プログラミング言語設計論

2014年度

第8回: プロダクトライン・
機能指向・文脈指向

担当: 増原英彦

問(1/2):要素の集合を表わすクラス群を設計せよ (15分)

- 共通の操作: 要素の挿入(insert), 削除(erase)
- ListSet: 要素を連結リストで管理
  - 挿入はO(1), 削除はO(*n*)
- TreeSet: 要素を二分木で管理
  - 挿入、削除はO(log*n*)
  - 追加操作: 要素を含むかの判定(contains)
- ConcurrentListSet: 複数スレッドから同時にinsert, eraseできるListSet
- ConcurrentTreeSet:複数スレッドから(略)TreeSet
- CountableListSet: 追加操作: 大きさ(size)
- CountableTreeSet:追加操作: 大きさ(size)
- CountableConcurrentListSet: (略)
- CountableConcurrentTreeSet: (略)

目標:
同じ定義が
複数回
登場しないこと

問(2/2):逐次にinsertをされた後、並列にinsert(など)が行われるListSetを効率化する方法を論ぜよ(10分)

- 逐次にしかinsertしないと分かっているときにロック操作をしないこと
- ListSetの定義側、利用側のどちらも変更可
- 具体的なクラス設計を示してもよい

```
//setup
ListSet students = new ListSet();
ListSet teachers = new ListSet();
readPeople(studentDB, students);
readPeople(teacherDB, teachers);
//main
new ThreadPool(10){
    void run() {
      ...students.insert(...)...
      ...teachers.insert(...)... }}
}
```

プロダクトライン (product line)

- 同じ会社から販売される一連の製品群やサービス群で、異なる仕様と異なる値段を持つ
  a range of similar products or services that are sold by the same company, with different features and different prices (Cambridge Dictionary Online)
- 同じ会社から市場に出される関連のある製品群
  a group of related products marketed by the same company (Collins)

