17-708 SOFTWARE PRODUCT LINES: CONCEPTS AND IMPLEMENTATION

FEATURE INTERACTIONS

CHRISTIAN KAESTNER
CARNEGIE MELLON UNIVERSITY
INSTITUTE FOR SOFTWARE RESEARCH

READING ASSIGNMENT NOV 16

tbd

LEARNING GOALS

Understand the nature of feature interactions and the optional feature problem; disentangle the different meanings of the terms

Identify common sources of feature interactions

Use appropriate strategies to avoid, mitigate, or detect feature interactions

Select and apply implementation strategies for the optional feature problem

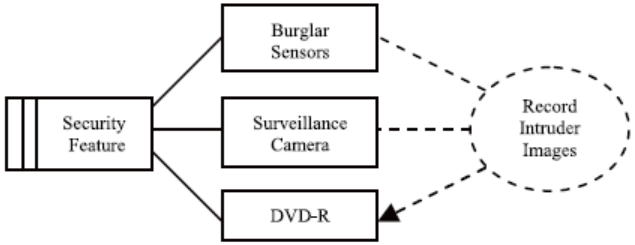


Fig. 2a Problem Diagram of Security Feature.

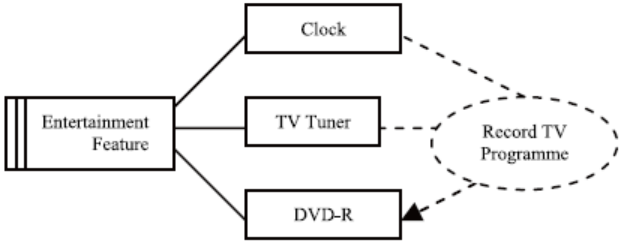


Fig. 2b Problem Diagram of Entertainment Feature.

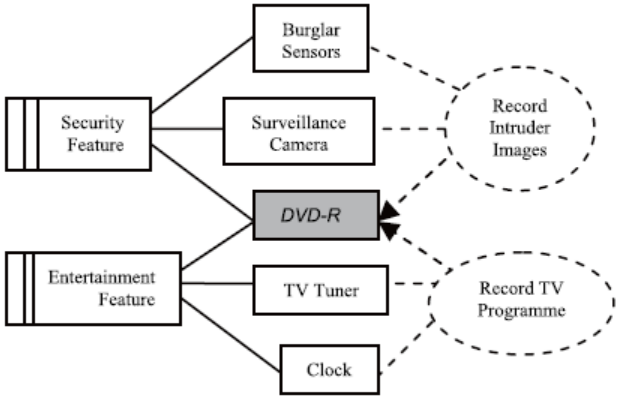


Fig. 2c Composition of Security and Entertainment Features.

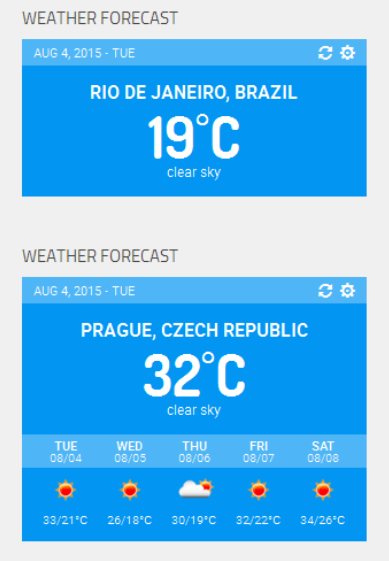
Nhlabatsi, Armstrong, Robin Laney, and Bashar Nuseibeh. "Feature interaction: the security threat from within software systems." *Progress in Informatics* 5 (2008): 75-89.


A/C
MAX A/C
USE


ENGINE
START
D
S
NAV
INFO

[:weather:]

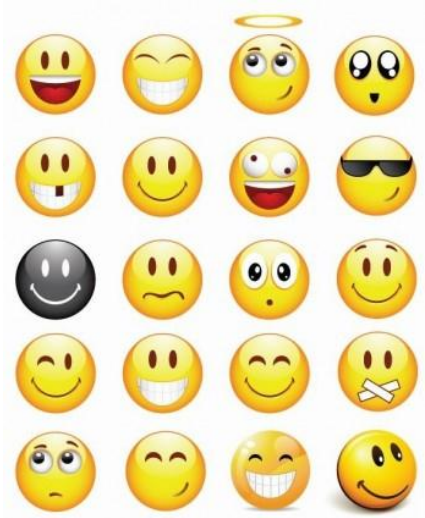
Plugin



WordPress

:) 8-) ;-) …

Plugin



Today's weather: [:weathe 

Hands-free entry
Night lock
Electronic operation
Intruder defense
…

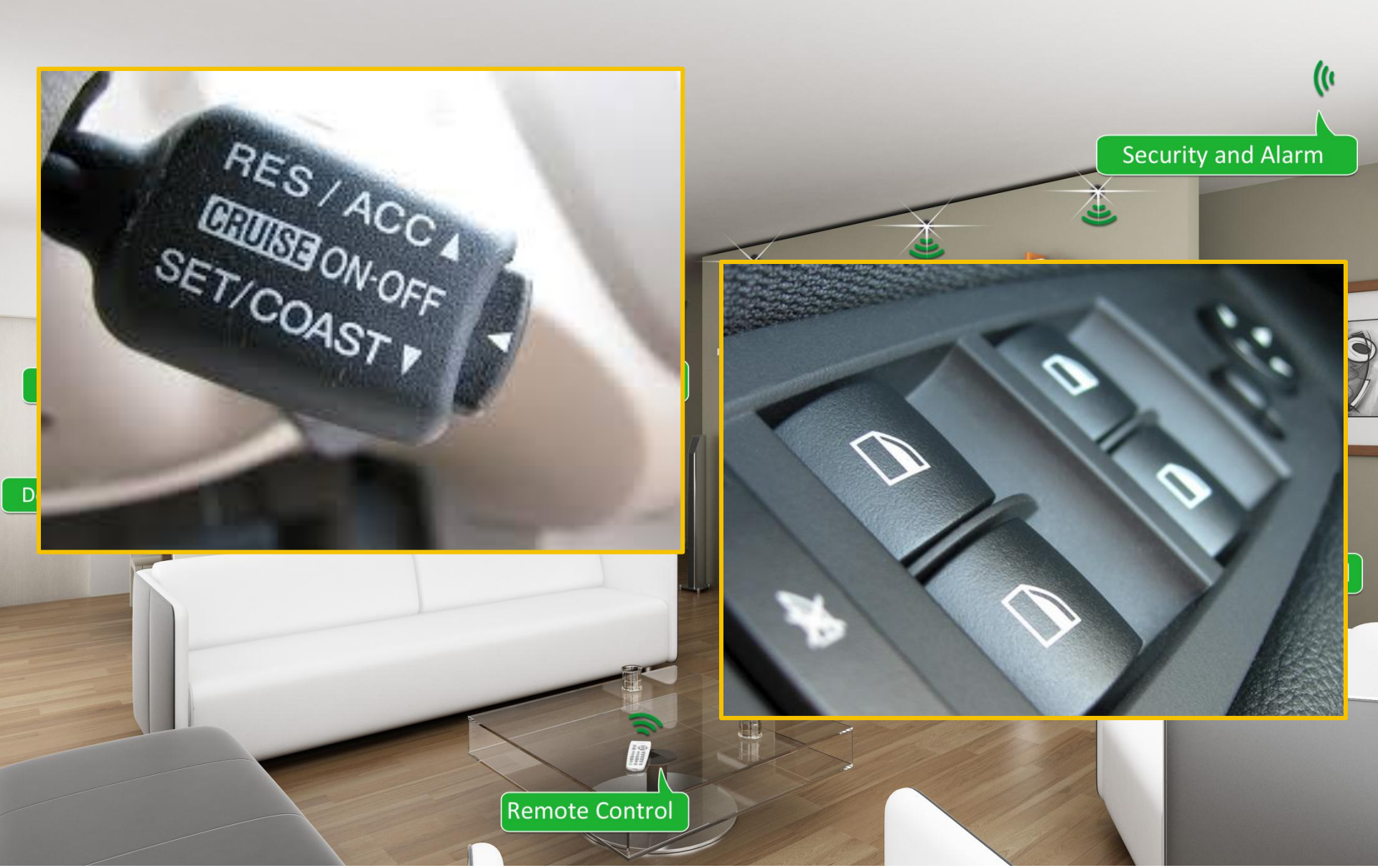
Example from P. Zav

Feature Interactions

Features designed in isolation
(divide and conquer)

Interact in intended and unintended ways
when composed

(Failure of compositionality
due to hidden underlying domain)



The physical world has no compositionality."

