Smalltalk – zum aktuellen Stand der ersten objektorientierten Programmiersprache

Partytime

Frank Müller

Ein Vierteljahrhundert ist seit Smallltalk-80 ins Land gegangen. Vielen gilt die Sprache als erste rein objektorientierte, andere sehen sie als abgehobenes Nischenprodukt jenseits des harten Programmiereralltags. Ungeachtet aller Vorurteile finden sich in einer von Java und C++ dominierten IT-Welt noch immer diverse Smalltalk-Anwendungen.

Zweifelsohne heißen die heute dominierenden Programmiersprachen Java, C# und C++. Im Bereich kleinerer Aufgaben machen diesen statisch typisierten Sprachen jedoch dynamisch typisierte Scriptsprachen wie Ruby oder Python Konkurrenz. All diese Sprache haben aber eine gemeinsame Wurzel, die heute etwas in den Hintergrund gerückt ist: Smalltalk.

Smalltalk gilt als erste rein objektorientierte Sprache, auch wenn die Objektorientierung selbst durch Ole-Johan Dahl mit Simula 67 in die Welt der Softwareentwicklung Einzug gehalten hat. Diese Simulationssprache kannte zwar bereits Klassen und Objekte, konnte den Begriff der objektorientierten Programmierung – kurz OOP – jedoch nicht so prägen wie Smalltalk.

Deren Entwicklung begann 1972 am Palo Alto Research Center von Xerox als Teil des Projekts Dynabook, geleitet von Adele Goldberg. Ziel dieses Projekts war keine neue Programmiersprache, vielmehr wollte man neue Konzepte für die Interaktion zwischen Mensch und Computer erarbeiten. Die Ergebnisse waren ihrer Zeit weit voraus. Neben der Sprache Smalltalk erarbeitete das Projektteam die Ideen für grafische Oberflächen, Pictogramme, Multimedia und die Bedienung mit der Maus. Die treibenden Kräfte hinter diesen Konzepten waren Alan Kay – Turing-Preisträger 2003 und Kyoto-Preisträger 2004 – sowie Dan Ingalls, der spätere Chefentwickler von Smalltalk 80.

Ein Smalltalk-System zeichnet sich aus durch seine einfache Sprache, die klare Struktur, eine umfassende Klassenbibliothek und die integrierte Entwicklungsumgebung. All dies findet sich in einer einzelnen Datei wieder, dem Image, das die virtuelle Maschine ausführt. Durch diese Abstraktion war Smalltalk schon früh portabel. Bis auf wenige direkte Aufrufe an die virtuelle Maschine liegt das gesamte System inklusive der Entwicklungsumgebung als Quelltext vor und ist dadurch an eigene Bedürfnisse anpassbar.

Über verschiedene Browser kann der Entwickler seinen Quellcode durchsuchen und bearbeiten. Eine Navigation über Pakete, Klassen, Protokolle und Methoden erleichtert die Arbeit. Im Editor bearbeitet er die Klassendeklaration oder einzelne Methoden, eine Aufteilung in Dateien findet nicht statt. Geänderter oder hinzugefügter Code einer Klassendeklaration oder einer Methode wandelt die Entwicklungsumgebung beim Speichern unmittelbar in Bytecode um (Just in Time – JIT).

Inspektoren zur Anzeige der Werte von Instanzvariablen sowie ein Debugger, der die interaktive Veränderung des Quelltexts inklusive eines Wiederaufsetzens im Stack erlaubt, runden die Entwicklungsumgebung ab. Da Unit Testing und Refactoring in Smalltalk ihre Wiege haben, werden heute praktisch alle Distributionen mit entsprechenden Werkzeugen ausgeliefert. Außerdem sind eigene Software-Configuration-Management-Lösungen und Anbindungen an externe Systeme Teil des Lieferumfangs.

Einfachheit ist oberstes Prinzip

Das Konzept von Smalltalk sieht eine einfach verständliche Sprache vor, Maßstab war die Nutzung durch Kinder. Inspiriert durch die Idee der Objektorientierung sowie die strukturelle Einfachheit von Lisp entwickelten sich sechs Hauptkonzepte:

- Alles ist ein Objekt.
- Objekte kommunizieren durch den Versand und Empfang von Nachrichten in Form von Objekten.
- Objekte haben ihren eigenen Speicher.
- Jedes Objekt ist eine Instanz einer Klasse, die wiederum ein Objekt sein muss.
- Die Klasse verwaltet das Verhalten aller ihrer Instanzen in Form von Objekten in einer Programmliste.
- Zur Auswertung einer Programmliste übergibt die virtuelle Maschine die Kontrolle dem ersten Objekt, den Rest behandelt sie als dessen Nachricht.

