CEREBELLUM FUNCTION FOR MSAAS

Erdal Cayirci[(a)], Hakan Karapinar[(b)], Lutfu Ozcakir[(c)]

[(a)] Electrical Engineering and Computer Science Department, University of Stavanger, Norway
[(b),(c)]Simulation, Training and Test Systems Department, HAVELSAN, Turkey

[(a)]erdal.cayirci@uis.no, [(b)]hakank@havelsan.com.tr, [(c)]ozcakir@havelsan.com.tr

ABSTRACT
Modelling and simulation as a service (MSaaS) offers elasticity, better utilization, higher reusability and many more advantages. However, MSaaS has a number of challenges studied in the literature. One of these challenges, namely time sensitivity of the services in an MSaaS federation, is introduced. Our scheme, called cerebellum function, is designed to address this new challenge. The conditions related to time sensitivity and the algorithm for the cerebellum function are also presented.

Keywords: modelling and simulation as a service, cloud, service, MSaaS, cerebellum function, quality of service, QoS

1. INTRODUCTION

Modelling and Simulation as a Service (MSaaS) is a model for provisioning, modelling and simulation (M&S) services on demand from an MSaaS provider (MP), which keeps the underlying infrastructure, platform and software details hidden from the MSaaS Customers (MC) (Cayirci, 2013a; Johnson, 2013). MP is responsible for licenses, software upgrades, scaling the infrastructure according to evolving requirements and accountable to the MC for providing grade of service (GoS) and quality of service (QoS) specified in the service level agreements (SLA). MSaaS introduces better utilization, ease in technical administration and therefore cost reduction. It also implies a big paradigm shift in computing and a long list of challenges related to both its ecosystem and technical requirements. Academia and industry have already tackled with many of those challenges (Cayirci, 2013a; Cayirci, 2013b; Cayirci, 2013c; Cayirci, 2014; Cayirci, 2015; Jensen, 2009; Subashini, 2012). In this paper, we focus yet another challenge for MSaaS: delay sensitivity of some M&S services, specifically military MSaaS; and present the preliminary results from our research on a new scheme that we call as cerebellum function for MSaaS.

A cloud can provide three basic service types (i.e., service models) as shown in Figure 1 (Ambrust, 2010; Badger, 2012; Cayirci, 2013a):

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS) and
- Software as a Service (SaaS).

There are also many other service types introduced in literature, such as, Network as a Service (NaaS), Trust as a Service, Authorization as a Service (Laborde 2013). These are derivations of PaaS and SaaS in various combinations and forms. MSaaS can be perceived as one of these derivatives. MSaaS is in essence a special form of SaaS. The inter-relations between MSaaS, SaaS, PaaS and IaaS are depicted in Figure 1. We consider three types of MSaaS:

- Modelling as a service,
- Model as a service and
- Simulation as a service.

An MC may develop models by using modelling as a service, use previously developed models to run simulations in their enterprise (i.e., model as a service) or run simulations by using simulation as a service.

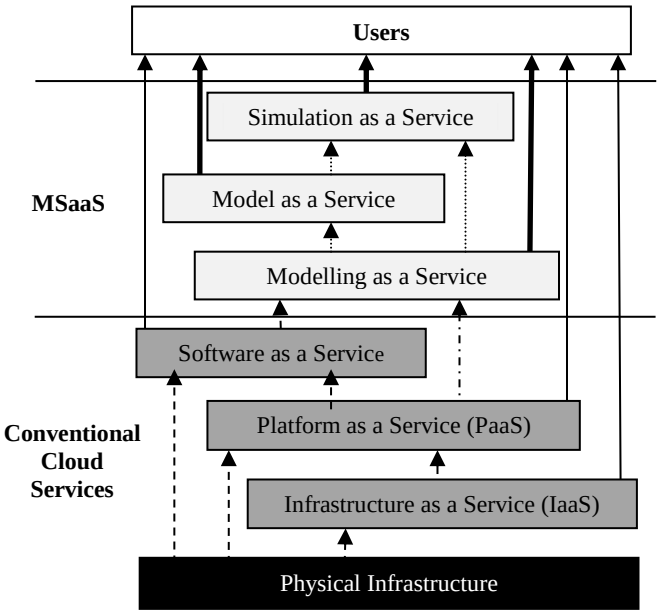

Figure 1: The Inter-relations of Cloud Services Including MSaaS

Please note that a cloud (i.e., a cloud service provider) typically maintains multiple data centers remotely located from each other. A data center is a facility that houses server pools and infrastructure to store, to process and to communicate large volumes of data.