M. Jackson, FI Dagstuhl 2014

SOURCES

Overlapping preconditions (nondeterminism)

Requirements inconsistency

Conflicting goals

Violations of assumptions

Resource contention

Nhlabatsi, Armstrong, Robin Laney, and Bashar Nuseibeh. "Feature interaction: the security threat from within software systems." *Progress in Informatics* 5 (2008): 75-89.

INTERACTION TYPES

Nondeterminism

Dependence

Override (same precondition)

Negative impact (same precondition)

Override (linked trigger events)

Negative impact (linked trigger events)

Order

Bypass

Infinite loop

Nhlabatsi, Armstrong, Robin Laney, and Bashar Nuseibeh. "Feature interaction: the security threat from within software systems." *Progress in Informatics* 5 (2008): 75-89.

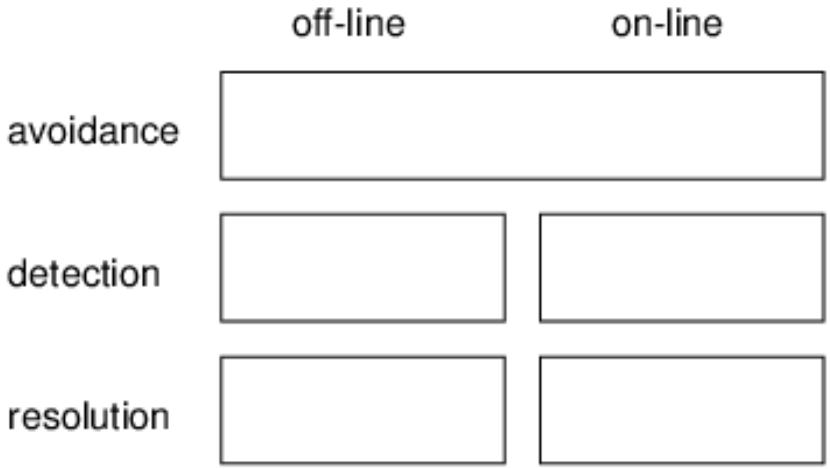


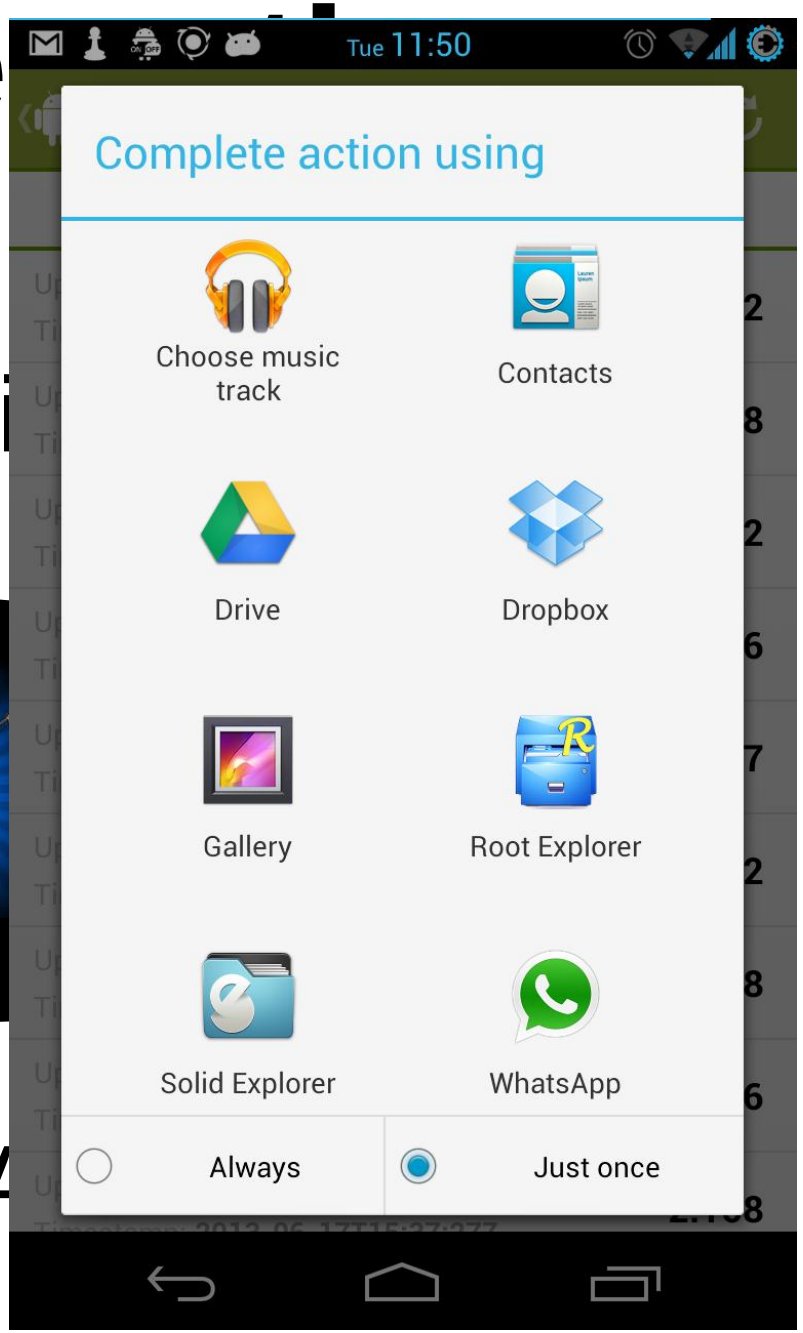
Figure 2: Categorization of approaches to addressing feature interactions (

Keck, Dirk O., and Paul J. Kuehn. "The feature and service interaction problem in telecommunications systems: A survey." *Software Engineering, IEEE Transactions on* 24, no. 10 (1998): 779-796.

Handling Inte

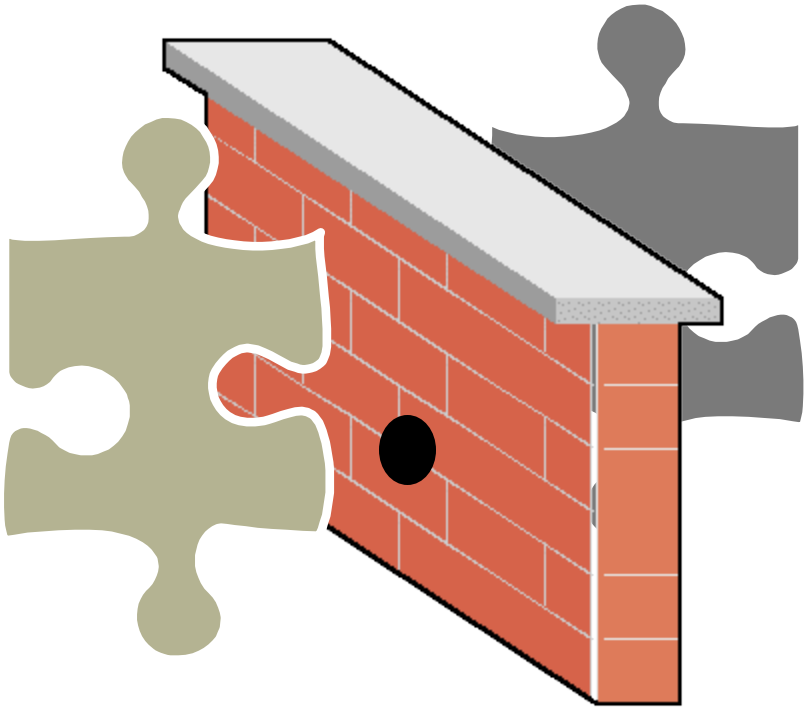
Composition mechani
resolution

(telecommunication, A
home automation)



Handling Interactions

Isolation, noninterference (to some degree)

