

# 1 Einleitung

**Günter Böckle** – Siemens AG

**Erik Kamsties, Klaus Pohl** – Software Systems Engineering, Universität Duisburg-Essen

Die Informations- und Kommunikationstechnologien beschleunigen Innovationen in Wirtschaft, Wissenschaft und Gesellschaft. Über 80% der Exporte Deutschlands hängen mittlerweile vom Einsatz moderner Informationstechnologien und elektronischer Systeme ab [BMBF 00]. Software wird dabei in zunehmendem Maße zum wettbewerbsbestimmenden Faktor in Produkten und Dienstleistungen [BMBF 00]. Funktionalität wird mehr und mehr in Software realisiert, sei es in Produkten wie Fahrzeugen, Hochgeschwindigkeitszügen und modernen Verkehrsflugzeugen oder im Rahmen von Dienstleistungen z. B. im Bereich Handel, Finanzwesen und öffentlicher Verwaltung.

Software-Produktlinien sind die Antwort auf die gegenwärtigen Ansprüche an Softwaresysteme und softwareintensive Produkte hinsichtlich hoher Funktionalität und Flexibilität zu geringen Kosten. Vor einem Jahrzehnt herrschte noch die Entwicklung von Individualsoftware vor, die aber nach und nach aus Kostengründen aufgegeben werden musste. Der Kunde konnte oder wollte in Individualsoftware nur noch selten investieren und Standardsoftware löste Individualsoftware mehr und mehr ab. Die Vorteile liegen auf der Hand, geringere Anschaffungs-, Einführungs- und Wartungskosten werden durch Massenfertigung möglich. Der Nachteil von Standardsoftware liegt in den oft mangelnden Anpassungsmöglichkeiten; außerdem ist sie in bestimmten Domänen, z. B. im Automobilbereich, kaum verfügbar.

Zwei der aktuellen Fallstudien, die in dem vorliegenden Buch ausführlich vorgestellt werden, verdeutlichen das Potenzial des Software-Produktlinienansatzes in gänzlich unterschiedlichen Situationen. So entwickelt die Robert Bosch GmbH Fahrerassistenzsysteme für Fahrzeuge, wie z. B. den Parkpiloten und die PreCrash-Sensierung (siehe Kapitel 15). Die große Produktvielfalt, die sich aus je nach Fahrzeugtyp verfügbarer Sensorik und geforderter Funktionalität ergibt, konnte nur mit Hilfe einer Software-Produktlinie ökonomisch beherrscht werden. Die Testo AG hat zwei Messgeräteserien am Markt, die sich historisch bedingt getrennt entwickelt haben. Zunehmender Kostendruck konnte durch Zusammenführung der beiden Serien in eine Produktlinie kompensiert werden.

Vorteile für den Kunden sind niedrigere Kosten, höhere Qualität durch eine gemeinsame und daher intensiver geprüfte Produktbasis und mehr Funktionalität (vgl. Kapitel 17). Diese Beispiele zeigen, dass der Produktlinienansatz für große Unternehmen ebenso erfolgreich sein kann wie für kleine und mittelständische Unternehmen.

Die *Produktlinienentwicklung* ist ein Ansatz zur Softwareentwicklung mit organisierter Wiederverwendung und organisierter Variabilität auf Basis einer gemeinsamen Plattform. *Organisierte Wiederverwendung* heißt, dass Software für die Wiederverwendung in Form einer so genannten *Plattform* entwickelt wird und dass einzelne Softwareprodukte durch Wiederverwendung aus dieser Plattform erstellt werden. Diese Herangehensweise stellt einen Umbruch in der Softwareentwicklung dar: Es werden nicht mehr einzelne Softwareprodukte *reaktiv* nach Markt- oder Kundenbedarf entwickelt, sondern der Fokus liegt auf der *proaktiven* Gestaltung einer gemeinsamen Plattform für eine Vielzahl von jetzigen und zukünftigen Produkten.

Der Ansatz der Software-Produktlinienentwicklung, die Entwicklung einer gemeinsamen Plattform, findet sich auch in der Fertigungsindustrie. In der Automobilindustrie beispielsweise hat die Entwicklung gemeinsamer Plattformen für unterschiedliche Fahrzeuge den Weg aus der Absatzkrise der 80er Jahre gewiesen und zu substanziellen Erfolgen geführt [Cusumano, Nobeoka 98].

