

- Komponenten
- Software-Dokumentation
- Architektur
- Tests (unter anderem Integrations-, Leistungs- und Komponententests)
- Anforderungsspezifikation
- Entwicklungsprozess, Methoden und Werkzeuge
- Budget-/Zeit- und Arbeitspläne
- Handbücher
- Entwickler

CelsiusTech: Familie von 55 Schiffssystemen

- Integrationstest of 1-1,5 Millionen SLOC benötigt 1-2 Leute
- Rehosting auf neue Plattform/Betriebssystem benötigt 3 Monate
- Kosten- und Zeitplan werden eingehalten
- Systemattribute (wie Performanz) können vorausgesagt werden
- hohe Kundenzufriedenheit
- Hardware-/Software-Kostenverhältnis veränderte sich von 35:65 zu 80:20

Nokia: Produktlinie mit 25-30 neuen Produkten pro Jahr
Produktübergreifend gibt es

- unterschiedliche Anzahlen von Tasten
- unterschiedliche Display-Größen
- andere unterschiedliche Produktfunktionen
- 58 verschiedene unterstützte Sprachen
- 130 bediente Länder
- Kompatibilität mit früheren Produkten
- konfigurierbare Produktfunktionen
- Änderung der Geräte nach Auslieferung

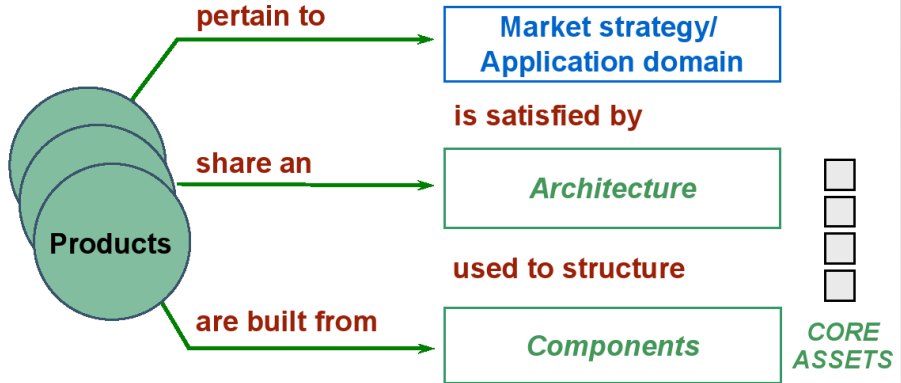
Definition

A **software product line** is a set of software-intensive systems

- sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission
- and that are developed from a common set of core assets in a prescribed way.

– Clements und Northrop (2001)

Übersicht über Produktlinien



Product lines

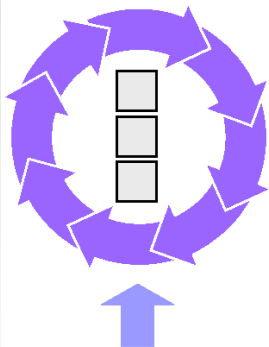
- take economic advantage of commonality
- bound variability

