ANSATZ FÜR EINEN SYSTEM SECURITY ENGINEERING-PROZESS ZUR ENTWICKLUNG EINES KABINENMANAGEMENTSYSTEMS DER NÄCHSTEN GENERATION

H. Hintze, R. God,
Institut für Flugzeug-Kabinensysteme, Technische Universität Hamburg-Harburg,
Nesspriel 5, D-21129 Hamburg, Deutschland

Zusammenfassung

Mit zunehmender Funktionalität und der damit einhergehend steigenden Komplexität der Systeme in der Flugzeugkabine, welche von verschiedensten Stakeholdern, wie z.B. dem Kabinenpersonal, den Passagieren und dem Wartungspersonal genutzt werden, gewinnt die System Security immer mehr an Gewicht. Dieser Artikel beschäftigt sich mit einem Model-Based Requirements Engineering (MBRE)-Ansatz, welcher einen System Security Engineering-Prozess erstmals methodisch integriert. Ein von uns so genanntes Drei-V-Modell dient als Vorgehensmodell und repräsentiert den Systementwicklungsprozess (SEP) zusammen mit dem Safety- und dem Security-Prozess. Alle drei Prozesse werden parallel durchlaufen und interagieren jeweils wechselseitig mit dem SEP. Der entwickelte Ansatz folgt einer für das MBRE bekannten Ontologie und erweitert diese um spezifische Modellelemente zur Security-Modellierung. Es wird gezeigt, wie die heute beim dokumentenzentrierten Requirements-Based Engineering (RBE) bestehenden Herausforderungen von einem MBRE-Ansatz besser adressiert werden können und wie eine modellbasierte Formulierung von Requirements für ein Kabinenmanagementsystem der nächsten Generation gelingt.

1. EINLEITUNG

Das Kabinenmanagementsystem übernimmt beim Betrieb der Flugzeugkabine wichtige Aufgaben bei der Kommunikation, der Steuerung und Anzeige sowie bei der Parametrierung von Kabinensystemen und präsentiert sich als komplexes System, welches wiederum eine Vielzahl von Wechselwirkungen zu anderen Flugzeugsystemen aufweist. Zur Umsetzung der bei EUROCAE ED-79 bzw. SAE ARP-4754 [1] gegebenen Empfehlungen für die Zulassung hochintegrierter oder komplexer Flugzeugsysteme folgt die Systementwicklung dem so genannten V-Modell. Die Entwicklung der eigentlichen Systemfunktion wird dabei von einem Safety Assessment-Prozess [1], [2] begleitet, welcher ebenfalls einem V-Modell folgt und die Zuverlässigkeit der Funktion sicherstellen muss. Ein kombiniertes Vorgehensmodell, welches einerseits die Systemfunktion und andererseits die Zuverlässigkeit des Systems (engl.: System Safety) gleichermaßen berücksichtigt, wird in der Literatur auch als „Zwei-V-Modell“ bezeichnet [3]. Zur Erfüllung von Anforderungen zum Schutz von Flugzeugsystemen gegenüber potenziellen Angriffen (engl.: System Security), wird bei EUROCAE ED-202 [4] erstmalig ein Ansatz für einen System Security-Prozess beschrieben, welcher künftig im Systementwicklungsprozess umgesetzt werden muss. Im hier vorliegenden Artikel greifen wir diesen Ansatz auf, entwickeln ihn weiter und integrieren ihn als wichtiges „drittes V“ in den Systementwicklungsprozess. Das daraus resultierende und von uns so gennannte „Drei-V-Modell“ stellt folglich ein Vorgehensmodell zur Systementwicklung dar, welches einen System Safety Engineering- und System Security Engineering-Prozess mit berücksichtigt.

2. DAS KABINENMANAGEMENTSYSTEM IM UMFELD VON SYSTEM SECURITY UND REQUIREMENTS-BASED ENGINEERING

Im zurückliegenden 20. Jahrhundert konzentrierten sich die Arbeiten in der kommerziellen Passagierluftfahrt auf die Weiterentwicklung des Fliegens mit größerer Nutzlast, größerer Reichweite, höherer Geschwindigkeit und auf mehr Komfort für den Reisenden. Wichtige Meilensteine wurden durch die großen und schnellen Fortschritte beim Flugzeugentwurf und der Antriebstechnik sowie durch den Einsatz von Druckkabinen zum Fliegen in großen Höhen erreicht.

Der Einsatz von Elektronik in Passagierflugzeugen, erst bei der Triebwerkssteuerung in den 1950er Jahren und dann in den 1970er Jahren bei der Flugsteuerung, erreichte mit der von Airbus im Jahr 1988 ausgelieferten A320 mit voll-digitaler Fly by Wire-Technologie einen Höhepunkt: Steuerimpulse wurden nicht mehr analog mit Drahtseilen an die Ruder übertragen, sondern von Computern digital verarbeitet und dann über ein Bussystem an die elektro-mechanischen Aktuatoren der Steuerflächen übermittelt.

Heute, im 21. Jahrhundert, widmet man der Flugzeugkabine mit ihren komplexen Systemen besondere Aufmerksamkeit. Sie stellt das Aushängeschild für jede Fluggesellschaft dar. Mit ihrer Ausrüstung und Technik ist sie das zentrale Element bei einer Flugreise. Im stetigen Wettbewerb um den Kunden muss die Fluggesellschaft in der Kabine während der primären Transportdienstleistungsaufgabe auch viele andere Anforderungen und Wünsche des Fluggastes erfüllen. Dem für den Betrieb der Kabine zentralen Kabinenmanagementsystem kommt dabei eine wichtige Rolle zu. Dessen Elektronik und Datennetz haben sich über fünfundzwanzig Jahre hinweg bis heute kontinuierlich weiterentwickelt. Aus dieser Evolution

resultierende Kabinenmanagementsysteme sind heute ausgereift und in Verkehrsflugzeugen allgemein verbreitet. Jedoch führen eine stetig wachsende Vielfalt von Kabinenfunktionen, Wartungsaufgaben und Passagierdienstleistungen und eine damit einhergehende wachsende Anzahl von Prozessen dazu, dass die bestehenden Architekturen hinsichtlich Flexibilität und Skalierbarkeit neu überdacht und weiter entwickelt werden müssen.
Die für ein Kabinenmanagementsystem der nächsten Generation zu betrachtenden Architekturen [5], [6] folgen bevorzugt den Prinzipien von verteilten Systemen, in welche für unterschiedliche Nutzergruppen auch drahtlos kommunizierende Subsysteme, wie zum Beispiel Ad-hoc-Sensornetzwerke, mobile PCs und andere mobile Geräte, eingebunden werden müssen. Damit rückt das Thema der System Security, also der Angriffssicherheit des Kabinenmanagementsystems, erstmals ganz deutlich in den Vordergrund.
In der Vergangenheit wurde dieses durch so genannte Physical Security-Mechanismen behandelt, was bedeutet, dass Hardware physisch abgeschottet, Softwarefunktionen weitestgehend gekapselt und Kommunikationswege geschützt und streng reglementiert wurden. Bei verteilten Architekturen mit drahtlosen Kommunikationsschnittstellen sind Physical Security-Mechanismen in dieser Form allein nicht mehr ausreichend. Künftig müssen auch Logical Security-Mechanismen, also Vorkehrungen gegen einen unberechtigten Zugang zu Kommunikationsnetzwerken, Software, Informationen und Daten des Systems oder gegen das Ausführen von Aktivitäten innerhalb des Systems, mit berücksichtigt werden. Ein System Security Engineering-Prozess, welcher in den Systementwicklungsprozess integriert ist, die etablierten Techniken des Requirements-Based Engineering [7] berücksichtigt und bei der Entwicklung künftiger Kabinenmanagementsysteme zum Einsatz kommen kann, wird nachfolgend beschrieben.

2.1. Das Drei-V-Modell

In BILD 1 ist das von uns so genannte Drei-V-Modell dargestellt. Das erste V repräsentiert den für die Flugzeug-Systementwicklung grundlegenden Systementwicklungsprozess (engl.: Systems Engineering-Prozess, SEP), welcher bei [1] beschrieben ist. Die Äste des V repräsentieren den Verlauf der charakteristischen drei Phasen, des Systementwurfs, der Implementierung und der Systemintegration. Das zweite V begleitet den SEP und beschreibt den bei [2] erläuterten Safety Assessment-Prozess, welcher der Umsetzung von System Safety-Anforderungen, d.h. Zuverlässigkeitsanforderungen dient.
Damit bei der Entwicklung von Kabinensystemen die System Security-Anforderungen noch umfassender dargestellt und transparenter in die Entwicklung eingegliedert werden können, schlagen wir ein prozessbegleitendes drittes V vor, welches einen System Security Engineering-Prozess repräsentiert. Eine grundlegende Spezifikation für einen Security-Prozess wurde bei [4] gegeben. Das von uns vorgeschlagene, kombinierte Vorgehensmodell, bestehend aus dem SEP zur Entwicklung der Systemfunktionen, einem Safety-Prozess zur Berücksichtigung der Systemzuverlässigkeit und einem Security-Prozess zum Schutz eines Systems vor Angriffen und gegen Missbrauch, bezeichnen wir als Drei-V-Modell.
BILD 1 soll verdeutlichen, dass sich über die Anforderungen für die Systemfunktionen die System Safety- und System Security-Anforderungen ergeben und dass diese in umgekehrter Richtung auf die Systemfunktionen zurückwirken. Konkret bedeutet dies, dass bei künftigen Architekturen von Kabinenmanagementsystemen zur Erreichung eines angemessenen Security Levels (SL) die Security-Anforderungen schon in einer frühen Phase definiert und dann während des Entwicklungsprozesses kontinuierlich weiter detailliert werden müssen.

