ISSN: 2321-7782 (Online)
Volume 3, Issue 11, November 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Development of Simulator for Simple Command Post Integrated Data Display using Assurance Case

Dipak Shamlal Gade[1]
Department of CSE,
Shri Jagdishprasad Tibrewala University, Chudela, India

Dr. Santosh Deshpande[2]
Department of CA, MES IMCC
Savitribai Phule Pune University, Pune, India

Abstract: ***Software design for mission critical systems is challenging and often needs utmost care in ensuring all the essential requirements of system are met without any compromise on quality and reliability of system deliverables. Present paper has explored an approach of developing software for command post integrated data display Simulator based on Assurance Case driven development. This paper has systematically illustrated how software development for a software system can be carried out using Assurance Cases. The developed code was tested using rigorous test cases to ensure that the goals and sub-goals are met with adequate reliability and safety and thus the overall simulator goal. The design was further verified with collected artifacts through actual testing to ensure that the developed Simulator is working as per the stated requirements under all specified operating conditions. We concluded with discussion of important results and suggestions for addressing the problems we faced while developing the Simulator software using Assurance Case.***

Keywords: Assurance Case, Simulator, Data Fusion, Software Assurance, WBS

I. INTRODUCTION

Command Post normally receives various sensors data including Radar, VOIP, Network Data, Record and Replay Data etc. over existing communication channels such as Serial Port, WI-FI, Ethernet or even through Wireless Radio sets. Command post processing system then suitably processes this data in timely manner and displays this integrated data on Multi-Function Console (MFC). The MFC serves the purpose of displaying consolidated data from various sensors in integrated fashion and also allows the operator of Command Post to select, filter and or display the selected data from selected sensors.

Receiving data from multiple sensors in real time and processing this data with in required time constraint is complex and very challenging. While dealing with and processing of such data, often requires handling following challenges

- » Accurate data analysis and prediction in real-time
- » Sensor wise data receiving on separate data channel and data buffering
- » Zero percentage data loss while receiving the data over multiple data channels
- » All sensors data processing with a given time constraint
- » All sensors data fusion and data formatting to suit common data processing protocol
- » Data filtering as per user requirements
- » Selection of specific sensors data in real time as per user requirements

Considering the above listed concerns and challenges, one can easily understands the complexity involved in integrated data receiving and processing for command post. If not dealt with care and by not using suitable methodology, there are significant amount of chances of wrong design & implementation and incorrect data handling which can lead to data loss and subsequently wrong results prediction. On battlefield, during wartime, it is not possible to afford to take any wrong decision

based on wrong data analysis. To ensure that operators are properly trained to operate the command post console, often they are trained with the help of simulator. Simulators need to simulate practical scenarios so that same can be presented to operators to facilitate them making aware of and giving feel of battle field war like situations. To ensure that such simulators are meeting their intended application purpose, they need to be designed and implemented with the same kind of care and due diligence which we require in case of any mission critical software system. In this paper we discussed development of simple command post display data simulator using Assurance Case based approach to ensure that intended goals of simulator are met successfully without any doubt & or ambiguity.

II. Simulator Overview

Command Post display data simulator is complex software and has same importance which can have to any critical software system. This requires designing of such simulator software with utmost care to ensure that it is fit for use and reliable. Instead of using conventional software design techniques which subsequently need certification of the software to ensure that all the functionalities are implemented defect free and reliably as per requirements, an innovative Assurance Driven Software Design methodology is used to ensure that all objectives and requirements goals of Simulator for Command post Integrated Data Display (CPIDD) are captured and taken care of in design and implementation phase. And thus the developed software meets all the functionalities defect free and reliably as per the final requirements without any compromise.

Figure 1 shows major blocks of CPIDD Simulator. As shown in the figure, the Simulator receives the data originated from three different Radars. The Radar Data Fusion and Processing (RDFP) block carries out radar data fusion and necessary processing in order to make it compatible for displaying on MFC. As shown in figure 1, MFC receives the data to be displayed from RDFP. The Operator Command Handler handles the operator commands and sends it to MFC for necessary actions. MFC, based upon operator choice, displays the relevant data. The Simulator has to receive the Radar data and carry out require processing and displaying of data as per operator interaction with in required time constraint. The challenge is to receive data from all radars without any data loss and process it within required scan time so that all data can be displayed without losing any target information.