Für die Umsetzung dieser Konzepte kommt Smalltalk mit nur fünf Schlüsselwörtern und drei ausführbaren Konstrukten aus. Von den Letzteren ist eins das genannte Senden von Nachrichten an Objekte. Die Zuweisung eines Objekts an eine Variable und die Rückgabe eines Objekts als Ergebnis einer Methode sind die anderen beiden. Alles Weitere bildet sich aus dieser überschaubaren Menge.

So stellen Objekte und Methoden sogar Kontrollkonstrukte zur Verfügung. Das immer wieder notwendige *if* für Verzweigungen beispielsweise liefert die Methode *ifTrue:* der Klassen *True* und *False*. Die Methode bekommt einen Codeblock – ebenfalls ein Objekt in einer einfachen Notation – als Argument übergeben. Allerdings führt nur *True* diesen Block aus, *False* ignoriert ihn. Für *else* gilt das Umgekehrte mit *ifFalse:* als Methode. So lautet zum Beispiel im Falle eines Vergleichs mit der Methode = der Rückgabewert je nach Ergebnis *true* als Instanz der Klasse *True* oder *false* als Instanz der Klasse *False*. Diese erhalten dann die Nachricht *ifTrue:* oder *ifFalse:*.

```
(x = 4711)
    ifTrue: [Transcript show: 'Ja!']
    ifFalse: [Transcript show: 'Nein!']
```

Für Umsteiger ist dieses Prinzip gewöhnungsbedürftig, Neulinge hingegen kommen schnell damit zurecht, da es durchgängig und einfach ist. Die Blöcke in den eckigen Klammern repräsentieren anonyme Objekte vom Typ *BlockContext*. Erst die Nachricht *value* führt sie aus. Dies ist auch mit Argumenten möglich und erlaubt so leistungsfähige Konstrukte. Im Zusammenhang mit dem mächtigen Collection Framework zeigt sich dies besonders deutlich.

```
customer allOrders do: [:each |
    Transcript
        show: 'Auftragsnummer: ';
        show: each id]
```

In diesem Fall fragt ein Kunde alle Aufträge ab. Dieser Menge übergibt das Programm den Block zur Anzeige der Nummer jedes einzelnen Auftrags. Die Methode *do:* iteriert über alle Objekte der Collection und sendet dem übergebenen Block die Nachricht *value:* mit dem Objekt als Argument.

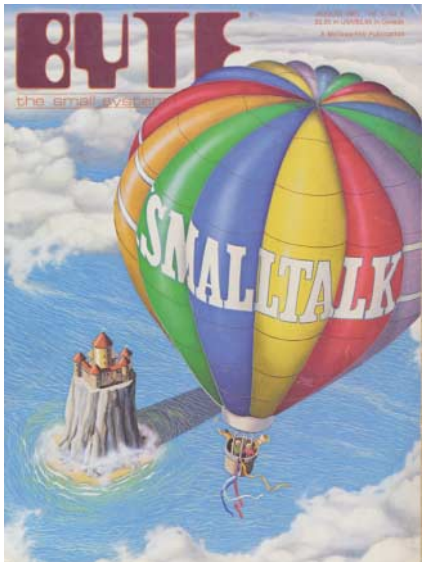
iX-TRACT

- Smalltalk gilt nicht nur als erste rein objektorientierte Programmiersprache, sondern hat auch Konzepte wie grafische Oberfläche eingeführt, die ihrer Zeit weit voraus waren.
- Das primäre Konzept von Smalltalk ist, dass alles ein Objekt ist und die Kommunikation zwischen Objekten durch Versand und Empfang von Nachrichten erfolgt.
- Im Vergleich zu Java, C++ oder C# führt Smalltalk ein Nischendasein, wird aber in vielen komplexen Anwendungssystemen in Banken, Versicherungen oder Produktionsbetrieben eingesetzt.

