NYHETSBLAD FÖR SESAM
Försvarssektorns Användargrupp för Software Engineering

RENDEZVOUS

Nr 2 augusti 2004

Innehåll

Vad är SESAM?

SESAM är ett samverkansnätverk för projektövergripande och företagsneutral kunskapsuppbyggnad och kunskapsspridning inom området programvaruintensiva försvarssystem.

SESAM skall genom organiserat samarbete mellan dess medlemmar (företag, organisationer, myndigheter och utbildningsinstitutioner) främja tillförlitlighet och effektivitet i utveckling och vidmakthållande av programvaruintensiva försvarssystem.

SESAMs verksamhet att samla, skapa och sprida information och kunskap sker huvudsakligen i arbetsgrupper som verkar inom avgränsade tekniska områden och som efter behov inrättas och avvecklas.

SESAM anpassar, profilerar och förnyar sin verksamhet med hänsyn till ändrade tekniska och andra omständigheter av betydelse för intresseområdet.

När SESAM startade sin verksamhet 1988 var s k inbyggda realtidssystem (i farkoster, sensorer, stridsledningssystem m fl) dominerande inom försvaret och svensk försvarsindustri. Det fanns ett allmänt behov att kunna driva utvecklingen av programvara för dessa alltmer komplexa system på ett mer organiserat sätt, d v s att mer konsekvent praktisera vad som börjat benämnas Software Engineering. SESAMs verksamhet inriktades därför från början på användningen av det speciellt för inbyggda och realtidssystem under början av 80-talet i brett internationellt samarbete framtagna och av ISO 1987 standardiserade programspråket Ada, vilket hade utformats särskilt för att stödja tillämpning av Software Engineering principer. Den fortsatta användningen av Ada, inkl i nya (Ada 95) och framtida versioner, är fortfarande av stort intresse och betydelse för många av medlemmarna i SESAM.

Efterhand har även andra typer av programvarusystem blivit mer förekommande inom försvarssektorn, t ex på informationssystemområdet. Utvecklingen inom kommunikationstekniken, t ex via Internet snabba expansion, har också tillfört nya möjligheter och typer av problemställningar för programvaruutvecklingen. Utbudet av kommersiellt tillgänglig programvara (COTS) som man önskar kunna använda i försvarssystem har också ökat. Inriktningen på uppbyggnad av ett nätverksbaserat försvar, behovet av interoperabilitet vid internationella insatser m m har aktualiserat nya problemställningar i systemutformningen. Denna utveckling gör att nya tekniker, processer och metoder, språk och andra hjälpmedel etc för framtagning och vidmakthållande av programvaruintensiva system av intresse att behandla inom SESAMs verksamhet ständigt tillkommer.

SESAM styrs av ett Råd med representanter för gruppens medlemmar. Rådet har till sin hjälp ett Verkställande Utskott (VU) och ett sekretariat.

Rådets ordförande är Claes Wadsten, AerotechTelub, tel 013-231652 .

VU

Andersson Tommy, Ericsson Microwave Systems AB

Carlsson Ingemar, adjungerad

Ekman Mats, Saab Aerospace AB

Hallén Johan, FMV

Merkell Curt, Saab Bofors Dynamics AB

Sandberg Leif, FMV

Sporre Lena, FOI

Wadsten Claes, AerotechTelub AB

Wåhlén Ulf, FMV

Arbetet utförs i två arbetsgrupper och en intressegrupp:

Ag Metodik

Håkan Edler, CTH/Datorteknik

Ag Teknik

Erik Dyrelius, Saab Combitech Systems

Ig Programvarusäkerhet

Inga-Lill Bratteby-Ribbing, FMV

Vilka kan vara med i SESAM?

Medlemmarna i SESAM är företag, organisationer och myndigheter (förvaltningar, utbildningsinstitutioner etc) i Sverige med anknytning till försvarssektorn. Medlemmarna indelas i följande kategorier
- ordinarie medlemmar
- arbetsgruppsmedlemmar
- informationsmedlemmar.

Enskild person kan endast komma ifråga som informationsmedlem.

Inträde i SESAM

För samtliga medlemskategorier gäller att inträde beslutas av Rådet. För inträde som ordinarie- eller arbetsgruppsmedlem krävs status som leverantör till FMV eller FM. Dessutom krävs en skriftlig förbindelse att uppfylla åtagande som ordinarie- resp arbetsgruppsmedlem. För inträde som informationsmedlem (erhåller endast informationsbladet) krävs status som leverantör till FMV eller FM eller status som myndighet inom totalförsvaret. Rådet kan emellertid anta annan part som informationsmedlem.

För ansökan om medlemskap i SESAM vänd er till sekretariatet.

SESAM-Sekretariatet
c/o Kåsjös Kontor, Anna Kåsjö
Odengatan 28, 4 tr
113 51 STOCKHOLM
Tel: 08-510 51866, 070-716 9702
Fax: 08-510 51932
E-post: kasjos.kontor@bredband.net

Ordföranden har ordet

Vi är nu uppe i full verksamhet för 2004 och med några år tillbaka som ordföranden vet jag att i slutet på sommaren och under hösten ökar de ”offentliga” aktiviteterna i SESAM. Det är då vi genomför seminarier, avslutar och redovisar projekt. Detta år är inget undantag.

Att samlas i det nätverk som SESAM utgör blir mer och mer aktuellt då alla industrier och myndigheter arbetar i samarbetsprojekt. De större system som utvecklas idag måste utvecklas så att de passar ihop med andra nya och gamla system. Vi ser också en globalisering inom utvecklingen och senare bland användare varför ett kontaktnät ökar i betydelse. Ingen kan idag utveckla sitt eget lilla system utan att titta på vad andra gör.

SESAM får därför inte bara en betydelse som kontaktnät för att diskutera vissa enskilda tekniska frågor eller titta på smarta nya lösningar utan är också en träffpunkt där man kan diskutera nya ”standards” inom datorarkitektur och datorsystemutveckling.

Kraven på robusta system blir också mer nödvändig i framtida system då dessa ofta är sammanlänkade med andra system. Det gamla ordspråket ” ingen kedja är bättre än den enskilda länken” får här en klar betydelse.

De tankar som fanns när SESAM startade med tonvikt på Ada, som då användes för robusta/säkerhetskritiska realtidsystem, är idag frågor som man kan tillämpa inom hela datorområdet. Betydelsen av SESAM är därför inget som minskar utan snarare tvärt om. Jag ser därför ljust på framtiden för den verksamhet SESAM bedriver.

Jag vill i denna ledare göra lite reklam för höstseminariet där vi också tillåter andra än medlemmarna i SESAM att deltaga som åhörare. Höstseminariet genomförs den 20 oktober och har rubriken ”INFORMATION - FRÅN BÖRDA TILL RESURS?; Informationshantering i framtida försvaret”. Det är ett gediget program med många föreläsare som behandlar frågor som berör många i nya stora projekt. Ämnet är lika aktuellt för utvecklare och beslutsfattare utanför försvaret.

Avslutningsvis hoppas jag att vi vid årets slut kan säga att vi genomfört ytterligare ett aktivt år i SESAM där många olika frågor belysts och redovisats.

Claes Wadsten
Ordförande i SESAM

Tfn. dir : 013-23 16 52
Mobil: 070-6000271
E-mail: claes.wadsten@aerotechtelub.se

Konferensrapport från Ada-Europe 2004

Årets konferens, som var den 9:e i ordningen med rubriken Reliable Software Technologies, hölls i Palma de Mallorca och samlade c:a 100 deltagare.