タイプ	3階建 4LDK		2階建 3LDK				1ルーム		ホテル	
型番	WZR-1750DHP2	WZR-1166DHP2	WZR-S900DHP	WSR-1166DHP	WHR-1166DHP	WZR-S600DHP	WHR-600D	WHR-300HP2	WMR-433-BK(ブラック) WMR-433-WH(ホワイト) WMR-433-BL(ブルー) WMR-433-PK(ピンク) WMR-433-YL(イエロー)	WMR-300(ブラック) WMR-300-WH(ホワイト)
	新価格 ¥18,700(税抜)	新価格 ¥16,500(税抜)	New ¥14,300(税抜)	New ¥10,000(税抜)	新価格 ¥7,200(税抜)	New ¥11,100(税抜)	新価格 ¥5,500(税抜)	新価格 ¥4,800(税抜)	新価格 ¥4,400(税抜)	新価格 ¥2,770(税抜)
スピード(規格値)	ac1300+na/ng450Mbps 有線1000Mbps	ac866+na/ng300Mbps 有線1000Mbps	na450+ng450Mbps 有線1000Mbps	ac866+na/ng300Mbps 有線1000Mbps	ac866+na/ng300Mbps INTERNETポート1000Mbps 有線LANポート100Mbps	na300+ng300Mbps 有線1000Mbps	na300+ng300Mbps 有線100Mbps	ng300Mbps 有線100Mbps	ac433/na/ng450Mbps	ng300Mbps
つながる	ビームフォーミングEX								出張先ホテルの有線LANをWi-Fi化	
	ビームフォーミング(Wi-Fiの安定性が向上)									
	High Power設計(電波がよく飛ぶ高出力設計)							High Power設計		
	アドバンスド QoS(動画視聴が快適)					アドバンスド QoS				
	無線中継機能(障害物があって、つながりにくい死角エリアへ電波を中継)※2									
									※出張・旅行用のため、PPPoE回線ではご利用いただけません。	
あんしん	ペアレンタルコントロール(通信可能時間帯設定&有害サイトブロックのWセキュリティー)					ペアレンタルコントロール				
	マルチセキュリティー(接続機器に合わせて自動で最高レベルのセキュリティーに設定します)									マルチセキュリティー
かんたん	Wi-Fiリモコン(スマホで設定変更可能)					Wi-Fiリモコン				
	AOSS2(3G回線やパソコンがなくてもスマホだけでインターネットWi-Fi初期設定が可能。しかも、タブレットやウルトラブックでも)					AOSS2				
	AOSS/WPS(パソコンだけでなく、テレビやプリンター、ゲーム機など広くデジタル家電品に採用されている、ボタン一発の接続・設定機能)								WPS	AOSS/WPS
				QRsetup			QRsetup			
みんなでつかう	USB共有接続(つなぐだけでUSB機器をワイヤレス化)					USB共有接続				
	USB3.0×1/USB2.0×1	USB3.0×1	USB2.0×1			USB2.0×1				
屋外でも便利	リモートアクセス(VPN)(外から自宅のPCにアクセス)※2					リモートアクセス(VPN)※2				
最大消費電力	18.2W	15.1W	23W	17.2W	10.8W	21W	8.2W	8.2W	2.5W	2.5W
外形寸法	W34×H212×D183mm	W34×H212×D183mm	W28×H185×D196mm	W160×H160×D36.5mm	W55×H159×D131mm	W28×H185×D196mm	W55×H159×D131mm	W55×H159×D131mm	W45×H45×D15mm	W58×H58×D20mm
質量	約580g	約590g	約520g	約340g	約280g	約520g	約275g	約265g	約19g	約51g
セットモデル	親機+中継機/LAN接続子機セット WZR-1750DHP2/E 特価 ¥36,400(税抜) ◆親機 中継機子機設定済み親機の2台セットモデル	親機+中継機/LAN接続子機セット WZR-1166DHP2/E 特価 ¥31,900(税抜) ◆親機 中継機子機設定済み親機の2台セットモデル				LAN接続子機セット(WLU-TX-G300) WZR-600DHP3/E ¥14,900(税抜) ※親機は新モデル(WZR-600DHP3)。		USB2.0用子機セット(WLU-G301N) WHR-300HP2/U 特価 ¥7,000(税抜)		便利セット(LANケーブル、収納ケースなし) WMR-300/S(ブラック) ¥2,430(税抜) WMR-300/S-WH(ホワイト) ¥2,430(税抜)

