Using Agile development practices for kernel development

A.K.A - Bringing sanity to chaos

Chase Maupin, system integration manager for the Linux Core Product Development (LCPD) team

Agenda

- Agile Manifesto
- Meet LCPD - Charter and team
- What’s the problem?
- Mmhmm, you can fix it right?
- Let’s make sausage
- Would you do it again?
- Continuous improvement

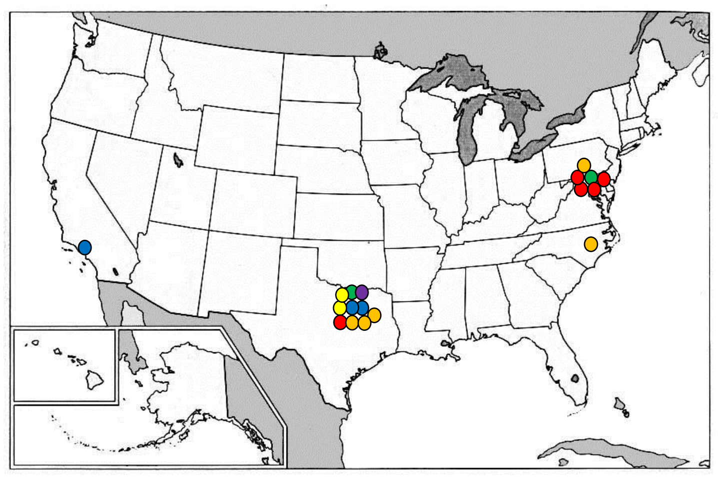
Agile Manifesto

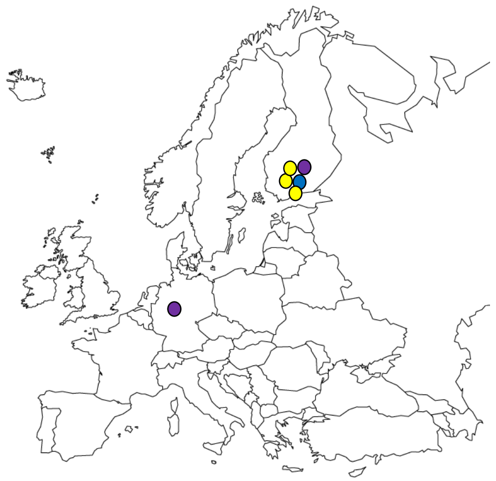
Agile Manifesto

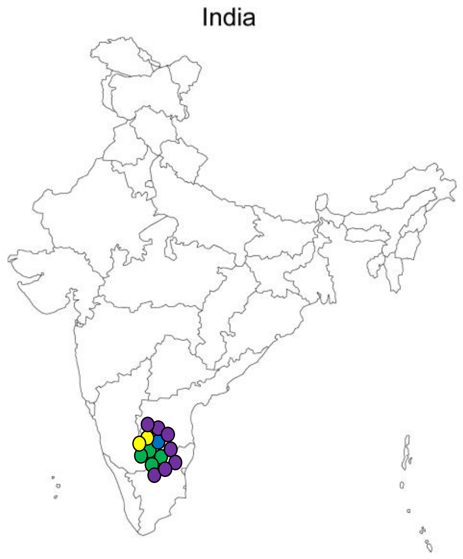
- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
  - **Individuals and interactions** over processes and tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan
- *That is, while there is value in the items on the right, we value the items on the left more*
- *http://agilemanifesto.org/*

Meet LCPD

Where in the world is LCPD?







- LCPD is spread out across the world in six time zones
  - West Coast US
  - Central US
  - East Coast US
  - Germany
  - Finland
  - India

LCPD Functional Teams

🟢	Baseport
🔵	Power & Thermal
🟣	Connectivity
🟡	Audio & Display
🔴	System Test
🟠	System Integration

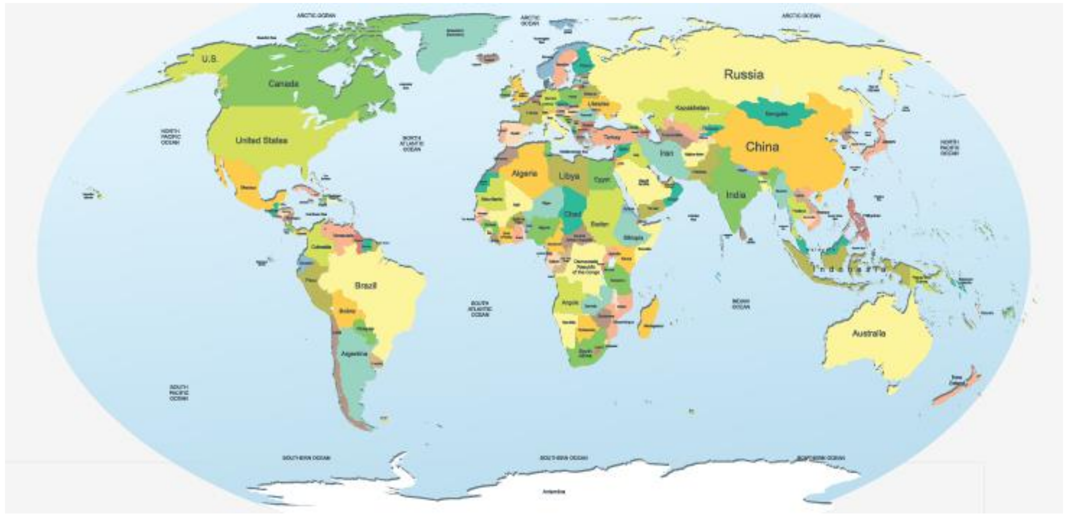
What would you say…you do here?