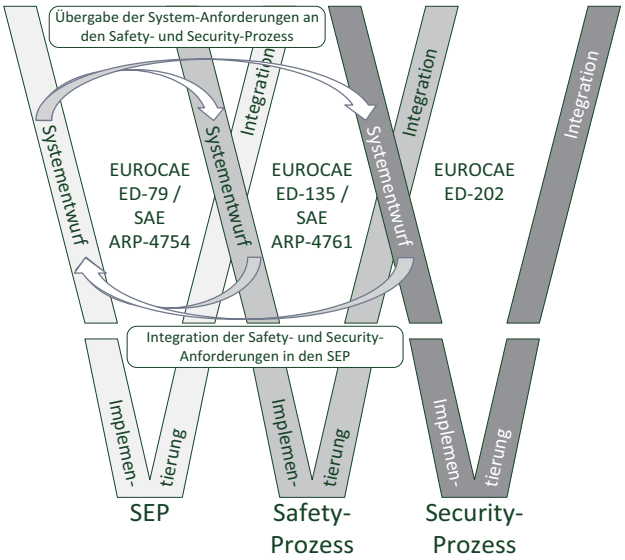

BILD 1. Das Drei-V-Modell mit den grundlegenden EUROCAE- bzw. SAE-Dokumenten für den Systems Engineering-Prozess (SEP), den Safety-Prozess und den Security-Prozess

2.2. Requirements-Based Engineering (RBE)

Neben einer verbesserten Eingliederung von System Security-Themen in den Entwicklungsprozess erfordert die stetig wachsende Funktionsvielfalt und zunehmende Komplexität moderner Kabinenmanagementsysteme auch eine erneute Betrachtung der im SEP eingesetzten Methoden.
Das so genannte Requirements-Based Engineering (RBE) ist eine heute etablierte Methode [7], [8], welche primär durch ein Anforderungsmanagement gekennzeichnet ist, dessen Handhabung durch computergestützte Werkzeuge erleichtert wird. In diesem Kapitel soll auf das RBE mit seinen wesentlichen zwei Bereichen und vier Phasen kurz eingegangen werden, bevor dann später in Kapitel 4 gezeigt wird, wie sich das RBE durch Modellierung unterstützen und zum so genannten Model-Based Requirements Engineering (MBRE) erweitern lässt.

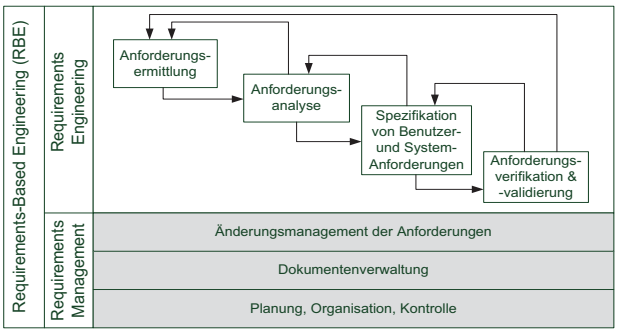

BILD 2. Übersicht zu Requirements-Based Engineering (RBE)-Aktivitäten nach [7]

BILD 2 gibt einen Überblick zu RBE-Aktivitäten mit den beiden parallel verlaufenden Bereichen des Requirements Engineering und des Anforderungsmanagements (engl.: Requirements Management). Dabei bildet der Bereich des Requirements Managements das unterstützende Fundament, welches ein Änderungsmanagement der Anforderungen, eine Dokumentenverwaltung sowie einen Planungs-, Organisations- und Kontroll-Prozess liefert.
Der Bereich des Requirements Engineering definiert die methodischen Phasen der 1) Anforderungsermittlung, 2) Anforderungsanalyse, 3) Anforderungsspezifikation und 4) Anforderungsverifikation und -validierung [7]:

1) *Anforderungsermittlung:* Während dieser Phase gilt es, die Wünsche und Forderungen aller Beteiligten einzusammeln. Dabei ist sicherzustellen, dass es während dieser Phase zu keiner Einschränkung oder Bewertung der Informationen kommt. Auch anscheinend redundante oder sich überschneidende Forderungen sind erlaubt, da sich daraus später im Prozess zusätzliche, differenziertere Forderungen ableiten können. Alle Informationen werden schriftlich festgehalten und stehen damit als Grundlage für die Anforderungsanalyse zur Verfügung.
2) *Anforderungsanalyse:* Während dieser Phase erfolgt die Auswertung aller Informationen, die bei der Anforderungsermittlung festgehalten wurden. Dabei gilt es, grundlegend Forderungen herauszuarbeiten und eine erste Bewertung bezüglich deren Erfüllbarkeit durchzuführen. Schwer zu erfüllende Forderungen dürfen nicht entfallen, sondern werden getrennt gesammelt. Im Rahmen eines Iterationsschrittes werden diese mit den Beteiligten erneut diskutiert, eventuell umformuliert, vereinfacht oder auf Beschluss gelöscht. Am Ende dieser Phase stehen die aufbereiteten Informationen als Eingabe für die nächste Phase bereit.
3) *Anforderungsspezifikation*: Während dieser Phase werden alle aufbereiteten Informationen nach festen Regeln in möglichst eineindeutige Anforderungen übersetzt und in einem Spezifikationsdokument festgehalten. Prinzipiell gibt es dabei zwei Anforderungstypen: die Benutzeranforderungen und die Systemanforderungen.
   Benutzeranforderungen beschreiben in der Regel auf einer hohen Abstraktionsebene und ohne Technikbezug mittels natürlicher Sprache eine Systemeigenschaft, welche von einem Nutzer des Systems benötigt oder gewünscht wird.
   Systemanforderungen stellen die in Technik übersetzten Benutzeranforderungen dar und dienen als technische Basis für den Systementwurf. Da die Systemanforderungen - im Gegensatz zu den Benutzeranforderungen - zur präzisen Beschreibung der Systemfunktionen bestimmt sind, können zu deren eindeutiger Spezifikation ein Modell oder Pseudocode benutzt werden.
4) *Anforderungsverifikation und -validierung*: Die Verifikation dient einer Überprüfung, ob „das System richtig“ entwickelt wird. Dabei wird geprüft, ob während der schrittweisen Detaillierung alle übergeordneten Anforderungen berücksichtigt werden. Die Validierung dient hingegen zur Überprüfung, ob „das richtige System“ entwickelt wird. Dazu wird auf allen Detaillierungsebenen eine inhaltliche Prüfung der Anforderungen durchgeführt. Ein gutes System, welches alle Erfordernisse eines Nutzers und an die Technik zu erfüllen vermag, muss also bei der Verifikation und Validierung durchweg positive Resultate liefern.

Die Entwicklung von Flugzeugsystemen folgt heute diesem methodischen Vorgehen des RBE. Dazu sei an dieser Stelle erwähnt, dass der linke Ast des V-Modells bei der Anforderungsspezifikation und Anforderungsvalidierung endet. Es folgt dann die Implementierung und, im rechten aufsteigenden Ast, die Integration des Systems. Die Anforderungsvalidierung, d.h. die Prüfung, ob das richtige System entwickelt wird, wird hauptsächlich während der Systemintegration durchgeführt.

3. DER SYSTEM SECURITY ENGINEERING-PROZESS

Ein erster Ansatz für einen System Security Engineering-Prozess bei Flugzeugsystemen wird bei EUROCAE ED-202 [4] beschrieben: Die Airworthiness Security Process Specification liefert wichtige Eckdaten für die innerhalb eines Security-Prozesses durchzuführenden Aktivitäten. Als Grundlage für diesen Prozess und die Aktivitäten soll im folgenden Kapitel 3.1. auf das heute verwendete Domänenmodell des Flugzeugs eingegangen werden, welches bei der Risikoanalyse, die in Kapitel 3.2. beschreiben wird, mit berücksichtigt werden muss. In Kapitel 3.3. wird dann erläutert, wie sich der Security-Prozess im Rahmen des etablierten Requirements-Based Engineering (RBE) gestalten würde. In Kapitel 3.4. werden dabei erwartete Herausforderungen diskutiert, welche von einer textbasierten RBE-Entwicklungsmethodologie nur unbefriedigend gelöst werden können. Schließlich wird dann in Kapitel 4. beschrieben, welches zusätzliche Potenzial eine modellbasierte MBRE-Methodologie bezüglich dieser bestehenden Herausforderungen bieten kann.

3.1. Das Domänenmodell des Flugzeugnetzwerkes

Im bekannten System Safety-Entwicklungsprozess wird zu einem frühen Zeitpunkt das so genannte Design Assurance Level (DAL) festgelegt. Dieses von der EASA bzw. FAA in den Bauvorschriften CS 25.1309 bzw. FAR Part 25.1309 [9] grundlegend verankerte DAL schafft eine Verbindung zwischen der zulässigen Wahrscheinlichkeit des Versagens einer spezifizierten Funktion und der Schwere und Tragweite der sich daraus ergebenden Folgen. Entsprechend darf der Totalverlust eines Flugzeugs nur extrem unwahrscheinlich sein, während eine geringfügige Einbuße an Reisekomfort, z.B. durch Ausfall des In-Flight Entertainments, jederzeit toleriert werden kann.