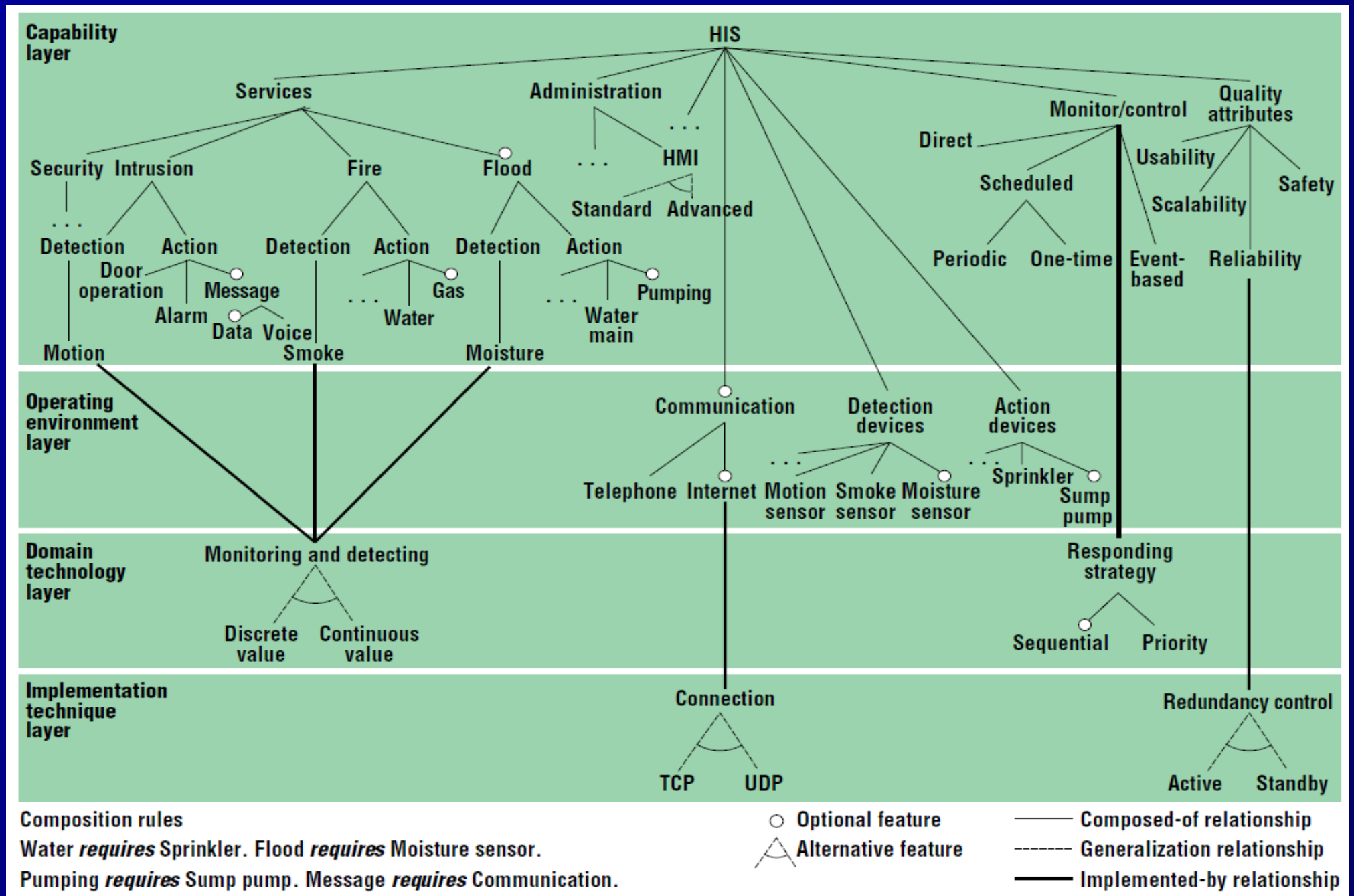
※1.屋内使用に限る。※2.WHR-1166DHP、WHR-600D、WHR-300HP2はWake On LANパケット転送非対応。

BUFFALO無線LAN親機製品比較表http://buffalo.jp/product/wireless-lan/ap/images/btn_hikakuhyo.jpg