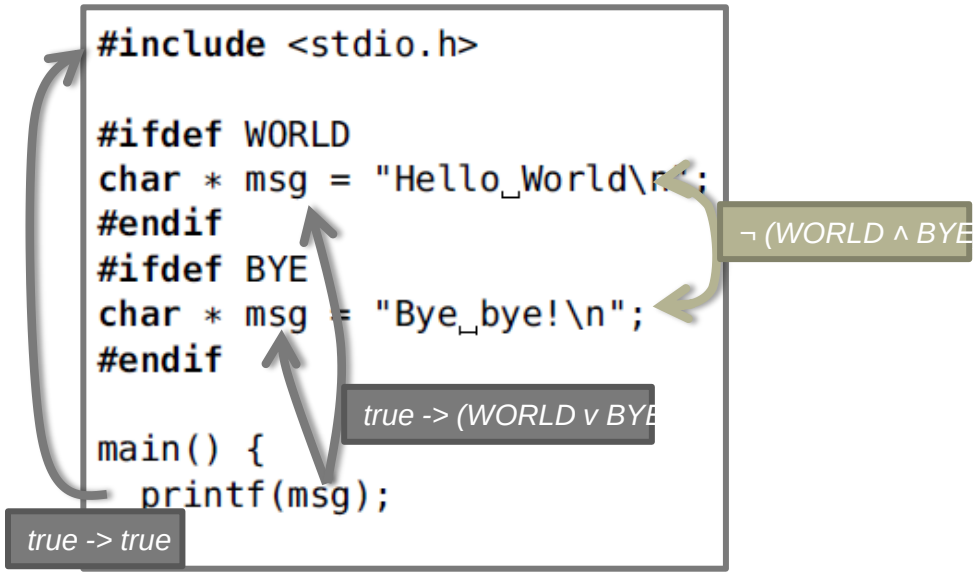


(Android, Kernel drivers, browser plugins)

Handling Interactions

Detection in requirements or implementations

```
127    public void timeShift() {
128        if (Configuration.overloaded) {
129            if (areDoorsOpen() && weight > maximumWeight) {
130                blocked = true;
131                if (Configuration.verbose) {
132                    System.out.println("Elevator blocked due to overloading (weight:" + w
133                }
134                return;
135            } else {
136                blocked = false;
137            }
138        }
139        if (stopRequestedAtCurrentFloor()) {
140            doors = DoorState.open;
141            // iterate over a copy of the original list, avoids concurrent
142            // modification exception
143            for (Person p : persons) {
144                if (p.getDestination() == currentFloorID) {
145                    leaveElevator(p);
146                }
147            }
148            env.getFloor(currentFloorID).processWaitingPersons(this);
149            resetFloorButton(currentFloorID);
150        } else {
151            if (doors == DoorState.open) {
152                doors = DoorState.close;
153            }
154            if (stopRequestedInDirection(currentHeading, true, true)) {
155                // continue
156                continueInDirection(currentHeading);
157            } else if (stopRequestedInDirection(currentHeading.reverse(), true, true)) {
158                // revert direction
159                continueInDirection(currentHeading.reverse());
160            } else {
161                // idle
162                continueInDirection(currentHeading);
163            }
164        }
165    }
```



DETECTION

Formal methods

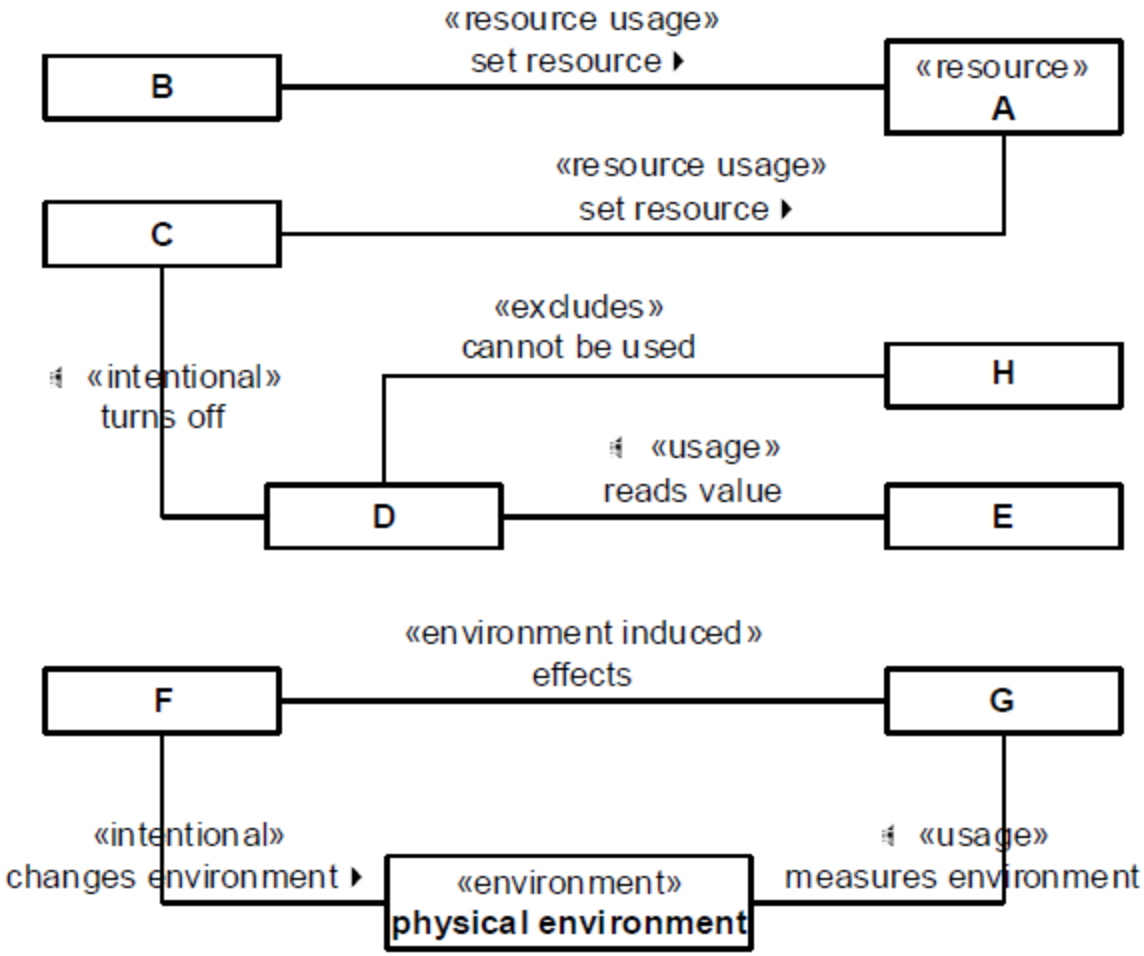
- **Model checking**
- **Detecting overlapping preconditions**
- **…**