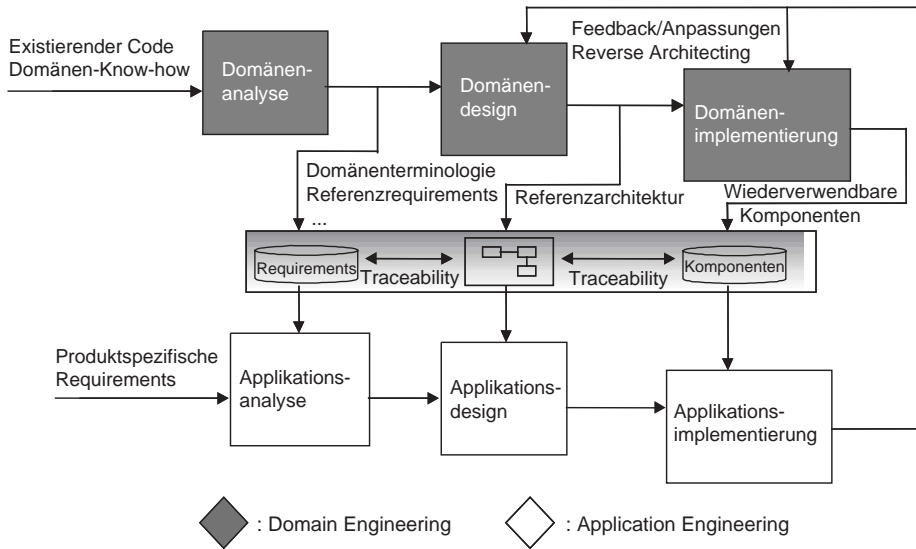
## 1.1 Grundlagen der Entwicklung von Software-Produktlinien

Die Produktlinienentwicklung für softwareintensive Systeme basiert auf zwei wichtigen Grundlagen:

- Beschreibung der Variabilität der Produktlinie
- Trennung von Domain und Application Engineering

Die Erfahrung mit der Softwarewiederverwendung in den 90er Jahren zeigt, dass ohne geeignete Planung die Kosten für die Wiederverwendung oft höher sind als der dadurch gewonnene Nutzen. Dies bedeutet, dass für Software-Produktlinien die Produkte geplant werden müssen, für die diese Wiederverwendung stattfinden soll, und zwar müssen die wesentlichen Charakteristika dieser Produkte (ihre *Features*) zu Beginn eines Projektes festgelegt werden und ihre Gemeinsamkeiten und ihre Unterschiede, d. h. die *Variabilität* der Produktlinie, identifiziert werden.

Einen Entwicklungsprozess für Produktlinien zeigt Abbildung 1–1. Es handelt sich um den Referenzprozess der Software-Produktlinienentwicklung, der im Rahmen des CAFÉ-Projekts entwickelt wurde [van der Linden 02]. Jeder Entwicklungsprozess für Software-Produktlinien in der Praxis basiert auf diesem Referenzprozess, wobei die einzelnen Aktivitäten verschieden stark gelebt werden.



**Abb. 1-1** Referenzprozess für die Software-Produktlinienentwicklung

Die Unterscheidung von Domain und Application Engineering ist grundlegend; beiden gemeinsam ist eine Anforderungsanalyse-, eine Design- und eine Implementierungsphase; Qualitätssicherungsaktivitäten sind auf diese Phasen verteilt, insbesondere Testaktivitäten werden unter Domänen- bzw. Applikationsimplementierung subsumiert. Unter einer Domäne versteht man einen zusammenhängenden Teilbereich einer Produktlinie, der gut wiederverwendbare Funktionalität für die Produkte der Produktlinie enthält. Während eine Anwendungsdomäne einen zusammenhängenden Funktionsbereich auf der Problemseite repräsentiert, wird eine technische Domäne meist (größenabhängig) durch eine Komponente, ein Teilsystem o. Ä. auf der Lösungsseite dargestellt.

