language</td>
</tr>
<tr>
<td>IDSS</td>
<td>Intelligent Decision Support Systems</td>
</tr>
<tr>
<td>MCDA</td>
<td>Multiple Criteria Decision Aid</td>
</tr>
<tr>
<td>MCDM</td>
<td>Multiple Criteria Decision Making</td>
</tr>
<tr>
<td>OO</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>VV&amp;A</td>
<td>Verification, Validation, and Accreditation</td>
</tr>
</tbody>
</table>
This page intentionally left blank
<table>
<thead>
<tr>
<th>DOCUMENT CONTROL DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)</td>
</tr>
<tr>
<td>1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)</td>
</tr>
<tr>
<td>Imen Bourguiba</td>
</tr>
<tr>
<td>DRDC Valcartier</td>
</tr>
<tr>
<td>2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)</td>
</tr>
<tr>
<td>UNCLASSIFIED</td>
</tr>
<tr>
<td>2b. CONTROLLED GOODS (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC APRIL 2011</td>
</tr>
<tr>
<td>3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)</td>
</tr>
<tr>
<td>Comprehensive Framework for the V&amp;V of Decision Support System Design</td>
</tr>
<tr>
<td>4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)</td>
</tr>
<tr>
<td>Bourguiba, I.</td>
</tr>
<tr>
<td>5. DATE OF PUBLICATION (Month and year of publication of document.)</td>
</tr>
<tr>
<td>December 2013</td>
</tr>
<tr>
<td>6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>6b. NO. OF REFS (Total cited in document.)</td>
</tr>
<tr>
<td>89</td>
</tr>
<tr>
<td>7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)</td>
</tr>
<tr>
<td>Contract Report</td>
</tr>
<tr>
<td>8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)</td>
</tr>
<tr>
<td>Defence Research and Development Canada – Valcartier</td>
</tr>
<tr>
<td>2459 Pie-XI Blvd North</td>
</tr>
<tr>
<td>Quebec (Quebec)</td>
</tr>
<tr>
<td>G3J 1X5 Canada</td>
</tr>
<tr>
<td>9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)</td>
</tr>
<tr>
<td>9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)</td>
</tr>
<tr>
<td>10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)</td>
</tr>
<tr>
<td>10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)</td>
</tr>
<tr>
<td>DRDC Valcartier CR 2013-372</td>
</tr>
<tr>
<td>11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)</td>
</tr>
<tr>
<td>Unlimited</td>
</tr>
<tr>
<td>12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.)</td>
</tr>
<tr>
<td>Unlimited</td>
</tr>
</tbody>
</table>
13. **ABSTRACT** (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The objective of this report is to introduce a framework supporting the verification and validation (V&V) of decision support systems (DSS) design. In this report, we propose a V&V framework with multi-dimensional facets to assist the decision maker in choosing the most appropriate approach/approaches for the V&V of DSS design. The framework serves as practice guidance for documentation of the V&V of architectural and detailed design. Also, our framework intends to facilitate the communication, and the validation of concepts before system development. Besides, it will be the basis for choosing and assessing models, methods and techniques for V&V in the next steps of a software life cycle.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

verification; validation; V&V; decision support systems; DSS
Defence R&D Canada

Canada’s Leader in Defence and National Security Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière de science et de technologie pour la défense et la sécurité nationale

www.drdc-rddc.gc.ca