Requirements

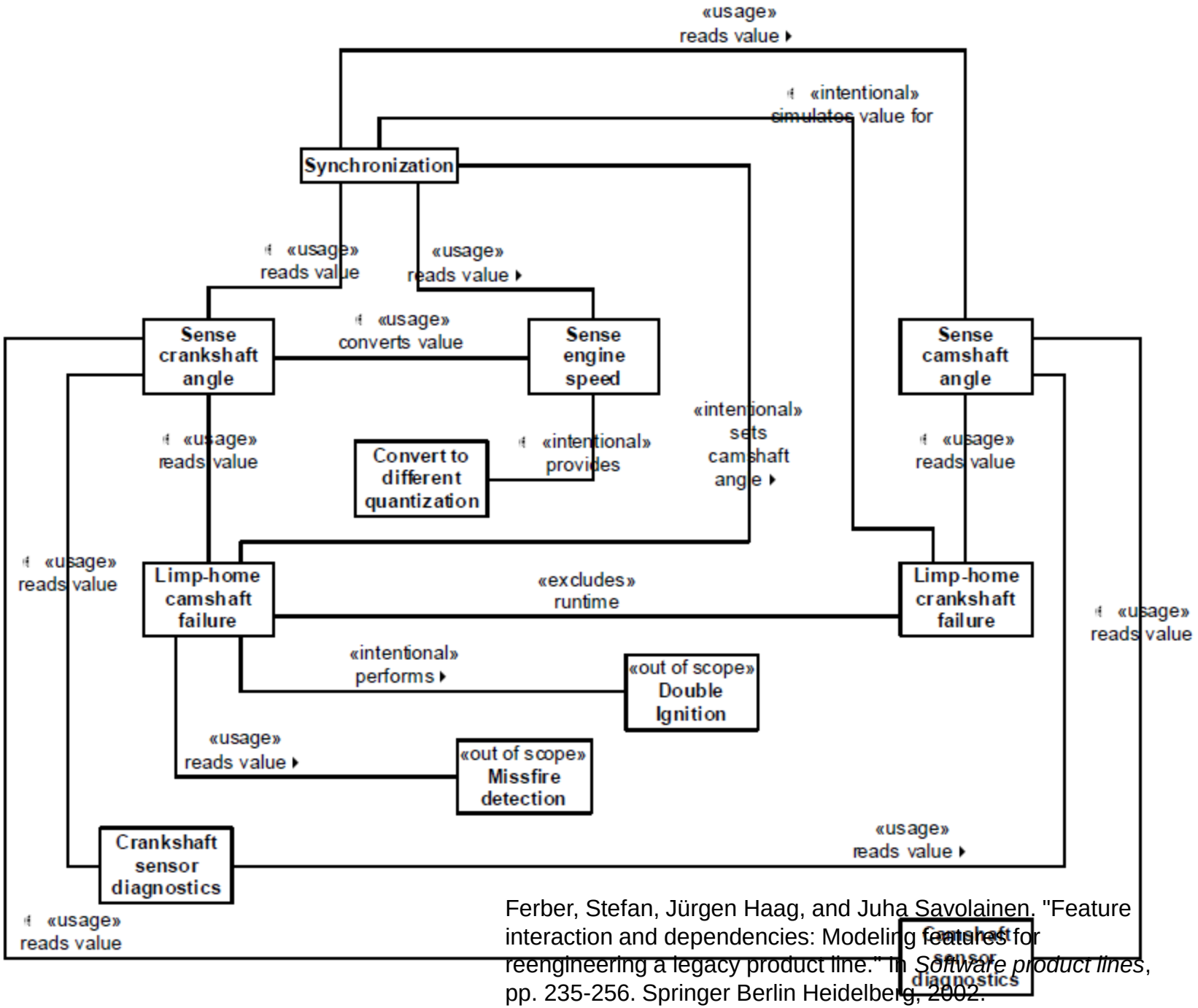
- **Identifying shared resources**

- At the requirements level, a typical strategy is to systematically search for shared resources. Two features that share resources may potentially interact over this resource. For example, the features FireControl and FloodControl from Example 9.2 both affect the resource water supply. A typical strategy is to model all resources relevant for each feature and subsequently investigate manually all pairs of features that share a resource.
- The strategy applied to resources can be used also for events (and preconditions of operations). Two features that react to the same event (or that have overlapping preconditions) are potential candidates for feature interactions. For example, the features CallForwarding and CallWaiting from Example 9.1 both react to the same event (that is, an incoming call on a busy line). Again, modeling events allows us to manually investigate all pairs of features reacting to the same event.
- Inconsistent requirements and conflicting goals of features revealed during domain engineering can also be an indicator of potential interactions. For example, features Acceleration to increase the speed of a car and AdaptiveCruiseControl to automatically adjust the distance to other cars by decreasing speed have conflicting goals. Again, requirements and goals need to be made explicit, for example, by modeling them.
- Making assumptions (or invariants) of features explicit can help detecting when an assumption is violated by other features. For example, feature Index in Example 9.3 assumes that the data structure is immutable, an assumption violated by feature Write.

Apel, Sven, Don Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines*. Berlin: Springer, 2013.



Ferber, Stefan, Jürgen Haag, and Juha Savolainen. "Feature interaction and dependencies: Modeling features for reengineering a legacy product line." In *Software product lines*, pp. 235-256. Springer Berlin Heidelberg, 2002.



Ferber, Stefan, Jürgen Haag, and Juha Savolainen. "Feature interaction and dependencies: Modeling features for reengineering a legacy product line." In *Software product lines*, pp. 235-256. Springer Berlin Heidelberg, 2002.

ONLINE TECHNIQUES

- *Feature Manager* based approaches. An entity, usually called the feature manager, is introduced into the network with the capability of observing and controlling the call processes. Hence, the control of the call is located with the feature manager. So far mainly centralised approaches featuring a single feature manager have been developed. However, distributed architectures for managers are also possible.
- *Negotiation* based approaches. Individual features have the capability of communicating their intentions to each other and negotiating an acceptable resolution. Most approaches advertise a direct communication where the call control resides with the features. However, if no resolution is possible the conflict can be forwarded to a third party to resolve.

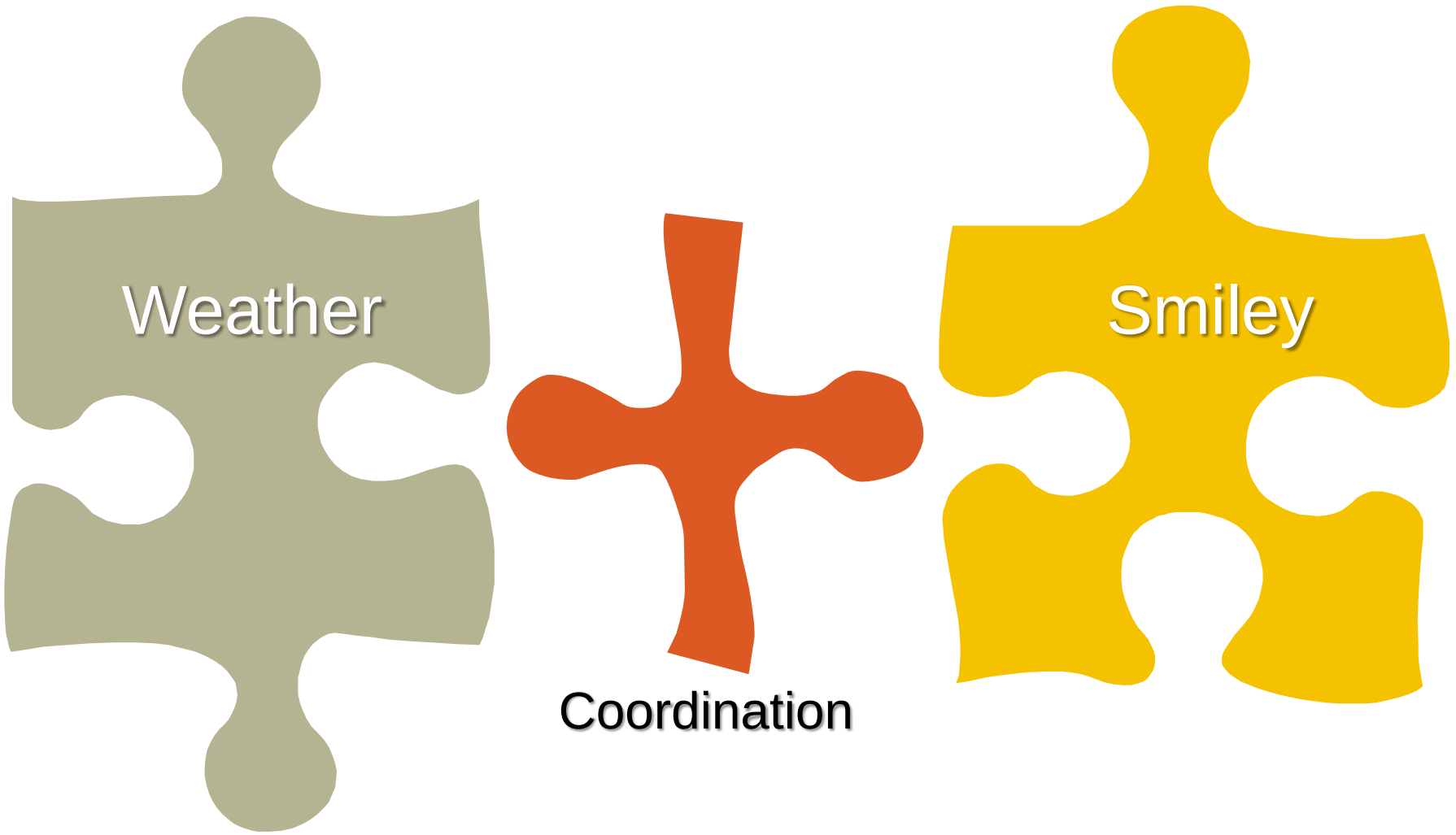
Calder, Muffy, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. "Feature interaction: a critical review and considered forecast." *Computer Networks* 41, no. 1 (2003): 115-141.