Im *Domain Engineering* werden die gemeinsamen und variablen Artefakte, die Bestandteile der Plattform werden, für eine oder mehrere Domänen entwickelt. Als Variabilität einer Produktlinie werden die möglichen Unterschiede ihrer Produkte bezeichnet, d. h., Variabilität definiert innerhalb einer Produktlinie jene Teile, die durch Selektion an unterschiedliche Kundenbedürfnisse, Geschäftsziele etc. angepasst werden. Die Artefakte modellieren und realisieren die identifizierte Variabilität. Beispielsweise kann Variabilität in den Anforderungen durch variable Szenarien, in der Architektur durch optionale/alternative Komponenten und in der Implementierung durch bedingte Kompilierung (make-files, #ifdefs und ähnliche Konstrukte) oder durch Plug-ins realisiert werden.

Im *Application Engineering* werden einzelne Produkte der Produktlinie entwickelt bzw. abgeleitet. Die Produkte werden so weit wie möglich aus den Artefakten der Plattform zusammengefügt (*konfiguriert*), so dass nur in geringem Maße produktspezifische Softwareentwicklung notwendig wird. Dabei werden

die Anforderungen, Architektur und Tests aus der Plattform abgeleitet und nur was dort nicht verfügbar ist, wird extra entwickelt.

Die Vorteile von Software-Produktlinien sind vielfältig. Zum einen lassen sich in kurzer Zeit neue Produkte auf den Markt bringen, d. h., die Time-to-Market kann deutlich reduziert werden. Zum anderen lässt sich das Ziel der individualisierten Massenfertigung von Software erreichen – in den Variabilitäten werden die Möglichkeiten der Individualisierung modelliert und durch die Plattformen wird die Massenfertigung ermöglicht. Durch die intensivere Nutzung der Plattformartefakte ist deren Qualität höher als die vergleichbarer Einzelproduktartefakte. Somit ist meist auch die Qualität eines aus einer Plattform abgeleiteten Produkts höher als die eines einzeln entwickelten Produkts.

Bei der Arbeit mit Software-Produktlinien können jedoch auch Probleme entstehen. So sollte zum Beispiel unbedingt darauf geachtet werden, dass im Falle sich ändernder Anforderungen nicht einzelne Produkte unkoordiniert modifiziert werden, damit die Konsistenz der Plattform erhalten bleibt. Das bedeutet, dass für jede Änderung eine explizite Entscheidung getroffen wird, ob die Produktlinie als Ganzes angepasst wird oder ob nur ein einzelnes Produkt modifiziert wird. Werden Entscheidungen falsch oder implizit getroffen, so besteht die Gefahr, dass Kunden auf relativ kleine Produktänderungen lange warten müssen (weil zunächst die Plattform ausgebaut wird), dass Wiederverwendungspotenziale nicht ausgeschöpft werden (weil Änderungen nur an einzelnen Produkten und nicht an der wiederverwendbaren Plattform vorgenommen wurden) oder dass beim Ableiten zukünftiger Produkte Probleme durch Inkompatibilitäten zwischen modifizierten Produktlinienartefakten entstehen. Alle diese Aspekte sind hochgradig miteinander verwoben; eine Maßnahme an einer Stelle kann zu unerwarteten Auswirkungen an ganz anderer Stelle führen. Dieses Buch soll einen Beitrag zu einem besseren Verständnis von Produktlinien leisten, damit ihre offensichtlichen Vorteile nutzbar werden, ohne dabei unerwartete Risiken einzugehen.

## 1.2 Aufbau des vorliegenden Buches

Das vorliegende Buch gliedert sich in fünf Teile: Einführung, Techniken, Transition und Evolution, Fallstudien und Zusammenfassung sowie einen Anhang. Es vermittelt die Grundlagen der Produktlinienentwicklung wie Variabilität und Organisationsaspekte und stellt Techniken und Methoden für alle technischen Aktivitäten, angefangen mit Scoping und Produktdefinition über Requirements Engineering und Architektur bis zur Implementierung und zum Test vor. Außerdem wird die Einführung der Produktlinienentwicklung in eine Organisation, insbesondere die Nutzung existierender Dokumente zur Erstellung von Produktlinienanforderungen und -architekturen, beschrieben. Abschließend werden Erfahrungen aus fünf industriellen Fallstudien zur Einführung der Produktlinienentwicklung berichtet.

### 1.2.1 Teil I – Einführung

Der erste Teil legt durch die Einführung von Variabilität und die Diskussion organisatorischer Produktlinienaspekte die Grundlage für die nachfolgenden Teile.

