2 claims

- what's new here
  - it is not separation of concerns
    - that is an old and well-established goal
  - it is not separation of concern technology
    - procedures, OOP, etc. (N.B. these work very well)
  - it isn't just any kind of tangling
  - ➢ solving tangling due to ***crosscutting concerns***
- "no dominant decomposition"
  = "strong AOP "
  = no time soon (if at all)
  - ➢ can support modular crosscutting code…
    …even with a dominant decomposition
    …that's the goal of AspectJ

error logging in AspectJ

work at PARC

```
aspect PublicErrorLogging {

  static Log log = new Log();

  pointcut publicInterface ():
    receptions(public * com.xerox..*.*(..));

  static after() throwing (Error e): publicInterface() {
    log.write(e);
  }
}
```

public methods in com.xerox package

FrontDesk
tellHouseKeeping checkIn checkout callSecurity

RoomService
acceptOrder deliver tellCook pickupTrays

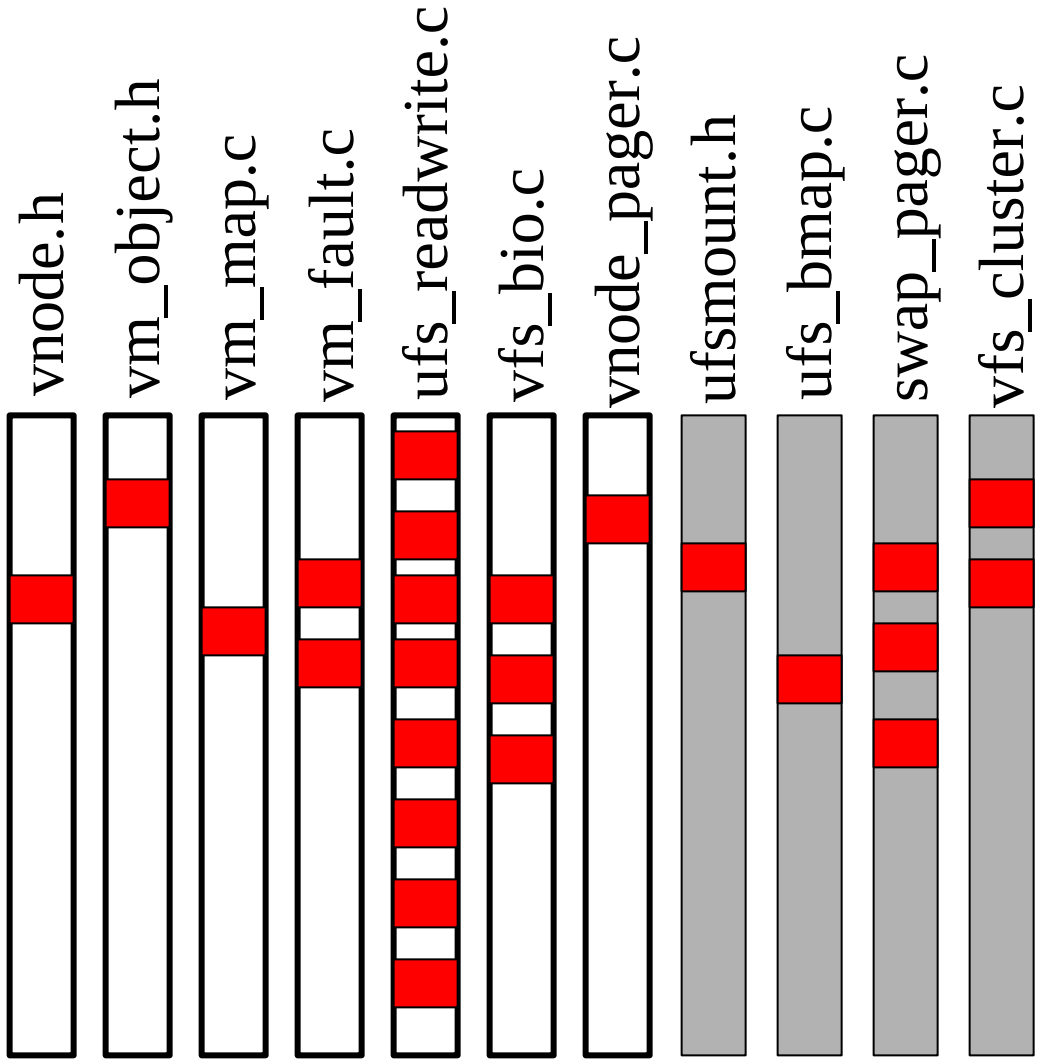
Cafe
seat takeOrder water giveCheck

with respect to functionality: 8 public operations

with respect to error logging: 1 public operation

pre-fetching in C

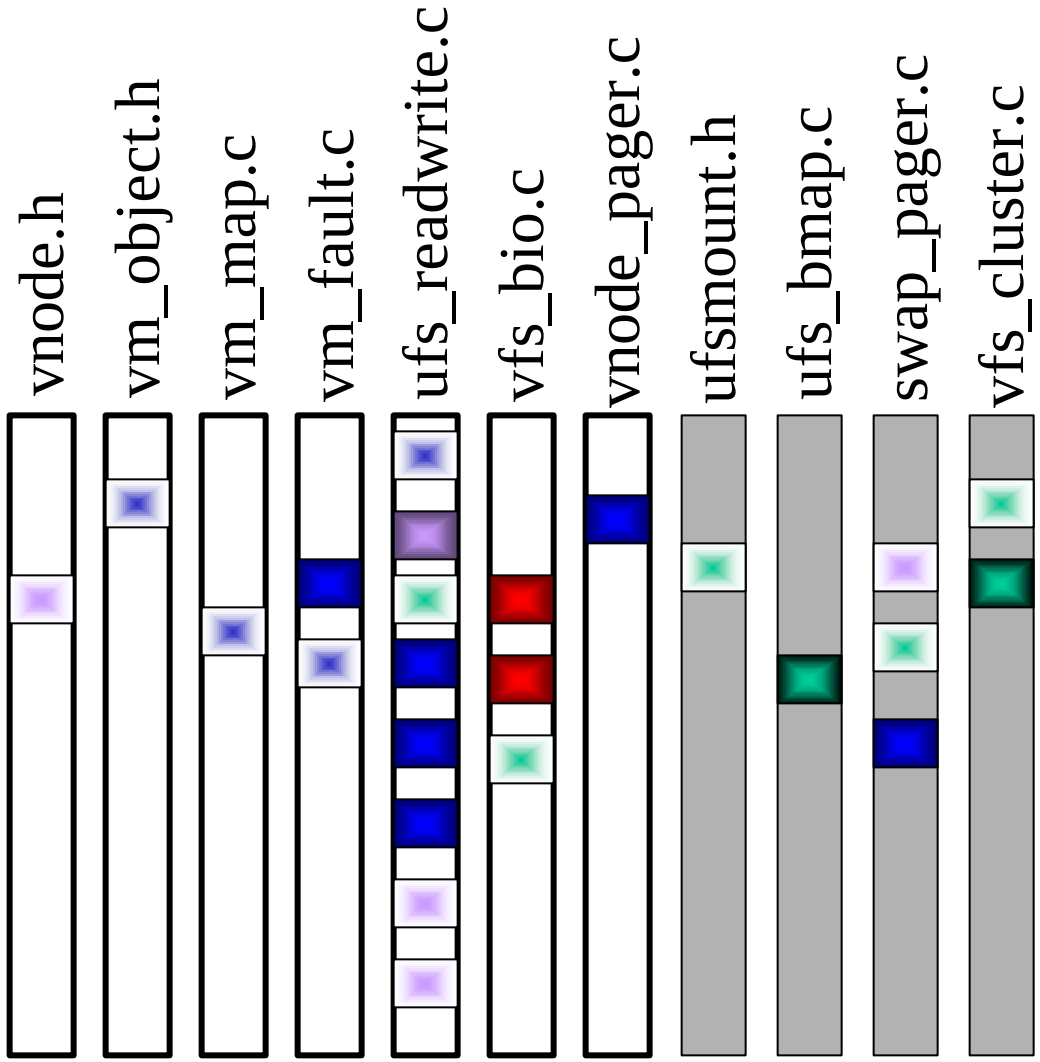
work at UBC



tangling becomes evident by just flagging all code that has to do with pre-fetching

