



Assessing the Malleability of Modular Design

Giuseppe (Peppo) Valetto

Drexel University - Dept. of Computer Science

Rationale

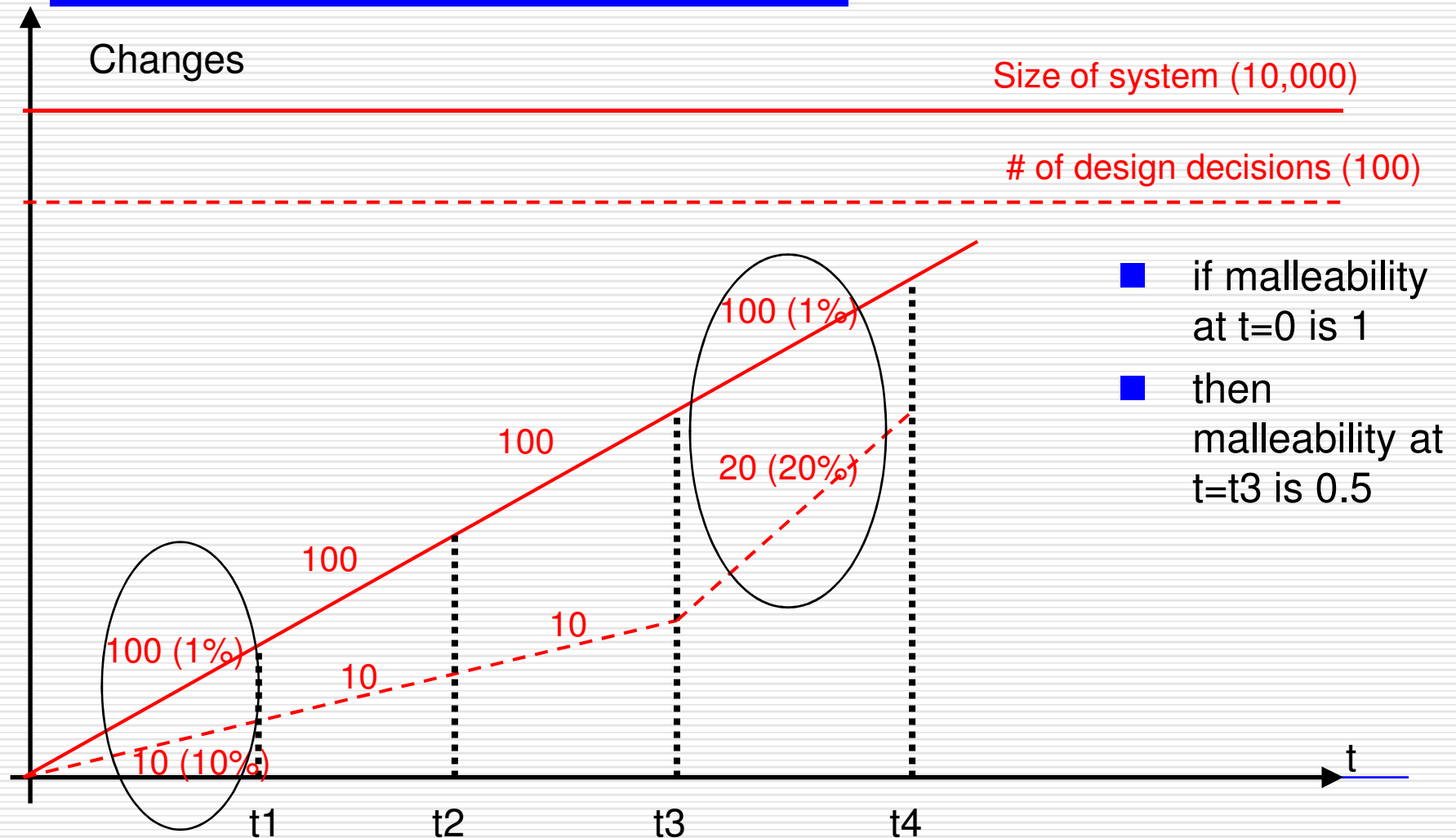
- SW change continuously threatens to obfuscate the design of the system
 - Well-known “SW aging” phenomenon
 - Remedies can be process-based
 - Compliance: aging control
 - Agility: continuous rejuvenation
 - **Remedies can also be product-based**
 - Good design should be able to withstand the “injury of time”
-