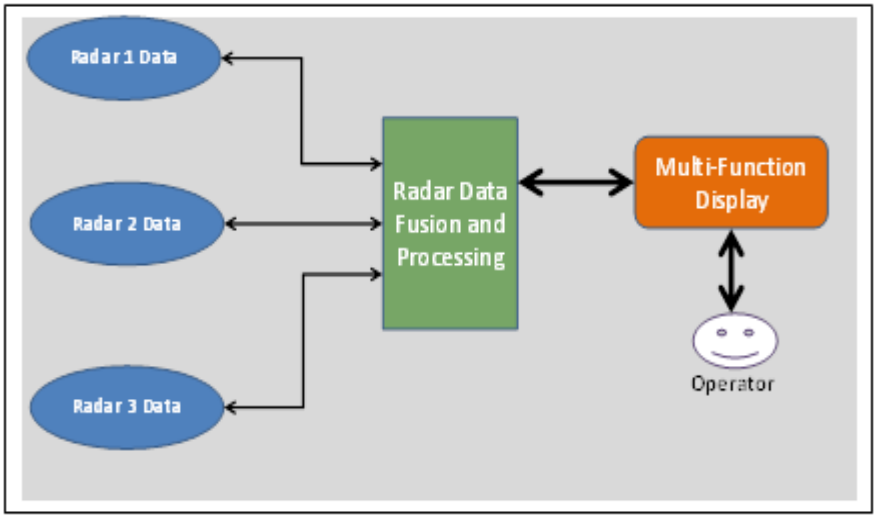

Fig. 1 Top Level Block Diagram for CPIDD Simulator

The various modules of CPIDD simulator are listed in Table1 along with their ID and specifying their implementation technique. As stated in Table 1 most of the modules are using simulated data and are implemented in C++.

TABLE I

CPIDD Simulator Major Modules

Sr. No.	Module	IMPLEMENTATION	ID
1	Radar 1 Data Generator	Simulated Data/ Fixed Data Buffer	RadSim1
2	Radar 2 Data Generator	Simulated Data/ Fixed Data Buffer	RadSim2
3	Radar 3 Data Generator	Simulated Data/ Fixed Data Buffer	RadSim3
4	Network Management	C++ Communication Data Handler	NM
5	Radar Data Fusion and Processing	C++ Application Software	RDFP
7	Integrated Display (Multi-Function Display)	C++ Application Software	MFC
8	Operator Command Handler	C++ Application Software	OCH

III. CPIDD SIMULATOR DESIGN USING ASSURANCE CASE

The Assurance Case, covering the CPIDD Simulator critical goals, sub-goals, strategies and solutions, is represented in figure 2. The Assurance case is developed by using Goal Structuring Notations (GSN). The Assurance case has provided all the critical goals and sub-goals of the CPIDD Simulator which are must to meet before declaring that CPIDD Simulator is absolutely fit for use under given operating conditions. The Assurance case has clearly stated the system contexts as well as assumptions made while making the simulator claims and providing evidences. This has provided clarity for better understanding while going through the Assurance Case. Note that figure 2 is presenting only part of the CPIDD Simulator Assurance Case. The complete Assurance Case has not been provided here due to size and length constraints, rather the Assurance Case has been shown in following figure considering key aspects of CPIDD Simulator.