pre-fetching in C

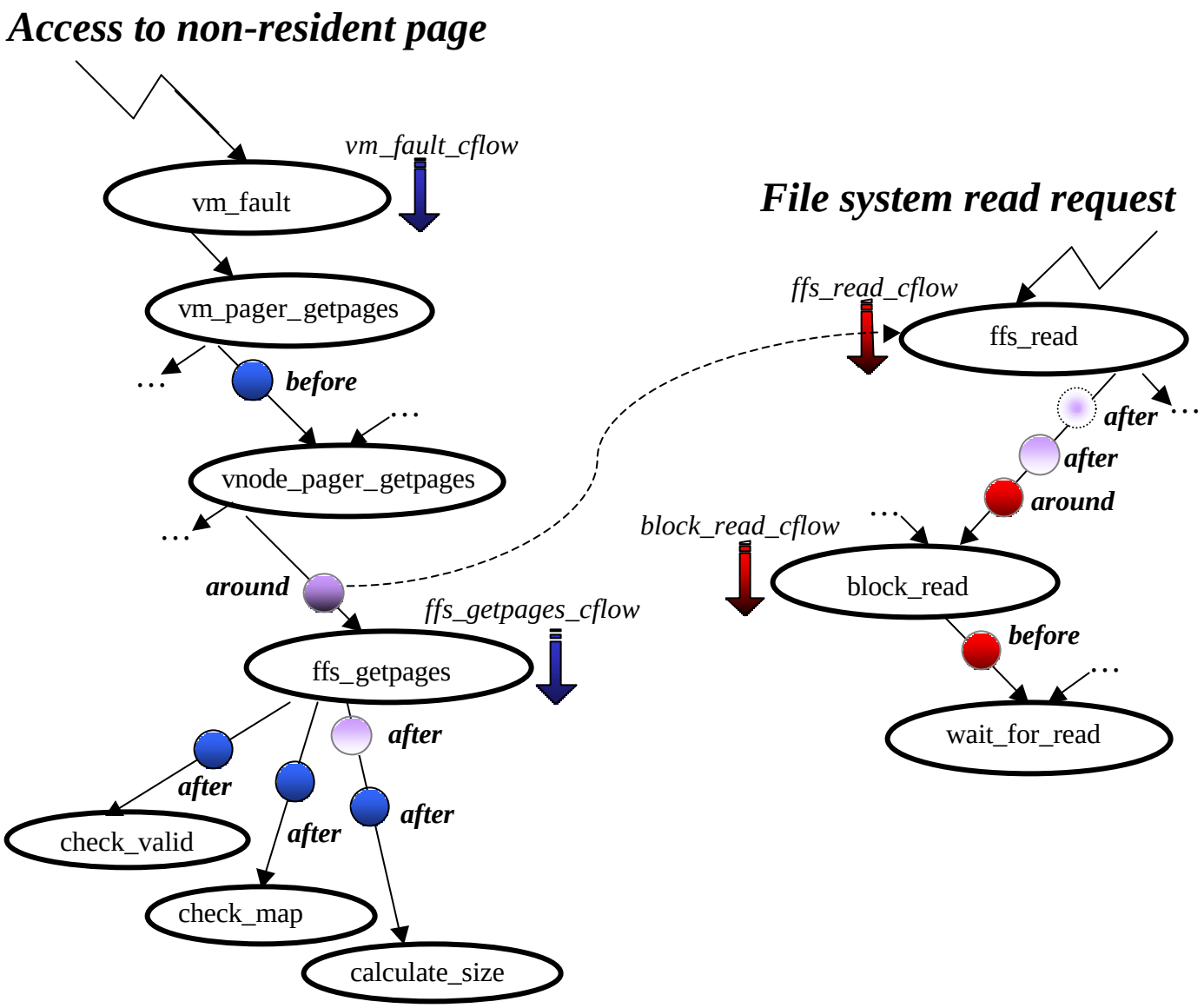
work at UBC



structure of crosscutting concerns emerges by sorting pre-fetching into modular sub-units

pre-fetching in ‘AspectC’

work at UBC

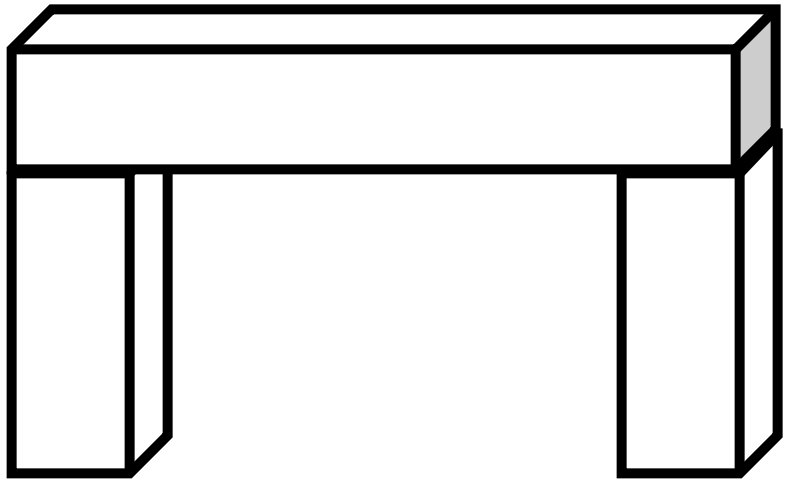


2 claims

- what's new here
  - it is not separation of concerns
    - that is an old and well-established goal
  - it is not separation of concern technology
    - procedures, OOP, etc. (N.B. these work very well)
  - it isn't just any kind of tangling
  - ➢ solving tangling due to ***crosscutting concerns***
- "no dominant decomposition"
  = "strong AOP "
  = no time soon (if at all)
  - ➢ can support modular crosscutting code…
    …even with a dominant decomposition
    …that's the goal of AspectJ