Wichtiger Faktor für die Machbarkeit derartiger Konstrukte ist die dynamische Typisierung. So kann man jedem Objekt jede Nachricht senden. Ein explizites Class Casting ist dafür nicht notwendig. Versteht ein Objekt

eine Nachricht nicht, führt das zur Ausführung der Methode *doesNotUnderstand:*. So kann die Anwendung oder das System in jedem Fall kontrolliert reagieren. Verschiedene Klassen nutzen dieses Verhalten explizit, zum Beispiel für Proxies. Andernfalls erfolgt jedoch eine saubere Unterbrechung mit der Möglichkeit, in den Debugger zu verzweigen. Da der Stack und seine Zustände ebenfalls Objekte sind, kann der Entwickler hier navigieren, Objektzustände inspizieren und verändern sowie bei Bedarf den Code im Stack modifizieren und den Programmablauf wieder aufnehmen. Dies ermöglicht kurze Entwicklungszyklen und die inkrementelle Programmierung stabiler Anwendungen.

Eine weitere für Umsteiger gewöhnungsbedürftige Eigenschaft ist die Arbeit in einem Image. Der Entwickler arbeitet so nicht mit einzelnen Quelldateien, die in den Binärcode für die Zielplattform übersetzt werden müssen. Vielmehr navigiert er mittels Browsern über die Klassendeklaration oder einzelne Methoden. Neu hinzugefügte oder geänderte Codeteile stehen dem System sofort zur Verfügung. Dieses Browsen über den Code ist mit der dateibasierten Arbeit nicht vergleichbar, geht jedoch schnell in Fleisch und Blut über. Eclipse hat mit der Ansicht „Java durchsuchen“ dieses Verhalten übernommen. Der Sinn erschließt sich bei intensiver Nutzung, wenn dem Entwickler deutlich wird, dass der Fokus in Smalltalk nicht auf dem Code in den Methoden, sondern auf den Klassenstrukturen, den Protokollen zwischen den Klassen sowie der sprechenden Namensgebung liegt. Ergebnis: ein wartbares und robustes System.



Man könnte meinen, das sei der Durchbruch gewesen: 1981 widmete die Byte Smalltalk-80 ein ganzes Heft.

Da das Image den gesamten Quellcode des Systems enthält, ist nicht nur das eigene Projekt sichtbar. Der Entwickler kann durch alle Klassen des Systems blättern und so deren Fähigkeiten ausloten. Zudem kann er von den Erfahrungen anderer Entwickler profitieren und deren Klassen an seine eigenen Bedürfnisse anpassen. Die Change Sets zum Austausch zwischen Images enthalten diese Anpassungen, sodass sie nicht manuell nachgezogen werden müssen. Insgesamt erhält der Entwickler so eine Umgebung, die ihm eine schnelle Einarbeitung, eine einfache Syntax, eine leistungsstarke Bibliothek und eine flexible Entwicklungsumgebung bietet. Alles ist auf eine schnelle und zielgerichtete Umsetzung von Anforderungen ausgelegt. Georg Heeg, der 1987 das gleichnamige Unternehmen mit Fokus auf Smalltalk gegründet hat, bringt es auf den Punkt: „In Smalltalk programming is modeling.“

Trotz ihres Alters und ihrer Stabilität ist die Entwicklung der Programmiersprache nicht stehen geblieben. Mit Traits findet ein neues Konzept Einzug. Traits ist eine Form der Komposition von mehreren Methoden mit Klassen, die voneinander unabhängige Klassen einfach um ein gemeinsames Protokoll erweitert. Dabei bleibt das Prinzip der Einfachvererbung bestehen, und die Methoden, die sich ihrerseits wieder auf Methoden der aufnehmenden Klassen stützen, werden in diese eingewoben.

Kommerzielle und freie Distributionen

Die mangelnde Präsenz von Smalltalk in der Presse lässt vermuten, dass es nur wenige Anbieter gibt. Das Gegenteil ist der Fall. Allerdings sind die Smalltalk-Distributionen nicht so zentral durch nur einen oder wenige Hersteller bestimmt wie bei Java, C#, Ruby oder Python. Vielmehr gibt es kommerzielle und freie Lösungen, teilweise plattformübergreifend, teilweise nur für ein Betriebssystem erhältlich.

