

Kapitel 7

Dynamische Verteilung von Daten in taktischen Netzen

Norman Jansen und Marc Spielmann

7.1 Einleitung

Aufgrund des veränderten Einsatz- und Aufgabenspektrums der Bundeswehr im internationalen Umfeld ergeben sich zur Unterstützung der Vernetzten Operationsführung (NetOpFü; BMVg, 2006) neue Anforderungen an FüInfoSys. Eine wesentliche Aufgabe zukünftiger FüInfoSys ist hierbei die verzugsarme und unterbrechungsfreie Informationsversorgung auf und zwischen allen Truppenteilen, d. h. die Schaffung eines streitkräftegemeinsamen, ebenenübergreifenden Kommunikations- und Informationsraums (BMVg, 2006). Für die Realisierung eines solchen gemeinsamen Informationsraums ist eine vollständige Integration aller Truppenteile auf allen Führungsebenen in ein durchgängiges Gesamtsystem notwendig.

Insbesondere die nahtlose Integration der taktischen Ebene hat einen besonderen Stellenwert, da grundlegende Informationen wie bspw. die Positionen und Status eigener Truppenteile ursprünglich von dieser Ebene bereitgestellt werden. Demnach muss die Architektur des FüInfoSys die besonderen Randbedingungen der taktischen Ebene berücksichtigen. So handelt es sich bei taktischen Netzwerken üblicherweise um Funknetze im UHF- bzw. VHF-Band. Diese zeichnen sich aufgrund der verwendeten Funkgeräte und der physikalischen Effekte der Funkübertragung (Interferenz, Abschattung etc.) durch sehr schmalbandige, oft unzuverlässige Kommunikationsverbindungen mit einer hohen und variablen Latenz aus. Aufgrund der Mobilität der Teilnehmer und der variierenden Kommunikationseigenschaften muss im taktischen Bereich von einer dynamischen Netzwerktopologie, in der Teilnehmer ad hoc miteinander kommunizieren, ausgegangen werden. Es liegen also mobile Ad-hoc-Netze (MANETs) vor, die eine besondere Herausforderung an die Kommunikationssysteme darstellen. Auch im zivilen Bereich zeichnet sich der Trend hin zu mobilen Endgeräten ab. Allerdings beschränkt sich die Dynamik der Netzwerktopologie im zivilen Bereich üblicherweise auf die Anbindung der Endgeräte. So sind beispielsweise die Endgeräte in heutigen Mobilfunknetzen über stationäre Basisstationen angebunden. Weiterhin stehen im zivilen Bereich zunehmend höhere Bandbreiten zur Verfügung (beispielsweise durch Technologien wie UMTS, Wi-

MAX), während die Einschränkungen der militärischen Netze auf absehbare Zeit hinsichtlich Übertragungskapazität bestehen bleiben. Aus den genannten Gründen sind Technologien aus dem zivilen Bereich zurzeit nur bedingt auf den militärischen Bereich übertragbar.

Für die Realisierung des von NetOpFü geforderten ebenenübergreifenden Informationsraums sind unter oben beschriebenen Kommunikationseinschränkungen neue Lösungsansätze notwendig. Da die Erreichbarkeit von einzelnen Teilnehmern/Knoten aufgrund der Kommunikationsproblematik nicht gewährleistet werden kann, müssen Redundanzen in das System eingeführt werden, um die systemweite Verfügbarkeit von Informationen zu sichern. Eine zentrale Datenhaltung hätte zwar die Vorteile eines geringen Wartungsaufwands und leicht zu realisierender Datenkonsistenz, ist aber aufgrund der eingeschränkten Kommunikation und der Forderung nach zeitweise autarken Endgeräten ungeeignet. Stattdessen sollte eine verteilte Datenbank eingesetzt werden. Diese besteht aus lokalen Datenbankinstanzen, die auf verschiedene Knoten des Netzwerks verteilt sind.

Wegen der Einschränkungen der taktischen Ebene muss besondere Sorgfalt getroffen werden, um die zwischen den Knoten zu übertragende Datenmenge, die für die Sicherstellung der Konsistenz der Daten notwendig ist, gering zu halten. Deshalb sollte jeder Truppenteil stets nur mit den Informationen versorgt werden, die er in der aktuellen Situation benötigt. Dies gestaltet sich unter den Randbedingungen von NetOpFü schwierig, da sich die Informationsbedarfe der verschiedenen Truppenteile dynamisch mit der Auftragssituation ändern können. Aus diesem Grund sollte die Datenverteilung einfach konfiguriert werden können. Zur Konfiguration des Systems wird ein *Datenmanagement-System* benötigt, welches insbesondere festlegt, welche Daten zwischen welchen Nutzern des Systems ausgetauscht werden. Mit Hilfe des Datenmanagement-Systems kann eine dezentrale Organisationseinheit zum Datenmanagement (Datenadministratoren) das System netzwerkbasierend mit einem geringen Aufwand, insbesondere ohne Programmierung, an geänderte Auftragslagen (und damit geänderte Informationsbedarfe) anpassen.

In diesem Beitrag wird ein Modell eines verteilten Datenhaltungs- und Datenmanagement-Systems vorgestellt. Das Modell kann als Grundlage für die Entwicklung von Datenmanagement-Systemen für militärische Netzwerke dienen. Wesentliches Merkmal des Modells ist eine verteilte Datenhaltung, die an die besonderen Randbedingungen militärischer Netzwerke angepasst ist. Zur Konsistenthaltung der lokalen Instanzen der verteilten Datenhaltung stellen wir ein Replikationsverfahren, welches in Netzen mit geringen Bandbreiten und instabilen Verbindungen eingesetzt werden kann, vor. Darüber hinaus umfasst das Modell Datenmanagement-Mechanismen zur Steuerung der Initialisierung des Systems und zur Steuerung der Verteilung (Replikation) der Daten zwischen den lokalen Instanzen der verteilten Datenhaltung. Damit ist eine einfache Konfiguration des Systems durch Datenadministratoren möglich.

Der Beitrag gliedert sich wie folgt: Die wesentlichen Aspekte des Modells werden im nächsten Abschnitt beschrieben. Eine Detaillierung folgt in Abschn. 7.3. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf zukünftige Arbeiten in Abschn. 7.4.