Quelle: Linda Northrop, SEI

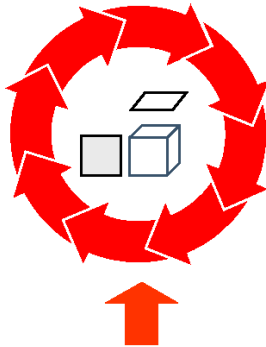
**Use of a core
asset base**

in production

**of a related
set of products**



Architecture



Production Plan



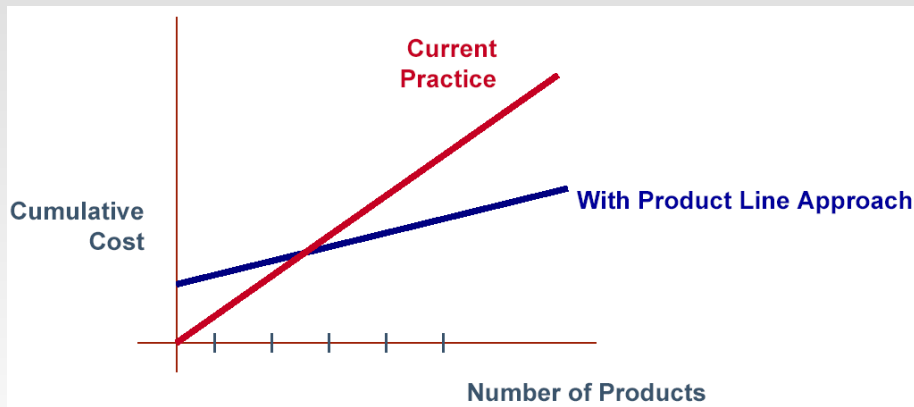
**Scope Definition
Business Case**

Quelle: Linda Northrop, SEI

Kostenaspekte einer Software-Produktlinie

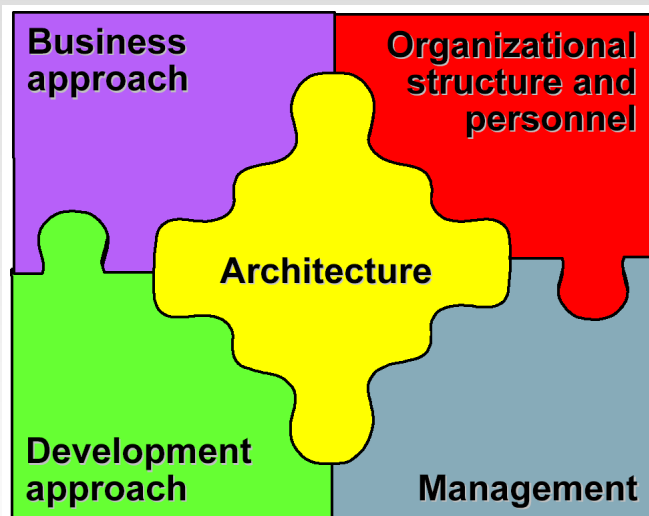
- Marktanalyse: muss eine Familie von Produkten betrachten
- Projektplan: muss generisch oder erweiterbar sein, um Variationen zu erlauben
- Architektur: muss Variation unterstützen
- Software-Komponenten: müssen generischer sein, ohne an Performanz einzubüßen; müssen Variation unterstützen
- Testpläne/-fälle/-daten: müssen Variationen und mehrere Instanzen einer Produktlinie berücksichtigen
- Entwickler: benötigen Training in den Assets und Verfahren der Produktlinie

Return-on-Investment



Quelle: Weiss und Lai, 1999.

