Continual Systems Development for Command, Control and Intelligence Systems

ABSTRACT

The traditional software development model is no longer adequate to meet the challenges of the fast-evolving needs of Command, Control and Intelligence (C2I) systems. To enable the transformation of the Third Generation Singapore Armed Forces (SAF), a continual systems development approach has been adopted by DSTA to develop C2I capabilities for the SAF. The key advantage of this approach lies in its flexibility to respond rapidly to meet changing needs and address emerging threats. The complexities of network-centric operations entail the fielding of systems quickly for operational trials and to continually discover capability gaps so as to evolve systems capabilities over its life cycle. This paper addresses the framework adopted by DSTA to develop the C2I system of systems.

Dr Yeoh Lean Weng
Teo Tiat Leng
Lim Horng Leong

Continual Systems Development

for Command, Control and Intelligence Systems

INTRODUCTION

The proliferation of network-centric operations concept has introduced layers of complexity in designing and developing Command, Control and Intelligence (C2I) systems to support military operations. After 9/11, the characteristics of asymmetrical threats have introduced new challenges in our understanding of the capabilities needed. We need to go through a process of learning and exploration to discover the new systems requirements and to translate these into the architecture of C2I systems. Given the evolving requirements and the emerging capability demands from known and hidden threats, there is a pressing need for technologists to shorten development cycles and put the systems into the hands of commanders and warfighters as quickly as possible. The results gathered through field trials and experiments enable the stakeholders to understand the needs better, explore new operational concepts and identify capability gaps. DSTA's Continual Systems Development approach leads to shorter and more frequent systems releases, resulting in better capabilities for the Singapore Armed Forces (SAF).

The traditional waterfall model is inadequate as many unknown requirements cannot be specified at the early developmental stages. During the late 1990s, Boehm's spiral model (Boehm, 1988) inspired numerous C2I systems developers to rethink and attempt to adopt it as a risk management process to deal with the continually changing requirements. High technical risks were mitigated through initiating multiple spirals to understand the risks better and synthesise solutions to address the risks. The various spiralling efforts must be orchestrated to generate the synergy to deliver systems capabilities to the commanders to conduct Integrated Warfare.

DEVELOPMENT CYCLES FOR C2I

System Life Cycle. In the 1980s when mainframes and Unix workstations dominated information technology, the life cycle of C2I systems was between 10 to 20 years. For the C2I systems that were installed onboard military platforms, the life span could extend beyond 30 years. During those years, the C2I systems were less complex and software upgrades were infrequent. A typical C2I system would be integrated into a suite of sensors and perform tracking and data fusion to present the situation to the commanders.

VADM Cebrowski conceptualised Network-Centric Warfare in 1998 (Cebrowski, 1998) to explore the effectiveness of networking command and operating nodes to achieve the benefits of speed of command and self-synchronisation. In 2002, the SAF began its journey to transform into the Third Generation SAF that can respond to the challenges of the 21st century. In tandem with this, Integrated Knowledge-based Command and Control (IKC2) was conceptualised. Leveraging the knowledge-centric paradigm, C2I systems would be network-enabled and organised around knowledge for effective command and control (Lee et al., 2003). The shift from the platform-centric to a knowledge-centric paradigm generated challenges for stakeholders to fully specify the complete capabilities of IKC2. Since the advent of IKC2, C2I systems have evolved through several iterations over their life cycles, constantly renewing themselves to maintain relevance and meet the challenges of future threats. A generation of renewal could happen between five to eight years, during which many enhancements would be incorporated to deliver new capabilities. With the challenges of impending asymmetrical threats, a shorter development timeframe has become a necessity for developers to build and deploy systems expeditiously to explore and validate the SAF operational concepts and identify capability gaps. Through the knowledge gained from

operational trials, developers can proceed to integrate or construct new capabilities to plug the gaps. The Continual Systems Development approach has resulted in a blurring of delineation between a prototype and an operational system. This developmental approach has become the choice for developing and delivering C2I capabilities to the SAF.

Continual Systems Development. The development model has to exhibit three characteristics to qualify as Continual Systems Development. Firstly, it has to possess the agility to handle changing requirements to maintain the relevance of the systems over time. Secondly, the time taken from conceptualising the system to developing the capabilities has to be short to enable rapid fielding of the system. This is not unlike commercial competitive pressures of time-to-market. Thirdly, it needs to be evolving to embrace technological opportunities. When the requirements are well understood, systems functionalities and capabilities can be incrementally integrated into the baseline C2I systems through a properly staged schedule for releases. Every incremental release adds to the widening range of operational capabilities of the system. Developers must have the agility to field the system in a shorter cycle and to respond effectively to evolving needs. As requirements are packaged into different releases, developers can work with stakeholders to prioritise the schedule to introduce these new functional capabilities.