OPTIONAL FEATURE PROBLEM

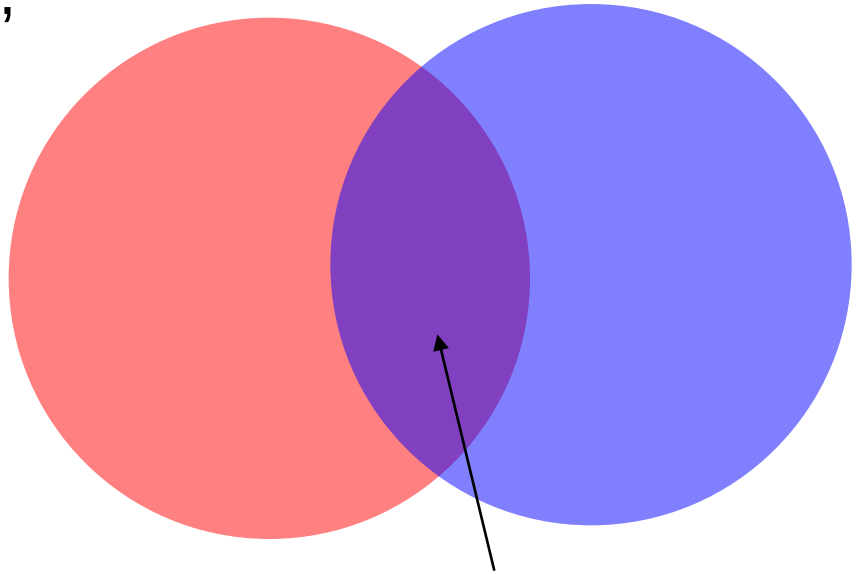
OPTIONAL FEATURE PROBLEM

Code focused view – how to implement coordination code between two features

Applicable only once interactions identified and detected


Weather
Smiley
Coordination

Statistics
(buffer hit ratio,
table size and
cardinality, …)

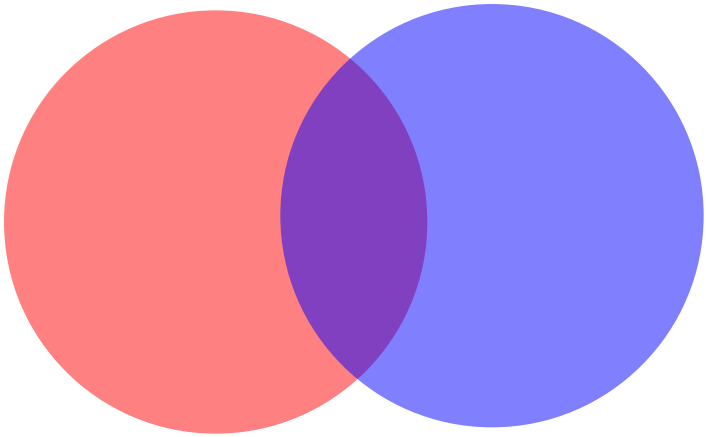


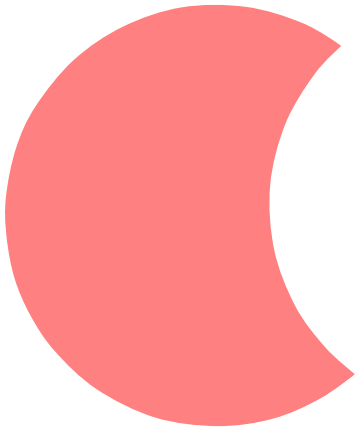
Transactions
(locks, commit,
rollback, …)

Transactions per second

PRODUCTS

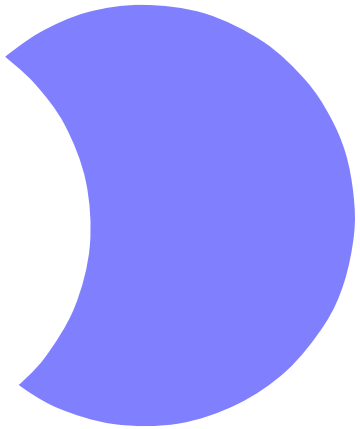
DB with statistics and transactions





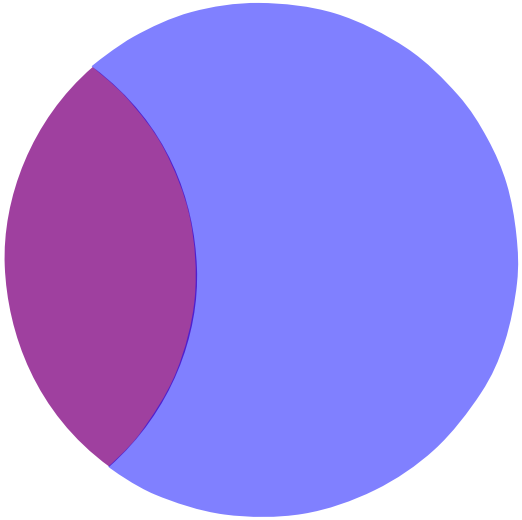
DB with statistics without transactions

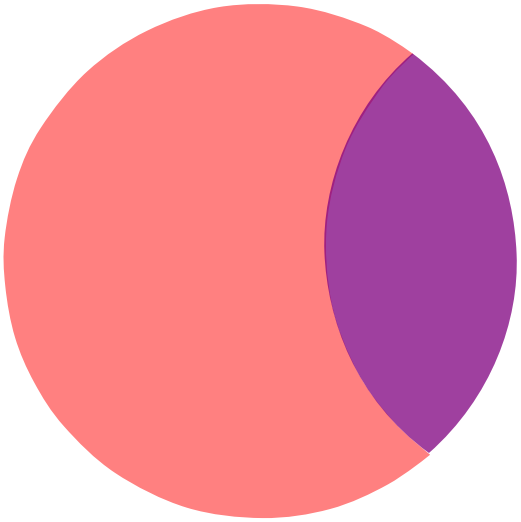
DB with transactions, without statistics



UNDESIRED PRODUCTS

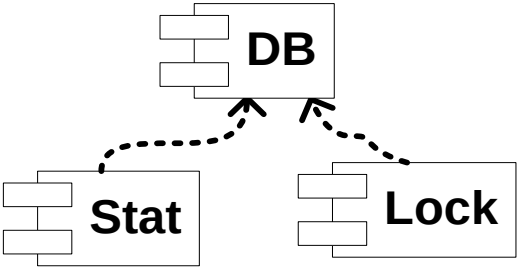
DB with transactions without statistics measuring transactions per second (larger and slower than necessary)





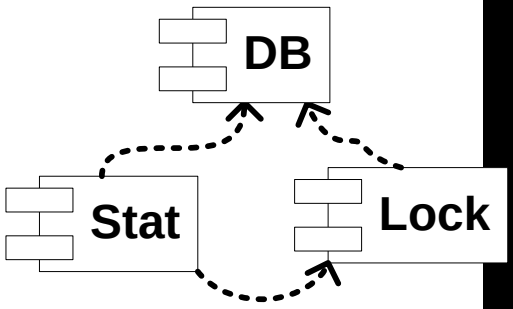
DB with statistics without transactions trying to measure transactions per second(?)