Hence, MSaaS can be designed in one of the following forms:

- Standalone MSaaS applications: Standalone applications, such as business process modelling and supply chain simulation (Rosetti 2012), are already available as simulation as a service in the Internet.
- Federated standalone MSaaS applications: Standalone MSaaS applications can be federated. These applications can be from the same data center or multiple data centers.
- Composed MSaaS: Not standalone applications, but services and data that can be integrated into a composite service are offered as MSaaS.
- Automatically composed MSaaS: As the technology and interoperability among the services mature, composed MSaaS become automatically discoverable and composable with each other.

In this paper, we focus on composed and automatically composed MSaaS. As explained in (Cayirci 2013a; Cayirci 2013b), composing an MSaaS from the services provided by multiple data centers is a challenging task. In the literature, composed cloud services are called as service mash-ups or service federations. We will use the term "MSaaS federation" for a composed MSaaS, and the term "federate" for each service that the federation is composed of. Federation has a different meaning in cloud computing from M&S. In cloud computing, the term "federation" is used not only for federating models but also for infrastructure or platforms, and therefore a federation may also mean a cloud service that integrates various resources in the form of IaaS (e.g., memory, processor time, etc) from multiple data centers (Buyya, 2010; Cayirci, 2013b; Singhal, 2013; Toosi 2011). On the other hand, in MSaaS domain, federations integrate multiple MSaaS either in standalone application or service module form. Please note that we do not imply any architecture, such as high level architecture (HLA), when we use the term federation, but integration of various M&S services for composing an MSaaS. We categorize MSaaS federations into four broad classes as depicted in Table 1 (Cayirci 2013a):

- Type 0: Federation of standalone applications located in the same data center (Toosi 2011)
- Type 1: Composite MSaaS of service modules located in the same data center
- Type 2: Federation of standalone applications from multiple data centers (Cayirci 2013b)
- Type 3: Composite MSaaS of service modules from multiple data centers (Cayirci 2013b).

Table 1: Types of MSaaS Federations

Nature of Federates	Intra datacentre	Inter datacenter
Standalone applications	Type 0	Type 2
Composed services (SOA)	Type 1	Type 3

MSaaS intrinsically introduces a new challenge, namely the physical distance and therefore the propagation delay in between the user device and the cloud. This challenge is exacerbated by the additional computational delay due to service federating. In the Internet, few hundred milliseconds of round trip times (RTT) can be expected. Based on the physical distance and other computational delays, RTT can be significantly higher than few hundred milliseconds, which may become an important issue for interactive MSaaS. In this paper, we elaborate the dynamics of this challenge, and introduce our new cerebellum function for MSaaS scheme, which is developed to tackle with that.

In the following section, we examine various potential services for military MSaaS federations. Their quality of service requirements are also discussed and listed qualitatively. In Section 3, we explain our cerebellum function scheme briefly and introduce a practical algorithm for configuring and locating cerebellum functions. We conclude our paper in Section 4.

2. POTENTIAL SERVICES IN A MILITARY SERVICE ORIENTED MSAAS

In this Section, we will focus on the potential services for a service oriented military MSaaS. Table 2 depicts a preliminary list of services. We expect that this list will have hundreds of services, and therefore, Table 2 is not even close to be exhaustive. However, Table 2 includes key MSaaS services which are already implemented by various organizations including HAVELSAN. These services can also be differentiated based on several factors, such as, level of fidelity, level of resolution, the type of simulation that they are designed for (i.e., live, virtual and constructive). Therefore, Table 2 is only a preliminary version of our work, and based on a focus group study. We are currently developing a more detailed and quantitative version of the same table.

In Table 2, we focus on four QoS parameters: reliability, bandwidth, jitter and delay. All the services in the Table are highly sensitive against reliability because they are based on the transfer of digital data. Bandwidth and jitter requirements/constraints are low for almost all services. On the other hand, almost all of these services have highly constrained delay requirements. Especially, the sensitivity of interactive visualization services (IVS) against delay is very high. It is clear that one of the most challenging M&S services with respect to the quality of service parameters is IVS. Therefore, we use IVS as our example in the later sections of this paper.