Då jag jämför med de Ada-Europe-konferenser jag varit på tidigare år, gav denna en känsla av att vara mer positiv och framåtsyftande. Kanske framför allt därför att en ny standard, Ada 0Y (kanske Ada 05?) kommer att vara verklighet inom en nära framtid, men också för att marknaden med kompilatortillverkare och verktygsmakare tycks ha stabiliserats.

Utställningsdelen omfattade 8 bord med deltagande från Ada Core (f.d. ACT), Aonix, Green Hill Software, I-Logix, LDRA, Praxis, Raincode och TNI.

Största antalet konferensdeltagare kom givetvis från Spanien (21 st.), Storbritannien (14), Frankrike (13), USA (11), Tyskland (7) och Sverige (6). Resterande deltagare i mindre antal kom från Australien, Belgien, Hong Kong, Italien, Norge, Portugal, Schweiz, Tjeckien, Ungern och Österrike.

Ada-Europe passade på att hålla sitt årsmöte under konferensen. Erhard Plöderer, föreningens ordförande sedan tre år tillbaka, ledde mötet med fast hand som vanligt. Även om föreningens ekonomi är stabil, beslutades om en höjning av årsavgiften till EUR 25 för indirekta medlemmar (via nationella organisationer) och EUR 35 för direktanslutna medlemmar. Sedan förra årsmötet har Ada UK upplösts som nationell organisation, och endast ett fåtal av de forna medlemmarna har direktanslutits. Kanske något att tänka på för oss i Ada-i-Sverige då medlemsantalet sjunker. Totalt har Ada-Europe idag 245 medlemmar.

Styrelsen beviljades ansvarsfrihet, och de styrelsemedlemmar som var på tur att avgå omvaldes enhälligt. (Se f.ö. hemsidan http://www.ada-europe.org/)

Kringarrangemangen var som väntat av högsta rang. Konferensen hälsades välkommen av staden Palmas borgmästare på 1300-talsslottet

Bellver, bara några hundra meters klättring uppför berget från konferenshotellet (varifrån en bekväm transport var ordnad). En busstur i Palmas omgivningar ledde oss till Valldemossa (där Chopin bodde åren 1838-39) och upp till bergen längre norrut där vi fick se goda exempel på Mallorcas fantastiska scenerier.

Som vanligt höll föredragen genomgående en mycket hög standard. Ofta med en lite utmanande titel. Bland de presentationer som stack ut speciellt vill jag notera:

- Tucker Taft (invited): Static Analysis Error Detection, där han konstaterade att industrin (i USA) lägger ut 1 GUSD på testverktyg, men 60 gånger mer på buggrättning!
- Ricky Sward: Extracting Ada95 Objects from Legacy Ada Programs, där han rapporterade hur man "översätter" strukturerad Ada till objekt-orienterad.
- Martin Gogolla (invited): Benefits and Problems of Formal Methods, där han återanvände Bowen&Hinchey's Ten Commandments of Formal Methods:

  1. Thou shalt choose an appropriate notation,

     - - -

  9. Thou shalt test, test and test again

  10. Thou shalt reuse!

- Alan Burns: Supporting Deadlines and EDF Scheduling in Ada, där förkortningen ska tolkas Early Deadlines First.
- Pascal Leroy (invited): An Invitation to Ada

Ordf Erhard Plöderer öppnar Ada-Europes årsmöte

2005, med en grundlig genomgång av det vi kan förvänta oss av den nya standarden.

- Peter Amey (Praxis): High Integrity Ada in a UML and C World, där han visade användbarheten och flexibiliteten i språket SPARK.
- Adrian Hilton: High-Integrity Interfacing to Programmable Logic with Ada, som belönades med pris för konferensens bästa presentation.
- Steve Vinoski (invited): Can Middleware Be Reliable? där han definierar middleware som kittet som binder samman olika programvaror.
- Andy Lapping, I-Logix, som i mitt tycke hade konferensens mest livfulla presentation, men där han samtidigt visade den senaste utvecklingen av verktyget Rhapsody.

Utmärkelsen för bästa dokumentation delades mellan två papper: Supporting Deadlines and EDF Scheduling in Ada (Alan Burns, Andy Wellings & Tucker Taft) och High-Integrity Ada in a UML and C World (Peter Amey & Neil White). Alla dokumentationerna finns förstås samlade i LNCS 3036 från Springer-Verlag, ISBN 3-540-22011-9.

Även i år gjorde Currie Colket reklam för nästa SIGAda konferens som hålls i Atlanta, USA, den 14-18 november i år. Se http://www.acm.org/sigada/ Undertecknad skickar på anmodan mer än gärna ut en inbjudan i pappersform.

De 8 tidigare Ada-Europe-konferenserna hölls i Montreux(CH) 1996, London(UK) 1997, Uppsala(SE) 1998, Santander(ES) 1999, Potsdam(DE) 2000, Leuven(BE) 2001, Wien(AU) 2002 och Toulouse(FR) 2003.

Nästa konferens kommer att hållas i York(UK) den 20-24 juni 2005, och 2006 års mötesplats blir Genève(CH) – vi ses väl?!

Rei Stråhle, SaabTech
rei.strahle@saabtech.se
tel. 073 437 7124

Förre ordf John Barnes håller högtidstalet vid banketten

A software architecture as a combination of patterns

Kent Petersson and Tobias Persson, Ericsson Microwave Systems, Sweden

Bo Sanden, Colorado Technical University, USA

Abstract

Ericsson Microwave Systems (EMW) in Sweden was confronted with the problem of constructing a radar system that could withstand the replacement of hardware and operating system software and be adaptable to different customers' functional requirements. This was accomplished by means of a software architecture with highly independent and flexible components that is a combination of four design patterns: Layers, Pipes and Filters, Observer and Model-View-Controller.

Artikeln har tidigare varit införd i Crosstalk October 2003.

1. Introduction

Design patterns have been introduced by the computer science community as a means to capture and document solutions to common software problems [1, 2]. An important aspect is the combination of patterns. The authors of "Design Patterns" [1] discuss how patterns "dovetail and intertwine" in good software. Nevertheless, combining patterns into a good overall architecture has received less attention than the individual patterns. In the literature, each pattern is often presented separately along with a discussion of how it fits in a resulting class diagram. This assumes that the class diagram exists when you present your patterns.

In this paper we present not only how different patterns are used in a software architecture but also how we evolved the architecture by successively integrating more patterns to deal with particular design problems. We present the individual patterns, their interaction and the questions that arise when they are combined.

The example is a software subsystem of a new generation of surveillance radar constructed by Ericsson Microwave Systems (EMW) in Mölndal, Sweden. This subsystem contains the modules that handle tracking, communication with external devices, threat evaluation, etc. A major problem with the software of an earlier radar system was that the modules were too interdependent. If you wanted to use only one part for a new product, you often had to include all modules even though the functionality was not needed.

When a new architecture was constructed, one premise was to maximize reuse. The main reason was the business situation for the surveillance radar system. The trend was that each customer bought a small series but still required considerable tailoring to their needs. To encourage reuse, the different components had to be made more independent and flexible. A function had to be coupled to as few other components as possible, and direct dependencies had to be eliminated or minimized.