Die Variabilität einer Produktlinie schlägt sich in allen Artefakten des Domain Engineering nieder, von den Anforderungen bis hin zu Testfällen. So genannte *Variationspunkte* beschreiben Stellen in den Artefakten, an der die Auswahl einer oder mehrerer Ausprägungen für unterschiedliche Produktvarianten möglich ist. Die Herausforderung liegt in einer phasenübergreifenden Beschreibung, so dass eine gemeinsame Betrachtung möglich wird. Kapitel 2 diskutiert die Grundlagen der Variabilität und beschreibt, wie Variabilität in Artefakten über alle Phasen des Lebenszyklus einer Produktlinie hinweg durchgängig erfasst und kontrolliert werden kann.

Die Software-Produktlinienentwicklung offeriert nicht nur ein neues Herangehen an die Softwareentwicklung, es impliziert auch neue geschäftliche und organisatorische Strukturen. Die Organisationsstruktur muss den Prozess widerspiegeln und vor allem die Verantwortlichkeiten, die sich im Domain Engineering auf die wiederverwendbaren, von mehreren Produkten genutzten Artefakte beziehen und im Application Engineering auf die produktspezifischen Artefakte. Kapitel 3 stellt Organisationsformen und Kriterien vor, die es ermöglichen, die jeweils am besten zu einer gegebenen Situation passende Organisationsstruktur auszuwählen.

### 1.2.2 Teil II – Techniken

In den frühen Phasen des Domain Engineering (zusammengefasst in der *Domänenanalyse* in Abb. 1–1) werden die Produkte mit ihren charakteristischen Leistungsmerkmalen (Features), sowohl den gemeinsamen als auch den unterscheidenden, definiert. Die Produkte werden in einem Produktportfolio zusammengestellt, das als Grundlage für die Optimierung der Softwarewiederverwendung innerhalb der Produktlinie dient. In der Software-Produktlinienentwicklung wird dies auch als *Scoping* bezeichnet, der Bestimmung des Produktraums (engl. *product scope*). Dies wird in Kapitel 4 beschrieben. Das Scoping verbindet die externe Sicht der Produktplanung mit der internen Sicht der Entwicklungseigenschaften. Entsprechend verändert sich die Arbeitsweise von Produktmanagern in einer Produktlinienumgebung. Für eine systematische Umstellung wird zunächst ermittelt, welche Prozesse und Aktivitäten bisher durchgeführt werden und diese werden dann auf ihre Eignung für die Software-Produktlinienentwicklung bewertet. Dann werden notwendige neue Prozesse und Aktivitäten in die tägliche Arbeit eingebunden. Solche Bewertungsverfahren für das Produktmanagement schildert Kapitel 5.

Die Phasen *Domänenanalyse* und *Applikationsanalyse* aus Abbildung 1–1 umfassen vor allem das Requirements Engineering. In der Domänenanalyse werden im Anschluss an das Scoping auf der Basis der ermittelten Leistungsmerkmale die gemeinsamen und variablen Artefakte der Produktlinie festgelegt. Dazu werden die Anwendungsszenarien und Anforderungen unter Berücksichtigung technischer, gesetzlicher und sonstiger Rahmenbedingungen der Produktlinie ermittelt und unter Berücksichtigung der Produktlinienvariabilität repräsentiert. Die Artefakte aus der Domänenanalyse bilden die Basis sowohl für weitere Entwicklungsphasen im Domain Engineering als auch für die Applikationsanalyse. Im Requirements Engineering des Application Engineering werden die für das jeweilige Produkt spezifischen Anforderungen identifiziert und mit den bisher in der Produktlinie vorgesehenen Möglichkeiten abgeglichen. Können die produktspezifischen Anforderungen nicht ohne Änderung der Plattform erfüllt werden, wird in einer Trade-off-Entscheidung abgewogen, inwieweit die jeweilige Anforderung angepasst werden kann oder eine zusätzliche Entwicklung erfolgen muss. Schließlich werden die produktspezifischen Anforderungen sowie eventuelle Abweichungen von der Produktlinie dokumentiert und damit die Grundlage für das Produktdesign gelegt. All diese Aufgabenstellungen des Requirements Engineering werden in Kapitel 6 beschrieben.