IVS is a key service for immersive M&S, which visualizes the virtual environment including the simulated entities (i.e., all objects including people) interactively, such that the angle and point of projection can be changed instantaneously based on user commands. Its quality and responsiveness is paramount for the immersion of the virtual simulation users, such as, the trainees in the aircraft and tank simulators. Moreover, the response time of IVS to the controls must

be the same as the real system. Otherwise, although the virtual simulation provides a perfect immersion, negative training can be given. For example, if a virtual aircraft responds the controls more rapidly or slowly than the real aircraft, the trainee may develop wrongly conditioned reflexes.

Table 2: Potential Cloud Services for Military MSaaS

Service	Reliability	Bandwidth	Jitter	Delay
Weapon Effects	High	Low	Low	High
Exterior Ballistics	High	Low	Low	High
Common Effects	High	Low	Low	High
Synthetic Environ.	High	High	Low	Medium
Synthetic Dynamic Environment	High	Medium	Low	Medium
Weather	High	Low	Low	Medium
Geography/ Hydrograph	High	High	Low	Medium
Line of Sight	High	Low	Low	Medium
Order Translation	High	Low	Low	Low
Autonomous Planning	High	Low	Low	Low
Movement	High	Low	Low	Medium
Supply	High	Low	Low	Medium
Mainten.	High	Low	Low	Medium
Detection and Recognition	High	Low	Low	High
Comms	High	Low	Low	Medium
Human/ Social Behavior	High	Low	Low	Medium
Troop/ Platform/Unit Behavior	High	Low	Low	Medium
Threat Network	High	Low	Low	Medium
Recognized Picture	High	Medium	Medium	Medium
C4 Population and Stimulation	High	Medium	Medium	Medium
Visualiz.	High	High	High	High
Interactive Visualiz.	High	High	Very High	Very High

A cloud has two ends: front-end and back-end as shown in Figure 2. The front-end is where the user interfaces are. That is the only part of a cloud visible to the users, and should not need any special hardware. The user sees the back-end as a cloud without knowing any details about its internal architecture. The back-end includes various components (i.e., storage space, processors and platforms) loosely coupled to each other through a mechanism that allows elasticity. The delineation between front and back ends introduce longer propagation delays between the user interface and the service comparing to conventional computation schemes where user interface and server are physically co-located. In addition to that, a composed MSaaS (i.e., an MSaaS federation) may have many MSaaS received from multiple data centers. Even an MSaaS federation may have a federate, which itself is a federation with federates from the other data centers. This nested architecture can definitely introduce jitter and delay, which is not manageable for services like IVS.

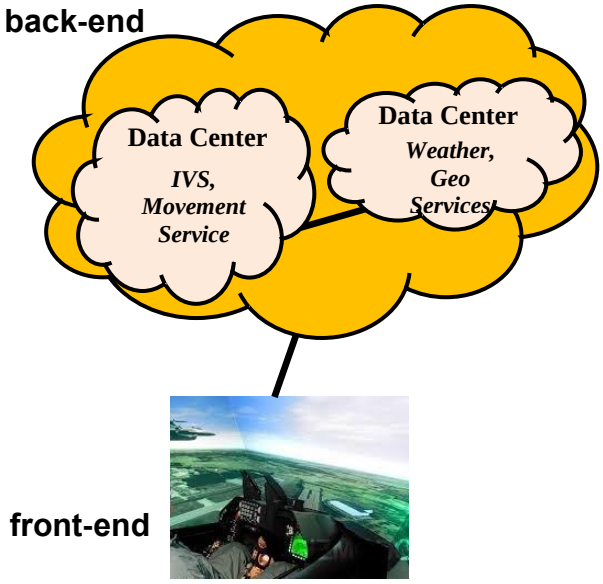

Figure 2: An Example for Back-end and Front-end in MSaaS (the services in the figure are just an example)

3. CEREBELLUM FUNCTION

Cerebellum function includes the part of an MSaaS federation which is time sensitive in responding the user commands (i.e., inputs). Please note that we do not mean shortest delay by time sensitivity, but the delay arranged the same as the delay in response by the real system to the user commands. For example, if the delay in real system d_r is between 90 and 100 msec, the delay in virtual system needs to be within exactly 90-100 msec window. It is expected that r is a random variable, which (i.e., both distribution and parameters) may change for various systems. Our scheme is based on the idea that the maximum delay between the user interface and cerebellum function d_{max} must be shorter than the lower bound of the real life system delay $r_{\alpha min}$ according to a given confidence level α. Hence, we can manage the delay such that negative training is avoided and immersion is maintained. The maximum delay d_{max} includes not only the propagation delay p_{max} introduced

by the physical distance between two ends of a communications link but also all sort of computational delays c_{max} due to processes, such as encryption, decryption, routing, service federating, etc. It is clear that we need to treat also $d_{\alpha max}$ as a random variable, and make our computations based on the upper bound according to the given confidence level α.