An important guideline in the architectural design was the requirement for modifiability. Especially the following sources of modifications were considered:

1. Replacement of hardware and operating system
2. Different customers' functional requirements

In the following we first discuss how the system was made adaptable to new hardware and software and then how it was designed to accommodate changes in customer requirements in general and requirements for presentation and control in particular.

2. Adaptability to new hardware and operating system

To facilitate future changes of hardware and operating system, a layered architecture was introduced. The system was structured in terms of components, which were placed in three layers with application oriented components at the top and hardware and operating system dependent components at the bottom. This is a time-honored architectural style with the Open Systems Interconnection (OSI) protocol stack as a classic example [3].

2.1 The Layers architectural pattern

The *Layers* pattern is described in [2]. In it, the system is structured as a number of layers representing different levels of abstraction. We introduced the following three layers, from top to bottom:

The application layer containing the functionality required of the system.

The support layer containing parts shared by the applications. An important role for this layer is to act as a data store for the applications. For maximum independence between applications, their input and output data is stored in the support layer. That way, the applications need not be directly aware of each other but instead depend on the database components in the support layer.

The core layer containing components depending on the operating system or the hardware.

A principle of the *Layers* pattern is that the components in each layer only know of and use components in lower layers. In our case, this means that the components in the application layer only know and use components in the support layer (and possibly the core layer). This principle automatically makes the components in the application layer independent, but sometimes forces you to introduce additional layers, potentially complicating the solution. For simplicity, it is sometimes reasonable to compromise and let some components in a layer know of each other. Still, a component should never need to know of a component in a higher layer.

2.2 Architectural components

The layered architecture is shown in Fig. 1. The following are some of the components in the Core layer:

System time (*Time* in Fig. 1) This component allows the system to run at user defined time in a simulation mode. With multiple instances of this component, real time and simulated time can be used concurrently.

Time synchronization (*TSync* in Fig. 1) With different parts of the system running on different hardware nodes, synchronizing the system time on each node is critical.

Communication protocols (*Com* in Fig. 1) This component includes components dealing with low level communication protocols tailored for specific applications.

Built-in test (BiT in Fig. 1) These modules handle the testing of the hardware.

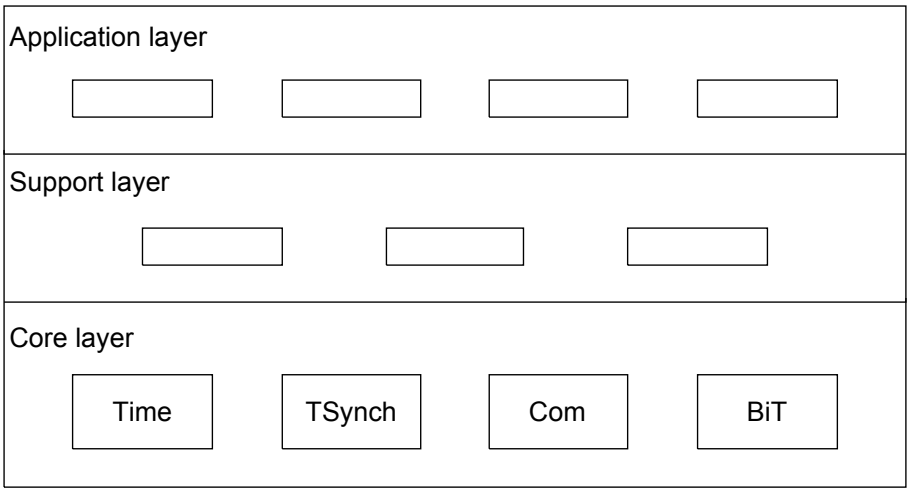

Fig. 1. Layered architecture with some of the Core layer components

3. Adaptation to customer requirements

Customers have differing functional requirements for at least two reasons. First, different customers need more or less functionality depending on how they intend to use the radar system in their overall system structure. Some want a basic system while others want a deluxe version.

Second, customers must integrate the system they buy into a larger system, whose interfaces vary considerably. This has led to a lot of reconstruction of the radar system over the years. The interface to the human operators is particularly variable.

To solve the problem with differing customer requirements, we designed an architecture for a complete system with full functionality. It defines how the entire system would work even if EMW would only realize parts of it. This architecture is broken into subsystems that are sufficiently independent to allow for all reasonable variations. Designing a system that includes all possible variants is not a design pattern but a widely applicable design principle.

3.1 The dataflow principle

The decomposition into subsystems was done according to the dataflow principle. In this common architectural style, you study how data flows through the system and divide it into subsystems at suitable points in the dataflow. For example, a radar can be divided into transmitter, antenna, receiver, signal processing, data processing and presentation, reflecting the path of a radar pulse through the system. The radar data processing has traditionally been structured along the same lines. One common structure is: target tracking, data fusion, threat evaluation and engagement planning. Most of those involved know and understand this structure.

3.1.1 The design patterns *Pipes* and *Filters*

The pattern that supports dataflow is *Pipes and Filters* [2]. It is intended to solve the problem of structuring a big system that transforms a stream of data. Designing such a system as a single component is unwise and makes the system inflexible and vulnerable to future changes. For this reason, the system is divided according to how data flows as shown in Fig. 2. We primarily use this pattern to structure the application layer but it also has implications for the support layer.

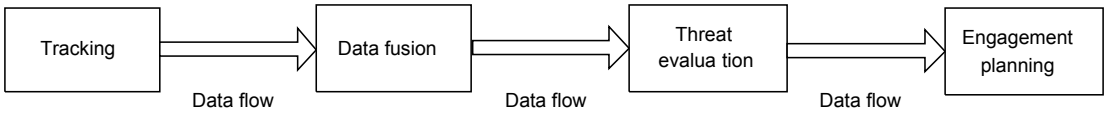

Fig 2. The Pipes and Filters pattern. Arrows indicate the data flow between components.

3.2 Composition of *Layers* and *Pipes and Filters*

The problem is to fit both the *Layers* pattern and the *Pipes and Filters* pattern into the architecture. *Layers* structures the system vertically while *Pipes and Filters* structures it horizontally. We had to find an architecture that retains the desirable features of each pattern. Using *Pipes and Filters* directly on the application would make those application components that represent the filters aware of each other. This defeats one of the goals of the *Layers* pattern, namely application independence.

The solution was to put data store components that work as buffers between the different applications – that is, the pipes in the *Pipes and Filters* pattern -- in the support layer. Fig. 3 shows the architecture with the *Layers* and *Pipes and Filters* patterns combined. This solution reduces the coupling between applications radically. By also making the components independent in terms of synchronization we minimize the coupling between the components that produce and consume data. A real-time

system sometimes produces data faster than it can consume it, and a direct coupling between producer and consumer may then lead to the use of stale data. It is often better to discard the old data and continue with the relevant information. With a separate data store component we can isolate the problem and store valid data only without involving the producer or consumer. The producer can produce data at one rate, and the consumer can consume it at a different rate. The data store components ensure that the most current data is used.

application that is dependent on a certain control would have to query the data store component for its status very often. Most of the time, the control would be unchanged. This is inefficient, but on the other hand, allowing the data storage component or some other application to send the information directly to another component in the application layer would create a strong dependence between components.