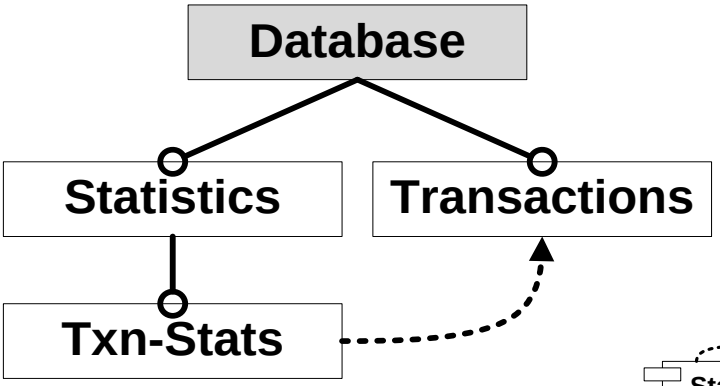
MODULES



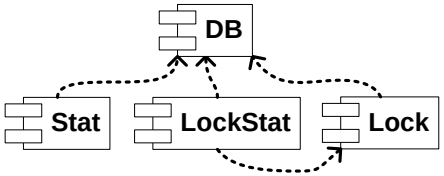
Where to implement transactions/sec

How to create product without transactions but with statistics

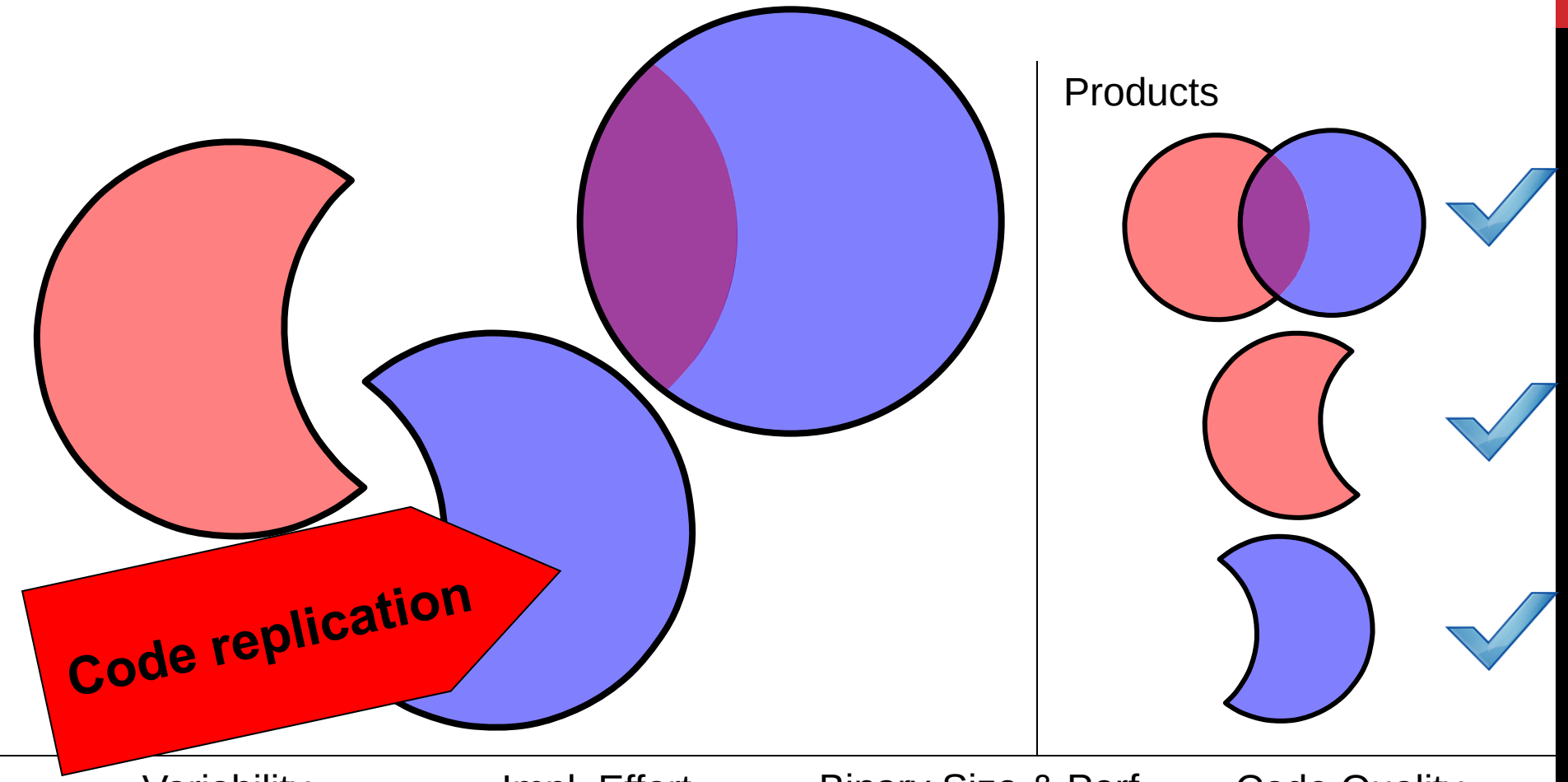




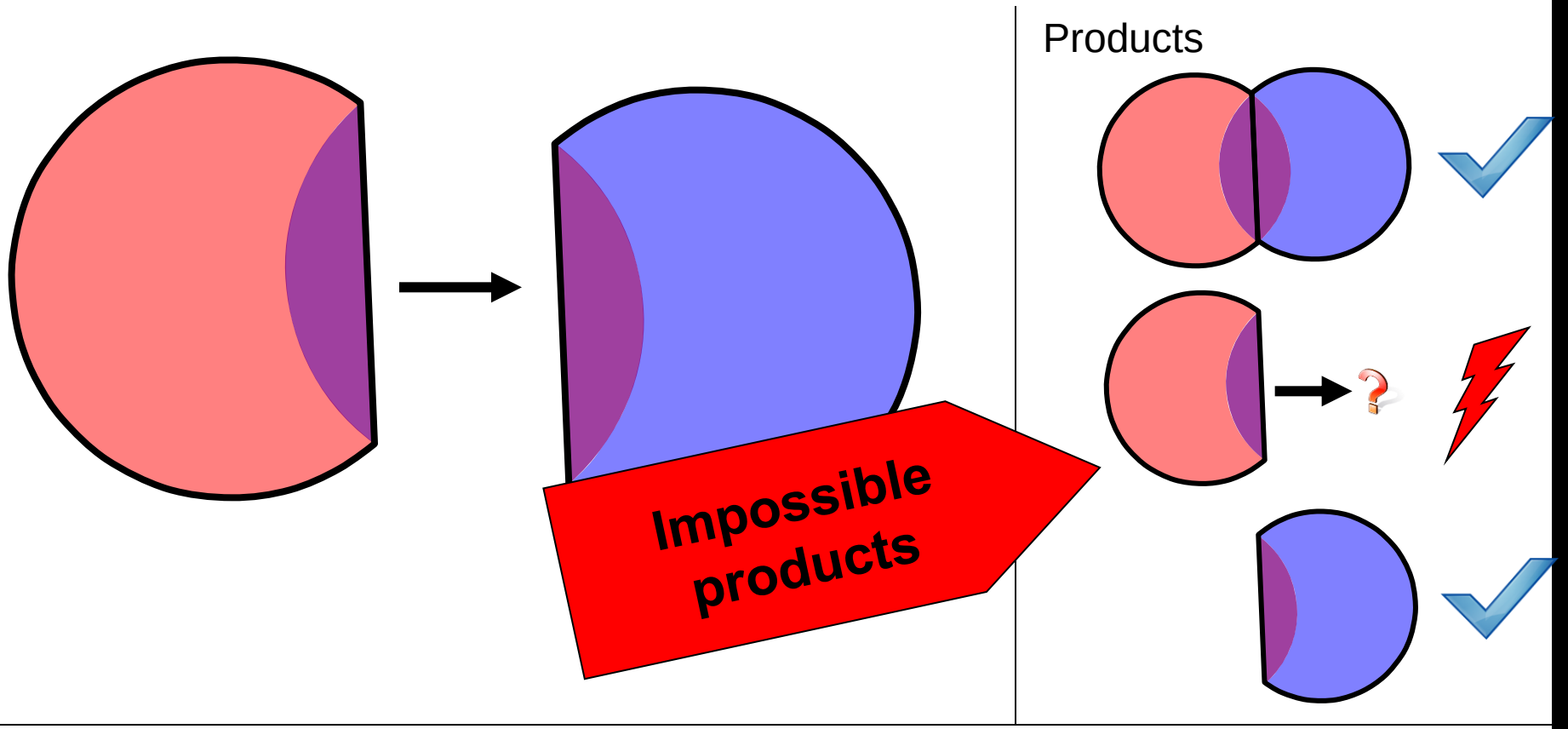
transactions/sec really a feature?



MULTIPLE IMPL.