Malleability of design

- Good design should be malleable
 - In the “bend-but-not-break” sense
 - *Malleability: ability to accommodate lots of system changes with little need for design modification*

 - Can we measure the malleability of a given design?
 - Can we say a design is more malleable than another?
 - Can we say the same thing for a design paradigm?
-

Measuring malleability



- if malleability at $t=0$ is 1
- then malleability at $t=t_3$ is 0.5

Defining malleability operationally

- What is system change?
 - Perhaps use “implemented changes”
 - Code churn: mod. + added + deleted LOCs
 - *Relative code churn*: provides the rate of implementation change in a given time interval

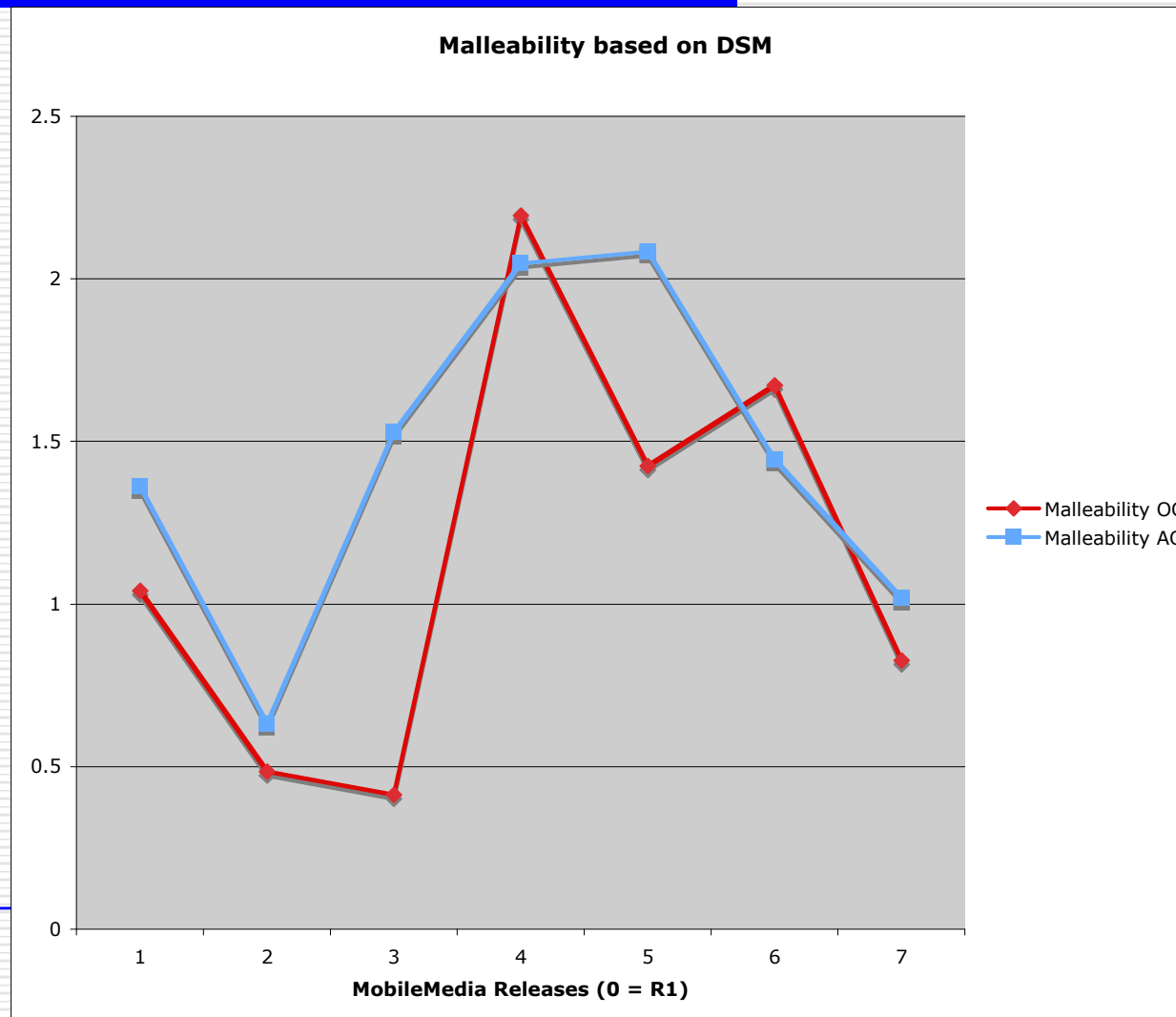
 - What is design change?
 - Count modifications in set of design decisions
 - ➔ Specific to the modularization mechanism adopted in the project
-