The solution is to introduce an event handling mechanism. Applications subscribe to an event

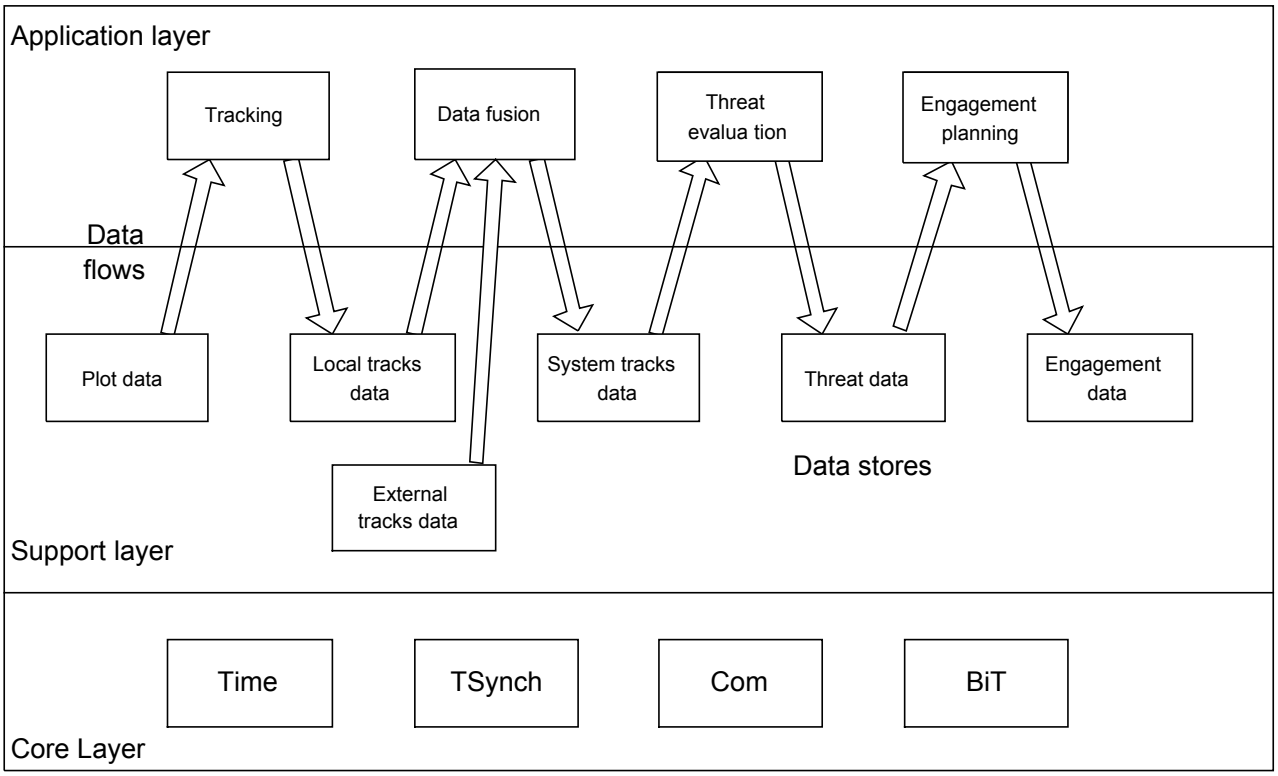

Fig 3. Combining the Layers pattern with Pipes and Filters. Data store components have been added at the Support layer.

3.3 Event handling

The simple model with the components in the application layer decoupled by means of data store components in the support layer works for data that is produced and consumed continuously. Other data, which may be associated with different events or operator controls, does not fit in this model. This is because changes occur rarely, but the application must react very quickly to them. In a dataflow solution, an

by calling an event handler. When another application generates an event, which may carry with it other (changed) information, the subscribers are informed of the occurrence and can retrieve the additional information from the data storage components.

This kind of event handling reduces the coupling between the event generating component and the subscribers; they need not be aware of each other.

The approach is quite resilient to system changes. The event handling mechanism works even if a certain subscriber is absent in a given variant of the system. Not even the event generator has to be present in all system variants, which will then lack certain functionality. A drawback is that event handling is an unstructured and dynamic way of information exchange comparable to exception handling and should be used restrictively.

3.3.1 The design pattern *Observer*

A design pattern that captures the idea behind the event handling described above is called *Observer* [1]. It is also referred to as *Publish/Subscribe* [2] and used to synchronize the state of co-operating components. The component that detects the state change publishes a message that is then forwarded to any number of subscribers.

The pattern solves the problem where many different components must be informed of a relatively rare event in a flexible way. The salient feature of *Observer* is the reduced coupling between the publisher and the subscribers.

4. Presentation and control

One area that is particularly susceptible to rework due to different customer requirements is presentation and control. It has proven difficult to predict the changes because the requirements for presentation and control tend to be unique to each customer. The structuring of the user interface is often fundamental to the architecture of any application where it is a part. There are essentially two possibilities:

1. Separating presentation and functionality with the rationale that the presentation represents one cohesive unit and functionality another. That is, certain functionality is considered more closely coupled to other functionality than to its own presentation. The *Model-View-Controller* (*MVC*) pattern captures this approach [2].

2. Tying functionality more closely to its presentation than to other functionality. A pattern that captures this view is *Presentation-Abstraction-Control* (*PAC*) [2].

In our application, the concrete presentation and control are collected in one component, Operator Presentation and Control (OPC). This component is the entire program's interface to the human user. Changes concerning the concrete presentation and control are localized in one component according to *MVC* [2].

4.1 The design pattern *Model-View-Controller (MVC)*

The *MVC* pattern divides the world into a model, a presentation part and a control part. For compatibility with our own earlier designs, we used a variant, also described in [2], where the presentation and control parts are combined, while the model remains separate and consists of all the data store components in the support layer.

The problem that *MVC* is intended to solve is that user interfaces are especially vulnerable to change requests, leading to many program modifications. A change to the presentation of one part of the system often forces a change to another part's presentation. It is hard to build a presentation with the required flexibility. Instead one can accept the situation and just structure the system so that all parts that handle the presentation are separated from the model.

4.2 Combining *MVC* with the earlier patterns

In the final architecture, *MVC* must obviously be combined with the earlier patterns. Fig. 4 shows how it fits. The control/view part consists of two components, Operator Presentation and Control for the actual presentation and Communication/Distribution for the distribution of data to external systems. The *MVC* pattern supports the idea that external communication is just another form of presentation. This means that external communication, which also tends to vary drastically between projects, can be handled in the same way as presentation. Fig. 4 shows how the components for presentation and external distribution are incorporated in the architecture. The data that is presented and distributed is in the data support layer, which is in accordance with *MVC*.