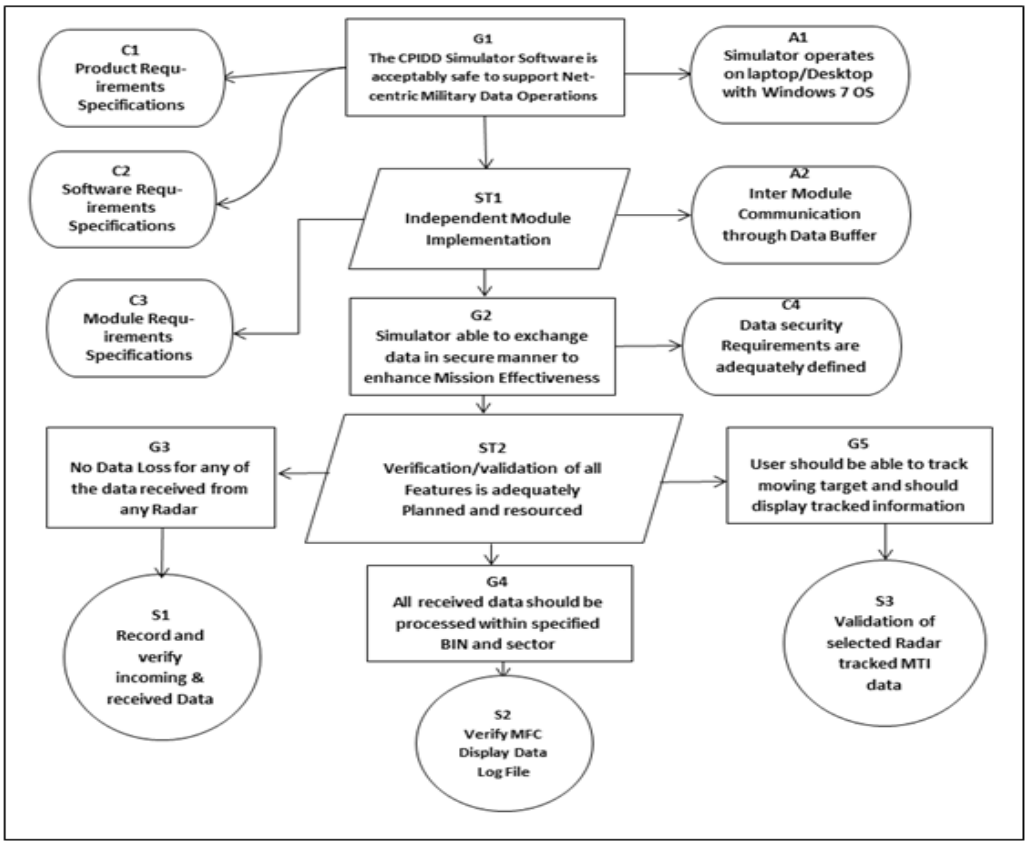

Fig. 2 CPIDD Simulator Assurance Case

While developing Assurance Case for CPIDD Simulator, our main focus was to demonstrate highest level of assurance for the CPIDD Simulator and use of modular approach. We were also interested in reducing overall development time frame and thus the efforts and cost associated with Simulator development. Therefore we moved ahead with building of independent

development modules and its associated components. This required us breaking down the entire Simulator into different required modules and sub-modules. Please refer Table1 for module details. Since the CPIDD Simulator is intended for mission critical military application system such as Command Post Integrated Data Display, we kept all the assurance goals meant for Simulator such as related to operation safety, data security, system reliability, operability, maintainability etc., in line with goals for any practical defense system.

It was ensured that all the relevant assurance issues are adequately addressed by the developed Assurance Case. To start with, the highest assurance goal demands Simulator compatibility with net-centric military data operations to ensure that all the communication ports data are securely handled without losing any of the communicated data. This highest level goal further decomposed in to sub-goals and subsystem goals such as zero data loss, secured data communication, meeting data processing time constraints and handling operator interactions effectively.

Please refer Table –II which has listed down all the GSN labels used in the CPIDD Simulator. The labels description is also provided for easy reference. It is ensured that the Simulator requirements specifications capture all the functional and non-functional requirements so that all the relevant features of CPIDD simulator are recorded and tracked. The design also demands modular development approach as stated by Strategy 1. This was ensured by developing an independent modules and sub-modules identified as the design evolves. This posed an additional requirement of developing the complete work breakdown structure (WBS) for the CPIDD Simulator. The WBS helped in identifying required development modules and sub-modules. It is possible to add module identifier for the specified goal. For each independent module separate verification and validation plan was developed to ensure that the module requirements are met without any ambiguity. To fulfill this demand, Strategy 2 suggested adequate plan in place with required resources. To ensure that all goals are met without any doubt, actual test case results were used as evidence wherever possible. The facility of logging system messages and communication packets helped in verifying data contents of processed communication messages. These artifacts facilitated in convincing the Assurance Case Reviewer and for accepting it.