When C2I systems exhibit medium to high risks, a flexible model is needed to manage these risks. Risk areas could be due to emergent and evolving operational concepts, leading and bleeding-edge technologies and the resultant architectural risks. Boehm's spiral model is a risk-driven process that guides multi-stakeholders to engineer software-intensive systems (Boehm, 2001). The cyclic approach leads developers to incrementally implement the system while decreasing the risks. While the well-known spiral figure showing the radial and angular growth at each progression seems to suggest that there is a single thread of development, Boehm has highlighted that parallel spiral cycles could be spun off for each software component. The parallel spirals create complexity and project managers need to manage these parallel spirals in congruence.

To better manage the risks holistically, engineering master plans are developed to lay out the approach, risk mitigation strategy and estimate the number and frequency of the spirals. Usually, three to five spirals are needed for the system to stabilise. In a single spiral, there will be several mini spirals executed in parallel or in series. Each mini spiral can last between three to six months while some can be as short as two weeks. Unlike the agile development model that follows a strict timebox control, the mini spirals create the flexibility, time and space for the stakeholders to manage the risks effectively and deliver several releases for rapid operational trials. Developers are not burdened with full-scale documentation but would produce only sufficient artefacts to capture the design considerations, risks, and decisions made which can be referenced to guide future spirals.

A single spiral is typically planned to be completed within a year so that the C2I system can be fielded for at least one major operational exercise. Through the exercises, capability gaps and system deficiencies would be identified and addressed in future spirals. While the spiral model may adequately manage the risks of developing a single C2I system, System-of-Systems (SoS) architecture risks inherently straddle various systems at the enterprise level and pose a different set of challenges. While developing C2I SoS, multiple spirals for each individual system will be concurrently executed. The collective effort to construct these component systems and integrate them into the SoS to achieve the intended result has to be carefully planned and orchestrated. Otherwise, the SoS could become dislocated and would fail to demonstrate its intended Integrated Warfare capabilities.

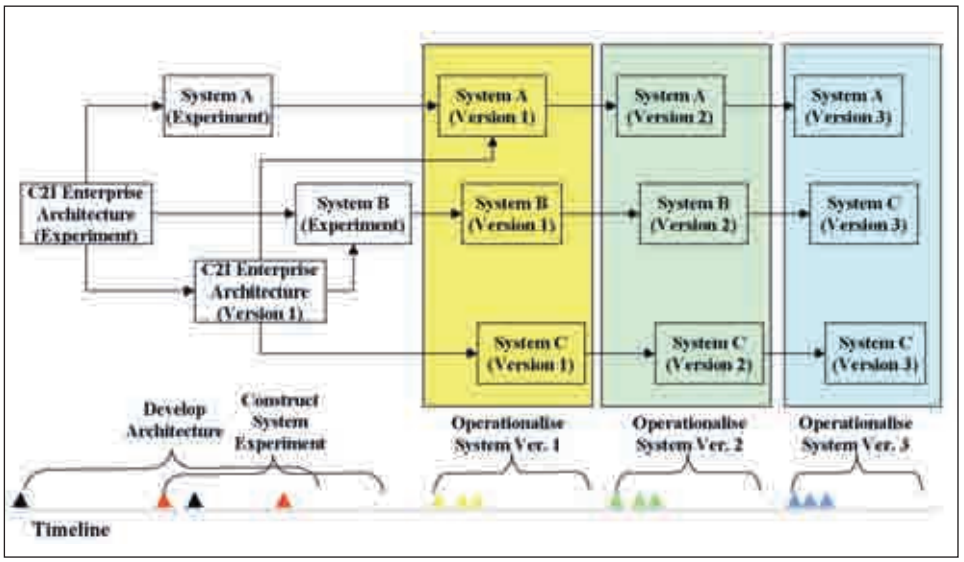

Figure 1. Concurrent Spirals Development for C2I System-of-Systems

Managing Concurrent Spirals. Before embarking on a massive development of the C2I SoS, deliberate and comprehensive planning is carried out. An enterprise architecture has to be constructed so that it can provide the strategic framework to deal with the integration issues pertinent to the development of the C2I SoS (Yeoh et al., 2007). To mitigate the architectural risks, an experimental approach is taken to construct a baseline C2I Enterprise Architecture for experimentation. An experimental System A is then developed to verify the completeness and correctness of the C2I Enterprise Architecture. The C2I Enterprise Architecture continues to evolve into version 1 for the basis of constructing the remaining individual systems. Version 1 of the architecture is verified through developing System B for experimentation. Figure 1 shows the concurrent spirals development for the C2I SoS.