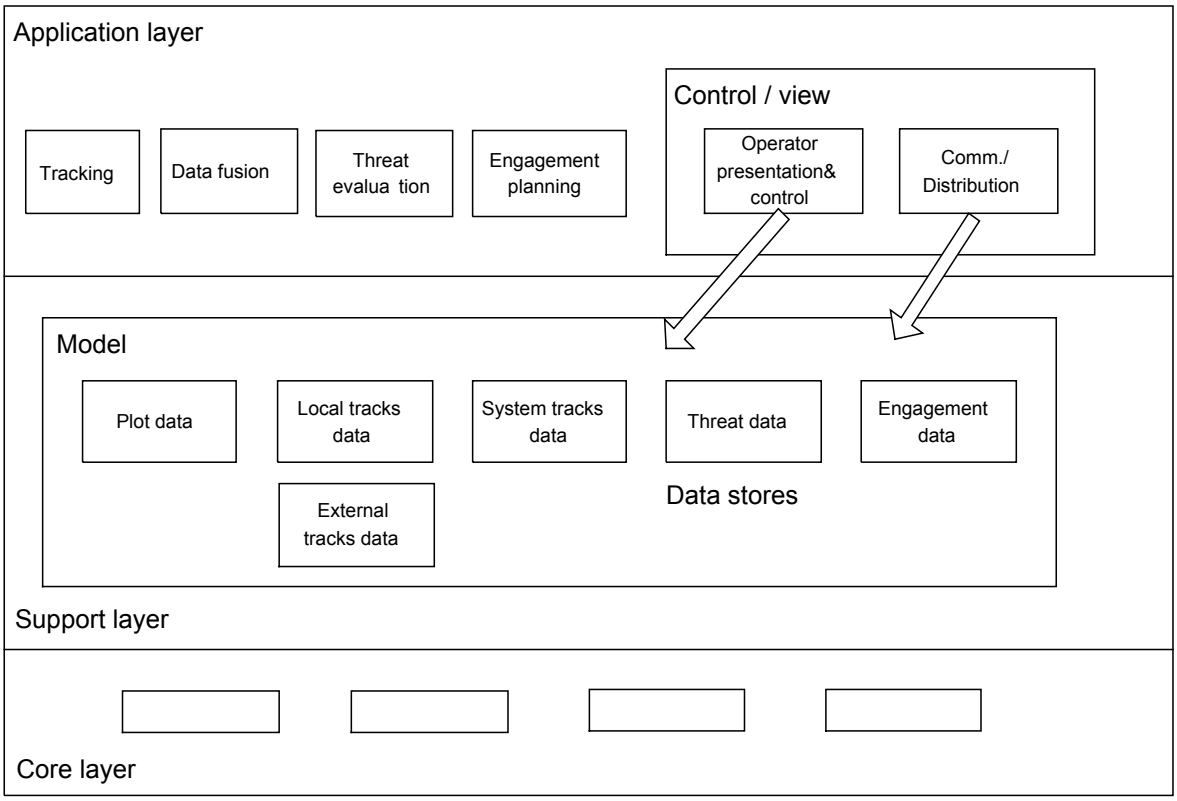

Fig 4. The architecture after applying the MVC pattern

5. Conclusion

The resulting architecture shows that the four patterns, *Layers*, *Pipes and Filters*, *Observer* and *Model-View-Controller* can be combined quite elegantly in the radar software. We have also shown that it is possible and beneficial to use design patterns in a large and very complex application. The same overall architecture has been used in seven different projects with quite different functionality in the human-machine interface, external communication, and command and control parts. In five of these projects, the design and implementation of the software is now completed. The flexible and independent structure of the architecture was very useful when combining components to meet the customers' different needs and resulted in reduced cost as well as faster delivery of the projects.

We believe that you can expect to continuously improve and refactor the software architecture during its entire lifetime. For example, the OPC needs to send commands and controls to the application components in a more efficient way. Another example is the inner structure of the OPC component. Finally, it may be desirable to split the view and the controller of the MVC pattern into separate components.

In this paper we have only described a logical view of the components and ignored the mapping to physical processes and machines. Further work has been initiated to see how components can be structured and delivered as distributed applications that can execute on different nodes in a network.

Acknowledgment. The patterns work was partially supported by the Swedish Defence Materiel Administration (FMV) through the FOTA project.

References

[1] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides: *Design Patterns -Elements of Reusable Object-Oriented Softwar*e, Addison Wesley, 1995.

[2] Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal, *Pattern – Oriented Software Architecture. A System of Patterns,* John Wiley, 1996.

[3] Day, J. D. and H. Zimmermann: The OSI Reference Model. *Proceedings of the IEEE*, vol 71,

1334-1340, Dec 1983.

Verkligt arbete, nödvändig friktion och kaos

I Carlsson har läst en tankeväckande artikel i av ***Phillip G. Armour*** *i Communications of the ACM, June 2004/vol. 47. No* 6.

I artikeln diskuterar Armour de utmaningar man står inför när man skall skatta programvarustorlek i termer av erforderlig arbetsinsats för att utveckla den.

Han menar att när man försöker mäta storleken på ett programvarujobb så är det egentligen kunskap man försöker mäta och det går inte att göra empiriskt.

Vad vi inte vet att vi inte vet

Ännu svårare blir det genom att omfattning och tidplan för ett projekt inte styrs av det vi vet, eller av vad vi vet att vi inte vet, utan av det vi inte vet att vi inte vet. Att klara ut detta senare tenderar enligt Armour dessutom att vara den mest arbetsintensiva delen i ett utvecklingsprojekt, och därför den viktigaste faktorn i skattningen av jobbomfattningen.

Att ange programvarustorlek som uppskattad arbetsinsats i t ex personmånader i stället för som t ex bedömt antal rader programkod, är annars troligen mer meningsfullt för de flesta intressenterna i ett programvaruprojekt. Men det finns alltså problem i att försöka göra detta.

Armour pekar på ett par sådana problem som han stött på. Det ena var att befintliga verktyg för programvaruskattning inte tar arbetsomfattning som *indata*, utan i stället i allmänhet levererar den som *utdata*. Ett annat, knepigare problem är att arbetsinsats inte är invariant, d v s om man t ex försöker att snabba upp projektet så går det åt mycket mer insatser än med en långsammare tidplan.

Ett populärt metrikverktyg använder relationen t^{-4} för sambandet mellan utvecklingsinsats och utvecklingstid vilket indikerar hur dyrbart det kan bli att leverera ett system före ”sin tid”.

Verkligt arbete

Armour förklarar att orsaken till att samma organisation kan leverera ett system med samma funktionalitet med mycket lägre arbetsinsats, är företeelser som friktion och kaos. Allt arbete som utförs i projekt är nämligen inte produktivt. Eftersom det sällan går att göra rätt första gången omsätts inte allt arbete i projektet direkt till exekverbar kod. Det arbete som verkligen ger sådan kod kallar Armour för ”verkligt arbete”. Det är en funktion av storlek, komplexitet och funktionalitet i det slutliga systemet och varierar inte med den tid det tar att utveckla systemet samt representeras tämligen väl av t ex antal kodrader.

Nödvändig friktion

Övriga arbetsinsatser som måste utföras om man inte vet hur man konstruerar rätt första gången, t ex att hitta information, göra utredningar, experiment, prototyping och verifiering kallar Armour för ”nödvändig friktion”. Det är det arbete som behövs för att upptäcka den för utvecklarna okända kunskapen som behövs för att bygga systemet. Hur mycket sådant arbete man måste lägga ner beror bl a på företagets utvecklingsmetod och processer. Denna friktion är endast delvis relaterad till systemets omfattning, men kan bero av hur lång tid det tar att utveckla systemet; den ökar om man försöker göra jobbet snabbare och omvändningen men den kan inte elimineras.

Variabelt kaos

Den tredje faktorn som Armour identifierat som bidragsgivare till arbetsomfattningen kallar han ”variabelt kaos” (egentligen ”optional chaos”). Det orsakas t ex av stress som uppstår om man försöker accelerera ett projekt och medarbetarna måste ta snabba och ibland overifierade beslut eller när man ökar antal deltagare i projektet med åtföljande ökning av kommunikationsbehov (och felkommunikation). Armour säger att han sett projekt med gott om personal, men ont om tid

där folk arbetet hårt men åstadkommit mycket litet per capita, därför att man varit frenetiskt sysselsatt med att reagera på ändringar som andra medarbetare gjort. Den som skattar ett systems storlek efter utvecklingsinsatsen föreställer sig antagligen något för organisationen normalt arbetsmiljöscenario med en ”normal” omfattning på det variabla kaoset. T ex för ett marknadsdrivet företag som alltid försöker minimera leveranstiden till priset av betydande kostnader.