TABLE II

CPIDD Simulator Assurance Case GSN Labels and Description

Sr. No.	Labels	Description
1	G1, G2, G3, G4, G5	Simulator Goals and Sub-goals
2	A1, A2	Assumptions made
3	C1, C2, C3, C4	Simulator relevant Context Information
4	ST1, ST2	Strategies adopted
5	S1, S2, S3	Solutions for meeting simulator specified goals

The Successful implementation of CPIDD Simulator meets all the required functionality. Figure 3 is showing the CPIDD Simulator display where it is shown that 3 moving targets (represented by small rectangular boxes) as captured by Radar 1, Radar2 and Radar3 respectively are displayed. The CPIDD Simulator Display has simulated the Radar Plan Position Indicator (PPI) and has displayed the entire coverage by three different radars in simple consolidated fashion. The reported clutter and chaff is also displayed. The operator has tracked target 3 and in Track Information Window (TIW) all the relevant details with respect to the tracked target such as its track Number, IFF Status, Velocity, Range and Azimuth are displayed. It is also shown that operator can do the radar range selection and can play the simulation as per requirements.

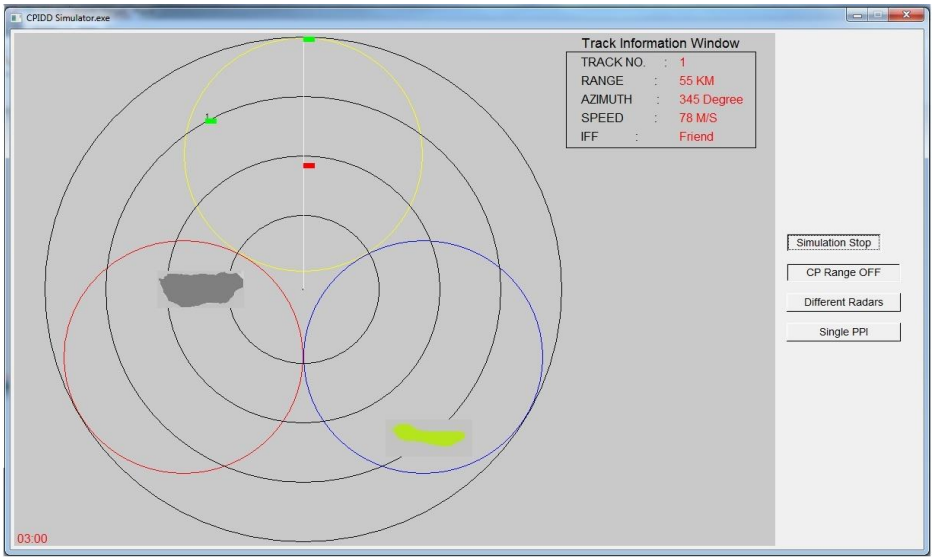

Fig. 3 CPIDD Simulator Display

IV. IMPORTANT RESULTS AND DISCUSSION

We created the Assurance Case to design and implement CPIDD Simulator and to demonstrate that it is safe to operate, reliable in operation and offers zero data loss. The intention behind developing CPIDD Simulator by using Assurance Case is to verify if Assurance Case based design approach can provide trustworthy and reliable software which is fit for operation and use under stated operating conditions. It is observed that the developed simulator using Assurance Case approach successfully meets all the required functionalities and fulfilled all the stated claims as mentioned in the Assurance Case. The CPIDD Assurance case underwent multiple review cycles by internal and external stakeholders just to ensure that there are no unattended goals or sub-goals and no unaddressed arguments results which can pose significant threat and risk to the CPIDD Simulator Software. The CPIDD Simulator Assurance Case evolved as the project progresses, and finally it was made comprehensive enough to take care of each and every sub-goal identified during the development phase. The Assurance arguments and evidences were structured properly which helps reviewers to understand and review the assurance case as per the requirements. Most of the evidences were provided through actual testing, functionality demonstration and test results which provided justified confidence that the Assurance Case has addressed all the relevant Assurance Issues. It is verified that the implemented CPIDD Simulator using an Assurance Case based approach has successfully met all the critical requirements as stated in Assurance Case. This finally convinced the reviewers to accept the CPIDD Simulator Assurance Case.