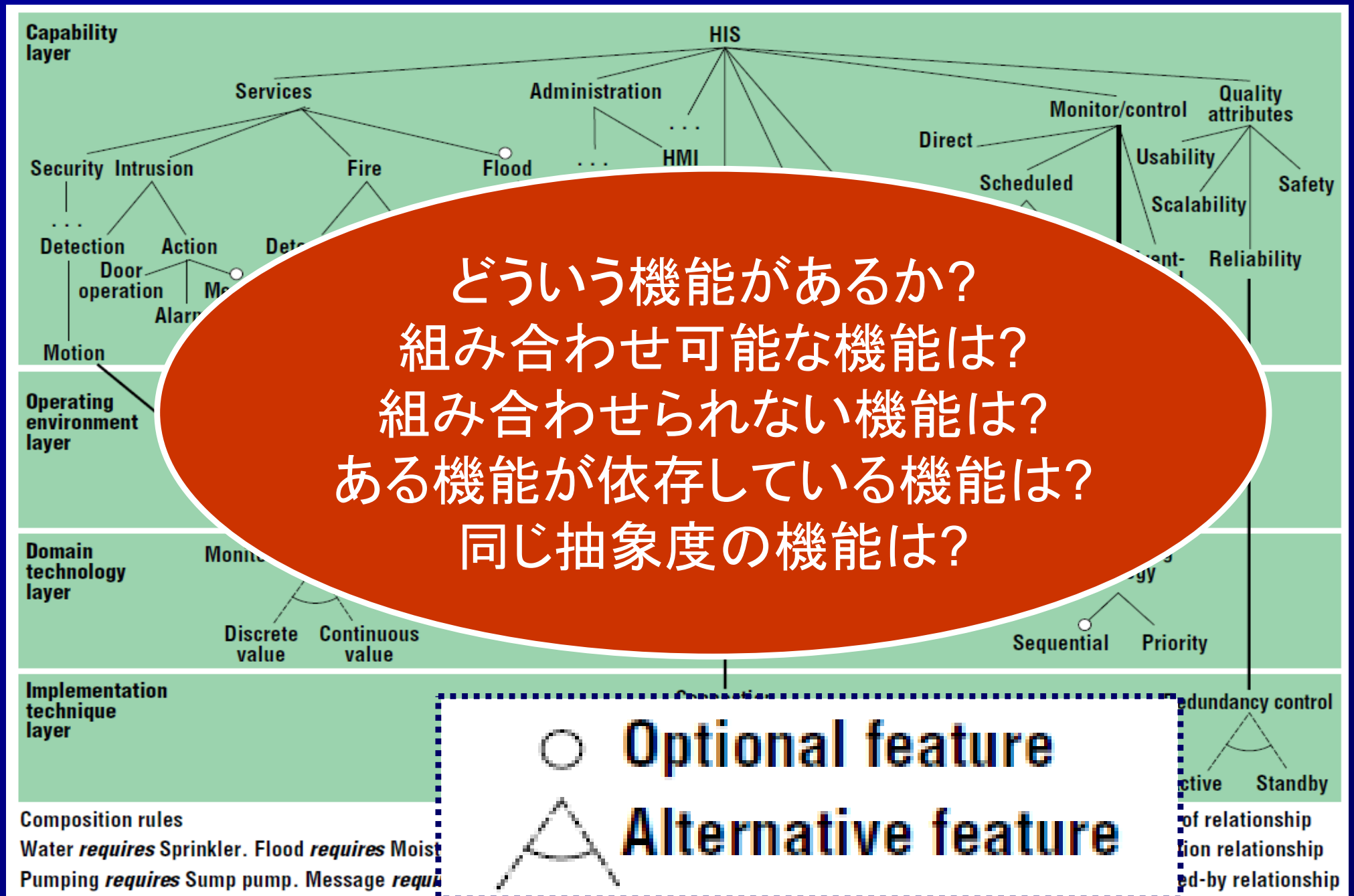
機能指向プロダクトライン工学[KLP02]
(Feature-oriented product line engineering)

- プロダクトラインを前提に、再利用性が高くなるように問題分析、モデル化、設計を行う手法
  - 可変性管理(variability management)
- 製品の機能を中心に考える
  - 機能間の依存関係を整理
  - 非モノ指向!?

機能分析の例: Home Integration System [KLD02]



機能分析の例: Home Integration System [KLD02]



どういう機能があるか?
組み合わせ可能な機能は?
組み合わせられない機能は?
ある機能が依存している機能は?
同じ抽象度の機能は?