7.2 Übersicht über das Modell

Zentrale Aufgabe eines verteilten Datenhaltungs- und Datenmanagement-Systems für militärische Netze ist die Steuerung der Verteilung sowie die Wahrung der Konsistenz von Datenobjekten. Aufgrund der Kommunikationsbeschränkungen in militärischen Netzen sollten die Datenobjekte so verteilt werden, dass jede Anwendung möglichst lokal auf die von ihr benötigten Datenobjekte zugreifen kann. Die Verteilung der Datenobjekte bietet jedoch nur Vorteile, wenn der Mehraufwand für die Wahrung der Konsistenz (Replikation) die Vorzüge der lokalen Speicherung nicht übersteigt. Mit dieser Zielsetzung wurde das folgende Modell entwickelt.

Lokale Objektdatenbanken. Konzeptionell wird jedem Nutzer (Person, Truppenteil, Rolle, Organisationseinheit etc.) eine lokale Datenbankinstanz zugeordnet. In dieser lokalen Datenbankinstanz sind die für den Nutzer relevanten Daten gespeichert. Der Nutzer greift ausschließlich über seine (lokale) Datenbankinstanz auf die Daten zu. Möchte der Nutzer auf Daten zugreifen, die nicht in seiner Datenbankinstanz vorgehalten werden, werden diese Daten zu seiner Instanz repliziert. Wir setzen voraus, dass die Anwendungen eines Nutzers und die Datenbankinstanz des Nutzers auf dem selben Knoten im Netzwerk ausgeführt werden. Außerdem setzen wir voraus, dass die Daten gemäß eines objektorientierten Datenmodells (vgl. Abschn. 7.3.1) repräsentiert sind. Wir bezeichnen daher im Folgenden die Datenbankinstanz eines Nutzers als *Objektdatenbank* (kurz: *ODB*).

Man beachte, dass unsere Annahme, dass jedem Nutzer eine eigene ODB zugeordnet ist, nicht bedeutet, dass ODBs verschiedener Nutzer zwangsläufig auf verschiedenen Knoten des Netzwerks liegen müssen. Vielmehr handelt es sich bei ODBs um logische Einheiten, die beliebig auf Knoten des Netzwerks verteilt werden können.

Dieser Sachverhalt ist in Abb. 7.1 dargestellt. Ein Rechteck repräsentiert eine ODB eines Nutzers mit seinen zugehörigen Anwendungen. In der ODB sind die Datenobjekte gespeichert, die für den Nutzer (bzw. die zugehörigen Anwendungen)

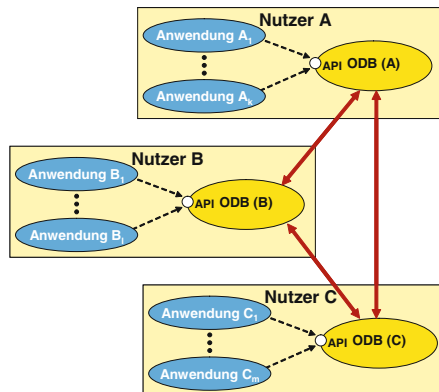


Abb. 7.1 Schematische Darstellung der ODBs von drei verschiedenen Nutzern

relevant sind. Die Datenobjekte werden mittels eines Replikationsverfahrens, welches Gegenstand von Abschn. 7.3.2 ist, zwischen den einzelnen ODBs verteilt. Auf die in der Abbildung rot dargestellten Kommunikationsverbindungen wird weiter unten eingegangen.

Initialisierungskomponente. Wir unterscheiden zwischen einer Initialisierungsphase und der eigentlichen Laufzeitphase. Die Initialisierungsphase wird durch eine *Initialisierungskomponente* (vgl. Abb. 7.2) gesteuert. In der Initialisierungsphase werden zuerst initiale Grunddaten, die in der Initialisierungskomponente verwaltet werden, zu den einzelnen ODBs repliziert. Hierfür besitzt die Initialisierungskomponente ebenfalls eine ODB. Nach der Initialisierungsphase befinden sich alle ODBs in einem wohldefinierten Anfangszustand. Das System wechselt unmittelbar darauf in die Laufzeitphase, in der Daten zwischen den (eigentlichen) ODBs repliziert werden. Die beiden Phasen werden in Abschn. 7.3.3 genauer betrachtet.

Kommunikationsverbindungen. In unserem Modell wird zwischen Kommunikationsverbindungen, die zur Laufzeit des Systems für die Replikation verwendet werden (in Abb. 7.2 rot dargestellt) und Kommunikationsverbindungen, die für die Initialisierung des Systems bzw. einzelner ODBs genutzt werden (grün dargestellt), unterschieden. Die zur Laufzeit für die Replikation verwendeten Verbindungen werden aufgrund der Einschränkungen der taktischen Ebene als schmalbandig und instabil angenommen, während die zur Initialisierung des Systems genutzten Verbindungen als stabil und breitbandig angenommen werden. Die Anwendungen können über eine spezielle Schnittstelle (API) auf ihre jeweilige ODB zugreifen. Diese Schnittstelle sollte Methoden für den Zugriff auf Datenobjekte und für die Erstellung, Änderung und das Löschen von Datenobjekten bereitstellen. Des Weiteren sollte die Schnittstelle Methoden für die Konfiguration der ODB vorsehen.

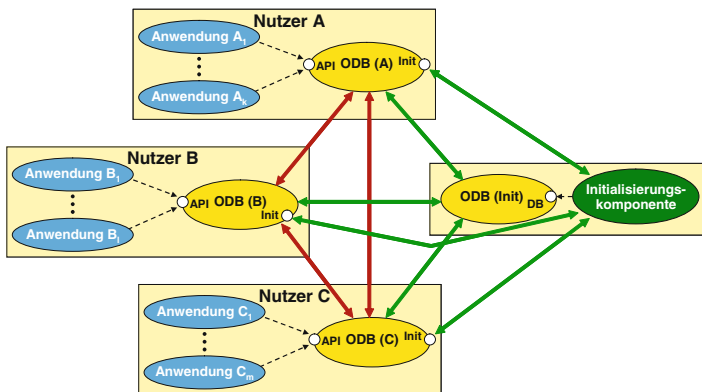


Abb. 7.2 In der Initialisierungsphase erhält jede ODB durch die Initialisierungskomponente initiale Grunddaten. Man beachte, dass Kommunikationsverbindungen zwischen ODBs als durchgezogene Pfeile dargestellt sind, im Gegensatz zu lokalen Datenbankzugriffen, welche durch gestrichelte Pfeile repräsentiert werden

7.3 Detaillierung des Modells

In diesem Abschnitt werden die wesentlichen Aspekte des Modells ausführlicher dargestellt. Dazu werden im nächsten Unterabschnitt einige Grundbegriffe aus dem Bereich der objektorientierten Datenmodelle wiederholt bzw. neu eingeführt. Das bereits oben angedeutete Replikationsverfahren und die Datenmanagement-Mechanismen zur Steuerung der Initialisierung des Systems und zur Steuerung der Replikation der Daten sind Gegenstand der darauf folgenden Unterabschnitte.