Zusammenspielende Komponenten



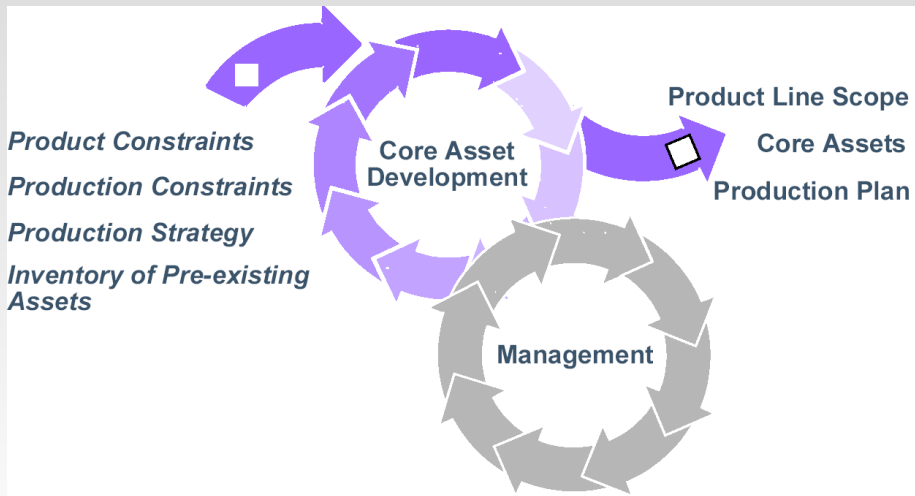
Quelle: Linda Northrop, SEI

Practice Areas



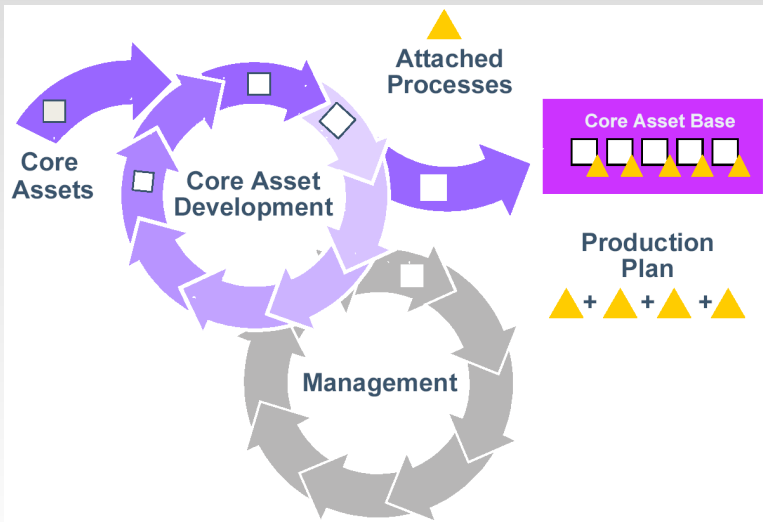
Quelle: Linda Northrop, SEI

Entwicklung der Core-Assets



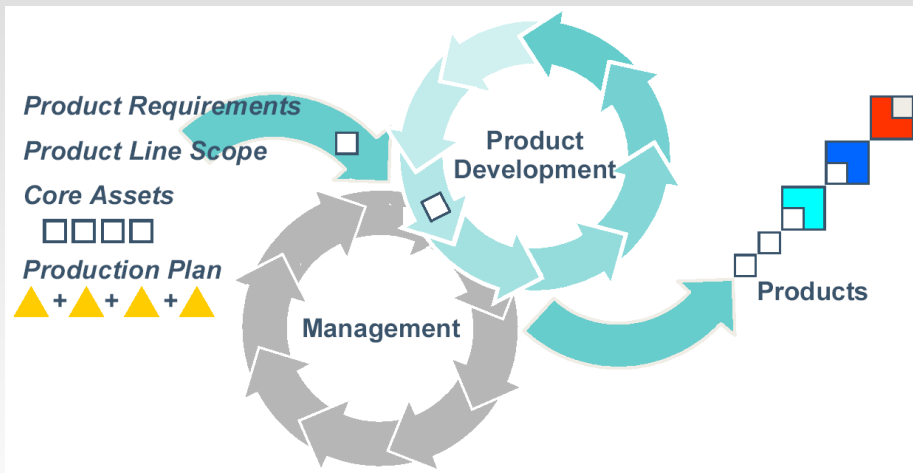
Quelle: Linda Northrop, SEI

Entwicklung der Core-Assets: Assoziierte Prozesse



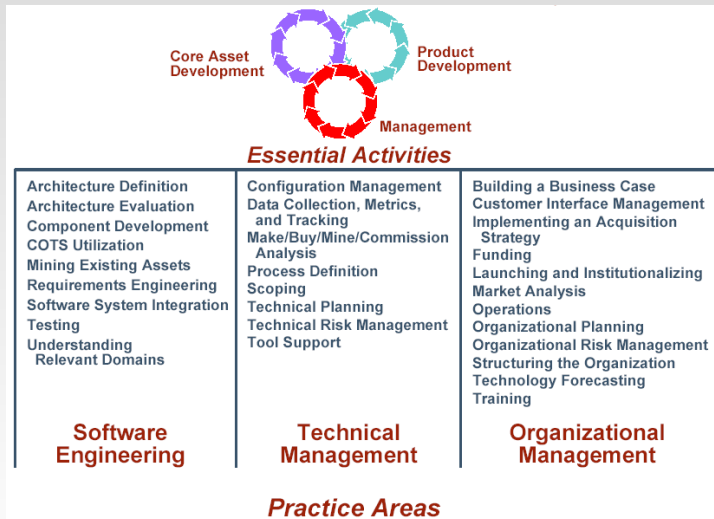
Quelle: Linda Northrop, SEI

Produktentwicklung



Quelle: Linda Northrop, SEI

Essentielle Aktivitäten



Quelle: Linda Northrop, SEI

Einführung von Produktlinien I

Proaktiv:

- definiere zuerst Scope: was gehört zur Produktlinie?
 - Scope leitet die weitere Entwicklung
 - Entwickle zuerst Core-Assets
- + Produkte können rasch entwickelt werden, sobald die Produktlinie steht
- hohe Vorausleistung und Vorhersagefähigkeit verlangt

Einführung von Produktlinien II

Reaktiv:

- Beginne mit einem oder mehreren Produkten
- Extrahiere daraus Core-Assets für die Produktlinie
- Scope entwickelt sich dabei stetig
- + niedrige Einstiegskosten
- + größerer Einfluss von Erfahrung
 - Architektur könnte suboptimal sein, wird schrittweise weiterentwickelt
 - Restrukturierungsaufwand notwendig

Einführung von Produktlinien III

Inkrementell (sowohl bei reaktiver als auch proaktiver Entwicklung möglich):

- schrittweise Entwicklung der Core-Assets mit initialer Planung der Produktlinie:
- entwickle Teile der Core-Asset-Base einschließlich Architektur und Komponenten
- entwickle ein oder mehrere Produkte
- entwickle weitere Core-Assets
- entwickle weitere Produkte
- entwickle Core-Asset-Base weiter
- ...

Produktlinien . . .

- haben Gemeinsamkeiten
- und definierte Unterschiede: Variabilitäten

Produkt wird aus Core-Assets zusammengebaut.
Variabilitäten werden festgelegt.

Produktlinien ...

- haben Gemeinsamkeiten
- und definierte Unterschiede: Variabilitäten

Produkt wird aus Core-Assets zusammengebaut.
Variabilitäten werden festgelegt.

Bindungszeitpunkt der Variabilitäten

- zur Übersetzungszeit
- zur Bindezeit
- zur Laufzeit

Architekturmechanismen für Variabilitäten

Kombination, Ersetzung und Auslass von Komponenten (auch zur Laufzeit)

Frontend {C, C++, Java}	Middle End {ME}	Backend {i386, Motorola 68000}
C	ME	i386
C	ME	Motorola 68000
C++	ME	i386
C++	ME	Motorola 68000
Java	ME	i386
Java	ME	Motorola 68000

Architekturmechanismen für Variabilitäten

Parametrisierung (einschließlich Makros und Templates)

```
1  generic
2    type My_Type is private;
3    with procedure Foo (M : My_Type);
4  procedure Apply;
5
6  procedure Apply is
7    X : My_Type;
8  begin
9
10     ...
11     Foo (X);
12     ...
13 end Apply;
```

Parametrisierung (einschließlich Makros und Templates)

```
1 typedef (*FP)( int );  
2 void Apply (FP fp) {  
3     ...  
4     fp (X);  
5     ...  
6 }
```