Vilket är nyttigast

Det är inte givet att ”verkligt arbete” alltid är den ur företagets synpunkt mest värdefulla sysselsättningen. Det kan vara den ”nödvändiga friktionen” för att den tar fram nya kunskaper som kan användas för att ta fram nya banbrytande produkter.

”Variabelt kaos” är huvudsakligen onyttigt brus, men kan ibland vara det som vi lägger ner mest tid på.

Slutsatsen är i alla fall att inte allt arbete som vi lägger ner på ett projekt är nyttigt och nödvändigt.

Läsarna rekommenderas att läsa artikeln i sin helhet inkl. förklarande diagram i juni-nr av Communications of the ACM.

Svensk forskning i programvaruteknik i internationellt samarbete

Claes Wohlin, professor vid Blekinge Tekniska Högskola, deltar de närmaste två åren i ett samarbete inom **International Process Research Consortium** med syftet att utröna vad som kan tänkas komma härnäst inom forskningen om programvaruprocesser. Det är Software Engineering Institute vid Carnegie Mellon University i Pittsburgh som håller samman programmet. Det har som mål att formulera en forskningsagenda, en ”vägkarta” för vad som kan väntas bli nästa heta områden inom processforskningen, så att forskningsvärlden och ”early adopters” kan förbereda sig för nästa generation utmaningar.
Wohlin har sällskap i forskningsgruppen av ledande forskare inom området från ett tiotal länder, inkl t ex Barry Boehm och Victor Basili från USA.

Verksamheten sponsras förutom av SEI av ett antal organisationer/företag bland vilka kan nämnas SAIC och Tatas amerikanska dotterbolag. Tata är Indiens ledande IT-organisation och Asiens största oberoende globala programvaru- och tjänsteföretag. Detta borde bädda för intressant korsbefruktning. Totalt deltog sex företagssponsorer på första mötet och ytterligare några kan tillkomma.

IPRC anordnar sex stycken workshops på olika platser i världen under de två åren, där man skall diskutera och komma fram till uppfattningar om processutvecklingens framtida utveckling. Där kan förutom forskarna även representanter för sponsorerna delta. Den första ägde rum alldeles nyligen i USA.

SEI är välkänt i svenska programvarukretsar, inte minst inom försvarsområdet, eftersom deras mandat från början gällde just försvarssystem. Återanvändning (ex. SEIs studier av BEAB/Celsiustechs fartygssystemfamilj) och processutveckling (CMMs olika varianter) torde höra till de mest kända hos oss. Claes Wohlins deltagande i IPRC-projektet kommer förhoppningsvis att ge nya tillfällen till kontakter med SEIs och andra deltagares insatser inom programvaruteknik.
Projektets hemsida finns på http://www.sei.cmu.edu/iprc

SIGAda 2004 – värd en tripp över Atlanten

Årets SIGAda äger rum den 11-14 nov i Atlanta, Ga. En hel del intressanta ämnen som kan motivera en resa över Atlanten finns på det preliminära programmet, t ex följande: (abstract ur konferensens hemsida www.acm.org/sigada/conf/sigada2004/)

Can Ada Stand Up to the Challenges of C/C++ and JAVA?

Pam M. Thompson, Lockheed Martin Aeronautical Systems

My presentation will consist of a discussion on results from several Lockheed Martin Ada company studies that investigated the best ways to address concerns about the future of Ada as a defense aerospace supportable programming language. I will also be referencing material from published external sources.

Lockheed Martin Aeronautics has several large software systems implemented using Ada which will require maintenance conceivably for the next 10 years. Concerns include compiler viability and availability, tool vendor interest in writing Ada applications, an aging Ada programming community and software development environment upgrades that omit Ada compatibility requirements.

Emerging technologies like model-driven architecture and object oriented design replacing structured applications have the ability to change significantly the role of programming languages like Ada in the future. Several key factors related to programming language selection will be identified as well as comparisons between Ada, Java and C/C++.

Security Changes Everything

Watts Humphrey, The Software Engineering Institute, Carnegie Mellon University

With the pervasive use of the Internet, security is becoming a critical requirement for almost all systems. In this talk, Mr. Humphrey reviews the security situation and how the evolving threat of security attacks on our national infrastructure will necessarily change the way we design, develop, test, and support software systems. He closes with some comments on what this increased emphasis on security means to the software development community.

Why can't engineering good software be like building a house?

Stephen E. Cross, Ph.D., Georgia Institute of Technology, Georgia Tech Research Institute

It can! Our management, customers, and even our fellow engineers believe that the engineering of software intensive systems is different and hard. I've exploited this false perception for years. After 30 plus years of building software and supervising software projects, two things now seem obvious in hind sight. First, designing and building great software is not really that much different than designing and building a good house. Both have architectures, both leverage building codes and standards, both benefit from new approaches such as re-use, and both are fundamentally dependent on the skills of the designers and builders and the disciplined processes upon which they base their work. Second, we live in society that expects good houses, but routinely accepts bad software. In this (hopefully) humorous lunch time talk, I hope to enlist you in joining me to convince our stakeholder communities that engineering software is no different than the engineering of other artifacts. And I also hope to enlist your help in convincing the marketplace that they should demand and expect high quality software just like they demand and expect such of other engineered artifacts.

Information Systems Security Engineering: A Critical Component of the Systems Engineering Lifecycle

James F. Davis (University of Maryland)

Microsoft Solutions Framework and the Microsoft Operations Framework (tutorial) och Microsoft Vendor Presentation: Microsoft with Ada in the Embedded World

Rick Conn (Microsoft)
[Anm. Conn är en välkänd person i Ada-världen, som tydligen fått en ny intressant arbetsgivare]

Ada 2005 Panel

The ISO technical group responsible for the Ada standards is the ISO/IEC JTC1/SC22 WG9. WG9 is currently working to amend the Ada language to support the evolving needs of the user community. In 2000, the WG9 ARG started looking into possible changes for the next revision of the standard. An Amendment to the Ada 95 Standard (ISO/IEC 8652:1995) incorporating these changes is expected to be approved by ISO in late 2005. This revision is a great opportunity to further enhance Ada by providing new capabilities for embedded and high-reliability applications (e.g., the Ravenscar Profile, new scheduling algorithms, new predefined package Ada.Execution_Time); by integrating new programming practices (e.g., Safety in the Object Oriented Programming (OOP) area, Access Type improvements); by including new Interfaces (e.g., container library, incorporation of advanced arithmetic facilities, adopting a practical international character set, Unchecked C Unions), and by addressing other issues encountered by Ada 95 users. This will be a substantive and important revision to the Ada Programming Language.

The technical work for the Ada 2005 Amendment is carried out by the Ada Rapporteur Group (ARG) under the auspices of WG9. Key ARG members will participate in a 3-hour panel on the significant improvements that WG9 has approved for inclusion in the Ada 2005 Amendment. The Panel will be chaired by IBM's Pascal Leroy, Chair of the WG9 ARG. The panel will consist of a number of mini-briefings presented by members of the ARG.

The ARG members who will be participating in the panel include: John Barnes, Alan Burns, Pascal Leroy, and Tucker Taft.