7.3.1 Objektorientiertes Datenmodell

Im Folgenden setzen wir voraus, dass die Informationen gemäß eines objektorientierten Datenmodells repräsentiert sind. Da die hier beschriebenen Konzepte weitgehend unabhängig von der exakten Definition des gewählten objektorientierten Datenmodells anwendbar sind, führen wir hier nur die grundlegende Terminologie von objektorientierten Datenmodellen auf. Dabei stützen wir uns auf den Standardisierungsvorschlag *ODMG 3.0* der Object Data Management Group (vgl. Cattell & Barry, 2000), der von vielen Herstellern objektorientierter Datenbanksysteme verwendet wird.

Objekte. Ein Objekt ist eine Entität bestehend aus

- einem Objektidentifikator (kurz *OID*), der das Objekt eindeutig identifiziert,
- einer Menge von Attributen (siehe unten) und
- einer Menge von Methoden (siehe unten).

Attribute. Ein Attribut hat einen Wert aus einer zugeordneten Wertmenge. Die zugeordnete Wertmenge wird als der *Datentyp* des Attributs bezeichnet. Jeder Datentyp ist entweder

- ein atomarer Datentyp (zum Beispiel: integer, real, char oder string),
- eine Referenz auf ein anderes Objekt,
- eine Liste von Referenzen oder
- eine aus atomaren Datentypen zusammengesetzte (endliche) Struktur (zum Beispiel: Liste, Menge).

Methoden. Methoden sind auf Objekten ausführbare Operationen (Programme).

Typen. Ein Typ (auch *Klasse* genannt) spezifiziert eine Menge von Attributen und eine Menge von Methoden. Intuitiv bestimmt ein Typ die Struktur und das Verhalten (durch die Methoden) von Objekten. Jedem Objekt ist hierfür ein Typ zugeordnet. Ein Objekt wird als Instanz des zugehörigen Typs bezeichnet.

Spezialisierung. Eine Spezialisierung ist eine Beziehung zwischen zwei Typen. Der spezialisierte (abgeleitete) Typ erbt alle Attribute und Methoden des *Obertyps* und

ergänzt diese eventuell durch neue Attribute und Methoden. Wir erlauben keine Mehrfachspezialisierung (Mehrfachvererbung). Das heißt wir gehen davon aus, dass jeder Typ höchstens einen Obertyp besitzt. Durch die Spezialisierungsbeziehungen können die Klassen in Hierarchien angeordnet werden.

Objektnetze. Objekte können zusammen mit ihren Referenzen auf andere Objekte als Graph aufgefasst werden. Jedes Objekt entspricht dabei einem Knoten des Graphen. Eine Referenz von einem Objekt zu einem anderen wird als eine beschriftete Kante zwischen den zugehörigen Knoten dargestellt. Als Beschriftung der Kante dient hierbei der Name des referenzierenden Attributs. Wir bezeichnen im Folgenden einen solchen Graphen als *Objektnetz*. Abb. 7.3 zeigt ein Objektnetz, das (einen Teil) einer Führungshierarchie (Blue-Force-Hierarchie) modelliert. In dem Objektnetz steht der Knoten „Btl“ für ein Objekt, das ein Bataillon repräsentiert (Attribute sind aus Platzgründen nicht dargestellt). Die Position des Bataillons ist in einem separaten Objekt, welches im Objektnetz als Knoten „absPt59“ dargestellt ist, repräsentiert (man beachte die mit „position“ beschriftete Kante, welche vom Knoten „Btl“ zum Knoten „absPt59“ führt). Der Gefechtsstand des Bataillons wird durch ein weiteres Objekt, welches als Knoten „Btl_CmdPost“ dargestellt ist, repräsentiert (man beachte die Kanten „unit“ und „commandPost“ zwischen den Knoten „Btl“ und „Btl_CmdPost“). Auf die gleiche Weise sind zwei Kompanien, die dem Bataillon unterstellt sind, modelliert. Das Bataillon und die unterstellten Truppenteile sind durch entsprechende Objektreferenzen, die das Unterstellungsverhältnis nachbilden, miteinander verbunden (vgl. die Kanten „commands“ und „underCommandOf“).

Pfade. Ausgehend von einem Objekt ist es möglich, in einem Objektnetz über die Objektreferenzen zu anderen Objekten zu gelangen. Man spricht hierbei von einem *navigierenden Zugriff* auf die Daten. Die Liste der Namen der Referenzen (bzw. Namen von Attributen, die die Referenz als Wert haben) auf diesem Weg bezeichnen

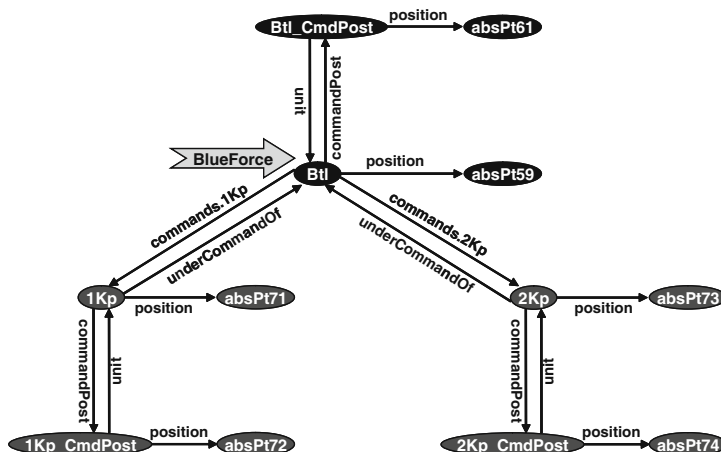


Abb. 7.3 Ein Objektnetz, welches militärische Einheiten repräsentiert

wir als *Pfad*. So erreicht man beispielsweise (vgl. Abb. 7.3) ausgehend vom Objekt „2Kp“ durch das Folgen des Pfades „underCommandOf.commandPost.position“ das Objekt „absPt61“.