Experiment: MobileMedia

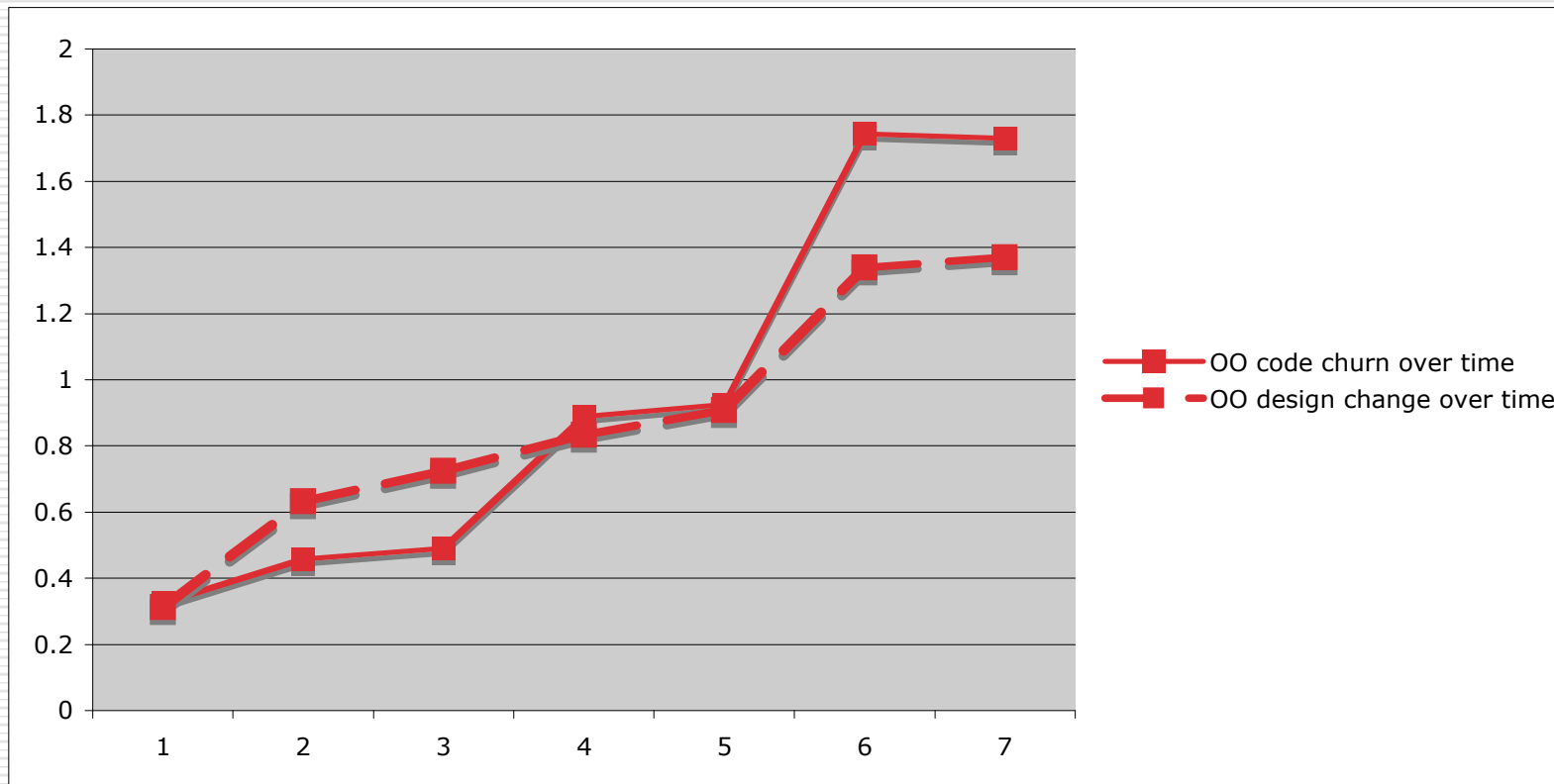
- Measured relative code churn in between pairs of releases
 - OO (Java) and AO (AspectJ) versions
 - Then counted corresponding design modifications
 - Used the Design variables defined for the DSMs of those releases

 - Both normalized
 - Malleability measured as the inverse of design change over relative code churn
-