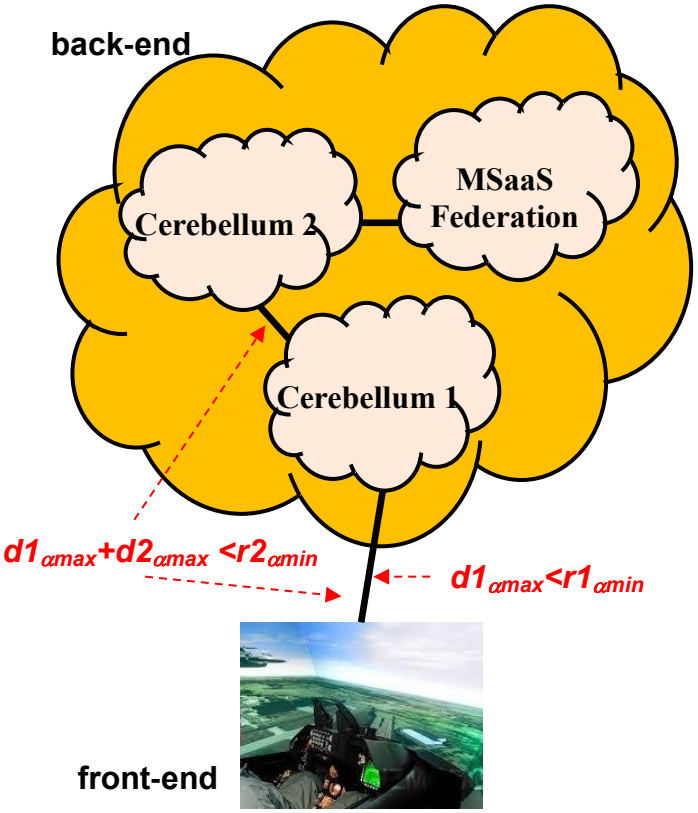

Figure 3: Cerebellum Function

When the services are designed, the designer should think about if a service is time sensitive, and if the time sensitive part of the service can be separated from the rest of the service. If there is a time sensitive part of a service, that part needs to be designed decomposable. Hence we do not need migrating all the service and data closer to the front end but only the time sensitive part of the service. For example, the part of IVS that fetches the terrain data and weather conditions and creating three dimensional virtual environments can be designed separately from the part that makes the projections based on the user commands. The later part, which is time sensitive, becomes the cerebellum function for IVS. Please note again that this is only a simplified example to clarify what we mean by cerebellum function. In some cases, not only the cerebellum function of a service, but all of the service may be treated as a cerebellum function depending on the configuration of an MSaaS federation. If an input of Service s_a uses another Service s_b, which has a part that needs to be treated within the cerebellum function, s_a as a complete service has to be within the cerebellum function. Moreover a cerebellum function may also have a nested structure, which means that the inputs of a cerebellum function may be coming from another cerebellum function. Therefore, the location of a cerebellum function is selected such that the conditions in Equations 1 and 2 are met.

$$dn_{\alpha\max} = \sum_{k=1}^{n} u(pk_{\alpha\max}) + u(ck_{\alpha\max}). \quad (1)$$

$$dn_{\alpha max} < u(rn_{\alpha max}). \quad (2)$$

where n-1 is the number of cerebellum functions preceding the cerebellum function *n* in the nested structure.