Pfadausdrücke. Ein Pfadausdruck ist ein regulärer Ausdruck, der aus

- dem Namen einer Objektreferenz (entspricht einer Kante in einem Objektnetz),
- dem Symbol ε (steht für den leeren Pfad) und
- dem Symbol ? (ist ein Platzhalter für eine beliebige Objektreferenz (Kante im Objektnetz))

mittels Konkatenation (\cdot), Alternative ($+$) und Repetition ($*$) aufgebaut ist. Ein Pfadausdruck spezifiziert eine Menge von Pfaden. Ein Beispiel für einen wohlgeformten Pfadausdruck ist: „commandPost.position + position“.

Die folgenden Begriffe benötigen wir, um unser Replikationsverfahren (vgl. Abschn. 7.3.2) und unsere Datenmanagement-Mechanismen (vgl. Abschn. 7.3.3) zu beschreiben.

Datenkategorien. Für den Zugriff auf ein Objekt der Datenhaltung wird der Objektidentifikator (OID) des Objekts benötigt. Da dieser der Anwendung üblicherweise nicht bekannt ist, bietet sich die Verwendung von mnemonischen (einprägsamen) Bezeichnungen für Objekte an. Wir nennen derartige Bezeichnungen *Datenkategorien*. So ist in Abb. 7.3 beispielsweise eine Datenkategorie „BlueForce“, die für das Objekt „Btl“ steht, dargestellt. Der Begriff Datenkategorie resultiert aus der Idee, dass das Objekt, welches einer Datenkategorie zugeordnet ist, als Wurzelobjekt eines Objektnetzes, das Daten einer speziellen Kategorie enthält, angesehen werden kann. So sind in obigem Beispiel ausgehend von dem Objekt „Btl“ durch das Folgen von Objektreferenzen alle Datenobjekte, die der Kategorie „BlueForce“ angehören, erreichbar.

Durch Angabe einer Datenkategorie und eines Pfadausdrucks kann eine Teilmenge der Datenkategorie spezifiziert werden. Der Pfadausdruck beschreibt hierbei Pfade, die von dem Wurzelobjekt der Datenkategorie ausgehen. So werden beispielsweise in Abb. 7.3 durch Angabe der Datenkategorie „BlueForce“ und des Pfadausdrucks „commandPost.position + position“ die Objekte „absPt61“ und „absPt59“ spezifiziert.

Besitzer. Wir gehen davon aus, dass jedem Objekt ein Besitzer zugeordnet ist. Nur der Besitzer eines Objekts ist befugt, dieses Objekt zu verändern. Besitzer ist üblicherweise die ODB, die das Objekt erzeugt hat.

Klone. Um die Änderung von Objekten, die zu einer ODB repliziert wurden (und damit nicht dieser ODB gehören), zu ermöglichen, sehen wir das „Klonen“ von Objekten vor: Ein Klon eines Objektnetzes (bzw. eines Teils eines Objektnetzes) ist eine Kopie dieses Objektnetzes und ist im Besitz der ODB, die den Klon erstellt hat.

7.3.2 Replikationsverfahren

In diesem Unterabschnitt beschreiben wir unser Replikationsverfahren, welches mit der Zielsetzung entwickelt wurde, den Kommunikationsbedarf, der für die Konsistenthaltung von replizierten Objekten anfällt, möglichst gering zu halten. Eine Möglichkeit, den Kommunikationsbedarf einzuschränken, ist es, zu jeder ODB nur die Objekte zu replizieren, die in dieser ODB benötigt werden. Somit fallen geringere Kommunikationskosten für die Konsistenthaltung der Objekte an. Aus diesem Grund erlauben wir Datenadministratoren vergleichsweise feingranular zu spezifizieren, welche Teilmengen der Objekte zwischen den verschiedenen ODBs repliziert werden sollen. Weiterhin lockern wir die Konsistenzbedingungen, um eine weitere Reduzierung des Kommunikationsbedarfs zu erzielen. Dies erreichen wir durch die Verwendung einer speziellen Strategie für die Aktualisierung von Objekten (*Update-Strategie*). Im Folgenden beschreiben wir die von uns als Update-Strategie verwendete *Primärkopie-Strategie*.

7.3.2.1 Update-Strategie

Ein wesentliches Unterscheidungsmerkmal von Replikationsverfahren ist die verwendete *Update-Strategie*. Diese bestimmt, wie replizierte Objekte (*Kopien* genannt) aktualisiert werden. Allgemein stellt eine Update-Strategie stets einen Kompromiss zwischen der maximalen Verfügbarkeit von Daten und der Datenkonsistenz im Gesamtsystem dar. So können beispielsweise bei dem *Read-One-Write-All-Verfahren* (ROWA-Verfahren; vgl. Beuter & Dadam, 1996) Leseoperationen stets lokal durchgeführt werden, da alle Kopien stets auf dem aktuellen Stand gehalten werden. Dies wird aber durch einen hohen Aufwand für die Konsistenthaltung der Daten erkauft. Jede Schreiboperation muss hier auf allen Kopien gleichzeitig durchgeführt werden. Dies bedeutet, dass bei der Nicht-Erreichbarkeit einer einzelnen Kopie (d. h. Datenhaltung) keine Schreiboperation durchgeführt werden kann.

Eine entgegengesetzte Strategie ist die Verwendung einer zentralen Datenbank. Hier entstehen keine Inkonsistenzen, da Schreiboperationen stets nur auf einer einzigen Kopie durchgeführt werden. Im Gegenzug ist die Verfügbarkeit der Daten sehr eingeschränkt, da Leseoperationen nur durchgeführt werden können, wenn die zentrale Datenbank erreichbar ist. Eine Beschreibung von weiteren Update-Strategien findet sich in (Beuter & Dadam, 1996; vgl. auch Jansen & Spielmann, 2007).