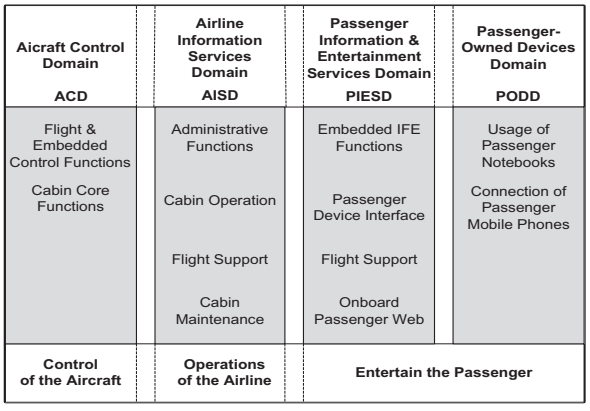

BILD 3. Domänenmodell des Flugzeugnetzwerkes mit zugeordneten Funktionen und Diensten

Ganz analog muss im System Security-Entwicklungsprozess ein Security Level (SL) festgelegt werden, was heute durch eine Kategorisierung der Kabinensystemfunktionen erfolgt. Im Detail geschieht diese Einordnung anhand des System Security-Domänenmodells des Flugzeugnetzwerkes.
Dieses Domänenmodell nach ARINC Specification 664P5 [10] einer Flugzeug-Netzwerkarchitektur setzt sich, wie in BILD 3 gezeigt, aus vier Domänen zusammen, die nach ARINC Report 811 [11] für die System Security-Domänenkategorisierung herangezogen werden. Die Aircraft Control Domain (ACD) ist dabei die besonders kritische und beherbergt alle flugrelevanten und eingebetteten Steuerungsfunktionen. In der ACD sind auch die für den Kabinenbetrieb grundlegenden Sicherheitsfunktionen mit ihren jeweiligen Unterdomänen angesiedelt. Die Airline Information Services Domain (AISD) enthält unter anderem die für den allgemeinen Kabinenbetrieb erforderlichen, jedoch nicht sicherheitskritischen Bedien- und Wartungsfunktionen. Damit beherbergt die AISD weniger kritische Dienste als die zuvor beschriebene ACD. Das niedrigste Kritikalitätsniveau haben die Passenger Information and Entertainment Services Domain (PIESD) mit der potenziell dort angeschlossenen Passenger Owned Devices Domain (PODD). Diese beiden letztgenannten Domänen unterstützen die Informations- und Unterhaltungsfunktionen für den Passagier und bieten Schnittstellen für Notebooks und Mobiltelefone an.
Es gilt jetzt zu berücksichtigen, dass ein generischer Ansatz für einen System Security Engineering-Prozess auf alle Kabinenfunktionen und Dienste anwendbar sein muss, unabhängig davon, welcher Domäne diese angehören. Es ist also eine Domänenunabhängigkeit der Methode zu gewährleisten. Auf eine an dieses existierende Domänenmodell angelehnte System Security-Analyse zur Bestimmung eines Security Level (SL) nach [4] wird im folgenden Kapitel eingegangen.

3.2. Vorgehen bei der Risikoanalyse

In diesem Kapitel wird beschrieben, wie zur Bestimmung eines Security Levels (SL) eine Risikoanalyse durchgeführt wird. Begriffe und Prinzipien der Risikoanalyse werden dazu erläutert.
Eine Risikoanalyse identifiziert nach [12] potentielle Bedrohungen (Threats), ordnet diese dann den eventuell vorhandenen Schwachstellen (Vulnerabilities) des Systems zu und bewertet abschließend deren Umfang des Einflusses auf das System sowie die Wahrscheinlichkeit des Eintretens. Ähnlich wie beim Design Assurance Level (DAL), wo eine Verbindung zwischen Versagenswahrscheinlichkeit, Schwere und Tragweite der Folgen geschaffen wird, ergibt sich hier das Risiko aus einer Verbindung der Wahrscheinlichkeit eines spezifischen Angriffs und dem Umfang des möglichen Einflusses auf das System. Folgende Begriffe werden bei der Risikoanalyse verwendet:
Schwachstelle (Vulnerability) [12]: Eine Schwachstelle ist ein Fehler oder ein Mangel beim Systementwurf oder bei der Systemimplementierung. Versehentlich oder absichtlich ausgenutzt, gelangt man über diese Schwachstelle zu einer Sicherheitslücke im System. Solche Schwachstellen können zu Sicherheitsvorfällen, wie zum Beispiel ungewollter oder verfälschter Datenkommunikation, führen. Schwachstellen lassen sich in technische und nichttechnische aufteilen. Nicht-technische Schwachstellen entstehen durch nicht ausreichende oder nicht vorhandene Sicherheitsstrategien, Prozesse oder Standards. Technische Schwachstellen umfassen bei der Entwicklung eingebrachte Ungenauigkeiten, Fehler oder Schwächen, welche z.B. durch falsche Implementierung oder falsche Konfiguration von Systemfunktionen auftreten können.
Bedrohung (Threat) [12]: Eine Bedrohung resultiert aus der Möglichkeit, dass eine Person oder ein Gerät versehentlich oder absichtlich eine Schwachstelle des Systems nutzt und im System eine Aktivität auslöst oder durchführt. Dabei gibt es mehrere Typen von Bedrohungen, die während des Betriebes eines Systems auftreten können. Prinzipiell lassen sich diese in natürliche, menschliche und aus der Umgebung resultierende Bedrohungen einteilen:

- Natürliche Bedrohungen ergeben sich im Zusammenhang mit Naturkatastrophen, welche zu extremen Skalenwerten bei Vibration und Erschütterung, Wassermenge, Windgeschwindigkeit, Temperatur, Elektrizität, Elektromagnetismus usw. führen können.
- Menschliche Bedrohungen resultieren aus versehentlich oder absichtlich durch Personen ausgelösten Aktivitäten. Versehentliche Aktivitäten können zum Beispiel unbeabsichtigte oder weggelassene Dateneingaben sein. Absichtliche Aktivitäten können beispielsweise netz- und rechnergestützte Angriffe, schadhafte Software-Uploads oder nicht autorisierte Systemzugriffe sein.
- Bedrohungen aus der Umgebung können sich beispielsweise durch ein Versagen der Energieversorgung oder der Kühlung und aufgrund von Verschmutzungen, Chemikalien oder Flüssigkeitsleckagen ergeben.

Risiko (Risk) [15]: Das Risiko beschreibt eine Wahrscheinlichkeitsfunktion für das Eintreten einer gegebenen Bedrohung zusammen mit der erfolgreichen Ausnutzung einer oder mehrerer Schwachstellen. Die Wahrscheinlichkeit wird in Verbindung mit der Schwere und Tragweite der resultierenden Auswirkungen auf das System betrachtet. Das Risiko stellt folglich eine Kombination von Faktoren dar, die, wenn sie auftreten, eine nachteilige Auswirkung auf die gewollten Systemfunktionen haben können. Nachteilige Auswirkungen können Loss of Integrity (auch Misuse genannt), Loss of Availability (auch Denial of Service genannt) und Loss of Confidentiality (auch Disclosure oder Offenlegung genannt) sein.

Die Risikoanalyse wird typischerweise in folgenden Schritten durchgeführt [12]:

1) Charakterisierung des Systems
2) Identifikation und Dokumentation von Bedrohungen
3) Identifikation und Dokumentation der Schwachstellen
4) Analyse und Dokumentation bereits spezifizierter Gegenmaßnamen
5) Bestimmung und Dokumentation der Eintrittswahrscheinlichkeit einer Bedrohung unter Berücksichtigung von Schritt 4)
6) Bestimmung und Dokumentation von Schwere und Tragweite der Auswirkung beim Eintreten einer Bedrohung
7) Bestimmung und Dokumentation des sich aus Schritt 5) und 6) ergebenden Risikos
8) Identifikation und Empfehlung von ergänzenden Sicherheitsmaßnahmen
9) Gesamtdokumentation der Risikoanalyse

Im wichtigen Schritt 1) zur Charakterisierung des Systems wird der Umfang der Risikoanalyse definiert. Dieser Um-

fang lässt sich, wie in BILD 4 gezeigt, in Form einer Matrix ausdrücken, welche durch Wahrscheinlichkeits- und Einflussstufen aufgespannt wird. Innerhalb dieser Matrix wird dann festgelegt, welche Eintrittswahrscheinlichkeit eines Ereignisses und welche Schwere und Tragweite der Auswirkungen ohne Gegenmaßnahme beim Betrieb des Systems akzeptiert werden können. In BILD 4 ist diese Festlegung der Akzeptanz in unterschiedlichen Graustufen dargestellt.

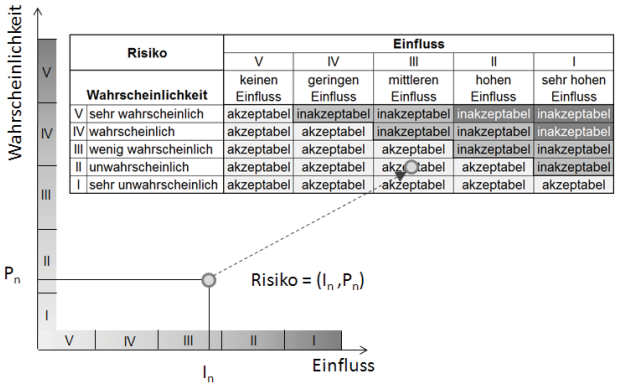

BILD 4. Risikobestimmung und -matrix nach [4]

Nach Abarbeitung der darauf folgenden Schritte 2) bis 6), welche u.a. zur Bestimmung der Eintrittswahrscheinlichkeit P_n und des Einflusses I_n durchgeführt werden, wird dann in Schritt 7) das durch ein gegebenes Systemdesign vorhandene Risiko anhand der aufgestellten Matrix bestimmt (BILD 4). Sollte ein auf diese Weise bestimmtes Risiko (I_n, P_n) inakzeptabel sein, kann durch die Spezifikation und Implementierung von geeigneten Gegenmaßnahmen eine Verringerung der Eintrittswahrscheinlichkeit und damit eine Verschiebung in den akzeptablen Bereich möglich werden.
Nach dieser Einführung von Begriffen und Prinzipien der Risikoanalyse wird im nächsten Kapitel beschrieben, wie sich ein Security-Prozess auf Basis eines RBE-Ansatzes implementieren ließe.