Agile Requirements; möjlighet eller motsägelse

Ken Orr känd metodguru inom IS-området har i maj/juninumret 2004 av IEEE Software en intressant betraktelse kring kravhanteringens återupprättelse.

Orr noterar att såväl systemanalys som kravgenerering, som de lärdes ut och praktiserades för 10-15 år sedan har haft det svårt under senare år. Detta har speciellt gällt i samband med tillämpning av ”Agile” (lättrörliga?) metoder. I och med spridningen av UML och Modelldriven Arkitektur börjar enligt Orr branschen återvända till var den var för ca tio år sedan.

Orsaken till att kravgenerering blev så ute en tid, tror Orr var att den innebar att man måste avsätta mer tid för att analysera affärs- och tillämpningsaspekter, vilket ansågs ”ta tid” från det riktiga arbetet - kodningen. En del företrädare för Agile Development ansåg att det inte behövdes någon kravdokumentation i den snabbt föränderliga miljö man verkade i utan att man i stället skulle ge användarna det de sa att de ville ha så fort som möjligt.

Orr säger att detta inte är något skäl för att inte ”do requirements”, utan endast

forts sidan 17

STC har blivit SSTC

2005 Systems & Software Technology Conference:
Capabilities: Building, Protecting, Deploying
A Forum for Systems & Software Professionals
18-21 April 2005, Salt Lake City, Utah

Den säkerligen största programvaruteknikkonfe rensen och utställningen på det militärtekniska området har länge varit Software Technology Conference (STC), som anordnats varje år sedan ett femtontal år kring slutet av april i Salt Lake City med ett årligt deltagande på senare tid av 2500 personer.

Konferensen hade i många år ett rätt livligt deltagande av SESAM-iter och andra från både FMV och industrin. 1995 fanns det t o m en stor svensk utställningsbooth som FMV sponsrade på konferensen. På senare år verkar det som om det svenska deltagandet varit blygsamt eller uteblivit helt. Det kanske är en följd av den allmänna ekonomiska kärvheten inom försvarssektorn, men det är inte säkert att det är så bra att spara in på denna typ av kunskapsinhämtning ens i knapphetstider.
Detta särskilt inte med det mycket tydliga fokus på internationella operationer, internationell för svarsmaterielsamverkan etc. som kännetecknar den nuvarande säkerhetspolitiska inriktningen. Där finns mycket att hämta av intresse och betydelse för svenska myndigheter och företag. Inte minst slår det amerikanska försvarets nätverksinriktning numera igenom tydligt i konferensföredragen.

Från och med 2004 års konferens har organisatörerna lagt till Systems i namnet så att den numera heter Systems & Software Technology Conference (SSTC), vilket innebär att den allt bredare täckning mot systemfrågor som konferensen haft under senare år nu också återspeglas i namnet. Denna breddning har också gjort konferensen än mer intressant för svenska förhållanden. Även SESAM har ju på senare år breddat sin täckning åt systemhållet.

Konferensen anordnas av Utah State University i samverkan med USAFs Technology Support Center STSC och Ogden Air Logistics Center i Ogden utanför Salt Lake City. Men det viktigaste är att konferensen numera sponsras av alla försvarsgrenar och DISA: United States Army, United States Marine Corps, United States Navy, Department of the Navy, United States AirForce och Defense Information Systems Agency. Därför kan man räkna med att vad som framförs vid konferensen rätt väl avspeglar inriktning och vad som sker inom hela amerikanska försvaret.

Nästa års konferens har temat ”Capabilities: Building, Protecting, Deploying” och man anger att de ”topics” man avser ta upp är:

SSTC 2005: Acquisition, Agile, Architecture, Future Technologies, Information Technologies, Management, Net/Web, People, Policy, Process Solutions, Research, Security, Software Solutions, Systems Engineering Solutions, Wireless

Man välkomnar också ytterligare förslag till ”topics”.

Tidpunkten för inlämnande av Abstracts till föredrag går ut den 10 september, så det är brådtom om någon vill anmäla sig som talarkandidat. (Det har funnits svenska föredragshållare vid tidigare konferenser!)

I första hand kan man se deltagande i SSTC som ett tillfälle att bli up-to-date med vad som pågår i branschen (inte bara i USA), träffa kollegor från andra länder samt inspireras i sin egen verksamhet.

Det är ett tag fram till april 2005, men inte så långt ur planeringssynpunkt. Dessutom bör man vara ute i god tid med hotellbokning om man vill bo i närheten av det stora konferens- och utställningscentret. (Det går att göra redan nu på konferensens hemsida.)

Programmet kan väntas bli publicerat i november. Kolla in hemsidan med jämna mellanrum! http://www.stc-online.org

Red.

forts från sidan 15

undanflykter. Varje system av någon betydelse behöver bra kravbeskrivningar; utan sådana ökar projektrisken dramatiskt. Ju bättre kravbeskrivningen är, dess bättre förstår folk vad som skall utvecklas eller anskaffas. Men det är inte längden på kravdokumentet som är avgörande.

Orr argumenterar sedan för att koncentrera kraven till att huvudsakligen handla om utdata och begränsningar (outputs and constraints), vilka också har den fördelen att de är testbara. Vidare hävdar han, liksom en tidigare skribent i ämnet, att duktiga kravanalytiker inte skall vara rädda för att ”uppfinna krav”. De enskilda användarna vet nämligen ofta inte vad som är möjligt, vad konkurrenterna gör eller var ”state of the art” ligger. Det behövs att någon tillför dessa aspekter.

För dem som sysslar med ”Agile Development” och kanske har en stark motvilja mot kravdokument, föreslår Orr prototyping som en lämplig metod. Orr ser alltså ingen nödvändig motsägelse mellan lättrörlighet och kravananlys etc, speciellt inte med alla de verktyg som är tillgängliga numera.

Ämnet är så intressant att jag föreslår att Ni studerar hela artikeln, som alltså finns i IEEE Software May/June 2004.

I Carlsson

Aktuella konferenser i Sverige

2004-09-08--10	2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Stockholm
2004-09-16--19	VikingPLoP Patterns konferens i Uppsala 3rd Nordic Conference of Pattern Language of Programs, VikingPLoP http://www.plop.dk/vikingplop/

SESAM Kalender

2004-08-26	VU-möte, kl 1000
2004-09-21	Människa-System-interaktion i säkerhetskritiska system
2004-09-30	VU-möte, kl 1000
2004-10-20	SESAM höstseminarie
2004-10-21	SESAM Rådsmöte, kl 0830
	Redovisningar från SESAM arbetsgrupper

SESAM HÖST-SEMINARIUM, 20 OKTOBER 2004, Hotell Birger Jarl

INFORMATION - FRÅN BÖRDA TILL RESURS?