Als Beispiel für ein im militärischen Bereich verwendetes Replikationsverfahren sei der im *Multilateral Interoperability Programme* (MIP; vgl. Kap. 16) verwendete *Data Exchange Mechanism* (DEM; MIP, 2008b) erwähnt. Hierbei handelt es sich um ein Publish-Subscribe-basiertes Replikationsverfahren. Es ist zu beachten, dass dieses Replikationsverfahren nicht für die Verwendung in taktischen Netzen geeignet ist, da bei der Konzeption des DEM ein Datenaustausch zwischen verschiedenen Nationen auf strategischer bzw. operativer Ebene Zielsetzung war. So unterstützt der DEM beispielsweise nur TCP als Transportprotokoll und unterstützt somit kein Broadcast bzw. Multicast. Die Verwendung von Broadcast bzw. Multicast ist jedoch

in taktischen Netzen von großer Bedeutung, da sie den Bandbreitenbedarf bei einer Gruppenkommunikation deutlich reduzieren kann. Eine weitere Einschränkung des DEM ist, dass nur eine Menge von vordefinierten operativen Informationsgruppen (OIGs) abonniert werden kann, was für die taktische Ebene nicht feingranular genug ist.

Das hier vorgestellte Replikationsverfahren verwendet als Update-Strategie eine *Primärkopie-Strategie* (Beuter & Dadam, 1996). Bei dieser Strategie wird jede Änderung eines Objekts stets auf einer ausgezeichneten Kopie (*Primärkopie*) durchgeführt. Die anderen Kopien werden bei einer Änderung der Primärkopie asynchron aktualisiert. Wir gehen davon aus, dass sich jedes Objekt im Besitz einer ausgezeichneten ODB befindet (vgl. Abschn. 7.3.1). Die Primärkopie eines Objekts wird in dieser ODB gespeichert. Schreibenfragen einer Anwendung an die Primärkopie eines Objekts werden direkt von der besitzenden ODB ausgeführt und anschließend asynchron an die ODBs, die ebenfalls Kopien des Objekts halten, propagiert. Durch die Verwendung einer Primärkopie-Strategie werden Inkonsistenzen der Daten bei gleichzeitiger Bearbeitung durch mehrere Nutzer vermieden, da jedes Objekt stets nur in der besitzenden ODB bearbeitet werden kann.

Wir haben diese Update-Strategie gewählt, da sie den Bandbreitenbedarf im Netzwerk reduziert. Lesetransaktionen können stets effizient auf der lokal vorliegenden ODB ohne Netzwerkkommunikation durchgeführt werden. Dies gilt analog für Schreiboperationen auf „eigenen“ Objekten, da diese lokal in der „eigenen“ ODB (als Primärkopien) vorliegen.

Ein Nachteil unserer Update-Strategie ist, dass Objekte zeitweise in einigen ODBs veraltet sein können, da in unserem Ansatz Updates asynchron übertragen werden. Ein weiterer Nachteil ist, dass ein Nutzer keine Objekte bearbeiten kann, die er nicht besitzt. Um dieses Problem zu umgehen, nutzen wir das Konzept des Klonens (vgl. Abschn. 7.3.1). Es bleibt zu untersuchen, ob die beschriebenen Nachteile angesichts der speziellen Einschränkungen taktischer Netze zugunsten einer schnelleren Zugriffsmöglichkeit und besseren Verfügbarkeit akzeptierbar sind.

7.3.2.2 *Subscribe/Download-Verfahren*

Um den Bandbreitenbedarf weiter zu reduzieren, nutzt unser Replikationsverfahren den Umstand aus, dass ein Nutzer üblicherweise nicht an allen Objekten, die im Netzwerk verfügbar sind, interessiert ist.

Wir führen ein spezielles Publish/Subscribe-Verfahren (*Subscribe/Download-Verfahren* genannt) ein. In diesem Verfahren bezeichnen wir dateninteressierte Nutzer als *Subscriber* und Anbieter von Daten als *Publisher*. Ein Subscriber kann eine Teilmenge der Datenobjekte eines Publishers abonnieren. Nach einer Änderung von Objekten beim Publisher werden nur Änderungsinformationen über Objekte, die der Subscriber abonniert hat, asynchron zu seiner ODB repliziert. Ein Publisher von Objekten muss nicht zwangsläufig der Besitzer dieser Objekte sein. Zudem kann jeder Nutzer gleichzeitig als Subscriber und Publisher von Objekten agieren.

Abbildung 7.4 zeigt einen typischen Nachrichtenfluss des Subscribe/Download-Verfahrens. Zuerst abonniert der Subscriber mittels einer *Subscription* (Abonne-

ment) eine Menge von Objekten beim Publisher (siehe Schritt 1 in Abb. 7.4). Wird ein abonniertes Objekt vom Publisher modifiziert, schickt dieser allen ODBs, die dieses Objekt abonniert haben, mittels eines Push-Verfahrens eine Änderungsbenachrichtigung (Schritt 2). Man beachte, dass die Änderungsbenachrichtigung im Vergleich zu einer Aktualisierungsnachricht (Update) sehr klein ist. Anschließend kann der Subscriber bei Bedarf die Änderungen (Update) mittels eines Pull-Verfahrens herunterladen (Schritte 3, 4).

Eine Subscription (vgl. Abb. 7.4, Schritt 1) besteht aus einer Datenkategorie und einem Filter. Der Filter ist ein Pfadausdruck und bestimmt die Teilmenge der Objekte der Datenkategorie, an welcher der Subscriber interessiert ist (vgl. Abschn. 7.3.1). Dadurch können die zu replizierenden Objekte feingranular bestimmt werden, wodurch der Bandbreitenbedarf reduziert werden kann.