Im Domänenendesign (siehe Abb. 1–1) wird auf Grundlage der Anforderungen eine gemeinsame Architektur für die Produkte der Produktlinie entwickelt, welche die Geschäftsziele und Kundenwünsche unterstützt. Diese *Referenzarchitektur* setzt insbesondere die architekturrelevanten Anforderungen um und enthält Variationspunkte zur Realisierung kundenspezifischer Produkte im Application Engineering. In Kapitel 7 wird eine Methode vorgestellt, die eine systematische Entwicklung tragfähiger Architekturen für softwareintensive Produktlinien unterstützt.

Kein komplexes Produkt kann zu 100 % neu entwickelt werden. Man wird immer versuchen, Komponenten zu verwenden, die aus früheren Produkten oder anderen Abteilungen einer Firma kommen. Gleichermaßen wird man am Markt erhältliche Komponenten verwenden, wenn sie Teilfunktionen des Produkts erfüllen. Damit lassen sich Kosten und Entwicklungszeit sparen. Man bezeichnet solche (kommerziell erhältlichen) Komponenten auch als *COTS – commercial-off-the-shelf*. Bei Software ist es jedoch erheblich schwieriger als bei Hardware, geeignete Komponenten zu finden und eventuell notwendige Anpassungen zu identifizieren. Bei Produktlinien muss darüber hinaus geprüft werden, ob sie den Gemeinsamkeiten der Produkte aus der Produktlinie genügen und ob sie die Anforderungen der Produktlinie bezüglich Variabilität in ihrem Funktions- und Leistungsbereich erfüllen. Die Auswahl solcher COTS-Komponenten in Abhängigkeit von den Anforderungen an die Produktlinie wird in Kapitel 8 beschrieben.

Die Implementierung eines Produkts aus der Produktlinie unterscheidet sich signifikant von der Implementierung eines Einzelprodukts. So weit wie möglich werden Produktkomponenten aus vorhandenen, eventuell variablen Komponenten *abgeleitet*. Dabei werden Variationspunkte an konkrete Werte gebunden. Die Techniken der Modellierung von Variationspunkten in Softwarekomponenten entstammen alle der klassischen Softwaretechnik – von bedingter Kompilierung über dynamische Bindung bis zu Plug-ins. Diese Art der Produktableitung ist erheblich schneller als eine herkömmliche Produktentwicklung und ermöglicht so eine sehr kurze Time-to-Market; dies ist einer der wichtigsten Vorteile der Software-Produktlinienentwicklung. Wie man in der Phase der Domänenimplementierung Variationspunkte in generischen Komponenten implementiert, beschreibt Kapitel 9.

Schon im konventionellen Software Engineering wird versucht, die Ableitung von Testfällen von den späten Phasen hin zu den frühen zu verlagern. Dadurch können Fehler früher erkannt und behoben werden, was eine Steigerung der Qualität der Software mit sich bringt. In der Produktlinienentwicklung wird dies noch wichtiger, da Teile der Produktlinie in vielen Produkten verwendet werden. Bereits im Requirements Engineering der Domänenanalyse werden daher für den Systemtest Testfälle abgeleitet. Ein wichtiger Vorteil der Software-Produktlinienentwicklung ist die Reduktion von Aufwand und Verkürzung von Entwicklungszyklen durch gezielte Wiederverwendung. Kapitel 10 beschreibt die Erstellung von wiederverwendbaren Testfällen sowie deren Wiederverwendung. Es wird eine Methode zur Ableitung von Testfällen auf Basis von Use Cases erläutert, die die Variabilität der Produktlinie berücksichtigt, um eine Reduktion des Testaufwands zu erzielen.

### 1.2.3 Teil III – Transition und Evolution

Wenn eine Firma oder eine Abteilung sich dazu entschließt, ihre Produkte im Rahmen einer Produktlinie zu entwickeln, um die dadurch zu erzielenden wirtschaftlichen Vorteile zu erreichen, so müssen dafür diverse Aktivitäten und Strukturen dieser Organisation umgestellt werden. Jede Organisation folgt gewissen Prozessen, entwickelt Produkte und sammelt dabei Erfahrungen, setzt bestimmte Methoden ein und nutzt bestimmte Werkzeuge. Außerdem hat sie eine bestimmte Organisationsstruktur. Von all dem muss der aktuelle Stand bestimmt werden und es muss festgelegt werden, was davon gleich bleiben kann und was geändert werden muss, um bestmögliche Software-Produktlinienentwicklung zu ermöglichen. Zur Bestimmung des aktuellen Zustandes der Organisation werden Assessmentverfahren eingesetzt. Ein Ansatz zur Entwicklung eines Transitionsprozesses für den Übergang von der Einzelsystementwicklung zur Software-Produktlinienentwicklung wird in Kapitel 11 geschildert.