slide from OOPSLA’94

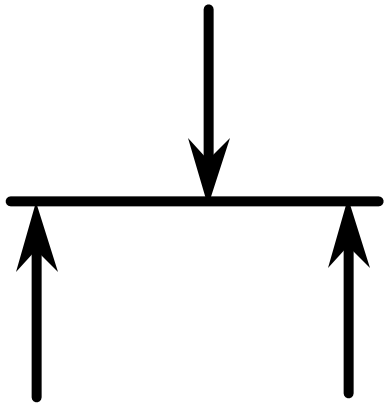
an analogy



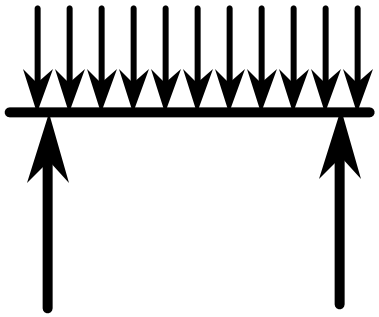
designing and building a simple bridge...

slide from OOPSLA'94

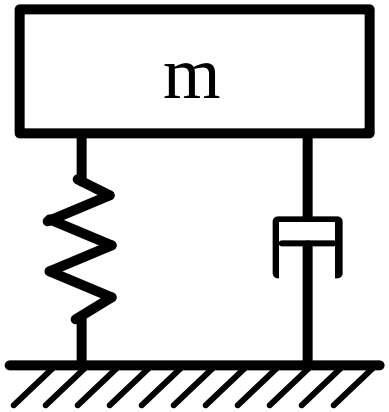
different kinds of picture



simple
statics



more
detailed
statics

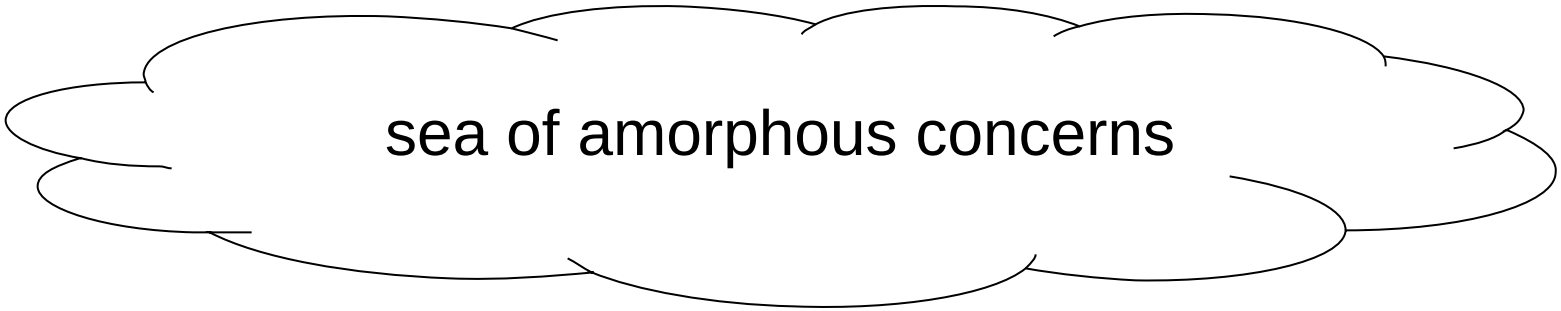


simple
dynamics

2 claims

- what's new here
  - it is not separation of concerns
    - that is an old and well-established goal
  - it is not separation of concern technology
    - procedures, OOP, etc. (N.B. these work very well)
  - it isn't just any kind of tangling
  - ➢ solving tangling due to ***crosscutting concerns***
- "no dominant decomposition"
  = "strong AOP "
  = no time soon (if at all)
  - ➢ can support modular crosscutting code…
    …even with a dominant decomposition
    …that's the goal of AspectJ

P.S. concerns, aspects…



---design---	---code---
modular design units (for encapsulating concerns): object, method also use case, pattern…	modular code units (for encapsulating concerns): class, method
aspect (ala AOD)	aspect, pointcut, advice, method
hyperslice	hyperslice