Marktführer auf dem kommerziellen Sektor ist Cincom mit VisualWorks, verfügbar für Windows, Mac OS X, Linux und diverse Unix-Varianten, sowie ObjectStudio, das nur in einer Windows-Version vorliegt. Dass dieser Hersteller zwei Systeme anbietet, liegt daran, dass er ObjectStudio von dessen Entwickler Easel gekauft hat. VisualWorks legt den Schwerpunkt auf kommerzielle Multi-Plattform-Applikationen, auch im Bereich Intranet oder E-Business. Neben einer Vielzahl gängiger Internetprotokolle, inklusive SSL, umfasst VisualWorks einen eigenen Application Server. Dieser bietet dem Entwickler einen Rahmen wie Servlets, JSP und ASP. Eine Systemverwaltung über SNMP ist ebenfalls vorgesehen. Die Möglichkeiten für verteilte Anwendungen, die Integration von Web Services und die Anbindung von Datenbanken wie dem MS SQL Server, Oracle, Sybase, DB2 und PostgresSQL runden das System ab. Mittels Store unterstützt es die verteilte Entwicklung an unterschiedlichen Standorten. ObjectStudio zeigt sich zwar ähnlich leistungsfähig, hat seinen Fokus jedoch eindeutig als Client in der Microsoft-Welt.

Einer der größten Anbieter war jahrelang IBM mit Visual Age for Smalltalk. Inzwischen hat Instantiations diese Software übernommen und bietet sie sls VASmalltalk7 in aktualisierter Form an. Verfügbare Plattformen sind Windows, AIX, Solaris und Linux. Neben der bekannt leistungsfähigen Basis erhält der Nutzer Erweiterungen der Entwicklungsumgebung sowie Frameworks für Widgets und die Generierung und Ausgabe von Grafiken.

Gemstone/S von Gemstone Systems ist ein Smalltalk Application Server, der auch Java- und CORBA-Anwendungen als Clients akzeptiert. Mittels des GemBuilder for Smalltalk lassen sich VisualWorks und Visual Age for Smalltalk als Clients einbinden. Ihnen steht damit ein zentraler Objektserver zur Verfügung. Durch einen leistungsfähigen Replikationsmechanismus können die Clientanwendungen mit der entsprechenden Performanz auf lokalen Objekten arbeiten. Eine transaktionsgesicherte Synchronisation gleicht die Änderungen transparent mit dem Server ab.

Vom deutschen Anbieter Exept Software stammt Smalltalk/X, das für eine Vielzahl von Unix-Versionen sowie Windows verfügbar ist. Neben der ANSI-Kompatibilität und der in Smalltalk üblichen umfangreichen Klassenbibliothek bietet Smalltalk/X die Erzeugung nativen Maschinencodes, die direkte Einbindung von C-Code, die flexible Steuerung von Prozessen und Threads, ein flexibles GUI-Handling inklusive der Unterstützung mehrerer

Online-Quellen

Smalltalk-Umgebungen	www.whysmalltalk.com/smalltalk_environments/
Smalltalk im Einsatz	www.whysmalltalk.com/production/
Unternehmen mit Smalltalk	www.stic.org/companies/companies.htm

Bildschirme sowie Anpassungen an lokale Besonderheiten.

Ein neuer Vertreter auf dem Smalltalk-Markt ist Ambrai mit einer Distribution speziell für Mac OS X. Dieses System ist noch nicht weit entwickelt, dafür jedoch gut in das Betriebssystem integriert. Es nutzt das native Mac OS X Framework für die Benutzerschnittstelle, sodass sich die Applikationen nicht von denen anderer Sprachen unterscheiden. Gleichzeitig hat Ambrai Zugriff auf die Bibliotheken und Frameworks.

Ähnlich gut integriert ist Dolphin von Object Arts, allerdings in Windows. Verschiedene Editionen decken die Ansprüche von Einsteigern bis zur professionellen Software-Entwicklung ab. Der Zugriff auf ADODB-Quellen ist ebenso möglich wie ActiveX-Scripting, man kann COM-Server implementieren und Executables erzeugen.

Dies sind nicht die einzigen kommerziellen Distributionen, es gibt weitere für Windows und .Net und Linux. Daneben existieren freie Systeme. Eins ist GNU Smalltalk, ein schlankes Paket mit Fokus auf Programme und Scripts auf Unix-Servern.