The concurrent spirals have to be managed in a concerted effort to mitigate architectural, technical, process and scheduling risks. Multiple synchronisation points have to be established during the planning phase so that the systems can converge throughout the development and implementation phases. The experimentation-to-operationalisation approach separates the risks into manageable pieces during the concurrent spirals. After verifying the C2I Enterprise Architecture, the individual systems are then staged for development and managed under configuration control. As shown in Figure 1, each version is an incremental release of capabilities into the C2I SoS. The incremental release also reduces integration risks and enables faster time-to-fielding for operational trials. The C2I Enterprise Architecture itself will also evolve with version changes.

Operations and Support (O&S). Prior to the continual development, the development team will hand over the system to a dedicated support team to provide systems maintenance during the O&S phase. The transitioning to O&S entails training of the support team by the development team. The support team would need to re-learn the design from the development team and would invariably be less competent than the development team in fixing the bugs. Therefore, such an arrangement might affect the system and operational readiness.

In the Continual Systems Development framework, the same development team continues to support the system throughout the exercises and daily operations. The team will fix software bugs and develop additional

functions to support the operations. There is no learning curve and we have benefited from the leanness that can be achieved as the knowledge and experience is retained within the team. The veteran developers are around to groom the juniors and pass on the knowledge through on-the-job coaching and supervision. The team continues to support the system and then dovetail to the next spiral development or major upgrade.

ENABLING CONTINUAL DEVELOPMENT FOR C2I SYSTEMS

To ensure proper governance, DSTA has established the Enterprise Architecture Framework to guide the innovation and experimentation of new operational concepts (Yeoh et al., 2007).

Enterprise Technical Architecture (ETA). The early generations of C2I systems for the SAF were developed with a low degree of connectivity among the three Services, namely the Army, Air Force and Navy. With the advancement in information and networking technologies, C2I systems were rapidly networked across the three Services through sharing and integrating common services. Over time, an unwieldy mesh would be created if there were no clear framework for proper governance. As such, DSTA established the Enterprise Architecture Framework to ensure that the C2I systems are developed in compliance with this framework so that the systems are interoperable by design. The framework offers better scalability through a layered and integrated design to create a well-integrated and interoperable development environment.

The ETA is one of the components in the Enterprise Architecture Framework (Yeoh et al., 2007), and is adapted from the Service-Wide Technical Architecture established by the Infocomm Development Authority of Singapore. It aims to establish the principles, standards and development guidelines in the design, development and acquisition of Information Technology (IT) systems that range from ubiquitous corporate IT systems to specialised systems such as C2I systems. There are eight principles to guide the developers to construct the enterprise architecture. Table 1 shows the eight architecture development principles.

The ETA is further organised into a nine-domain architecture to provide the guidance and standards for the developer to construct an enterprise system. Figure 2 shows the nine-domain architecture.