3.3. System Security Engineering mit RBE

Bereits in Kapitel 2.1. wurde anhand von BILD 1 erläutert, dass bei der Systementwicklung die drei V-Modelle für den Systems Engineering-Prozess (SEP), den Safety-Prozess und den Security-Prozess parallel durchlaufen werden. Dabei interagieren jeweils der Safety-Prozess und/oder der Security-Prozess wechselseitig mit dem SEP. Anhand von BILD 5 soll nun weiter vertieft werden, wie sich im Rahmen des Systementwurfs, repräsentiert durch den linken Ast im V-Modell, die spezifische Interaktion des Security-Prozesses mit dem SEP darstellt.
Der Security Prozess beginnt auf Flugzeugebene. Schritt 1 in BILD 5 zeigt, dass hier im ersten Entwicklungsschritt die im SEP spezifizierten Benutzeranforderungen an den Security Prozess in Form eines Dokumentes zu übergeben sind, welches den Umfang für den Security-Prozess spezifiziert und den für den Security-Prozess zutreffenden Aufgabenbereich und das Aufgabenvolumen festlegt. Der ermittelte Umfang dieser durchzuführenden Aktivitäten wird zusammenfassend dokumentiert und an den SEP zurückgekoppelt.
Ebenfalls auf Gesamtflugzeugebene erfolgt nun die Ableitung der Funktionsanforderungen aus den Benutzeranforderungen. Schritt 2 in BILD 5 zeigt, wie auf Basis der vom SEP gelieferten Funktionsanforderungen im Security-Prozess mittels einer funktionalen Impactanalyse mögliche Bedrohungsszenarien für das Flugzeug erarbeitet werden. Hierüber lassen sich die im Entwicklungsprozess zu berücksichtigenden Bedrohungen festlegen und deren Einfluss beschreiben. Nach weiteren Detaillierungsschleifen, bei denen es gilt, auf Flugzeugebene die System Security-Anforderungen den entsprechenden Systemen zuzuordnen, werden die Ergebnisse dokumentiert und an den SEP zurückgegeben.

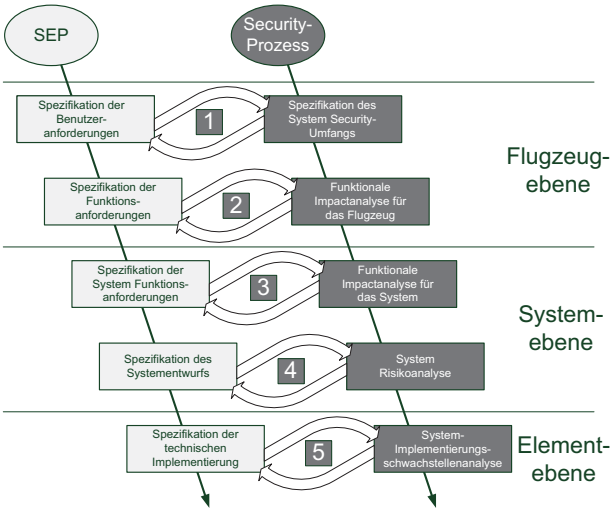

BILD 5. Wechselseitige Interaktionen zwischen dem Systems Engineering-Prozess (SEP) und dem Security-Prozess

Auf der obersten Systemebene zeigt BILD 5 in Schritt 3, wie mit den vom SEP gelieferten Anforderungen für die Systemfunktionen und der bei Schritt 2 auf Flugzeugebene erarbeiteten Impactanalyse jetzt eine Security Impactanalyse für die einzelnen Systemfunktionen durchgeführt wird. Hierbei wird der mögliche Einfluss von Bedrohungen auf einzelne Systeme im Zusammenhang mit der globalen Flugzeugarchitektur analysiert. Die hier resultierenden, funktionalen System Security-Anforderungen werden dem SEP zurückgeliefert und dort in die Dokumentation integriert.
Da im Schritt 3 bislang nur die funktionalen Anforderungen, nicht aber das Risiko betrachtet wurde, wird nun auf Basis des Systementwurfs im SEP im Schritt 4 eine Risikoanalyse nach dem in Kapitel 3.2. beschriebenen Vorgehen durchgeführt. Als Ergebnis steht dann eine Risikoeinstufung der einzelnen Systemfunktionen bezogen auf den Systementwurf zur Verfügung. Wie in Kapitel 3.2. erläutert, kann es bei einem als nicht akzeptabel eingestuften Risiko nötig werden, geeignete Gegenmaßnahmen im SEP zu spezifizieren, was dann zu einem erneuten Iterationsschritt bei der Risikoanalyse führt und eine Verschiebung in den akzeptablen Bereich ermöglicht.
In Schritt 5 in BILD 5 wird im SEP die technische Implementierung spezifiziert, die seitens des Security-Prozesses eine weitere, tiefergehende technische Schwachstellenanalyse der vorgesehenen Implementierung auslöst. Diese muss neben der technischen Spezifikation auch die Ergebnisse der Risikoanalyse aus Schritt 4 berücksichtigen, um am Ende für ein Systemelement die Einhaltung aller System Security-Maßnahmen nachweisen zu können und eine Zertifizierung zu ermöglichen.
Alle hier anhand von BILD 5 für den SEP und den Securi-

ty-Prozess beschriebenen fünf Schritte folgen den in Kapitel 2.2. erläuterten Methoden des RBE und werden heute auf Basis von Textdokumenten erledigt. Ein ebenfalls in Kapitel 2.2. beschriebenes Requirements Management unterstützt dabei den Prozess mit einem Änderungsmanagement, einer Dokumentenverwaltung sowie einem Planungs-, Organisations- und Kontroll-Prozess.

3.4. Herausforderungen bei der Implementierung

Bei der Implementierung eines verbesserten und handhabbaren System Security-Prozesses existieren folgende vier grundsätzliche Herausforderungen, welche bei einem textbasierten Vorgehen nach RBE auftreten und die es die es in Zukunft besser zu beherrschen gilt:

Informationstransfer: Im Gegensatz zur System Safety, wo das nötige Know-How, bezogen auf Flugzeugsicherheit vollständig beim Flugzeughersteller vorliegt, lässt sich im Bereich System Security sagen, dass es nicht ausreicht, wenn der Flugzeughersteller nur auf eigenes Know-How vertraut. Dies resultiert aus einer hohen Dynamik und Bandbreite des System Security-Umfeldes und sich fast täglich verändernden Bedrohungsszenarien. Daher ist es wünschenswert, dass Wissen aus Kooperationen mit externen Partnern zusätzlich eingebracht werden kann. Für solche Kooperationen können andere Wirtschaftszweige, wie zum Beispiel der e-Payment Sektor, überaus hilfreich sein. Dort existieren vergleichbare Risiken und es wurden über Jahrzehnte hinweg Erfahrungen zum Umgang mit System Security aufbaut. e-Payment ist heute ein neues Thema bei den Kabinenfunktionen. Bislang gab es jedoch kaum Berührungspunkte zwischen Experten für e-Payment-Systeme und Experten für Kabinensysteme. Es besteht also heute die Herausforderung, dass für die Erstellung einer System Security-Analyse die funktionalen Systemanforderungen eindeutig zwischen allen Instanzen kommuniziert werden müssen, was angesichts vielfältiger Abhängigkeiten und Wechselwirkungen in textbasierter Form nur schwer lösbar ist.

Identifikation und Nutzung von Negativanforderungen: Die Spezifikation einer Systemfunktion wird üblicherweise als so genannte Positivanforderung formuliert, da das zu entwickelnde System sich später durch die geforderten und implementierten Funktionen auszeichnet. System Security-Anforderungen erfordern jedoch in der Regel eine Negativformulierung, da es häufig darum geht, eventuell implizite, mitgelieferte Systemfunktionen, die nicht spezifiziert sind, konkret auszuschließen. Ein solcher Ausschluss von Funktionen kann nötig werden, um ein spezifisches Security Level (SL) zu erfüllen. Ein Beispiel dafür ist die Beschreibung eines Betriebssystems (OS). So erfordert die Systemfunktion eine spezielle Systemschnittstelle, die im Rahmen des SEP als Positivanforderung in die Systemspezifikation einfließt. Im Rahmen der System Security-Analyse kann es jedoch dazu kommen, dass über die geforderte Systemschnittstelle hinaus das gewählte OS weitere Schnittstellen mitliefert, die zu einer nicht akzeptablen Systemschwachstelle führen könnten. In einem solchen Fall müssten dann so genannte Negativanforderungen formuliert werden, welche kritische Schnittstellen im OS explizit ausschließen. Die Herausforderung besteht hinsichtlich der Identifikation und Beschreibung solcher impliziten Systemfunktionen, die ausgeschlossen werden sollen. Da sie sehr stark von der technischen Implementierung abhängen, kommen diese im Entwicklungsprozess erst sehr spät ins Blickfeld. An dieser Stelle wird eine Methode benötigt, welche die Spezifikation und Nutzung von Negativanforderungen, die im Entwicklungsprozess erst an später Stelle abgeleitet werden können, unterstützt.

Traceability aller Security-Anforderungen: Für eine vollständige System Security-Analyse werden sowohl die Benutzeranforderungen als auch die Systemanforderungen benötigt.

Die Benutzeranforderungen sind eng an die jeweiligen Nutzergruppen angelehnt. Unterschiedliche Nutzergruppen wirken sich bei der System Security-Analyse aufgrund ihrer zum Beispiel stark differierenden Qualifikation und Vertrauensstellung beim Umgang mit dem System sehr unterschiedlich aus.

Die Systemanforderungen werden für die Betrachtung der Systeminteraktionen sowie der Systemfunktionen inklusive deren technischer Realisierung benötigt.

Da die beitragenden Informationen zu unterschiedlichen Zeiten gesammelt werden und vorliegen und weil die Benutzer- und Systemanforderungen auf verschiedenen Ebenen des V-Modells spezifiziert werden, besteht eine Herausforderung bei der Nachverfolgbarkeit der Anforderungen. Die wechselseitige Interaktion des Security-Prozesses mit dem SEP (vgl. BILD 5) erschwert die Nachverfolgbarkeit zusätzlich, sodass auch hier eine handhabbare Methode zur Verbesserung der Nachverfolgbarkeit von Security-Anforderungen gewünscht ist.