プロダクトラインの実現方法: #ifdef

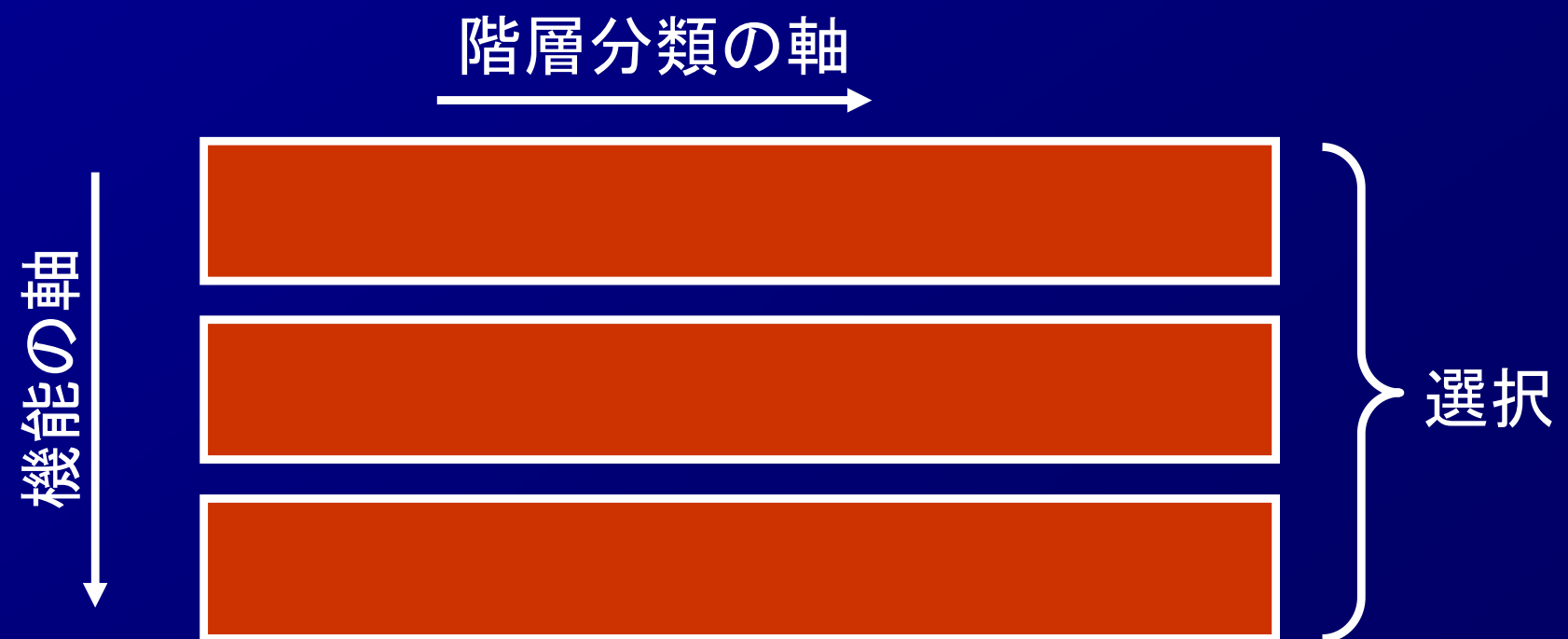
- Linux 3.2には約12000種類の「機能」[Reinhard14]
- 問題点
  - 分かりにくい
  - ifdef処理後でないと正しさを確認できない
  - 組み合わせが2^n通り

```
static int __rep_queue_filedone(dbenv, rep, rfp)
  DB_ENV *dbenv;
  REP *rep;
  __rep_fileinfo_args *rfp; {
#ifndef HAVE_QUEUE
  COMPQUIET(rep, NULL);
  COMPQUIET(rfp, NULL);
  return (__db_no_queue_am(dbenv));
#else
  db_pgno_t first, last;
  u_int32_t flags;
  int empty, ret, t_ret;
#ifdef DIAGNOSTIC
  DB_MSGBUF mb;
#endif
  // over 100 further lines of C code
#endif
}
```

OracleのBerkeley DBにおける使用例 ([KA13]より)