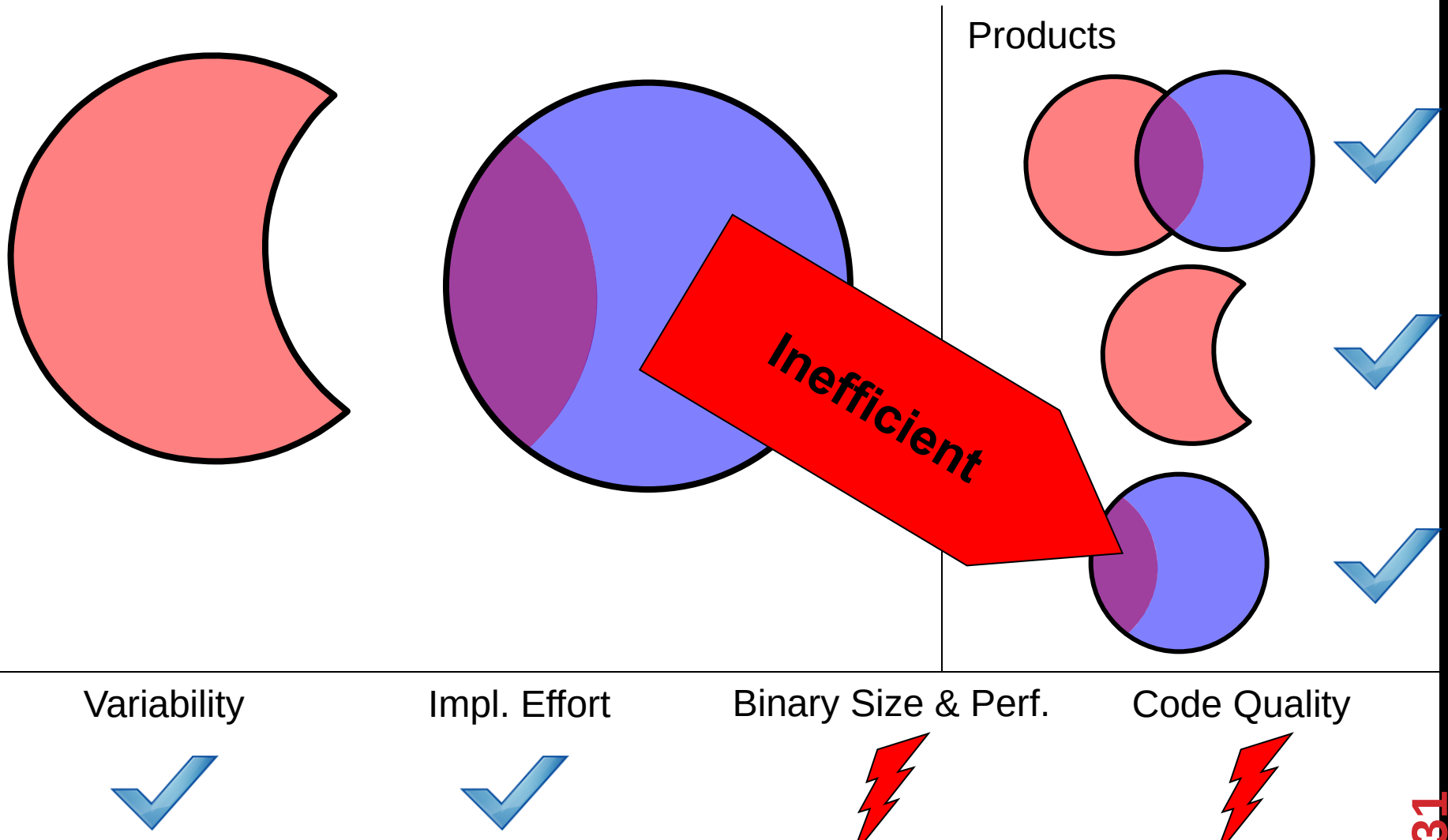


DOCUMENT DEPENDENCIES

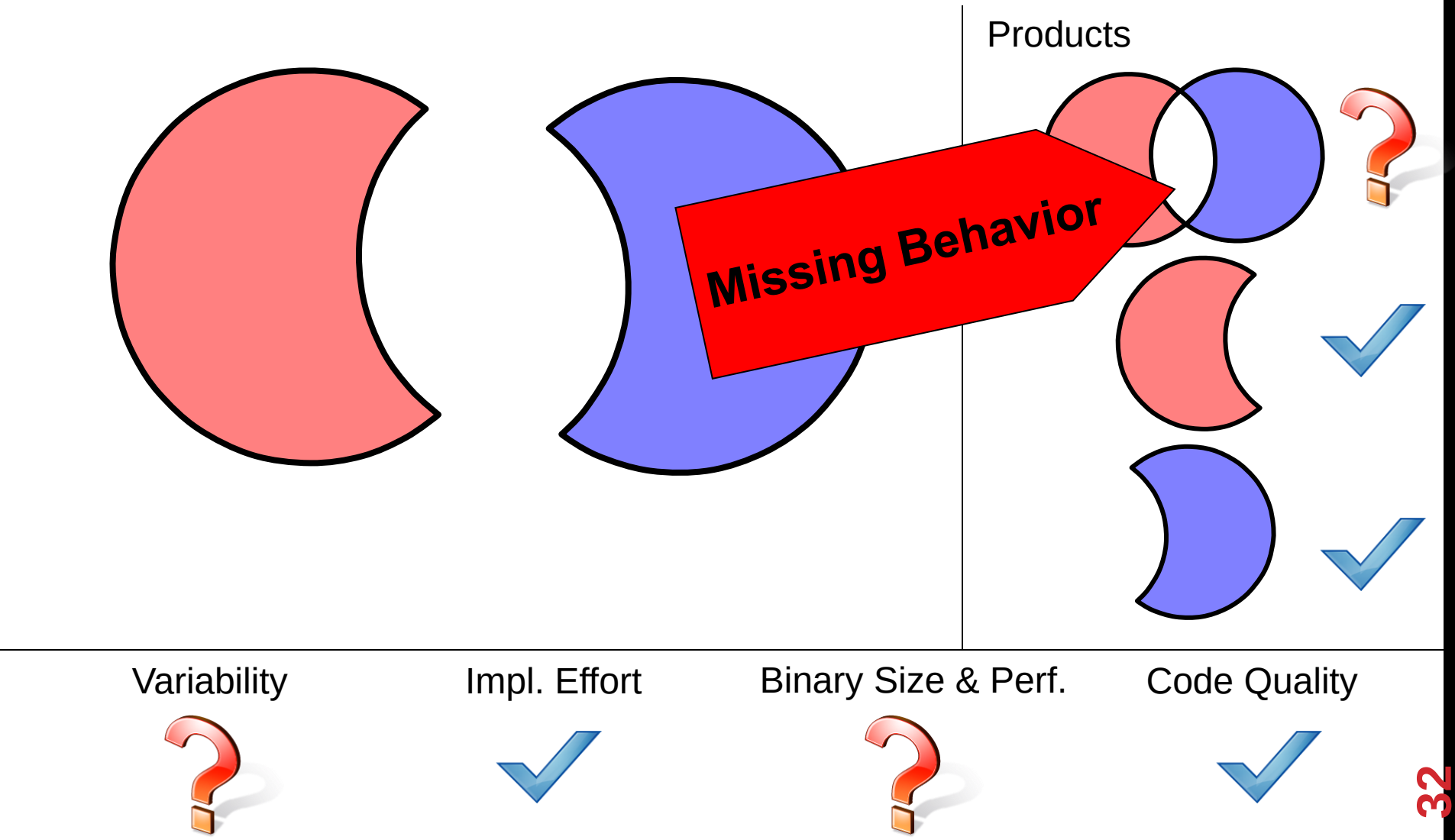


Variability	Impl. Effort	Binary Size & Perf.	Code Quality

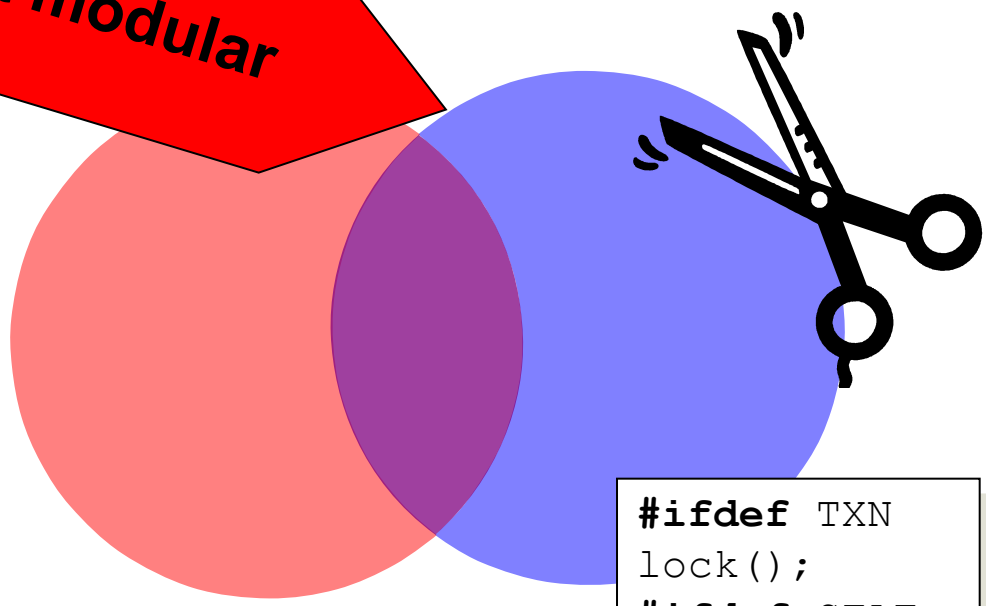
MOVING SOURCE CODE



CHANGE BEHAVIOR

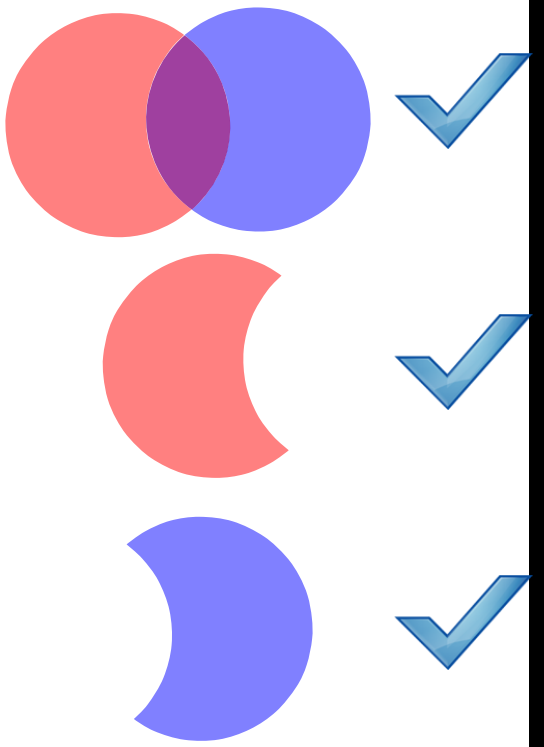


PREPROCESSOR



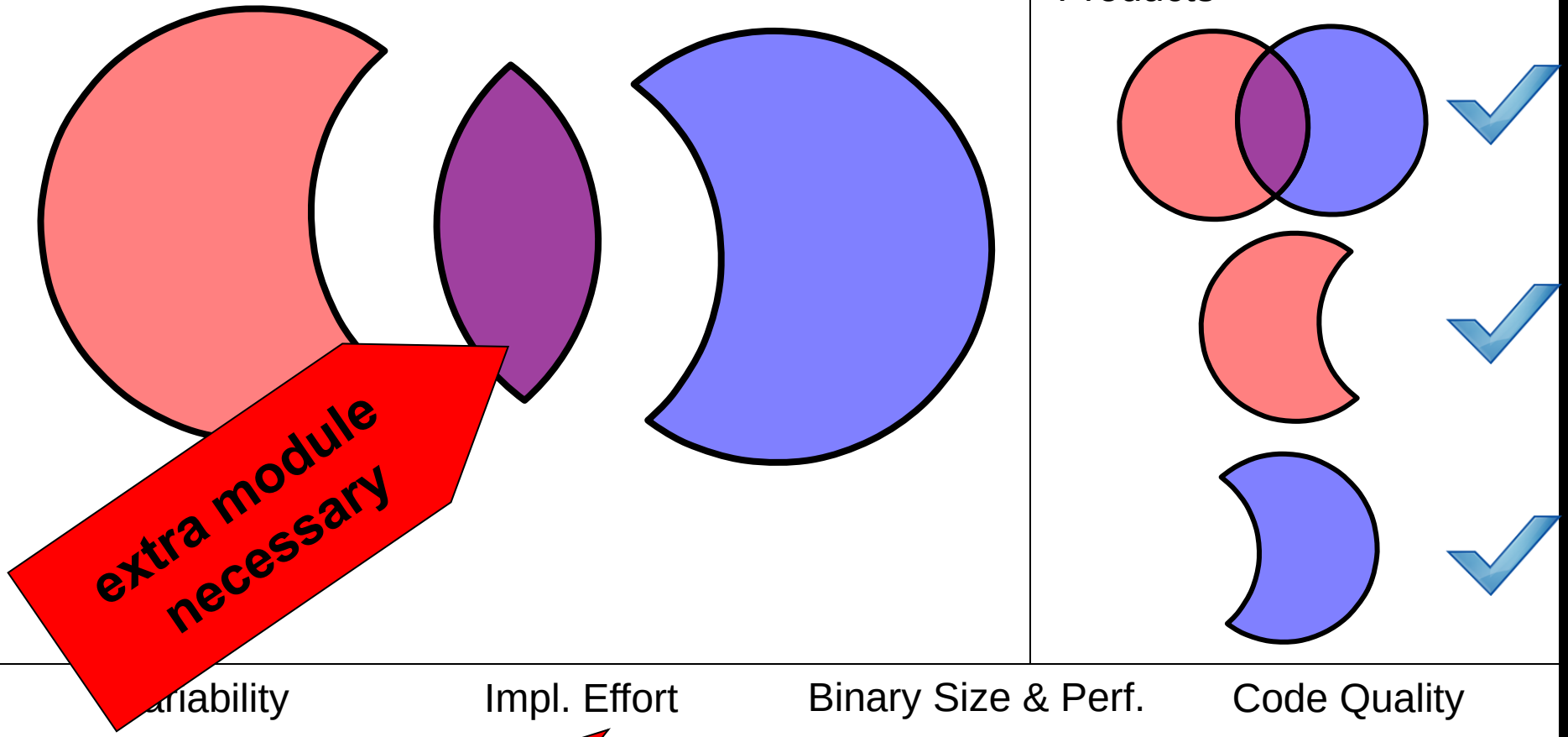
```
#ifdef TXN
lock();
#ifdef STAT
lockCount++;
#endif
#endif
```

Products



Variability	Impl. Effort	Binary Size & Perf.	Code Quality
✓	✓	✓	⚡⚡

GLUE CODE MODULES



OVERVIEW

Lösung	Variability	Effort	Size & Perf.	Quality
Multiple impl				
Dependency				
Move source code				
Change behavior				
Preprocessor				
Glue code modules				

EXPERIENCE BERKELEY DB