Auswirkungen von Systemänderungen: Ähnlich, wie den System Safety-Prozess kann man auch den System Security-Prozess in eine Systementwicklungsphase und eine Systemnutzungsphase aufteilen. Während bei der System Safety die Wahrscheinlichkeiten des Versagens einer Systemfunktion z.B. durch beschleunigte Alterungstests ermittelbar sind und so für die spätere Nutzungsphase vorhergesagt werden können, stellen bei der System Security gerade die Vorhersagbarkeit und der mögliche Erfolg von Angriffen bei der Systemnutzung die größte Herausforderung dar. Dieses resultiert aus einer hohen Dynamik im Bereich der IT-Entwicklung und den damit verbundenen veränderlichen Systemschwachstellen. Als ein Beispiel kann hier das Aufdatieren eines Betriebssystems dienen. Da das Aufdatieren am Ende immer auch eine Veränderung am System bedeutet, muss in jedem Fall mit dieser Änderung eine Risikoanalyse einhergehen. Diese kann abhängig von der Veränderung mehr oder weniger umfangreich sein. Es besteht hier ein Bedarf an einer im Idealfall automatisierbaren Risikoanalyse, welche die Auswirkungen auf alle von der Veränderung betroffenen Elemente berücksichtigt.

Diese vier grundsätzlich bestehenden Herausforderungen können von einer textbasierten RBE-Entwicklungsmethodologie heute nur unbefriedigend gelöst werden. Es soll deshalb im folgenden Kapitel gezeigt werden, welches zusätzliche Potenzial eine modellbasierte MBRE-Methodologie bieten kann.

4. ANSATZ FÜR EINEN MODELLBASIERTEN SYSTEM SECURITY ENGINEERING-PROZESS

Das in Kapitel 2.2. beschriebene und auf Textdokumenten beruhende methodische Vorgehen des RBE stößt zunehmend an seine Grenzen, wenn es um die Sicherstellung der späteren Qualität von komplexen Kabinenmanagementsystemen der nächsten Generation geht. Ein jederzeit detaillierter Überblick und eine entsprechend tiefgehende Berücksichtigung von Systemfunktion, System

Safety und System Security stellen eine große Herausforderung dar, wenn Anforderungen und Funktionen textuell niedergelegt werden. Die Vielzahl der zu berücksichtigenden Elemente und deren vielfältige Wechselwirkungen untereinander sind mit einer textbasierten Dokumentation für den Entwickler kaum zu durchdringen. Zur verbesserten und effizienten Beherrschung von hardware- und softwarebasierten komplexen Systemen und zur Früherkennung von möglichen Fehlern im Entwicklungsprozess kann das so genannte Model-Based-Requirements-Engineering (MBRE) eingesetzt werden [13]. Dieses verknüpft den textbasierten und durch ein Anforderungsmanagementsystem gekennzeichneten RBE-Ansatz mit dem des Model-Based Systems Engineering (MBSE) [7]. Kerngedanke von MBRE ist es, den Prinzipien des RBE mit seinem Anforderungsmanagement zu folgen, jedoch frühzeitig im Entwicklungsprozess abstrakte Systemmodelle zu erzeugen, welche die unterschiedlichen Aspekte eines Systems umfassend repräsentieren bzw. spezifizieren [13]. Ein im Rahmen von MBRE erzeugtes Systemmodell kann also eine bislang in natürlicher Sprache aufwändig formulierte Systemspezifikation prinzipiell ersetzen.
Im folgenden Kapitel 4.1. wird daher ein Modellierungsansatz für den Security-Prozess entworfen, der bei der Entwicklung von Kabinenmanagementsystemen der nächsten Generation den System Security Engineering-Prozess besser integriert. Die Anforderungen an eine solche MBRE-basierte Vorgehensweise und wesentliche Elemente des Modellierungsansatzes werden in den folgenden Kapiteln schrittweise erarbeitet.

4.1. System Security Engineering mit MBRE

Anders als das RBE nutzt das Model-Based-Requirements-Engineering (MBRE) Modelle zur Formulierung von Anforderungen und zur Systemspezifikation. Zur Modellierung eingesetzte formale Sprachen, wie z.B. SysML oder UML erlauben eine standardisierte Systembeschreibung [14]. Man gelangt damit zu einer so genannten formalen Spezifikation, welche eine formale Verifikation der Anforderungen erlaubt. Bei Bedarf kann aus der formalen Spezifikation eine ausführbare Spezifikation abgeleitet werden, d.h. ein simulierbares Modell des Systems, welches die Stellung eines Prototyps einnimmt und getestet werden kann.
Einerseits sind Anwendungen und Vorgehensweisen zum MBRE in der Literatur ausreichend bekannt [7], [14], [15], jedoch fokussieren diese vor allem auf den SEP und die Systemfunktionalität. Andererseits gibt es Aufsätze zur Modellierung von System Security [16], [17], allerdings wird dort nicht die Implementierung eines Security-Prozesses gelöst. In diesem Kapitel soll deshalb gezeigt werden, wie sich ein System Security Engineering-Prozess mittels MBRE gestalten und implementieren lässt. Als generischer Ansatz zum Vorgehen wurde die bei Jon Holt et. al. beschriebene ACRE Ontology (Approach to Context-Based Requirements Engineering) [15] ausgewählt. BILD 6 zeigt dazu ein zur Darstellung der ACRE-Ontologie verwendetes SysML-Blockdiagramm, welches um die spezifischen Aspekte des Security-Prozesses ergänzt wurde.
Die mit den Nummern 1 bis 4 gekennzeichneten Modellelemente werden nachfolgend erläutert und spezifisch auf den System Security Engineering-Prozess ausgerichtet. Damit wird es möglich, das Drei-V-Modell geschlossen nach einem MBRE-Ansatz zu bearbeiten und zugleich den zuvor unter 3.4. beschriebenen Herausforderungen zu begegnen.

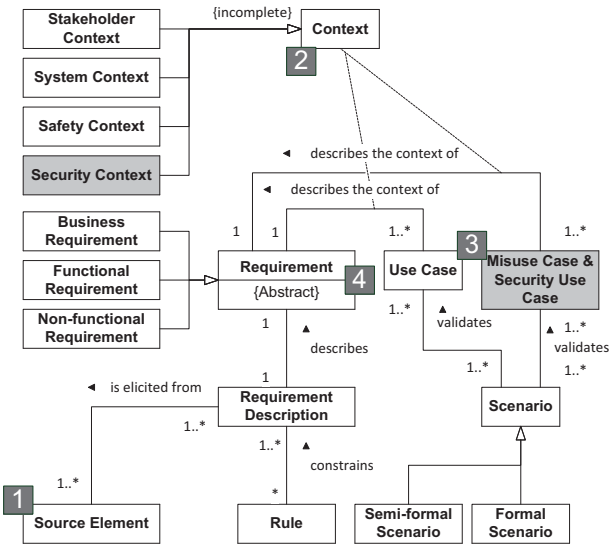

BILD 6. ACRE-Ontologie nach [15]. Für einen Security-Prozess mittels MBRE wurden die Modellelemente Security Context sowie Misuse Case & Security Use Case eingeführt.

4.1.1. Das Source Element

Jedes Projekt beruht auf Informationen, die unterschiedlichen Quellen entspringen. Ohne diese Quellenelemente wären keine Anforderungen formulierbar. In der ACRE-Ontologie nach [15] wird eine jeweilige Informationsquelle als Source Element bezeichnet. BILD 7 zeigt dessen Verbindung zu den anderen Modellelementen innerhalb der Ontologie.

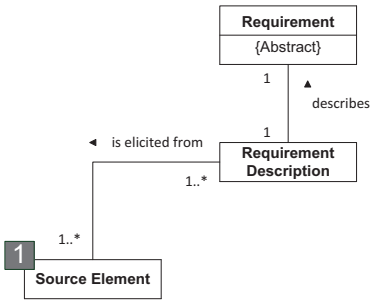

BILD 7. Das Source Element

Es ist zu erkennen, dass ein oder mehrere (vgl. 1..*) Anforderungsbeschreibungen sich aus einem oder mehreren Source Elementen ergeben (vgl. is elicited from). Mit der indirekten Verknüpfung über die Anforderungsbeschreibung an die Anforderung wird eine bessere Nachverfolgbarkeit (Traceability) dadurch erreicht, dass es innerhalb der Anforderungsbeschreibung wesentlich einfacher ist, die verschiedenen Facetten eines Source Elements informativ zu erhalten. Bei der Dokumentation von Source Elementen ist es wichtig, darauf zu achten, dass diese quantifizierbar sind. Es können so Redundanzen vermieden und ein Änderungsmanagement ermöglicht werden. Die Liste möglicher Source Elemente kann in einem Projekt einen beliebigen Umfang annehmen. Aus diesem Grund sollen hier nur die für einen System Security Engineering-Prozess wichtigen Source Elemente angeführt werden:

– *Businessanforderungen*: Da sich die im Prozess später zu spezifizierenden funktionalen Anforderungen aus den Businessanforderungen ableiten, bilden diese einen Teil der Source Elemente. Bezogen auf den System Security-Prozess wäre eine Businessanforderung beispielsweise die Einrichtung eines Security Life Cycle Managements.
– *Standards*: Die Flugzeugindustrie ist geprägt von Standards und Vorschriften der Behörden, aus denen sich Anforderungen für die Produktentwicklung ableiten. Diese Anforderungen bilden damit weitere wichtige Source Elemente. Bezogen auf den System Security-Prozess bedeutet dieses, dass beispielsweise für ein Vorgehen der EUROCAE ED-202 Standard [4], für ein Risikomanagement der NIST Standard [12] und für die Risikoanalyse die Norm ISO 27001 [18] einzuhalten sind.
– *Bestehende Systeme*: Gerade modernere Flugzeugsysteme existieren nicht isoliert und gekapselt, sondern interagieren mit vielen anderen Systemen. Diese umgebenden Systeme liefern in der Regel nichtfunktionale Anforderungen, zum Beispiel in Form von Schnittstellenanforderungen. Für die Betrachtung der System Security nehmen diese einen wichtigen Stellenwert bezüglich ihrer Security-Domänenzuordnung (vgl. Kapitel 2.3) ein. Ein Systembeispiel ist das Dataloading and Configuration Management System (DLCS) als zentrales System für Softwareupdates im Flugzeug, welches Schnittstellen zu vielen anderen Systemen bedient.
– *Informationen:* Weitere Source Elemente sind alle Informationen, die Einfluss auf das Systemdesign haben. Hierzu zählen sowohl die wissenschaftliche Primär- und Sekundärliteratur als auch aktuelle Reports of Findings in der Luftfahrt. Bezogen auf den System Security-Prozess können hier zum Beispiel Security Konferenzen und Security-Foren als wichtige Informationsquelle für aktuelle Bedrohungen und mögliche Gegenmaßnahmen wichtig sein.