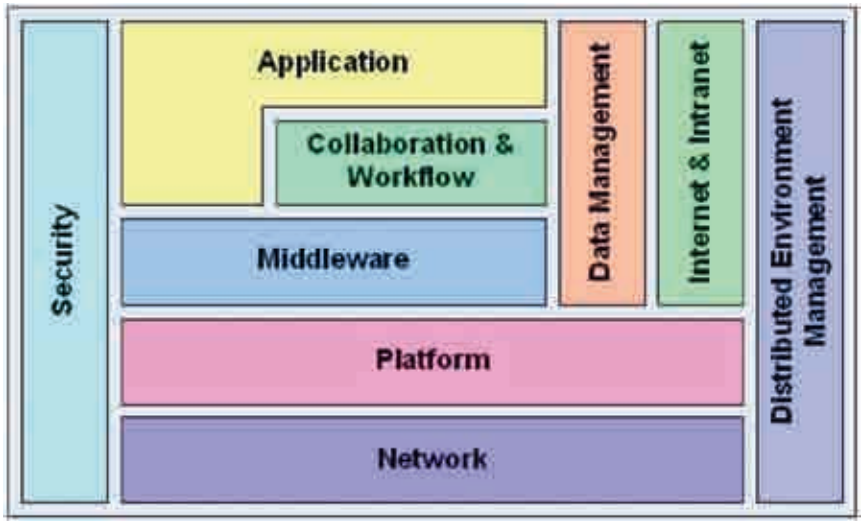

Figure 2. Nine domains of the Enterprise Technical Architecture

Architecture Development Principles		
S/N	**Principle**	**Definition**
1.	Information is a critical asset. It must be effectively collected, managed, exploited, shared and protected.	DSTA should be linked as a single virtual network in order to provide all personnel with on-demand access to authoritative, relevant and sufficient information to perform their duties effectively.
2.	Adequate security	Security must adequately protect information from unauthorised access and protect systems from attacks, both internal and external.
3.	Reduced integration complexity through standards	Interoperability and ease of integration (both intra and inter agency) should be achieved through adherence to open standards with wide industry acceptance and implementing simplified and well-defined interfaces. In the case where no open standards exist, the organisation should adopt a product standard with wide industry acceptance. In addition, standard user interfaces and access protocols should be used for all systems, where standards are available.
4.	Reuse through component-based model	Systems should evolve to employ and share reusable components and infrastructure services across DSTA.
5.	Highly granular	Systems should be engineered to be 'highly granular' and 'loosely coupled'. This can be achieved through N-Tier logical partitioning and implementing event-driven systems and message-based interfaces.
6.	Architecture and infrastructure robustness, scalability, adaptiveness and performance	Architecture design and development should incorporate degrees of robustness, scalability, and adaptiveness to support continuity, growth, and evolution of the business respectively. Performance requirements of a system should be considered in totality which may lead to design trade-offs of components within each domain.
7.	Cost-effectiveness and operational efficiency	Minimising total cost of ownership should be a goal of architecture development. Both initial development costs and ongoing operational costs like system administration and maintenance must be considered in totality. Operational efficiency of the architecture should be considered.
8.	Minimise configuration support	Create a small number of consistent configurations for deployment across the enterprise.

Table 1. Eight principles for developing architecture

• The **Application** domain describes the framework for the design of applications for interoperability, maintenance of a high level of distributed systems integration, reuse of components and rapid deployment of applications to enable a high responsiveness to changing operational requirements.

• The **Collaboration and Workflow** domain defines the environment for the automation of the distribution of ideas, notices and documents throughout workgroups and the entire organisation. The nature of the collaboration and workflow among the users and machines could be based on the processes needed in the workgroup or the conversational type of the interactions.

• The **Internet and Intranet** domain defines the technologies, standards and guidelines for seamless and platform-independent communications among the internal and external nodes.

• The **Data Management** domain defines the mechanics for managing, securing and maintaining the integrity of every data entity. It also describes the structure of authoritative databases and provides the standards to access decision support data.

• The **Distributed Environment Management** domain defines the hardware and software components of the environment that will be controlled through configuration management. The broad disciplines of the domain also include fault detection and isolation, testing, performance measurement, problem reporting and software upgrades and control.

• The **Middleware** domain defines an integration environment between workstations and servers in order to improve the overall usability of the distributed infrastructure. It creates a uniform mechanism for application integration independent of network and platform technologies.

• The **Platform** domain defines the technical computing components of the infrastructure for the client and server hardware to interact with the operating systems. It also describes the storage, backup and high availability components that constitute the hardware infrastructure.

• The **Network** domain defines the communication infrastructure for the distributed computing environment. It describes the logical elements such as topology, physical hardware components and protocols for the networking infrastructure.

• The **Security** domain defines the technologies, standards and guidelines to ensure the availability, integrity and confidentiality of data. The elements include identification, authentication, authorisation, access control, administration and audit.

Common Repository. The common repository is an asset that DSTA has created to support the Continual Systems Development. The repository preserves the intellectual capital of C2I systems business application and technical component services from which developers could draw upon to rapidly assemble and deploy C2I systems for the SAF. As the repository applications and services are thoroughly tested for operational deployment, the assembled C2I systems would achieve a high degree of assured quality for operational trials.