Algorithm 1 is designed for configuring and locating cerebellum functions. After selecting all the services in an MSaaS Federation, a dependency tree is constructed starting from the service that directly interacts with the user. That is typically the user interface for an MSaaS Federation. Then all the services in the service fan in (the services used by the service) of each service are inserted into the tree as a child node to the service.

```
Select all the services required for an MSaaS federation
Create dependency tree for the services in MSaaS Federation
    Insert the top level service as the root node
    n=0
    Repeat until none of the services at level n has another service in their service fan in
      Insert all the services in the service fan in of a service at level n into the tree
      n=n+1
Run depth first traverse of the tree and create cerebellum function structure
    If Service a has a cerebellum function
      If Service a has a parent Service b
        If u(ra_max) < u(rb_max)
          Merge cerebellum functions
          Service a = Service a – Cerebellum a
          Cerebellum b = Service b + Cerebellum a
          u(rb_max) = u(ra_max)
          Remove Cerebellum a
Locate the cerebellum functions according to their time sensitivity
```

Algorithm 1: Cerebellum Function Configuration Algorithm

When the dependency tree is complete, it is traversed in the depth first order. If a cerebellum function of a service has more stringent time sensitivity constraint comparing to its parent service, all of the parent service is treated as a cerebellum function together with the cerebellum function of the child service. The time sensitivity of the merged cerebellum function is assigned with the time sensitivity parameter of the child service. After the traverse of the tree and merging of the cerebellum functions are complete, the cerebellum

functions are migrated to appropriate machines. A cerebellum function can be located in any data center that accepts the function and satisfies the constraints in Equations 1 and 2 within the cloud or in the front end machine itself if none of the data centers or servers in the Cloud can provide the required conditions.

The cerebellum function for IVS can also support the design of architectures that fulfil security constraints of military MSaaS. Although the environmental data, and specifications of military equipment, such as maximum speed and altitude that a military aircraft can reach, are unclassified, the turn rates and similar data about the aircraft may be classified. Since the effects like turn rates are time sensitive and therefore will be typically treated by a cerebellum function in IVS, the cerebellum function approach may become useful also for tackling with the security related challenges of MSaaS. Since this is a different topic, we do not further elaborate on the cerebellum function for enhanced security in MSaaS in this paper.

4. CONCLUSION

MSaaS is an emerging approach for M&S following the latest trends in information technologies. It promises many advantages, such as rapid elasticity, ease in technical administration and licensing, better utilization, pays per use, and therefore enables considerable cost reduction. However, it also introduces many challenges including security, privacy, accountability, risk and trust management and service composition. Industry and academia have tackled with these challenges for almost a decade. In this paper, we introduce yet another challenge related to quality of service guarantees, specifically delay and jitter. When a service is a time sensitive service, it needs to be physically close enough to the front end. However, migrating all the service and data closer to the front end may not always be feasible. In such a case, a service may be divided into two parts: a time sensitive part, and the other part which is not time sensitive. We call the time sensitive part that we need to locate closer to the front end, and sometimes at front end, as cerebellum function of the service. A practical algorithm that is designed for selecting and configuring the services with cerebellum function is presented. We are currently implementing a cerebellum function for IVS as a prototype for experimentation.

REFERENCES

Armbrust M., A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. 2010. A view of Cloud computing. *Communications of the ACM, vol. 53, no. 4, pp. 50–58*.

Badger L., T. Grance, R.Patt-Corner and J.Voas. 2012. Draft Cloud Computing Synopsis and Recommendations. *National Institute of Standards and Technology, Special Publication 800-146*.

Buyya R., R. Ranjan, R.N. Calheiros. 2010. InterCloud: Utility-oriented federation of Cloud computing environments for scaling of application services. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'10)*, pp. 13–31.

Cayirci E. 2013a. Modelling and Simulation as a Cloud Service: A Survey. *In Proceedings of the 2013 Winter Simulation Conference, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl*. Washington DC, December.

Cayirci E. 2013b. Configuration Schemes for Modelling and Simulation as a Service Federations. *Simulation Transactions of the Society for Modelling and Simulation International*, Vol. 89, Issue 11, pp. 1388 – 1399, November 2013.

Cayirci E. 2013c. A Joint Trust and Risk Model for MSaaS Mashups. *In Proceedings of the 2013 Winter Simulation Conference, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl*. Washington DC, December.

Cayirci E. , A. Garaga, A.S.Oliveira and Y. Roudier. 2014. Cloud Adopted Risk Assessment Model. *International Workshop on Advances in Cloud Computing Legislation, Accountability, Security and Privacy (CLASP)*.

Cayirci E. and A.S.Oliveira. 2015. Modelling Risk and Trust for Cloud Service Mashups. *IEEE Transactions on Cloud Computıng* (submitted).

Jensen M., J.Schwenk, N.Gruscka and L.L.Iacono. 2009. On Technical Security Issues in Cloud Computing. *IEEE International Conference on Cloud Computing*, pp. 109-116.