Dependencies?

- Would render important features de-facto mandatory

Change behavior?

- undesired

Glue code module?

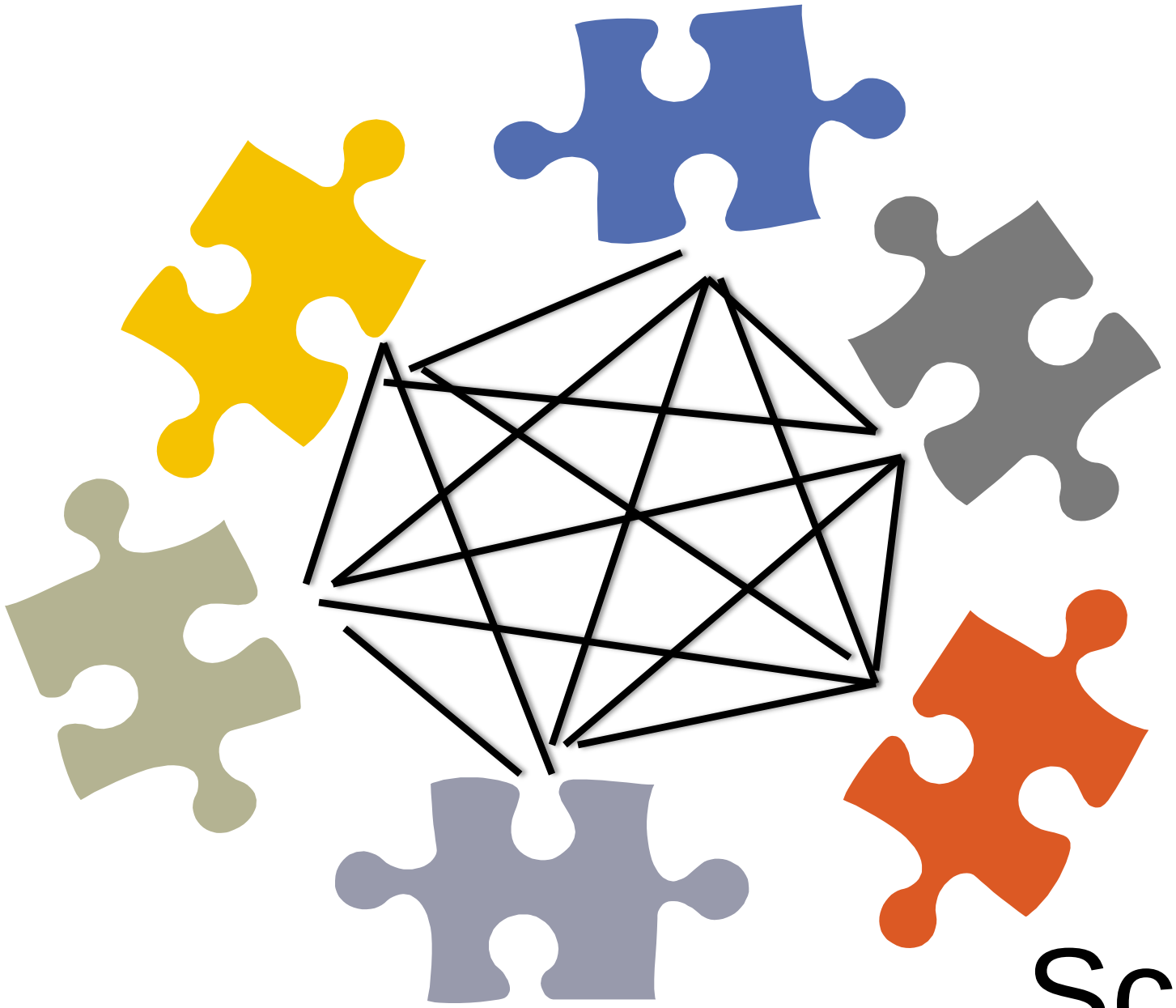
- Extracted 76% of statistics code into 9 modules
- → possible but labor intensive

Preprocessor

- Faster, easier
- Scattered code

Scale

Scale



Scale

REMINDER: VARIABILITY MANAGEMENT

see mass customization in automotive

Identify relevant variability

Reduce unnecessary variation to avoid interactions and optional code problems

FURTHER READING

Calder, Muffy, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. "Feature interaction: a critical review and considered forecast." *Computer Networks* 41, no. 1 (2003): 115-141.