Während Updates bei den bekannten Publish/Subscribe-Verfahren üblicherweise mittels eines Push-Verfahrens zu den Subscribern gesendet werden, kombinieren wir die Vorteile eines Push- mit denen eines Pull-Verfahrens. Unser Ansatz hat den Vorteil, dass der Subscriber entscheiden kann, welche Objekte er zu welchem Zeitpunkt herunterladen möchte. Dies erlaubt eine Priorisierung von „wichtigen“ Daten im Falle von eingeschränkten Kommunikationsressourcen. Mechanismen für die Priorisierung von Objekten sind jedoch nicht Gegenstand dieses Beitrags. Zur weiteren Reduzierung der über das Netzwerk zu übertragenden Daten sollte ein Publisher einen Subscriber über Änderungen eines Objekts nur einmal informieren, auch wenn es in der Zwischenzeit mehrfach geändert wurde. Somit wird ein Subscriber, der momentan an einem Objekt nicht interessiert ist, nicht laufend über die Änderung dieses Objekts informiert. Wenn der Subscriber zu einem späteren Zeitpunkt wieder ein Update des Objekts anfragt, erhält er nur die Änderungsinformationen, die er benötigt, um den aktuellen Zustand des Objekts zu erhalten.

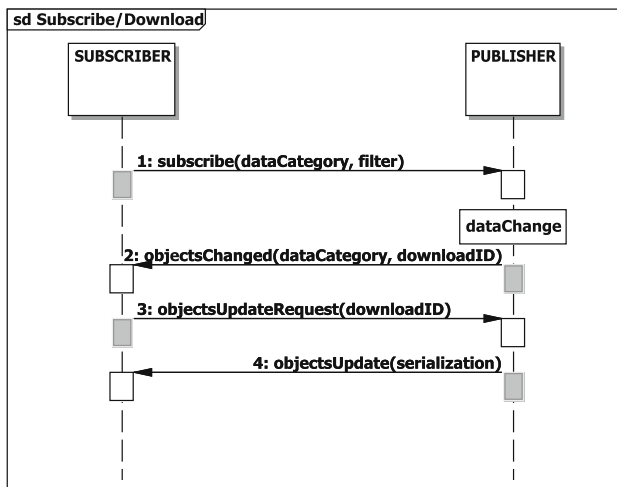


Abb. 7.4 Subscribe/Download-Verfahren (UML-Sequenzdiagramm)

7.3.3 Datenmanagement

Während unser Replikationsverfahren eine flexible, technische Basis für die Replikation von Datenobjekten bildet, bleibt zu spezifizieren, welche Objekte für welche Nutzer (beispielsweise Truppenteile) relevant sind und welche Objekte zwischen welchen ODBs repliziert werden sollen. In diesem Unterabschnitt stellen wir erste Ansätze zur Entwicklung von Datenmanagement-Mechanismen vor, mit deren Hilfe die Initialisierung des Systems und die Konfiguration der Verteilung der Daten (Replikation) zwischen den ODBs gesteuert werden kann.

Ziel dieser Mechanismen ist es, die Konfiguration des Systems zu vereinfachen und es Datenadministratoren zu ermöglichen, die Verteilung der Datenobjekte schnell an geänderte Situationen anzupassen. Dies ist insbesondere bei der Vernetzten Operationsführung (NetOpFü), die zu schnelleren Führungs- und Entscheidungsprozessen und damit zu einer höheren Dynamik der Informationsbedarfe der einzelnen Truppenteile führt, wichtig.

7.3.3.1 Konfiguration und Initialisierung

Grundlage der Konfiguration des Systems bilden *Konfigurationsregeln*, die es ermöglichen, die initiale Verteilung der Daten zu beschreiben und die Replikation zwischen den ODBs zu konfigurieren. Diese Konfigurationsregeln sind durch Datenadministratoren in Vorbereitung eines Einsatzes des Systems zu erstellen. Dem eigentlichen Einsatz des Systems (Laufzeitphase) ist eine Initialisierungsphase vorgeschaltet. Die Initialisierungsphase wird durch eine Initialisierungskomponente (vgl. Abb. 7.2) gesteuert. Sie nutzt für die Steuerung der Initialisierung eine spezielle Schnittstelle der ODB. Im ersten Schritt der Initialisierung überträgt die Initialisierungskomponente initiale Grunddaten und eine Menge von Konfigurationsregeln zu den einzelnen ODBs. Anschließend wertet jede ODB ihre Konfigurationsregeln aus und konfiguriert den Replikationsmechanismus anhand dieser Regeln. In der Laufzeitphase werden Daten zwischen den ODBs der verschiedenen Nutzer repliziert.

7.3.3.2 Konfigurationsregeln

Die Konfigurationsregeln beschreiben den Informationsbedarf der einzelnen Truppenteile. Hierfür beschreibt eine Regel für einen Nutzer (beispielsweise Truppenteil) und für verschiedene operative Situationen, welche Informationen für den Nutzer in der jeweiligen Situation relevant sind. Dies ermöglicht es Datenadministratoren, die Informationsbedarfe von Truppenteilen einfach (insbesondere ohne Programmcodeanpassungen) festzulegen. Weiterhin können Konfigurationsregeln schnell angepasst werden, wenn neue operative Situationen entstehen.

Für jede ODB wird ein Regelsatz für die Initialisierungsphase und ein Regelsatz für die Laufzeitphase des Systems verwaltet. Bei einem Regelsatz handelt es sich um eine Sequenz von Konfigurationsregeln. In einem ersten Schritt werden Konfigurationsregeln entworfen, welche beschreiben,

- von welchen ODBs welche Daten zur eigenen ODB zu replizieren sind,
- wie replizierte Daten zu manipulieren sind (z. B. mittels Aggregation) bevor diese in die eigene ODB integriert werden und
- wie replizierte und ggf. manipulierte Daten in die eigene ODB zu integrieren sind.

Im Folgenden stellen wir eine Auswahl von Konfigurationsregeln vor (eine umfassendere Beschreibung findet sich in Jansen & Spielmann, 2007).

Eine *SUBSCRIBE-Regel* beschreibt, welche Datenobjekte von welchen ODBs repliziert werden sollen. Eine solche Regel ist wie folgt aufgebaut:

DestCategory = SUBSCRIBE (*Publisher*, *Category*, *Filter*),

wobei die vier Bezeichner *DestCategory*, *Publisher*, *Category* und *Filter* die folgende Bedeutung haben:

- *Publisher* ist eine Beschreibung der ODBs, von denen Daten bezogen werden sollen,
- *Category* beschreibt die Datenkategorie, aus der Daten bezogen werden sollen,
- *Filter* spezifiziert die Objekte der Datenkategorie, die für den Nutzer relevant sind,
- *DestCategory* gibt die Datenkategorie an, unter der die replizierten Datenobjekte in der eigenen ODB abgelegt werden sollen.