Die Liste der hier beispielhaft für einen Security-Prozess angeführten Source Elemente kann nahezu beliebig verlängert werden. Allerdings bilden diese Source Elemente nur eine wichtige Eingangsgröße zur Spezifikation von Anforderungen (vgl. BILD 7). Eine weitere wichtige Eingangsgröße liefern die Anwendungsfälle (Use Cases) sowie die für den Security-Prozess noch einzuführenden Missbrauchsfälle (Misuse Cases) und Gegenmaßnahmen (Security Use Cases). Zu deren Erstellung ist es nötig, den Kontext (Context), in dem sich die Use Cases und Misuse Cases bewegen, zu definieren. Im folgenden Kapitel wird deshalb gezeigt, wie dieser Rahmen aus unterschiedlichen Systembetrachtungsweisen entwickelt wird.

4.1.2. Der Context

Der Kontext (engl. Context) bildet die für die ACRE-Ontologie [15] wichtige Basis. Er entsteht aus verschiedenen Betrachtungsweisen des Systems mit seinen Akteuren, Prozessen und Funktionen. Ohne die grundlegende Context-Information ist es schwer, einen Use Case, einen Misuse Case oder Security Use Case sowie eine Anforderung richtig zu interpretieren. Da dies im weiteren Entwicklungsprozess zu Fehlern führen könnte, soll hier beschrieben werden, wie der Context ermittelt wird. BILD 8 zeigt die für die Kabinensysteme wichtigen Context-Typen. Diese Liste der Context-Typen kann bei Bedarf weiter ergänzt werden, was im BILD 8 mit der Randbedingung {incomplete} bezeichnet ist.

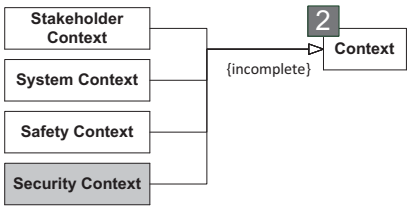

BILD 8. Für Kabinensysteme wichtige Context-Typen

Ein wichtiger Context-Typ nach [15] ist der Stakeholder Context. Jede Person oder Sache, die einen Nutzen, Einfluss oder ein Interesse am System hat, soll als Stakeholder berücksichtigt werden. Stakeholder sollten idealerweise in Anlehnung an ihre Rolle im Systemkontext bezeichnet werden. Wie bei [15] beschrieben, können in Anlehnung an ISO 15505 und ISO 15288 die Stakeholder in Customer, External und Supplier eingeteilt werden. BILD 9 zeigt diese Einteilung, die mit den für die System Security bei Kabinensystemen wichtigen Stakeholdern weiter detailliert wurde.

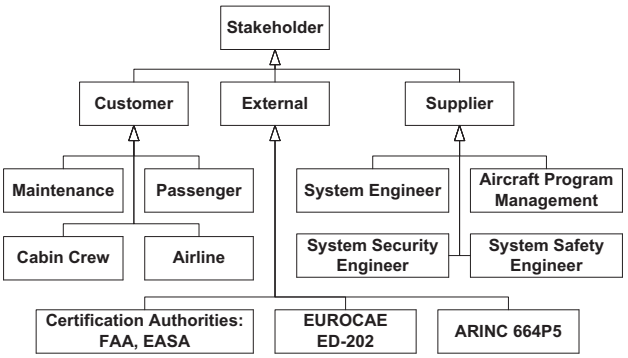

BILD 9. Der Stakeholder Context

Eine erste Gruppe beschreibt den Customer. Diese Gruppe umfasst alle Stakeholder, die das System später nutzen wollen oder bei dessen Betrieb involviert sind. Deren Anforderungen an das System gilt es zuallererst zu berücksichtigen. Im Sinne der System Security sind der Passagier, das Kabinenpersonal, das Wartungspersonal sowie die Airline mögliche Kunden.
Eine zweite Gruppe External beinhaltet alle regulierenden Instanzen als Stakeholder. Deren Anforderungen gilt es unbedingt zu erfüllen. Jedoch können diese verpflichtenden Regularien im Gegensatz zu den Kundenanforderungen nicht diskutiert oder variiert werden. Typische Externals sind zum Beispiel Standards, Normen und Bauvorschriften, nach denen entwickelt werden muss. Für die System Security sind hier die beiden maßgeblichen Zulassungsbehörden FAA und EASA sowie die Normendokumente ARINC 664P5 [11] und die in Europa erstmals initiierte EUROCAE ED-202 [4] zu nennen.
Eine dritte Gruppe Supplier repräsentiert den Flugzeughersteller. Sie enthält alle Stakeholder, die mit der Entwicklung und Lieferung des Systems beauftragt sind. Dies sind das der System Engineer, das Aircraft Program Management, der System Security Engineer und der System Safety Engineer.
Neben dem Stakeholder Context sind der System-Context, der Safety-Context und der Security-Context drei weitere in der Luftfahrtindustrie wichtige Context-Typen. Diese letztgenannten drei Context-Typen gilt es jeweils

auf der Flugzeugebene, der Systemebene und auf der Elementebene des Systems (vgl. BILD 5 und BILD 10) herauszuarbeiten. Alle tragen sie letztendlich zur richtigen und zweifelsfreien Interpretation der System-, Safety- und Security-Anforderungen bei.

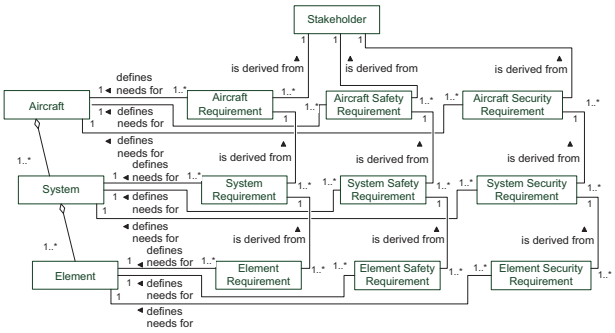

BILD 10. Beziehungen zwischen den drei Entwicklungsebenen (Aircraft, System, Element) und den jeweiligen Anforderungen der Stakeholder

Die jeweiligen Beziehungen zwischen den drei Entwicklungsebenen und den jeweiligen Anforderungen zeigt BILD 10. Es ist dort zu sehen, wie für die zuvor beschriebenen Stakeholder die funktionalen Systemanforderungen sowie die begleitenden Safety- und Security-Anforderungen abgeleitet werden. Im nächsten Prozessschritt folgt dann auf Systemebene und schließlich auf Elementebene für jedes der drei V-Modelle ein Verfeinern der Flugzeuganforderungen. Jede Entwicklungsebene besitzt dabei ihren individuell zu berücksichtigenden Context, was hier für jede Ebene erläutert wird:

– *Flugzeugebene (Aircraft)*: Für den Context auf Flugzeugebene wird betrachtet, wie das Flugzeug in Beziehung zu seinem Umfeld steht. Dabei sind z.B. andere Flugzeuge genauso wichtig wie die Flughäfen. Diese Betrachtungen sind alle anhand der Sichtweisen der hier genannten Stakeholder durchzuführen.
– *Systemebene (System)*: Der Context auf Systemebene erfordert eine Betrachtung, wie die Systeme eines Flugzeuges untereinander in Beziehung stehen. Ein prominentes Beispiel für die Flugzeugkabine ist das Kabinenmanagementsystem, welches als zentrales Kabinensystem vielfältigste Schnittstellen zu anderen Systemen, wie zum Beispiel zum Environmental Control-System, zum Water & Waste-System sowie zum In-Flight-Entertainment-System, unterhält.
– *Elementebene (Element)*: Auf Elementebene werden in weiter differenzierter Form die Beziehungen der Elemente eines Systems betrachtet. Beim oben angeführten Beispiel des Kabinenmanagementsystems mit seinen Elementen betrachtet man auf dieser Ebene unter anderem die Beziehung zwischen dem Zentralrechner des Systems und den verschiedenen Anzeige- und Bediengeräten.

Aufgrund der Erweiterung des Vorgehendmodells zum Drei-V-Modell muss der System Security Context für alle drei Entwicklungsebenen betrachtet werden. Ein so erweiterter Context wird für die in Kapitel 3.3. beschriebene Wechselwirkung zwischen dem SEP und Security-Prozess in möglichst standardisierter Form benötigt. Nur so können Reibungsverluste beim Informationsaustausch minimiert und ein standardisierter Prozessablauf gewährleistet werden.
Der notwendige System Security Context wird nun exemplarisch auf der Systemebene näher erläutert. Aus Sicht der System Security lässt sich jedes System in Anlehnung an [11] und wie in BILD 11 schematisch dargestellt, strukturieren:

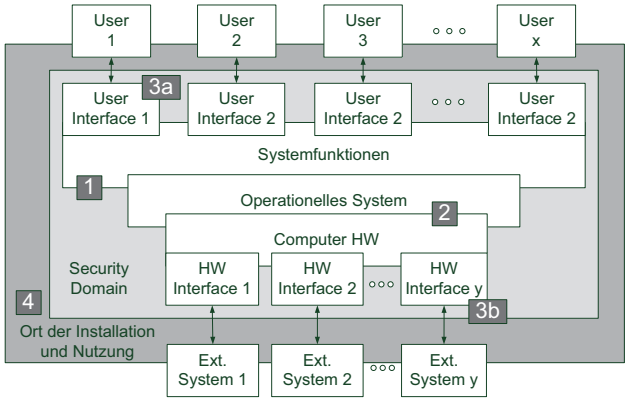

BILD 11. Schema eines Systems mit den vier wichtigen Kontextparameter-Bereichen 1) bis 4) für die System-Security-Analyse

Man erkennt, wie sich das Gesamtsystem, basierend auf der Computer Hardware (HW) mit seinen HW-Schnittstellen, dem operationellen System und den darüber liegenden Systemfunktionen aufbaut. Die Systemfunktionen stellen bei Bedarf eine geeignete Anzahl von Nutzer-Schnittstellen zur Verfügung. Gemäß Kapitel 3.2. wird eine Systemfunktion einer System Security-Domäne zugeordnet und es wird der Installationsort des Systems bestimmt. Die Nummerierung in BILD 11 markiert die folgenden Kontextparameter-Bereiche 1) bis 4), die für eine standardisierte System Security-Analyse heranzuziehen sind:

1) die Systemfunktionen mit der dazugehörigen Security Domänenzuordnung,
2) die Systemplattform mit ihrer Hardware und dem operationellen System
3) die physischen und logischen Schnittstellen inklusive des Verbindungstyps (3a) und die Mensch-Maschine-Schnittstellen (3b) und
4) die Orte der physischen Installation und der Hardwarenutzung.