Informationshantering i framtida försvaret

0845-0945	REGISTRERING och KAFFE	
0945-0955	Om SESAM, inledning till seminariet	*Claes Wadsten, Ordf SESAM*
0955-1015	IRM i Försvaret	*Jarl S Magnusson, FMV*

TEORI & FORSKNING

1015-1045	Informationsteori (-filosofi), informationens natur, dess relation till tänkande och kunskap etc.	*Johan Bendz, FMV*
1045-1115	IRM och kunskapshantering, forskn.proj på KTH/CID	*Kristina Groth, KTH*
1115-1130	Bensträckare	
1130-1200	Hur kan webben användas som kunskapskälla?	*Mats B Andersson, KTH*
1200-1230	Spårning av data Forensisk analys av Polisen och vid it-revision Spårbarhet i datorer och annan elektronisk utrustning	*Svante Nygren, teknisk analytiker och journalist*
1230-1330	LUNCH	

STRATEGISK ARBETE & TILLÄMPNINGAR

1330-1400	CIO Funktion på HKV	*Genmj, C Lidström, FM HKV*
1400-1430	Multisensorintegration inom JAS39 Gripen:	
	• Inledande presentation av uppdrag Multisensorintegration inom JAS 39 Gripen	*Patrick Sakurai, FMV*
	• Krysspejlfunktion och ID-fusion inom MSI39. *Aerosystems*	*Niklas Ferm, Saab*
	• Elektrooptisk modell integrerad i Saab Aerosystems Gripen Systemsimulator (PMSIM).	*Stefan Ringberg, Saab Aerosystems*
1430-1500	Unstructured Information Management	*Michael Thorson, Infosphere*
1500-1530	Kaffe	
1530-1600	Information och kunskap i en distribuerad värld.	*Olle Olsson, SICS*
1600-1700	US DOD Net-Centric Data Strategy	*Mike Krieger, Director, Information Management DASD(DCIO), OASD-NII or Dave Chesebrough, President AFEI*
1730-	BUFFÈ	

Med reservation för ändringar

Verksamhet inom Ig Programvarusäkerhet 02-10-24 – 04-06-09

En intressegrupp i programvarusäkerhet inrättades under SESAM:s höstseminarium 2002. Olika aspekter inom området Programvarusäkerhet har studerats och presenterats under gruppmöten samt genom inbjudan till seminarier över olika teman.

Möten och seminarier

Mötesform	*Datum*	*Tema*	*Kommentar*
Möte 1	02-10-24	• OS i säkerhetskritiska tillämpningar	3 externa talare.
Möte 2 + Sem 1	03-02-04	• Teknikprov inom FoTA P12 • Systemsäkerhetsanalys i olika företag	9 talare varav 2 externa.
Seminarium 2	03-03-26 --27	• Safeware:s SpecTRM-verktyg	2 Leveson-doktorander
Möte 3	03-05-07	• SpecTRM-prov, • Design by contract, • Återanvändning o interoperabilitet	Enbart interna talare
Seminarium 3 (Leveson)	03-08-20 -- 21	• System Safety for Software-Intensive Systems	Seminarium öppet även för icke-SESAM medlemmar
Möte 4 + Sem 4	03-09-18	• Tidsstyrd programmering/nätverk	5 externa talare.
Möte 5 + Sem 5 (höstsem)	03-10-21 -- 22	• Arbetsgruppsredovisningar • Informationssäkerhet i en NBF-miljö	SESAM Gruppmöten + Öppet SESAM-seminarium.
Möte 6 + Sem 6	04-02-11	• Programvara i obemannade farkoster	6 externa talare.
Möte 7 + Sem 7	04-06-08	• Människa-System-interaktion i säkerhetskritiska situationer	Gruppmöte + 7externa. Öppet SESAM-seminarium
Möte 8 + Sem 8	04-09-21	Säkerhetskritisk MSI (forts på föreg sem)	Gruppmöte + externa Öppet SESAM-seminarium

SESAM:s hemsida

Under rubriken ”Arbetsgrupper” på SESAM:s hemsida (http://sesam.tranet.fmv.se) har en struktur för IG Programvarusäkerhet lagts in, där dokumentation i form av inledande förutsättningar (upprop, programförklaring) samt mötesagenda med presentationsmaterial och mötesnotiser finns tillgängliga. Material från det projekt som varit mönster vid bildande av IG:et[1] har utgjort grunden för informationen under rubrikerna ”Utbildning och kurser”, ”Intresseorganisationer”, ”Publikationer”, ”Verktyg” samt ”Projekt”.

Tekniköverföring/-utbyte

Detta har realiserats främst genom diskussioner och info-utbyte under möten och seminarier. Teknikprov inom medlemsorganisationerna används för erfarenhetsutbyte både bland dem som utför provningarna och de som agerar som granskare/bollplank. Gemensamma frågeställningar formuleras och fördelas mellan olika provare. Angreppssätt och realisering av tidiga prov kommer p s s senare prov tillgodo, dubbelarbete undviks och ett snabbare genomförande möjliggörs. Såsom medlemmar i SESAM, en icke-vinstdrivande och utbildande organisation, har IG-deltagare även fått möjlighet att utan kostnad prova en produktionsversion av verktyget SpecTRM. Två företag valde att prova verktyget genom examensarbetare, något som stimulerat utbytet mellan deltagande industrier och högskolor.

Ett stående tema för IG-möten under 04 är MMI:er i säkerhetskritiska situationer. En kartläggning av erfarenheterna inom medlemsföretagen kommer att genomföras och ett antal personer från andra företag / organisationer kommer att inbjudas. En sammanfattande slutredovisning planeras inför höstseminariet.

Medlemmar

F.n. ingår 24 personer (se Medlems-/Intressentlista).

(Footnotes)

[1]FoTA P12 (Överföring till industrin av programvaruteknik för säkerhetskritiska system), ett projekt, vars karaktär av specialinriktat nätverk kombinerat med kunskaps- och teknikutbyte samt teknikprov visat sig vara en framgångsrik modell för hur tekniköverföring kan befrämjas mellan industri, företag, organisationer och högskolor. P12 avslutades 2002.

Inbjudan till seminarium tisdagen den 21 september 2004
Människa-System-interaktion i säkerhetskritiska system

Program

12.00	Inledning	*Inga-Lill Bratteby-Ribbing, FMV*
12.10	Prokrustes säng och HMI-design	*Erik Hollnagel, LiU*
13.10	Människans roll i komplexa tekniska system med exempel från flyget	*Lena Mårtensson, INDEK, KTH*
14.10	*Kaffe*	
14.40	Människa-maskin-interaktion på fartygsbryggan	*Margareta Lützhöft, LiU*
15.25	Analys och utvärdering av operatörsgränssnitt	*Anna Borg, Swedpower*
16.10	Diskussion	
16.40	Avslutning	

Bakgrund

Föreningen SESAM, Försvarssektorns användargrupp för Software Engineering, har en intressegrupp i Programvarusäkerhet, vilken studerar programvaruegenskaper av betydelse för säkerhetskritiska system, dvs system med risk att kunna skada person, egendom eller miljö.

En av gruppens verksamheter är att ordna seminarier inom olika tillämpningsområden. Inriktning under detta år har varit interaktion mellan människa-maskin i säkerhetskritiska system. Två seminarier på detta tema har arrangerats, varav detta utgör det andra.

Målgrupp

Seminariet är en del av IG Programvarusäkerhets kompetensutveckling. Även övriga personer från FM, FMV, FOI, industri och högskolor är i mån av plats välkomna.

Tid och plats

Seminariet genomförs 2004-09-21 kl 12.00 – 17.00 på FMV, Banérgatan 62, Stockholm i Filmsal C.

Anmälan

Anmälan med uppgifter om namn, organisation samt personnummer lämnas till Anna Kåsjö (via epost: kasjos.kontor@bredband.net eller fax: 08-510 519 32) senast 2004-09-17. Deltagandet är kostnadsfritt.

Inga-Lill Bratteby-Ribbing

Ordf IG Programvarusäkerhet, SESAM

Strategisk specialist i Programvarusäkerhet, FMV