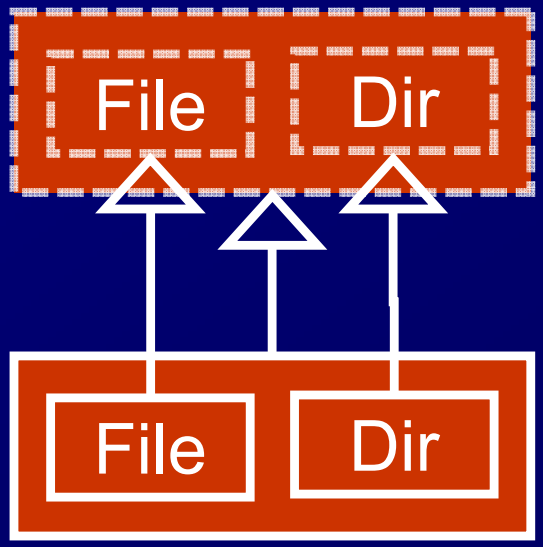
機能指向プログラミング

- 目標
  - 機能指向と階層分類の両立
  - 機能を選択的に組み合わせる
- 手法: 層(=機能)によるモジュール化

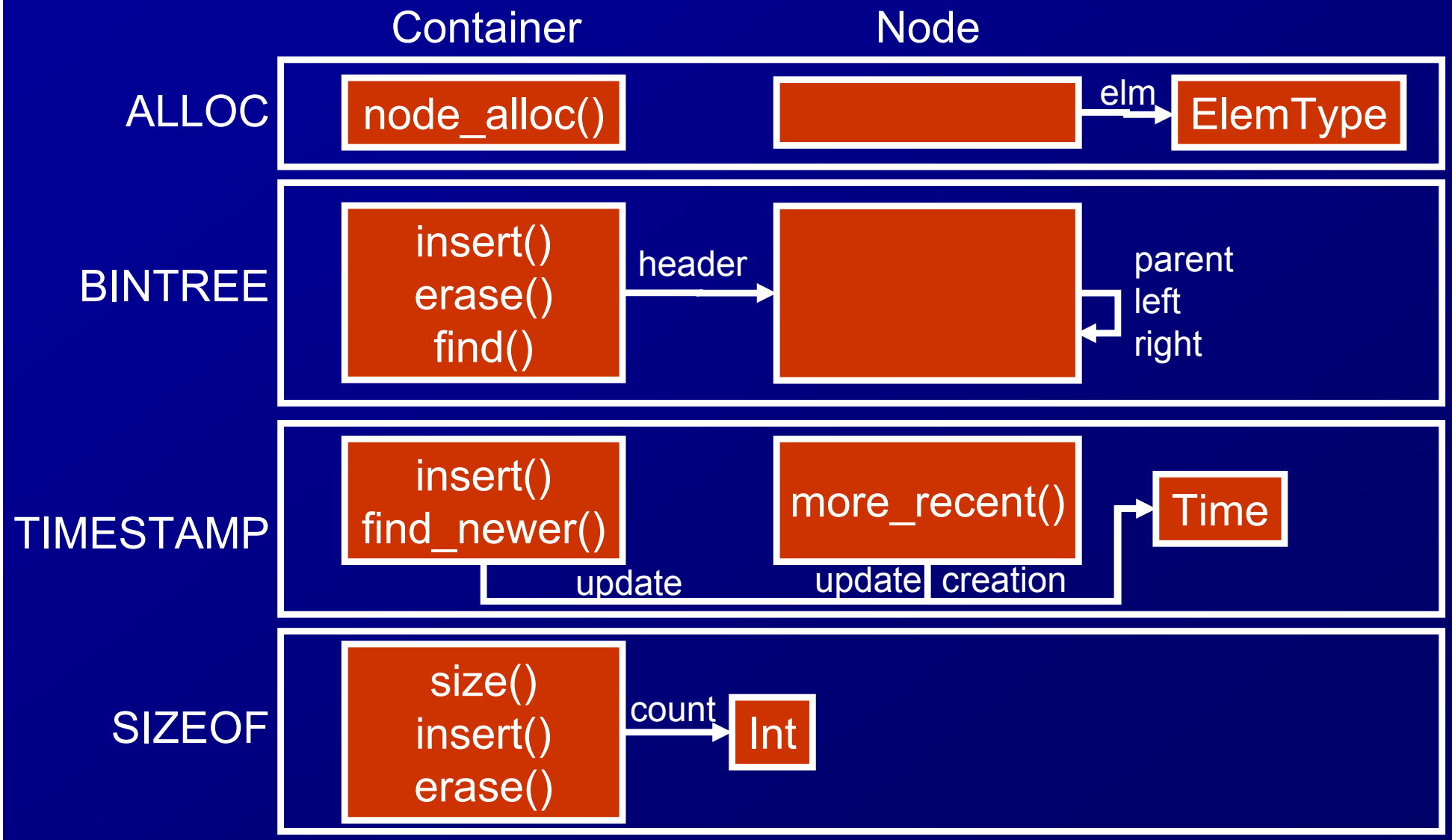


mixin layersによる機能指向プログラミング [YB98]