Eine Organisation wird bei diesem Übergang so viel ihrer bisherigen Entwicklungen wie möglich für die neue Produktlinie verwenden wollen, um Aufwand und Kosten zu sparen. Wenn existierende Produkte in die neue Produktlinie aufgenommen werden sollen, kann man beispielsweise aus der existierenden Dokumentation neue Produktlinienanforderungen ableiten. Wie dies geschieht, schildert Kapitel 12. Möchte man existierende Produkte in die Produktlinie aufnehmen, so muss deren Architektur bekannt sein, um wiederverwendbare Komponenten, bisherige Gemeinsamkeiten, aber auch Unterschiede identifizieren zu können. Wie man die Architektur existierender Softwaresysteme mit in die Entwicklung einer Referenzarchitektur für eine neue Produktlinie einbezieht, wird in Kapitel 13 beschrieben.

Große Softwaresysteme entwickeln sich im Laufe der Zeit häufig irregulär. Sich ändernde Anforderungen führen dazu, dass an vielen Stellen Änderungen oder Erweiterungen vorgenommen werden müssen, wobei sich einige dieser Änderungen quer über viele Architekturkomponenten hinziehen und oft ein ursprünglich elegantes Design zunichte machen. Daher müssen große Softwaresysteme oftmals im Laufe ihres Lebenszyklus von Grund auf neu entwickelt werden. In gewissem Rahmen ist es möglich, zukünftige Entwicklungen vorausszusehen und zu steuern und damit eine Neuentwicklung hinauszuschieben. So kommt man eher zu einer kontrollierten Evolution als zu planlosen Wucherungen. Da jede Änderung eines Softwaresystems von Anforderungen ausgeht, müssen solche Ansätze im Requirements Engineering beginnen. Im Fall einer Produktlinie ist es besonders wichtig, eine gezielte Evolution anzustreben, da hier von Änderungen oft nicht nur ein Produkt betroffen ist, sondern viele, was zu unerwünschten Änderungen der Plattform mit Seiteneffekten auf andere Produkte führen kann. Ansätze zur Evolution werden in Kapitel 14 beschrieben.

#### **1.2.4 Teil IV – Fallstudien und Zusammenfassung**

Die Software-Produktlinienentwicklung ist nicht auf bestimmte Anwendungen beschränkt, sondern sie wird schon in einer Vielzahl von Anwendungsdomänen eingesetzt, aus denen positive Erfahrungen zu berichten sind. In den Folgekapiteln wird dies in verschiedenen Domänen gezeigt: für Fahrerassistenzsysteme in Kapitel 15, für Börseninformationssysteme in Kapitel 16, für eingebettete Systeme in Kapitel 17, für Krankenhausinformationssysteme in Kapitel 18 und für die Medizintechnik in Kapitel 19.

Kapitel 20 bietet eine Zusammenfassung der verschiedenen Aspekte des Buches und einen Ausblick auf zu erwartende Entwicklungen in der Software-Produktlinienentwicklung.

### 1.2.5 Teil V – Anhang

Um die notwendigen Grundlagen für die Software-Produktlinienentwicklung zu erarbeiten, haben sich eine Vielzahl von europäischen Firmen, Forschungsinstituten und Universitäten zusammengeschlossen und gemeinsam Projekte dazu durchgeführt. Das vorliegende Buch stellt Ergebnisse dieser Projekte vor, mit Schwerpunkt auf den Arbeiten der deutschen Partner. Anhang A.1 gibt einen Überblick über diese Projekte und die beteiligten Partner.

Die in diesem Buch verwendeten Begriffe aus der Software-Produktlinienentwicklung sind in einem Glossar (Anhang A.3) zusammengefasst und von anderen Begriffen abgegrenzt. Es wurde bewusst darauf verzichtet, Begriffe im Glossar aufzuführen, die nur für einzelne Kapitel dieses Buches relevant sind.