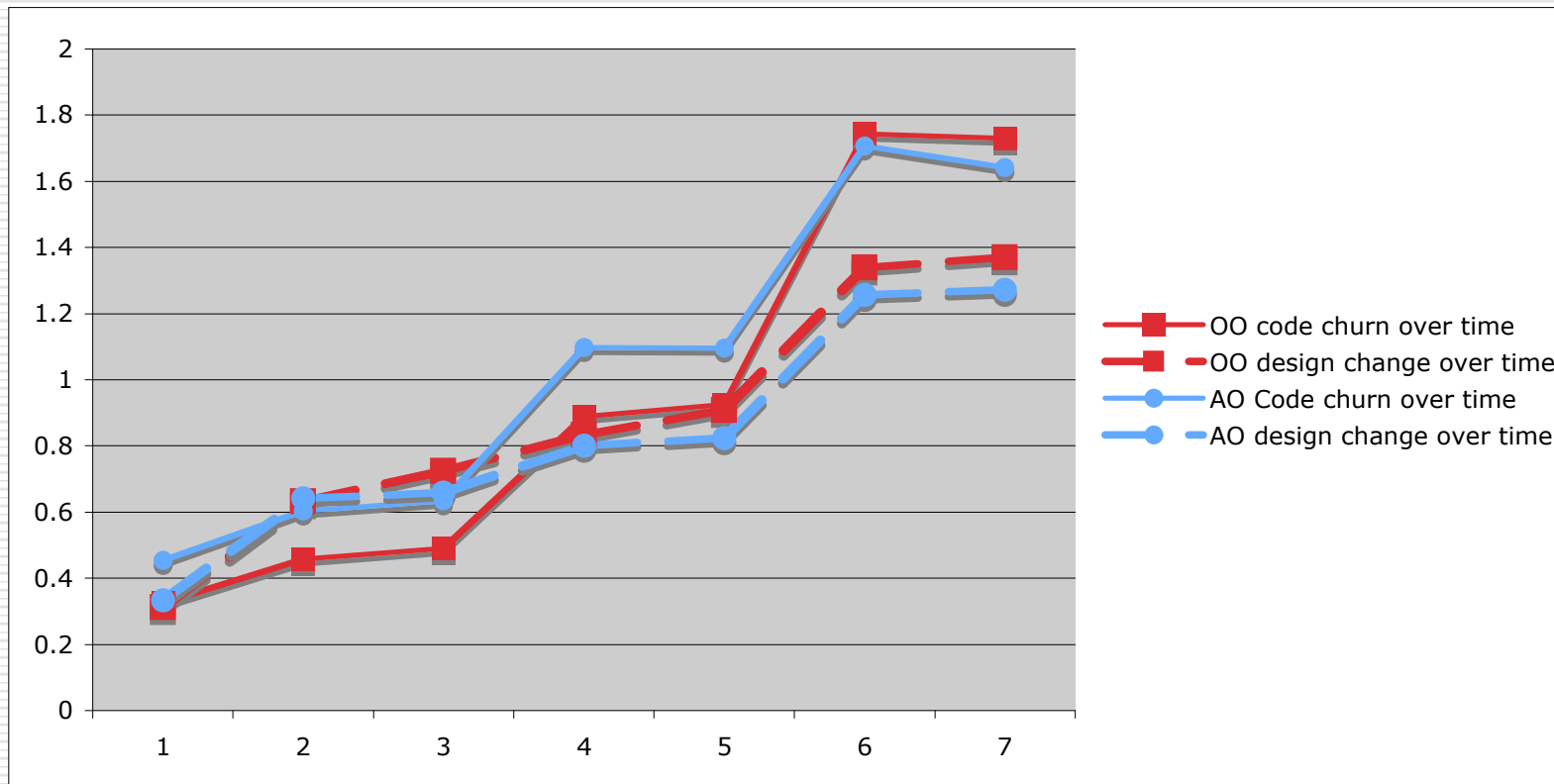
Experiment: MobileMedia



Interesting - but ...



Interesting - but ...



Considerations

- Measuring size?
 - Same set of features for OO and AO
 - Code churn depends on programming paradigm
 - Move towards more abstract notions of system size:
 - Number of change requests?
 - Function points?
-

Conclusions

- Malleability can be used as a comparative measure:
 - How different designs withstand the accumulation of change in a system
 - Operational definition must take into account the specificity of modularization paradigm
 - What is the right level of abstraction?
 - Investigate relationships with other design measures
-