Komplexe Anwendungen sind schnell

Hauptvertreter der freien Smalltalks ist Squeak von Alan Kay, das er für die Entwicklung neuer Konzepte zur Ausbildung in der Entwicklung konzipiert und selbst in Smalltalk implementiert hat. Squeak generiert C-Quellen für die Virtual Machine und ist so portabel. Es steht für Mac OS X, Windows, Linux, Unix, Acorn RISC OS sowie diverse PDAs zur Verfügung. Die aktive Entwicklergemeinde hat nicht nur SUnit und den Refactoring-Browser integriert, sondern auch eine Vielzahl von Bibliotheken für die Unterstützung von Datenbanken – relational und objektorientiert – sowie für Netzwerke, grafische Applikationen und Multimedia. Nicht zuletzt ist Squeak die Basis für Traits oder Umgebungen wie Open Croquet.

Traditionell findet Smalltalk Einsatz in der Produktion sowie in Banken und Versicherungen, vor allem, weil sich damit komplexe Anwendungssysteme performant und stabil realisieren lassen. So steuert Smalltalk die Chipproduktion von AMD in Dresden, bei Texas Instruments die Waferbearbeitung. Fuji Research automatisiert Herstellungsmechanismen mit Hilfe von Smalltalk, VW steuert damit die Produktion des New Beetle, und auch Honda und Mitsubishi nutzen die Sprache. Im Finanzbereich setzen beispielsweise die Bayerische Landesbank, die Landesbank Baden-Württemberg, die Bayerische Vereinsbank und die Graubündner Kantonalbank auf Smalltalk. Daimler Chrysler nutzt die OO-Sprache zur Verwaltung seiner Finanzen, und bei Ericsson Radio Systems bildet sie die Basis des Bestell- und Auf-

tragssystems für die Lieferanten. Die Deutsche Bahn koordiniert die Züge über eine Smalltalk-Anwendung, Federal Express die Lieferungen. Diese und viele weitere Beispiele zeigen, wie robust und performant sich dieses ausgereifte Entwicklungssystem im täglichen Einsatz präsentiert.

Entwickler schwören auf Smalltalk

Eine Vielzahl ehemaliger Smalltalk-Entwickler ist heute zu Java oder .Net gewechselt, da es Vorgaben seitens der Projektleitung oder der Auftraggeber so verlangen. Nahezu alle schwärmen jedoch von der Nutzung Smalltalks und würden sich sofort ein Projekt auf dieser Basis wünschen. Auf die Gründe hierfür angesprochen, nennen sie die Eleganz und Einfachheit der Programmierung selbst bei komplexen Aufgabenstellungen. Neuentwicklungen sind dank der hohen Produktivität schnell, die Wartung aufgrund eines großen Wiederverwendungsgrads und der leistungsfähigen Entwicklungsumgebung einfach. Diese Meinung teilen jedoch Projektleitung und Auftraggeber nicht immer.

Immerhin folgen nicht alle Unternehmen den Massentrends. Einige erhalten sich die Vorteile Smalltalks und vertreten ihre Entscheidung auch gegenüber ihren Kunden. So ist ihre „Liebe zur Sprache“ einer der Auslöser für Sabine Knöfel, Gesellschafterin der Classware GmbH und Anbieterin der ASP-Software HRworks, Smalltalk einzusetzen. Sie schätzt die Möglichkeit, neue Funktionen schnell und iterativ mit dem Kunden entwickeln zu können. Die durchgängige Objektorientierung im Zusammenhang mit einer objektorientierten Datenbank ermöglicht bei Classware eine fast nahtlose Integration von Daten und Logik.

Diese Begeisterung teilt Hans-Dieter Brenner von der Novatec GmbH, der Smalltalk während des Studiums kennen gelernt hat. Novatec bietet individuelle Lösungen in C, C++, Java, Smalltalk und C# an. Matthias Seeliger von der CSC hat sich bereits zu Zeiten der Firma Telenet für Smalltalk entschieden. Java war seinerzeit noch nicht verfügbar. Inzwischen hat CSC Telenet gekauft, setzt aber weiterhin Smalltalk ein.