Als Beispiel betrachte man Abb. 7.5, in der zwei Truppenteile (ein Bataillon und eine unterstellte Kompanie) in Form von Datenobjekten dargestellt sind. Jeder Truppenteil besitzt eine eigene ODB, welche die lagebildbezogenen Daten dieses Truppenteils verwaltet. In der Initialisierungsphase soll die Blue-Force-Hierarchie erzeugt und in den ODBs der Truppenteile gespeichert werden. Zunächst ist nur die Repräsentation des eigenen Truppenteils (und des zugehörigen Gefechtsstands) in der entsprechenden ODB gespeichert (siehe Abb. 7.5). Der eigene Truppenteil wird jeweils durch eine Datenkategorie „BlueForce“ referenziert.

Nehmen wir an, dass folgende Regel von der ODB des Bataillons ausgeführt wird, um Objekte von der Kompanie zum Bataillon zu replizieren:

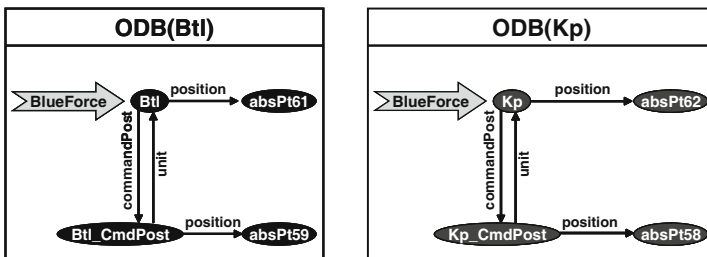


Abb. 7.5 Ausführung einer SUBSCRIBE-Regel (Anfangssituation). Anfangs sind in den ODBs von Bataillon und Kompanie nur die eigenen Truppenteile repräsentiert

BlueForceChild = SUBSCRIBE (ODB(Kp),
 BlueForce,
 $\epsilon + position$).

Die Regel beschreibt, dass das Bataillon an allen Objekten interessiert ist, die sich ausgehend vom Wurzelobjekt der Kategorie „BlueForce“ durch das Folgen von Pfaden, die mit dem Pfadausdruck „ $\epsilon + position$ “ spezifiziert werden, erreichen lassen. Da der leere Pfad „ ϵ “ für das Wurzelobjekt der Kategorie („Kp“) steht und die Referenz „position“ zu dem Objekt „absPt62“ führt, welches die Position der Kompanie repräsentiert, werden diese beiden Objekte zu dem Bataillon repliziert und dort in der Kategorie „BlueForceChild“ abgelegt (siehe Abb. 7.6). Jede Positionsänderung der Kompanie, die in der ODB der Kompanie durchgeführt wird, wird nun mittels Subscribe/Download-Verfahren zu dem Bataillon übertragen. Somit ist das Bataillon stets über die aktuelle Position der Kompanie informiert.

Eine weiterer Typ von Konfigurationsregeln ist die *MODIFY-Regel*. Eine solche Regel beschreibt, wie Datenobjekte automatisiert modifiziert werden sollen und ist wie folgt aufgebaut:

MODIFY (*Trigger*, *Operation*, *DestCategory*),

wobei die Bezeichner *Trigger*, *Operation* und *DestCategory* die folgende Bedeutung haben:

- *Trigger* hat die Form $\{Category_1, ..., Category_n\}$, wobei $Category_i$ eine Datenkategorie bezeichnet,
- *Operation* bezeichnet eine Methode und
- *DestCategory* bezeichnet eine Datenkategorie.

Die Semantik dieser Regel ist wie folgt: Wenn ein Datenobjekt einer der angegebenen Datenkategorien ($Category_1, ..., Category_n$) in der eigenen ODB geändert wird, wird die angegebene Operation ausgeführt und das Ergebnis dieser Operation unter der Datenkategorie *DestCategory* abgelegt.

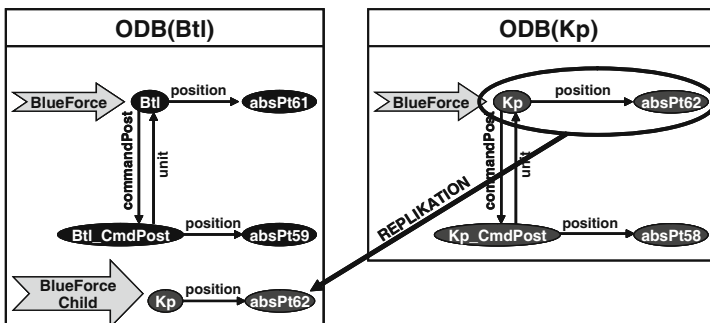


Abb. 7.6 Ausführung einer SUBSCRIBE-Regel (Fortsetzung). Die Objekte, die die Kompanie repräsentieren, werden von der ODB der Kompanie zur ODB des Bataillons repliziert

Als Beispiel für die Anwendung einer MODIFY-Regel betrachten wir die in Abb. 7.7 dargestellte Situation. In der ODB eines Bataillons ist (analog zum vorherigen Beispiel) die Blue-Force-Hierarchie repräsentiert. Die Datenkategorie „Blue-Force“ verweist auf den eigenen Truppenteil. „BlueForceChild0“ und „BlueForceChild1“ sind technische Hilfskategorien, die auf zwei Truppenteile (Kompanien), die dem Bataillon unterstellt sind, verweisen.

