XTreme Programming 2.0

Planning

* User stories are written at a high level; smaller details are defined and refined each day as **Context Items**
* Release planning and scheduling
* Create **Product Contexts** (PC) on several levels:
  + Aspects: security, performance, UX, cost
  + Business: feature level contexts
  + Engineering: tech debt
* Release in awareness:
  + for library creators: get metrics on how your library is being used, and plan accordingly
  + for consumers: get information about what’s coming from the library creators
  + exponential decay of strength as we get information from farther and farther levels of dependency

Managing

* The team has an open, semi-asynchronous communication
* Instead of standup meetings, we have written and voiced Context Descriptions, which include details of implementation, progress, and thread of QnA
* Move People Around; any developer can take over a piece of work from another developer, at a moment’s notice;
* Refactor Mercilessly
* Inspect your workflow and refine it periodically

Coding

* The customer is always aware of the Q/A; a privacy filter can be added to any Question, to direct it inside the team
* The code is automatically formatted to company standards, even though each developer can work with their preferred formatting
* Architectural details and changes are described first, then implementation details, then the user writes the code
* Sequential Context Integration: PC on different levels can be integrated in one click, and then those changes undone in one click
* Integrate Often; plan interfaces and write code such that exchanging versions does not involve any change in the other users’s code
* Model and Code are interchangeable, rather than sequential operations.
  + you can start with a model and evolve code-model from it
  + you can start from code (a prototype for example) and evolve code-model from it

Integrate with existing tools:

* git (already done)
* other versioning systems?
* trackers (Jira, Bugzilla, Asana, Redmine, etc)
* build tools (ionic, ghc, maven, cruiseControl, etc)
* modelers (UML tools like Visual Paradigm, Data Science tools like Jupiter Notebook, etc)
* knowledge management systems (Confluence, etc)
* communication tools (Slack, etc)
* design tools (canva, figma, adobe etc)

TODO: write paper

- a large group of people can make better guesses on average than a single person (citation needed)

- analyze code and offer metrics on artifact interdependencies (different levels)