Johnson H. and Tolk A., "Evaluating the Applicability of Cloud Computing Enterprises in Support of Next Generation of Modelling and Simulation Architectures," *Spring Simulation Multi-Conference*, April 2013.

Rosetti M. And Chen Y., "Cloud Computing Architecture for Supply Chain Network Simulation," *Winter Simulation Conference*, December 2012.

Singhal, M., S. Chandrasekhar, G. Tingjian, R., S. Sandhu, R. Krishnan, G-J. Ahn, E. Bertino. 2013. Collaboration in Multicloud Computing Environments: Framework and Security Issues. *IEEE Computer Magazine February 2013*, pp 76-84.

Subashini, S. and Kavitha, V. 2011. A survey on security issues in service delivery models of cloud computing. *Elsevier Journal of Network and Computer Applications*, Vol. 34, Issue 1, pp. 1-11.

Toosi A.N., R.N.Calheiros, R.K.Thulasiram, R.Buyya. 2011. Resource Provisioning Policies to Increase IaaS Provider's Profit in a Federated Cloud Environment. *HPCC 2011*.

AUTHORS BIOGRAPHY

Erdal Cayirci graduated from Army Academy in 1986 and from Royal Military Academy, Sandhurst in 1989. He received his MS degree from Middle East Technical University, and a PhD from Bogazici University both in computer engineering in 1995 and 2000, respectively.

He retired from the Army when he was a colonel in 2005. He was a faculty member and a researcher at Istanbul Technical University, Yeditepe University, Naval Sciences Institute and Georgia Institute of Technology between 2000 and 2005. He is currently Head, CAX Support Branch in NATO's Joint Warfare Center in Stavanger, Norway, and also a professor in the Electrical and Computer Engineering Department of University of Stavanger. His research interests include computer simulations, cloud computing, risk and trust modelling, mobile communications and sensor networks.
He received the "**2002 IEEE Communications Society Best Tutorial Paper" Award** for his paper titled "A Survey on Sensor Networks" published in the IEEE Communications Magazine in August 2002, the **"Fikri Gayret" Award** from Turkish Chief of General Staff in 2003, the "**Innovation of the Year" Award** from Turkish Navy in 2005 and the **"Excellence" Award** in ITEC 2006.
He co-authored two books titled as "Security in Wireless Ad Hoc and Sensor Networks," and "Computer Assisted Exercises and Training: A Reference Guide" both published by John Wiley & Sons in 2009.

Hakan KARAPINAR graduated from Hacettepe University Electrical & Electronics Engineering Department in 1996. He received his MS degree from Bilkent University in 1998 about antenna simulation. He started working at HAVELSAN Company; Simulation, Training and Test Systems department in 1998 and he is still working as Program Director in that division. He started his PhD studies at Hacettepe University Electronics Engineering Department and MBA studies at Cankaya University, Turkey. His research interests include Real Time Simulation, Modelling, Distribution Interactive Simulation, Electronics Warfare, Radar, Antenna, Tactical Environment Simulation and Sensor Simulation.
In HAVELSAN, between 1998 and 2005, he worked as System Engineer, System Team Leader, System Group Manager working actively on simulation of electronic warfare and weapon systems. Between 2005-2011, he worked as a Project Manager and Program Manager for fixed-wing and rotary-wing platform training simulators projects. Till 2011 he is working as Programs Director responsible for all Simulator and Simulation programs at HAVELSAN, managing around 22 projects and programs with more than 600 M $ value for local and foreign customers.

Lutfu OZCAKIR, graduated from Hacettepe University (Ankara) Electronics Engineering Department in 1996 as honour faculty student. He received his MS degree from of Bilkent University in Electronics Engineering Medical Signal Processing in 1998. In 1998 he has joined HAVELSAN company Simulation and Training Systems Division as responsible engineer of Air Force Wargaming systems.
He started his PhD studies about simulation of anatomical structures at Hacettepe University, Ankara.
His research interests include Medical Signal Processing, Tactical Environment Simulation, Real Time Modelling, Distributed Joint Simulation Systems and C2 Simulation.
Between 1998-2005 as a team leader, group manager and project manager he has completed many programs in simulation and training systems of HAVELSAN. From 2005-2011 he worked as Project Manager and Program Director of all projects at HAVELSAN Simulation Systems division. Till 2011 he is working as Executive Vice President of Simulation, Training and Test Systems division and board member of HAVELSAN Technology Radar (HTR).