To continually evolve and expand the repository, a rigorous process was adopted to build common applications and services on top of the existing services. When capability gaps are identified, C2I systems developers will develop the new applications and services on top of the baseline C2I systems. The C2I systems are then deployed for operational exercises to verify the implementation and validate that the gaps are satisfactorily covered. After the successful completion of experiments, the new applications and services will be enhanced to

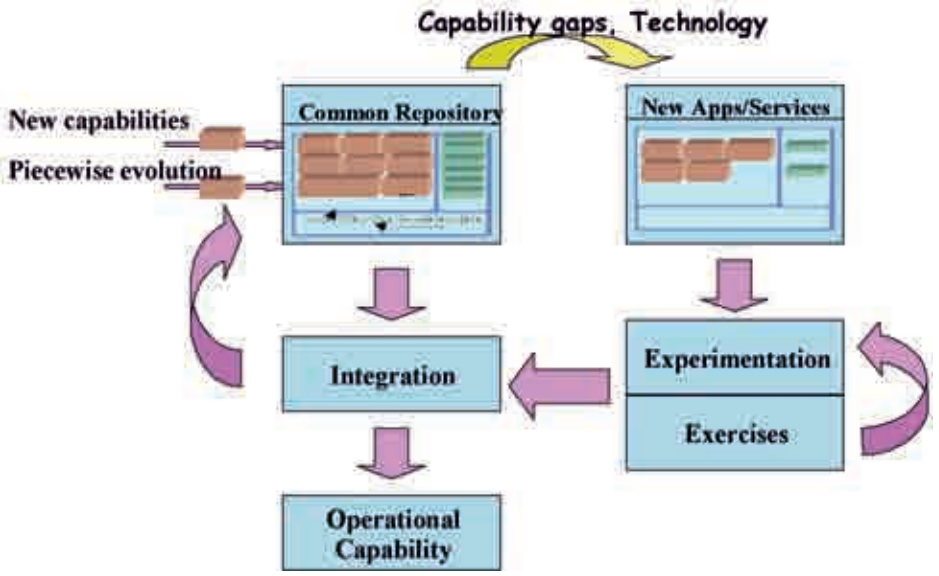

Figure 3. Process for maintenance of the Common Repository

incorporate the non-functional requirements such as exception handling and reliability. The validated services are then integrated into the next baseline C2I systems and added into the common repository for future development. Figure 3 shows the process of continually building up the common repository.

Using Modelling and Simulation (M&S). M&S has been exploited to assist the stakeholders to discover new requirements. As such, C2I developers will team up with M&S analysts to help users to define operational scenarios and develop simulation models to allow users to walk through their scenarios. Through the process of modelling and simulation, stakeholders can develop a better understanding of the operational issues, emerging threats and capability gaps. The system effectiveness could be analysed to establish the factors affecting the overall system performance. Leveraging the newly acquired knowledge, C2I developers can design different architectures. Developers can further employ M&S to evaluate the relative performance of various architectures to select the optimal architecture to meet the desired performance of the C2I SoS.

Using an M&S approach to discover new requirements and develop architectures in early spirals is critical to the continuity of developing C2I systems in future spirals. The right system level and functional level service components are identified for development in the early spirals. These service components can be assembled to form new capabilities and validated using M&S. Another benefit of employing M&S to study the needs is to level up all stakeholders' knowledge on the operational and technical issues. This helps to bring the operational and technical communities to a common footing to build better systems.

Emulator-based C2 Development (EC2D). The EC2D is another approach to address the difficulties of designing first-of-its-kind systems (Yeoh et al., 2007). In the early spirals, emulators can be constructed and assembled to provide emulated inputs from other interacting system components to the C2I systems. These system

components can be the missile system or the surveillance radar system. The C2I systems can be easily assembled using the components from the common repository and then integrated with the emulators to form an emulator-based C2I system environment to facilitate the exploration of operational concepts and evolving requirements. The EC2D provides a low cost and low complexity option in the early development phase when interacting system components are being specified. As and when the interacting system components are available, they will then replace the emulators in the follow-on spirals.

Resource Management System (RMS). Human resources are paramount to the growth and success of the organisation. This is especially true for a technology organisation like DSTA. To maintain leanness, utilisation of limited resources has to be properly planned and allocated to maximise the desired outcomes. The RMS was created to facilitate the planning and allocation of manpower. The developer competencies are kept up to date in the RMS and the system is able to recommend the right match between the resource requirements of the developers and the project. Matching the developers to the appropriate tasks enables a higher probability of success in meeting the system's requirements. In addition, individual developer workload is also captured in the system to avoid overloading the developer. Maintaining a healthy balanced workload allows the developers to spend time thinking creatively and delivering innovative solutions to the SAF.