- (mixin: 親クラスをパラメタ化したクラス)
- mixin layer: mixinを入れ子にしたもの
  - 外側のmixin = 機能に対応
  - 内側のクラス(mixin) = 階層分類
    - 外側のmixinの親にある同名クラスの子クラス
- 機能の合成
  = 外側のmixinの合成



mixin layersによるCollectionの機能指向実現 [YB98]



mixin layersによるCollectionの機能指向実現 [YB98]

上位層の
Containerを
継承

上位層はパラメタ = 後から決定

Container　　Node

ALLOC

BINTREE

TIMESTAMP

update creation

SIZEOF

size()
insert()
erase()

count → Int

```
template <class Super> class SIZEOF : Super {
  class Container : public Super::Container {
    int count;
    void insert ( EleType el ) {
      Super::Container::insert(el); count++; }
    void erase ( Node* node ) {
      Super::Container::erase(el); count--; }
    int size () { return count; } }; };
```

問(2/2):逐次にinsertをされた後、並列にinsert(など)が行われるListSetを効率化する方法を論ぜよ(10分)

- 逐次にしかinsertしないと分かっているときにロック操作をしないこと
- ListSetの定義側、利用側のどちらも変更可
- 具体的なクラス設計を示してもよい

```
//setup
ListSet students = new ListSet();
ListSet teachers = new ListSet();
readPeople(studentDB, students);
readPeople(teacherDB, teachers);
//main
new ThreadPool(10){
    void run() {
      ...students.insert(...)...
      ...teachers.insert(...)... }}
}
```

文脈指向プログラミング [HCN08]

- 背景
  - モノの振舞は場面によって変わる
  - OOPでのモデル化
    - 1つのクラス・メソッドに複数の振舞
      → 条件分岐が沢山
    - 場面ごとに異なるクラスを定義
      → 1つのモノが1つのオブジェクトでなくなる

OOPでのモデル化例 (1つのクラス)

```
if (isConcurrent) lock.get();
head = new Cons(elm,head);
if (isConcurrent) lock.release();
```

ListSet
isConcurrent
lock
insert(elm)
setConcurrent(c)

```
ListSet students = new ListSet();
ListSet teachers = new ListSet();
students.setConcurrent(false);
readPeople(studentDB, students);
teachers.setConcurrent(false);
readPeople(teacherDB, teachers);
students.setConcurrent(true);
teachers.setConcurrent(true);
new ThreadPool(10){
  void run() {
    ...students.insert(...)...
    ...teachers.insert(...)... }}
```