While doing the development using Assurance Case approach, we however had some difficulties while representing the Assurance Case with graphics notations. It was found very time consuming and inconvenient to draw some of the GSN symbols and adding text into it, when it was done in word document. Also the Assurance Case ran into multiple pages where we faced difficulties in keeping continuity with Assurance Case. Also when it was required to do modifications in some certain sections of Assurance Case when it was required to do so, based on the generated evidences, we found bit difficult to trace the required section of Assurance Case. Maintaining the lengthy Assurance Case was also found challenging specifically when it has to undergo changes frequently. To avoid such issues, it is recommended to go for off the shelf software tools which can assist in creating and maintaining the Assurance Case systematically and easily.

The critical functionalities successfully achieved by the CPIDD Simulator are stated in Table III below.

TABLE III

CPIDD Simulator Critical Goals/ Sub-Goals Status

Sr. No.	Simulator Critical Goals (Major Functionalities)	Status Remark
1	Sector wise processing of all Sensors data	Achieved with in scan time of respective Radar
2	Displaying all Sensor Data	All three Radars data was displayed
3	No Data Loss	All MTI Videos including chaff and clutter from all three radars were processed & displayed without any data loss
4	MTI Video along with IFF status Display	Displayed the target data of all three targets as reported by 3 radars respectively.
5	Target Tracking	Can track each target from each radar and the information including Track Number, Speed, Range, Azimuth, IFF Status, was displayed in target info window
6	Individual Sensor Data selection	Operator could select the individual Radar Data for display purpose on respective PPI
7	Start, Pause, Stop	Simulator Start/ Pause/ Resume and Stop Functionality worked well without any issue.

V. Conclusion And Future Work

Based on the designed and implemented CPIDD Simulator using Assurance Case and after verifying all its functionalities and performance, it is concluded that

- Assurance Case based design approach was found very effective in capturing and meeting CPIDD Simulator Critical goals.
- The generated artifacts were sufficient to verify and confirm that the goals are achieved without any compromise on quality and performance front.
- It is also observed that the Assurance Case based approach can be used to implement only critical functionalities of the simulator to ensure that these functionalities are met without any compromise whereas non-critical functionalities (For example colour combinations used in GUI) can be implemented using conventional software development approach. This has optimised the overall development time frame.
- Assurance Case based software design provides a rigorous development process and needs lot of detailing in form of documentation, evidence collection and generation, planning, implementation and testing. It is not a shortcut process which can reduce the development time frame, rather to provide trustworthy and reliable software, a comprehensive Assurance Case needs to be developed which can be time consuming and a complex activity.
- Modular Assurance Cases can be used as it is or with some fine-tuning in different projects promoting reuse concept. For example the part of CPIDD simulator Assurance Case was devoted for operator command handling and interactions, this part of Assurance Case can be used in different Assurance Case which requires operator command handling functionality similar to CPIDD Simulator. This can provide lot of flexibility in creating new Assurance Cases and can also save development time frame. The only care should be taken here is that such part of Assurance Case should be modular so that it can be used independently in different Assurance Case for different projects.

In future, further work is needed to explore Assurance Case based Software Design approach to check if generated evidences and development artifacts are sound enough to convince any software certification authorities that the generated software is robust enough and at par with any formally Certified software as far as quality/reliability/safety/security/robustness

and or availability of software is concern. This may help to adopt Assurance Case Design approach as an alternative choice to formal Software Certification process.

References

1. Graydon P.J., Knight J.C., Strunk E.A., "Assurance Based Development of Critical Systems", Dependable Systems and Networks, 2007
2. Nguyen E.A., Greenwell W.S., Hecht M.J., "Using an Assurance Case to Support Independent Assessment of the Transition to a New GPS Ground Control System", Conference proceedings for Dependable Systems and Networks With FTCS and DCC, IEEE Publication, June 2008
3. Jee E., Lee I., Sokolsky O., "Assurance Cases in Model-Driven Development of the Pacemaker Software", 4th International Symposium On Leveraging Application of Formal Methods Verification and Validation (ISoLA), Part II, pp. 343-356, 2010
4. GSN Community Standard Version 1, November 2011. [Online]. Available: www.goalstructuringnotation.info/documents/GSN_Standard.pdf
5. Dipak Gade, Dr. Santosh Deshpande, "A Literature Review on Assurance Driven Software Design", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 9, September 2015
6. Dipak Gade, Dr. Santosh Deshpande, "Assurance Driven Software Design using Assurance Case based Approach", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 10, October 2015