Developing Competencies. While developers are actively engaged in developing C2I systems, the organisation needs to create space and time for the developers to upgrade their skills and develop new competencies. The effort includes identifying emerging competencies needed for future challenges and committing resources, time and effort to develop the C2I professionals. DSTA established the Organisation Capability Development (OCD) entity in 2006 to embark on this endeavour. OCD works with the C2I developers to chart the individual annual learning plan. The plan highlights the training needs of each developer and determines the training methods, schedule and resources to build his/her capacity and competency. The investment in the developers and their training ensures the sustenance of Continual Systems Development.

CONCLUSION

Continual Systems Development for C2I systems is a risk management approach to developing well-architected and integrated systems. This approach has enabled the rapid development and deployment of C2I systems for the SAF, moving in tandem with its evolving and expanding scope of desired capabilities to tackle the challenges of the 21st century.

REFERENCES

Adams, G., Daniel, D. (2000). Managing Concurrent Development – A Systems Engineering Approach, Autotestcon Proceedings, IEEE.

Aoyama, M. (1993). Concurrent-Development Process Model, Software, IEEE, Vol. 10, Issue 4, July, pp 46-55.

Boehm, B. (1998). A Spiral Model of Software Development and Enhancement. Computer, Vol. 21, No. 5, May, pp 61-72.

Boehm, B., and Hansen, W. (2001). The Spiral Model as a Tool for Evolutionary Acquisition. CrossTalk.

Cebrowski, Vadm A. and Garstka, J. (1998). Network-Centric Warfare: Its Origin and Future. Proceedings of the Naval Institute, January.

Infocomm Development Authority of Singapore (2006). Service-Wide Technical Architecture. Retrieved on August 2007 from http://intranet.igov.gov.sg/GovernanceandManagement/PoliciesAndStandards/SWTA/

Krygiel, A. J. (1999). Behind the Wizard's Curtain. CCRP Publication Series.

Lee, J., Ong, M., Singh, R., Tay, A., Yeoh, L.W., Garstka, J., Smith, E. (2003). Realising Integrated Knowledge-based Command and Control – Transforming the SAF. Pointer Monograph No.2.

Yeoh, L.W., Chung, W.K., Cai, J. (2007). Emulator-Based C2 Development: A Systems Engineering Approach To Developing C2 Systems. Asia-Pacific Systems Engineering Conference, Singapore.

Yeoh, L.W., Lew, C. S., Cheung, D.S.K., Lee, C.C.W. (2007). Using Modelling and Simulation System-In-The Loop Solution to Enhance Productivity and Service Level for Military Communication Systems Development. Asia-Pacific Systems Engineering Conference, Singapore.

Yeoh, L., Syn, H., Lam, C. (2007). An Enterprise Architecture Framework For Developing Command and Control Systems. 17th Annual International Symposium of the International Council on Systems Engineering.

This paper was first presented at INCOSE 2008, 15 - 19 June 2008 in the Netherlands and has been adapted for publication in DSTA Horizons.

BIOGRAPHY

Dr Yeoh Lean Weng is Director (C4I Development and Systems Architecting). He is also concurrently the Deputy Director of the Temasek Defence Systems Institute and an Adjunct Professor at the National University of Singapore (NUS). Lean Weng has extensive experience working on large-scale defence engineering systems. As a systems architect, he played a key role in developing the Enterprise Architecture for defence applications. He also developed the systems architecting methodology for masterplanning and defence transformation. He is also the Vice-President of the INCOSE Singapore Chapter, INCOSE Region VI Representative to Member Board and the Chairman of Systems Engineering Technical Committee, Institution of Engineers, Singapore. Lean Weng received his Bachelor and Master of Science degrees from NUS in 1983 and 1987 respectively. He further obtained two Masters in 1990 and a PhD in Electrical Engineering from the Naval Postgraduate School (NPS), USA in 1997.

Teo Tiat Leng is currently Deputy Director (Industry) in the Defence Industry & Systems Office, Ministry of Defence. He has more than 15 years of experience in software engineering and systems engineering, developing C4I systems for the Singapore Armed Forces. His current work encompasses policy matters pertaining to the defence technology ecosystem. He received his Master of Science (Computer Science) with Distinction from NPS, USA and a Master in Defence Technology & Systems, Master of Technology (Software Engineering) and Bachelor of Science (Computer & Information Science) degree from NUS.

Lim Horng Leong is a Programme Manager (C4I Development). He led the development of several large-scale command and control systems and successfully fielded the systems for the Republic of Singapore Navy. He is currently applying Systems Engineering methodologies for C4I experimentation. Horng Leong received his Bachelor of Science degree (Computer and Information Sciences) from NUS in 1996. He also holds a Master of Science in Systems Engineering from NPS, USA.