Zur Erarbeitung der benötigten Kontextparameter, welche der späteren System Security-Analyse dienen, müssen folgende Leitfragen für jeden der vier Bereiche berücksichtigt werden:

1) Welche Funktionen sollen durch das System realisiert werden? In welcher Domäne des Flugzeugnetzwerkes sollen die Funktionen allokiert werden?
2) Wird für das System mit seinen Funktionen eine Hardware mit hoher Performance benötigt? Wird ein operationelles System als Plattform für die Funktionen angestrebt?
3) 3a) Welche physischen und logischen Systemschnittstellen müssen bereitgestellt werden? In welcher Domäne des Flugzeugnetzwerkes befinden sich die mit diesen Schnittstellen verbundenen Systeme? Für welche Systeme werden Informationen über welche Schnittstellen durch das System geleitet?
   3b) Welche Nutzerschnittstellen müssen bereitgestellt werden?
4) Wo im Flugzeug sollen die Nutzer-Schnittstellen benutzt werden? Wo wird das System mit all seinen

Systemelementen installiert? Wo sind die verbundenen Systeme im Flugzeug installiert?

Durch Beantwortung dieser Leitfragen können nun Kontextparameter für die System Security abgeleitet werden. Ein Beispiel für zwei Kontextparameter sind die Domänenzuordnung (vgl. Kapitel 2.3. ACD, AISD, PIESD mit PODD) sowie die Systeminteraktion mit anderen Systemen (z.B. keine Interaktion bzw. Interaktion mit Environmental Control-System, Water & Waste-System, In-Flight-Entertainment-System, etc.) Im Rahmen einer System Security-Analyse lässt sich jetzt mit Hilfe dieser beiden Parameter eine Aussage darüber machen, ob eine tiefergehende Analyse benötigt wird, weil beispielsweise eine domänenübergreifende Kommunikation besteht, oder ob dies entfallen kann, weil das System weitestgehend gekapselt und ohne externe Schnittstellen existiert. Ein System letzterer Art wäre dann mit physischen Security-Mechanismen ausreichend abzusichern.
Mit dem nun herausgearbeiteten Context, wobei wir speziell den Security Context intensiver betrachtet haben, ist es nun möglich, sich den Use Cases, Misuse Cases und Security Use Cases zu widmen.

4.1.3. Der Use Case, der Misuse Case und der Security Use Case

Auf Basis des herausgearbeiteten Kontexts lassen sich als weitere wichtige Eingangsgrößen für Anforderungen die Anwendungsfälle (Use Cases) sowie die für den Security-Prozess noch einzuführenden Missbrauchsfälle (Misuse Cases) und die entsprechend zu ergreifenden Gegenmaßnahmen (Security Use Cases) ermitteln. Die Use Cases führen zu den Anforderungen, indem sie diese in ihrem Context beschreiben und damit im weiteren Sinne selbst zu einer Art von Anforderungsbeschreibung werden. Dieser Zusammenhang ist in BILD 12 dargestellt.

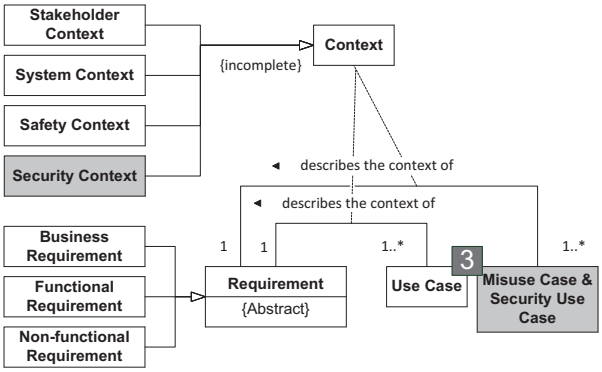

BILD 12. Use Case, Misuse- und Security Use Case

Ein großer Vorteil von Use Cases ist darin zu sehen, dass diese für den Customer eine ergänzende Beschreibungsform der Businessanforderungen anhand des Anwendungs- bzw. Nutzungsfalls des Systems bieten. Dies unterstützt auf Seiten des Suppliers eine Ableitung von funktionalen Anforderungen für das System. Darüber hinaus werden die Use Cases zur Verifikation von Anforderungen genutzt, d.h. zur Überprüfung, ob das für das System Geforderte durch die Anforderungen richtig abgebildet wird. BILD 13 gibt einige Use Case-Beispiele für die Nutzung eines Anzeige- und Bediengerätes des Kabinenmanagementsystems.

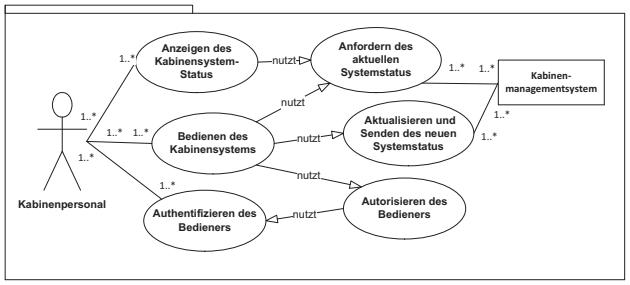

BILD 13. Use Cases für ein Anzeige- und Bediengerät des Kabinenmanagementsystems

Es ist gut zu erkennen, dass es bei diesem Beispiel möglich ist, sich den Status von Kabinensystemen auf dem Gerät anzeigen zu lassen und auch deren Systemparameter zu verändern, also beispielsweise die Kabinentemperatur anders einzustellen. Die Use Cases Authentifizieren und Autorisieren zeigen, dass eine Absicherung der Bedienung für den Fall der Parameteränderung benötigt wird, und dass sich diese Absicherung aus der Authentifizierung und der Autorisierung zusammensetzt. Diese beiden letztgenannten Anwendungsfälle beziehen sich auf den System Security Context des Systems. Aus System Security-Sicht ist die alleinige Betrachtung von Use Cases jedoch nicht ausreichend, da diese lediglich den gewollt geforderten Nutzungsbereich abdecken. Vielmehr muss die System Security besonders solche Fälle betrachten, die es erlauben, ungewollte Funktionen zu aktivieren. Hierzu haben Y. Chung and M. Yung den Misuse- und Security Use Case eingeführt [17]. Diese beiden, für den Security-Prozess besonders wichtigen Modellelemente sind in BILD 12 dargestellt und mit der Nummer 3 markiert. Für den Security-Prozess wurde also die ACRE-Ontologie von uns um diesen Bereich erweitert.
Mittels der Misuse Case-Modellelemente werden solche Funktionalitäten beschrieben, die als Systemfunktion nicht erlaubt sind. Die in das Gesamtmodell integrierten Misuse Cases lassen sich dann für die System Security-Analyse verwenden. BILD 14 liefert eine um zwei in dunkelgrau markierte Misuse Cases erweiterte Darstellung von BILD 13.

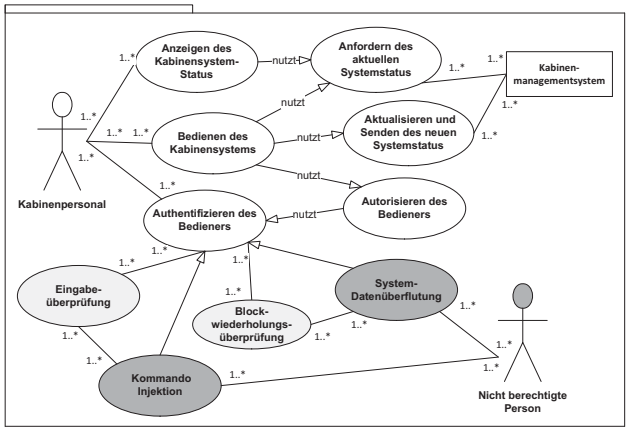

BILD 14. Misuse Cases (dunkelgrau) und Security Use Cases (hellgrau) für die Anzeige- und Bediengeräte des Kabinenmanagementsystems

Zur Bedienung der Kabinensysteme über das Anzeige- und Bediengerät ist es nötig, sich gegenüber dem System zu authentifizieren. Nach einer erfolgreichen Authentifizierung führt das System eine Autorisierung des Bedieners entsprechend dessen Benutzerrolle durch. Der Bediener

ist im gewollten Fall ein Mitglied des Kabinenpersonals. Für den ungewollten Fall ist beispielhaft ein Angreifer, also eine nicht berechtigte Person mit den denkbaren Misuse Cases System-Datenüberflutung und Kommando Injektion dargestellt.
Mit der Integration von Misuse Cases in das Systemmodell wäre es jetzt möglich, eine System Security-Analyse durchzuführen, wenn die System Security-Anforderungen in Ihrem Context wie gefordert beschreibbar wären. Da aber klassische Systemanforderungen als Positivanforderungen formuliert werden und der Misuse Case eine Negativanforderung beschreibt, müssen die sich aus der Risikoanalyse ergebenden Gegenmaßnahmen modelliert werden können. Bei [17] wurden dazu die so genannten Security Use Cases eingeführt, die sich als Gegenmaßnahmen aus den Misuse Cases ergeben und in BILD 14 in hellgrau dargestellt sind. Security Use Cases sind also Gegenmaßnahmen, die dazu dienen, ein als nicht akzeptabel erkanntes Risiko durch Reduktion der Eintrittswahrscheinlichkeit in einen akzeptablen Bereich zu bringen.
Die beiden in BILD 14 enthaltenden Security Use Cases zeigen, wie für den Misuse Case der Kommando Injektion als geeignete Gegenmaßnahme eine Eingabeüberprüfung und gegen die System-Datenüberflutung eine entsprechende Blockwiederholungsüberprüfung implementiert werden sollen. Mit der Einführung von Security Use Cases wird es also möglich, die System Security-Anforderungen in ihrem Context zu beschreiben und besser zu interpretieren. Auf die sich daraus ergebende Spezifikation von Security Anforderungen bzw. von Anforderungen ganz allgemein wird im nachfolgenden Kapitel eingegangen.