逐次で初期化

場面が変化したら振舞を切り替えてまわる

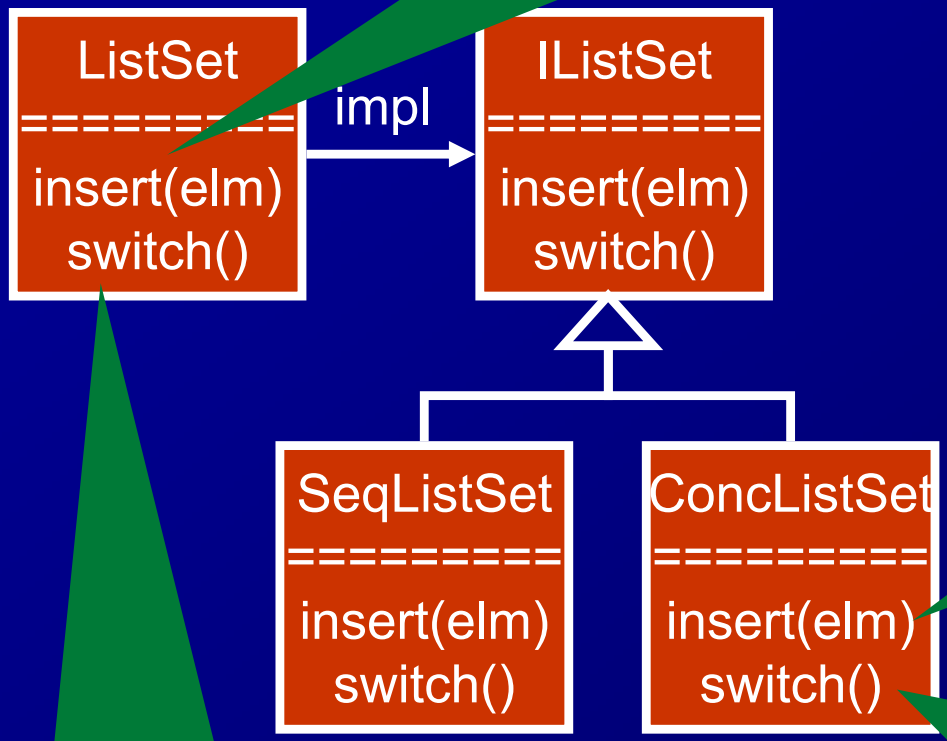
並列に更新

OOPでのモデル化例 (複数のクラス)

implに全てを委譲

```
impl.insert(elm);
```

※場面が変化したら
振舞を切り替えてまわる




```
lock.get();
...insert elm...
lock.release();
```

```
s=new SeqListSet();
s.head=this.head;
return s;
```

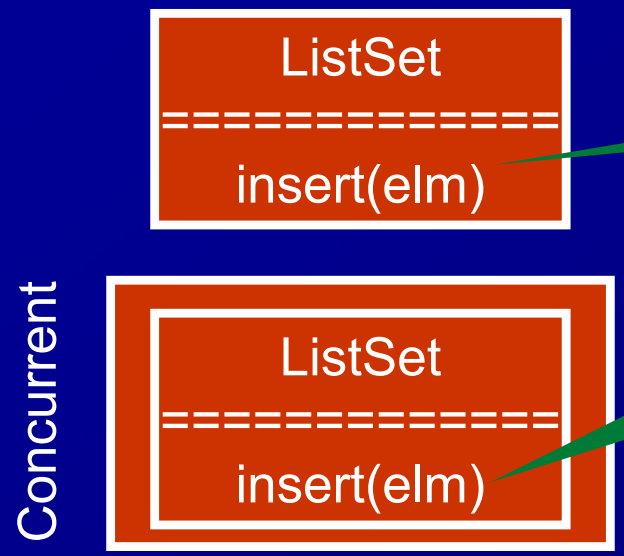
```
impl = impl.switch();
```

振舞の変化: 新しい
オブジェクトに置き換え

文脈指向プログラミング [HCN08]

- 場面ごとに異なる = 文脈依存
- 文脈依存の振舞を層の中に定義
  - mixin layersに類似
  - 同名クラスの同名メソッドを上書き
  - proceedによる上書き前の再利用 (cf. super)
- 層が活性化されているときだけ層内の定義による上書きが起きる
  - 活性化を制御する命令
  - 複数層の活性化 → 上書きの上書き

文脈指向プログラミングの例




```
head=new Cons(elm,head);
```

```
lock.get();
proceed(elm);
lock.release();
```

```
s.insert(p);
```

```
ListSet students = new ListSet();
ListSet teachers = new ListSet();
readPeople(studentDB, students);
readPeople(teacherDB, teachers);
with (Concurrent) {
  new ThreadPool(10){
    void run() {
      ...students.insert(...)...
      ...teachers.insert(...)... }}
}
```

層の活性化:
オブジェクトの振舞が
一斉に切り替わる

参考文献

[KA13] Kästner, Christian, and Sven Apel. "Feature-Oriented Software Development." Generative and Transformational Techniques in Software Engineering IV. Springer Berlin Heidelberg, 2013. 346-382.

[Reinhard14] Tartler, Reinhard, et al. "Static analysis of variability in system software: The 90,000# ifdefs issue." Proc. USENIX Conf. 2014.

[KLP02] Kang, Kyo C., Jaejoon Lee, and Patrick Donohoe. "Feature-oriented product line engineering." IEEE software 19.4 (2002): 58-65.

[YB98] Smaragdakis, Yannis, and Don Batory. "Implementing layered designs with mixin layers." ECOOP'98—Object-Oriented Programming. Springer Berlin Heidelberg, 1998. 550-570.

[HCN08] Hirschfeld, Robert, Pascal Costanza, and Oscar Nierstrasz. "Context-oriented programming." Journal of Object Technology 7.3 (2008).