Wir betrachten folgende MODIFY-Regel:

MODIFY ({BlueForceChild0, BlueForceChild1},
 geometricMean,
 BlueForce)

Die Regel beschreibt die Berechnung der Position des Bataillons durch Aggregation der Positionen der unterstellten Truppenteile. Bei jeder Änderung eines Objekts einer der angegebenen Kategorien „BlueForceChild0“ oder „BlueForceChild1“ wird die Operation „geometricMean“ ausgeführt. Die einzelnen Schritte der Ausführung der MODIFY-Regel sind in Abb. 7.8 dargestellt. Ausgelöst wird die MODIFY-Regel in diesem Beispiel durch eine replizierte Änderung der Position der 2. Kompanie (Objekt „absPt73“, vgl. Schritt 1 in Abb. 7.8). Dies führt zur Ausführung der Operation „geometricMean“. Diese liest die Positionen der unterstellten Truppenteile (Objekte „absPt73“ und „absPt71“, vgl. Schritte 2, 3), berechnet aus

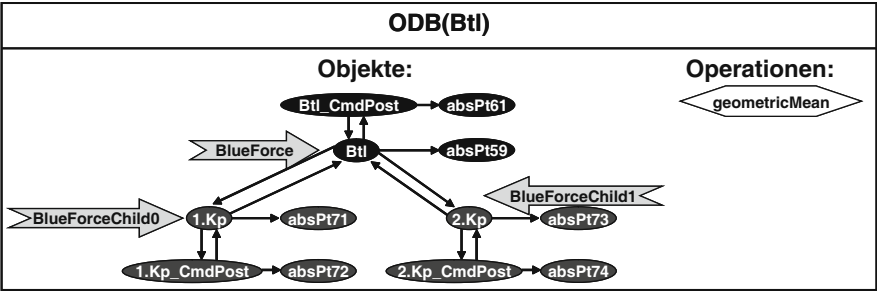


Abb. 7.7 Beispiel der Ausführung einer MODIFY-Regel (Anfangssituation)

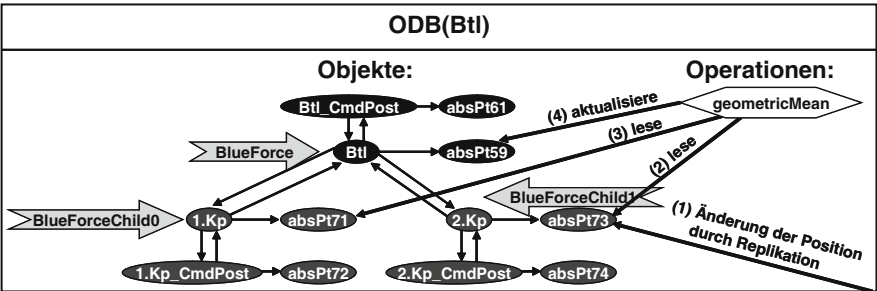


Abb. 7.8 Beispiel der Ausführung einer MODIFY-Regel (Fortsetzung)

diesen das geometrische Mittel und schreibt das Ergebnis in das Objekt „absPt59“, welches die Position des Bataillons repräsentiert (Schritt 4).

7.4 Zusammenfassung und Ausblick

Aufgrund der besonderen Einschränkungen militärischer Netzwerke (insbesondere auf der taktischen Ebene) sind beim Entwurf zukünftiger, verteilter FüInfoSys spezielle Lösungsansätze notwendig. In diesem Beitrag wurde ein Modell eines objektorientierten, verteilten Datenbanksystems vorgestellt, in dem Datenobjekte teilerredundant in lokalen Objektdatenbanken (ODBs) gespeichert werden. Jedem Nutzer (z. B. Person, Truppenteil, Rolle oder Organisationseinheit) wird dazu eine eigene ODB zugeordnet. Diese ODB verwaltet die Datenobjekte, die für diesen Nutzer relevant sind. Das Modell umfasst ein Replikationsverfahren (Subscribe/Download-Verfahren), welches an die besonderen Anforderungen taktischer Netze angepasst ist. Das Replikationsverfahren erlaubt es, feingranular zu spezifizieren, welche Datenobjekte zwischen welchen ODBs repliziert werden sollen. Somit reduziert das Replikationsverfahren die Kommunikationskosten für den Zugriff auf Objekte und für die Synchronisierung der Objekte. Das Modell enthält darüber hinaus Datenmanagement-Mechanismen zur Konfiguration der ODBs. Die Mechanismen ermöglichen es Datenadministratoren mittels Konfigurationsregeln festzulegen, wie das System initialisiert wird und welche Datenobjekte zur Laufzeit zwischen welchen ODBs repliziert werden. Exemplarisch haben wir eine Auswahl von Konfigurationsregeln vorgestellt.

Ausblick. Als Erweiterung der hier vorgestellten Konzepte bietet sich die Entwicklung einer formalen Sprache für die Konfiguration des Systems an. Diese Sprache muss einerseits ausdrucksmächtig genug sein, um die Informationsbedarfe von Truppenteilen für verschiedene Auftragssituationen beschreiben zu können und andererseits hinreichend einfach sein, um es Datenadministratoren zu ermöglichen, das Systemverhalten schnell an neue operative Bedürfnisse anzupassen. Eine sinnvolle Erweiterung der hier vorgestellten Regelsprache ist die Unterstützung von bedingten Konfigurationsregeln. Hierbei wird jede Regel mit einer Bedingung versehen, welche angibt, ob die Regel in der aktuellen Situation gültig ist. Dadurch wäre es möglich, den Informationsbedarf eines Truppenteils für verschiedene operative Situationen zu beschreiben. Das FüInfoSys könnte demnach die Replikation automatisch anpassen, wenn antizipierte Änderungen dynamischer Parameter (wie der operativen Situation oder des aktuellen Interessengebiets (area of interest)) stattfinden.

Literaturverzeichnis

Beuter, T. & Dadam, P. (1996). Prinzipien der Replikationskontrolle in verteilten Datenbanksystemen. *Informatik – Forschung und Entwicklung*, 11(4), 203–212.

- BMVg (2006). Teilkonzeption Vernetzte Operationsführung.
- Cattell, R. G. G. & Barry, D. K. (2000). *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann Publishers.
- Jansen, N. & Spielmann, M. (2007). *Konzeption eines verteilten Datenhaltungs- und Datenmanagement-Systems für lagerelevante Daten in FüInfoSys*. FKIE-Bericht Nr. 142, Forschungsgesellschaft für Angewandte Naturwissenschaften e.V., Wachtberg.
- MIP – Multilateral Interoperability Programme (2008b). MIP Technical Interface Design Plan (MTIDP), Edition 3.8. <http://www.mip-site.org>.

Verteilte Führungsinformationssysteme

Wunder, M.; Grosche, J. (Hrsg.)

2009, XXII, 319 S., Hardcover

ISBN: 978-3-642-00508-4