4.1.4. Das Requirement

Die Anforderung (engl. Requirement) ist das wichtigste Element einer Systemspezifikation, da über die Requirements die Funktionalität eines Systems eindeutig beschrieben und später nachgewiesen wird. Die Requirements bilden damit die wichtige vertragliche Basis für die Implementierung des Systems. Ausgehend von den in den vorangegangenen Kapiteln 4.1.1. bis 4.1.3. erarbeiteten Modellelementen und Informationen (Source Element, Context-Information, Use Case, Misuse Case und Security Use Case) können jetzt die Anforderungen für das Systems spezifiziert werden.
Die Formulierung von Requirements erfolgt nach einer festgelegten Form und sollte in Anlehnung [15] mindestens folgende Attribute enthalten:

- Requirement ID: eindeutige Nummerierung der Anforderung
- Requirement Text: standardisierte Formulierung der Anforderung.
- Origin: Stakeholder-Referenz (z.B. Customer) welche anfordert
- Validation Criteria: Use Case-Referenz
- Ownership: Stakeholder-Referenz (z.B. Supplier) welche realisiert

Da der von uns entwickelte MBRE-Ansatz dem Drei-V-Modell (vgl. Kapitel 2.1.) folgt und unter Zuhilfenahme der Misuse- und Security Use Cases auch den wichtigen System Security Engineering-Prozess mit einschließt, gelingt an dieser Stelle erstmals die modellbasierte Formulierung von Requirements, bei der auch Security-Anforderungen mit einbezogen sind. Einerseits werden die Requirements für die Systemfunktionen und die Zuverlässigkeit des Systems von den Use Cases in Verbindung mit dem System- und Safety Context hergeleitet. Andererseits leiten sich die Requirements für die System Security aus dem Security Context und den Misuse- und Security Use Cases ab. Entsprechend sind zur Validierung von Requirements zusätzlich zu den Use Cases auch die Misuse Cases und die Security Use Cases heranzuziehen. Das bei der Formulierung von Requirements genutzte Attribut Validation Criteria ist also wie folgt zu vervollständigen:

- Validation Criteria: Use Case-, Misuse Case- und Security Use Case-Referenz

Darüber hinaus schlagen wir vor, die Anforderungen zu kategorisieren und entsprechend dem Drei-V-Modell wie folgt einzuteilen:

- Requirement Category: System-, Safety- oder Security-Anforderung

Aufgrund der Erweiterung um den Security-Prozess ist es nötig, jedem Requirement im Entwicklungsprozess seine Security Parameter (vgl. Kapitel 4.1.2.) zuzuweisen. Dies geschieht mithilfe des Attributs Security Context, welches anhand seiner Parameter beschrieben wird:

- Security Context: Satz von verschiedenen Security-Parametern

BILD 16 zeigt einen Requirement-Block, welcher die zuvor beschriebenen Attribute enthält.

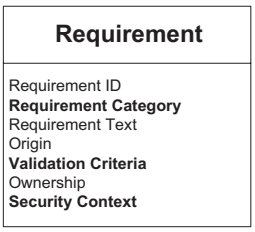

BILD 15. Requirement-Block mit Attributen

Zusätzlich wird das Requirement durch eine Requirement-Beschreibung ergänzt (BILD 16). Die Requirement-Beschreibung enthält abstrahierte Informationen und Merkmale der beitragenden Source Elemente, die zum besseren Verständnis des Requirement Textes genutzt werden, welche jedoch für die direkte Aufnahme in den Requirement-Block ungeeignet sind. Diese Informationen und Eigenschaften können zum Beispiel Komplexitätseinstufungen, zusätzliche Verfolgbarkeitsinformationen zu anderen Dokumenten oder auch vertragliche Themen sein.

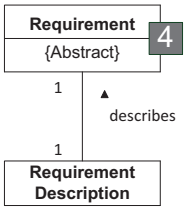

BILD 16. Requirement und Requirement Description

Bezogen auf die System Security wird es über die Requirement-Beschreibung möglich, diverse Hintergrundinformationen zum Security-Requirement einzubinden, die es erleichtern, dieses im Zusammenspiel mit anderen Security-Requirements richtig einzuordnen, umzusetzen und die Herkunft des Requirements besser zu verfolgen.

5. ERGEBNISSE UND AUSBLICK

Mit dem von uns entwickelten MBRE-Ansatz im Drei-V-Modell, welches den System Security Engineering-Prozess erstmals methodisch integriert, gelingt an dieser Stelle die modellbasierte Formulierung von Requirements für die Entwicklung eines Kabinenmanagementsystems der nächsten Generation. Vor Ausarbeitung dieses Ansatzes bestand eine Herausforderung bei der standardisierten Integration des Security-Prozesses in den Systementwicklungsprozess. Durch Erweiterung der eingesetzten ACRE-Ontologie (BILD 6) [15] um den Misuse Case und Security Use Case [17] sowie um den Security Context entsteht erstmals eine Kombination aus einem bei der Kabinensystementwicklung gelebten System Security Engineering-Prozess und einer standardisierten Security-Modellierungsmethode innerhalb einer MBRE-Ontologie.
In der Praxis ist es wichtig, den Security Context im Prozessablauf so mitzuführen und zu dokumentieren, dass wichtige Informationen zwischen den Stakeholdern besser kommuniziert werden können. Dies trägt ganz wesentlich zu einem verbesserten Verständnis aller Beteiligten und einer vereinfachten Implementierung der Security-Anforderungen bei. Zur Umsetzung innerhalb des Modells wurden von uns die Security Context Parameter eingeführt, die für die Erarbeitung der Misuse Cases und Security Use Cases wichtig sind und bei der Requirement Spezifikation im Attribut Security Context verankert werden. Die bei einem mittels RBE bearbeiten Security-Prozess bestehenden Herausforderungen beim Informationstransfer und bei der Nachverfolgbarkeit (Traceability) von Security-Anforderungen, bei der Identifikation und Nutzung von Negativanforderungen und bei der Analyse der Auswirkungen von Systemänderungen werden von der vorgeschlagenen MBRE-Methodologie besser adressiert. Weiterhin ist es damit möglich, alle in den drei parallel verlaufenden V-Prozessen anfallenden Informationen und Anforderungen innerhalb nur eines formalisierten Systemmodells abzubilden. Eine weitergehende spätere Nutzung dieses Modells bei der Implementierung, der virtuellen Integration und der Validierung durch Testfälle sollte daher gut möglich sein.

6. REFERENZEN

[1] EUROCAE / SAE: Certification considerations for highly-integrated or complex aircraft systems, EROCAE ED-79 / SAE ARP-4754, November 1996

[2] EUROCAE / SAE: Guidelines and methods for conducting the safety assessment process on civil airborne systems, EROCAE ED-135 / SAE ARP-4761, Dezember 1996

[3] S. Benz: Eine Entwicklungsmethodik für sicherheitsrelevante Elektroniksysteme im Automobil, Dissertationsschrift, Universität Karlsruhe (TH), 2004

[4] EUROCAE: Airworthiness security process specification, EUROCAE ED-202, Dezember 2010

[5] H. Hintze, A. Tolksdorf, R. God: Cabin core system - A next generation platform for combined electrical power and data services, Tagungsband zum 3rd International Workshop on Aircraft System Technologies - AST 2011, H, 221-231, März 2011

[6] B. Rosenberg: Cabin Management Systems, Avionics Magazine, 26-30, Mai 2010

[7] C. Ebert: Systematisches Requirements Engineering, 3. aktualisierte und erweiterte Auflage, dpunkt.verlag, 2010

[8] K. Pohl: Requirements Engineering, 2. korrigierte Auflage, dpunkt.verlag, 2008

[9] EASA Certification Standards bzw. FAA Federal Aviation Regulations: CS 25.1309 bzw. FAR Part 25.1309

[10] ARINC: Commercial aircraft information security concepts of operation and process framework, ARINC Report 811, Dezember 2005

[11] ARINC: Network domain characteristics and interconnection, ARINC 664P5 – Aircraft data network part 5, April 2005

[12] G. Stoneburner, A. Goguen, A. Feringa: Risk Management Guide for Information Technology Systems; Recommendations of the National Institute of Standards and Technology (NIST); NIST Special Publication 800-30, MD 20899-8930, Juli 2002

[13] R. God, H. Hintze: Komplexität beherrschen: Methodologie für die modellbasierte Entwicklung von Kabinensystemen, Ingenieursspiegel, 1, 34-36, Februar 2012

[14] T. Weilkiens: Systems Engineering mit SysML/UML, 2. aktualisierte u. erweiterte Auflage, dpunktverlag, 2008

[15] J. Holt, S. A. Perry, M. Brownsword: Model-Based Requirements Engineering, IET, 2012

[16] A. Maña, J. F. Ruiz, R. Harjani: SecFutur - Design of Secure and energy-efficient embedded systems for Future Internet applications, SecFutur, IST-256668, Deliverable 4.1, V1.0, 2011

[17] Y. Chung, M. Yung: Using Special Use Cases for Security in the Software Development Life Cycle, Springer-Verlag, WISA 2010, LNCS 6513, 122–134, 2011

[18] ISO/IEC 27001:2005 Information technology – Security techniques – Information security management systems – Requirements, 2005