Allerdings sieht Sabine Knöfel auch Nachteile. Diese sind jedoch nicht in der Sprache selbst begründet, sondern in herrschenden Vorbehalten ihr gegenüber. Dabei kommen diese seltener von den Fachabteilungen, deren Fokus die Umsetzung der Anforderungen ist, sondern von Seiten der Technik oder dem Management. Hans-Dieter Brenner hat diese Erfahrung ebenfalls gemacht. Smalltalk gilt als Randerscheinung und Investitionsrisiko, die Kunden folgen dem Java-Hype. Für Matthias Seeliger kommt noch die Ungewissheit bezüglich der Distributionen hinzu. So war lange nicht klar, wie es mit IBMs Visual Age weitergeht. Zudem wird die Community kleiner, was die Nachwuchssituation verschlechtert. Alle drei Unternehmen bilden daher selbst aus, teilweise durch eine Umschulung von Java-Entwicklern.

Ebenfalls allen drei gemein ist die durchweg positive Erfahrung in Bezug auf Performanz und Stabilität. Hier macht sich die lange Reifezeit der Software bemerkbar. Der Einsatz im kritischen 7 × 24-Stunden-Betrieb ist problemlos, Hotfixes sind bei Bedarf schnell umgesetzt, da keine aufwendigen Build- und Deploy-Prozesse notwendig sind. Die Inbetriebnahme dieser Fixes oder neuer Versionen geht entsprechend schnell. Über das Versionsmanagement ist jeder Schritt direkt nachvollziehbar, was zusammen mit der Möglichkeit, zur Laufzeit am Image zu arbeiten, auch Operationen „am offenen Herzen“ erlaubt. Dieses Vorgehen ist allerdings nur bei angepasster Qualitätssicherung gangbar und sollte sonst vermieden werden.

Smalltalk-Distributionen

Anbieter	Distribution	Web-Adresse
Cincom	VisualWorks	smalltalk.cincom.com
Cincom	ObjectStudio	smalltalk.cincom.com
Instantiations	VASmalltalk7	www.instantiations.com/VAST/
Gemstone Systems	Gemstone/S	www.gemstone.com/products/smalltalk/
Exept Software	Smalltalk/X	www.exept.de/exept/deutsch/Smalltalk/frame_uebersicht.html
Ambrai	Ambrai Smalltalk	ambrai.com
Object Arts	Dolphin Smalltalk XP	www.object-arts.com
GNU Genral Public License	GNU Smalltalk	www.smalltalk.org/versions/GNUSmalltalk.html
Apple Computer, Inc. Software License	Squeak	www.squeak.org

Fazit

Smalltalk ist ein mächtiges System, ausgereift, produktiv, wartungsfreundlich und performant. Dank einer Vielzahl von Bibliotheken können die Entwickler komfortabel auf Datenbanken, Application Server, Message Broker, Web Services und vieles mehr zugreifen. Programmierer, die Smalltalk kennen, sind in der Regel begeisterte Anhänger. Smalltalk lässt sich für nahezu jede Art von Applikation nutzen. Bereits als es herauskam, war es seiner Zeit voraus. Allerdings waren die Distributionen teuer und verlangten leistungsstarke Rechner. Dennoch gab es einen kurzfristigen Hype, in dem es als die neue Sprache galt. Zu einem echten Durchbruch kam es jedoch nicht. Java hat diesen Schwung mit seiner kostenlosen Lizenz und dem anfangs niedrigen Ressourcenverbrauch aufgenommen.

Heute sind kostenlose beziehungsweise preiswerte Smalltalk-Distributionen verfügbar, die in der Regel einen geringen Ressourcenbedarf haben. Damit sollte ein zweiter Anlauf möglich sein. Die Anzahl der Widersacher ist jedoch nicht nur durch Java und .Net, sondern auch durch Ruby und Python größer geworden. Mit den letzten beiden ist allerdings ebenfalls die Bereitschaft, sich mit Alternativen zu beschäftigen, gewachsen. Neue Smalltalk-Umgebungen erscheinen, und auch die Community wächst wieder, was sich in Blogs, Mailinglisten, Newsgroups, Foren und IRCs beobachten lässt. Motiv ist der Wunsch nach einem einfachen Weg aus der Komplexitätsfalle, der Wunsch nach einer leichtgewichtigen und kleinen Sprache. (ka)

FRANK MÜLLER

arbeitet als Systemspezialist bei der Thales Defence Deutschland GmbH in Wilhelmshaven.