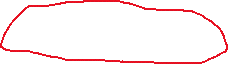
Phillip Thoendel - Module 6.2 - Strangler Fig Pattern at Blackboard Learn 2011

**Case Study Summary**

A screenshot of a computer

Description automatically generatedThe Blackboard Learn 2011 case study was of special interest to me as it is the platform we are all using to do our schoolwork. I have generally found the Blackboard platform to be stable and functional save a few design features that I don’t care for. The intrusive banner in the image below is a great example, however I usually just “inspect” the code and delete it in the browser code editor.



The case study talks about how the architecture for Blackboard was old and monolithic. The basis for the system dates back to 1997 using J2EE. At first the challenges were small and hardly noticeable but as the platform grew, testing and pushing changes became more and more complicated and resulted in unintended consequences. Basically, the code base kept growing but improvements and commits slowed to a near grinding halt.

A decision was made to decouple from the monolithic architecture in favor of smaller “building blocks” that operate in a more modular fashion. It is worth noting that this was not done all at once. A “strangler” method was used to implement the “building blocks”. In essence, little by little, a piece of the monolithic architecture was shut down and replaced with easier to maintain code blocks. This process was repeated until their system was decoupled completely.

These building blocks allowed developers to work faster and with more autonomy. This also significantly reduced feedback time and overall communication between developers.

**Lessons Learned**

There are a few key takeaways from this study that serve as great examples one may bring to the attention of leadership if an organization is facing similar challenges.

• Having your code based decentralized allows for much greater developer agility.

• Don’t let technical debt pile up, it’s better to address emergencies before they happen.

• Bottlenecks happening in your processes usually mean something is outdated

• Developers don’t like communicating with each other

**Conclusion**

I think this case study is a great example of not only identifying technical debt via process bottlenecks, but also of breaking a large problem up into smaller more manageable segments. The piecemeal approach of gradually adding “building blocks” as opposed to “rip it out and rebuild it” allowed for very intentional changes that allowed their platform to step into the modern age.