- LCPD charter
  - Creation of high quality, scalable Linux solutions for processors through upstream development of uboot, the Linux kernel, tool chain and file system
  - Insure maximum software reuse and device entitlement by working with silicon design teams in providing feedback and requirements on new SoC architectures
- Translation from manager to ‘techie’:
  - Work with the upstream communities for our software components to ensure that TI devices are supported in the mainline and work without additional patches
  - Ensure that we are addressing feedback from the community and regressions in the mainline to ensure continued quality
  - Work with our design teams to make sure simple design decisions don’t have ripple effects through the software

What’s the problem?

It’s a big world after all

- As mentioned previously we have team members around the world in six different time zones
- Furthermore within each functional area we have team members spread around the world
- This makes co-ordination difficult among team members due to limited overlapping work time
- IRC helps some but we needed more collaboration



Everyone wants a piece

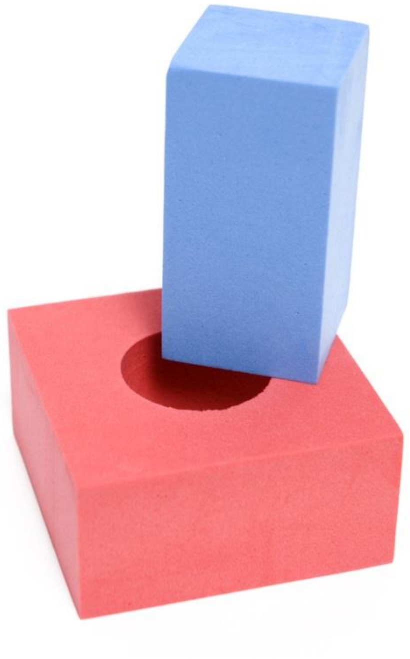
- LCPD services multiple customers each with:
  - Their own set of care about devices
  - Their own priorities and release schedules
  - Their own set of end customers with requirements and issues
- LCPD engineers care about the IP first, not the device
  - Develop the feature or fix the issue for the IP on all devices
  - This means that teams are not organized by device (i.e. a kernel team per device) but instead by IP and functional areas
- This leads to the same developers being requested to develop features for multiple customers and a need to have a single voice prioritizing and directing these efforts

A balancing act

- The LCPD charter is to develop support for TI devices upstream. This is how we ensure sustainable, quality software development
- The community provides us feedback and requirements as part of this which requires effort from TI
- This effort has to be balanced along side the requirements from our internal customers that LCPD serves
- Furthermore as merge windows approach, the priority of community tasks increases since missing a merge window means carrying patches out of tree for months

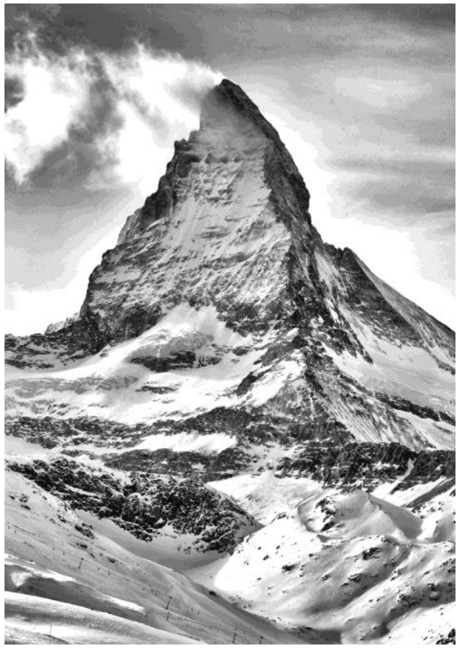
Square peg, meet round hole

- Many of our developers specialize in a particular IP or kernel subsystem
- Experts require less ramp time which improves efficiency
- This efficiency comes at the cost of cross functionality
  - We do not view developers as interchangeable cogs
  - Rather we would like to encourage developers to branch out into other interest areas



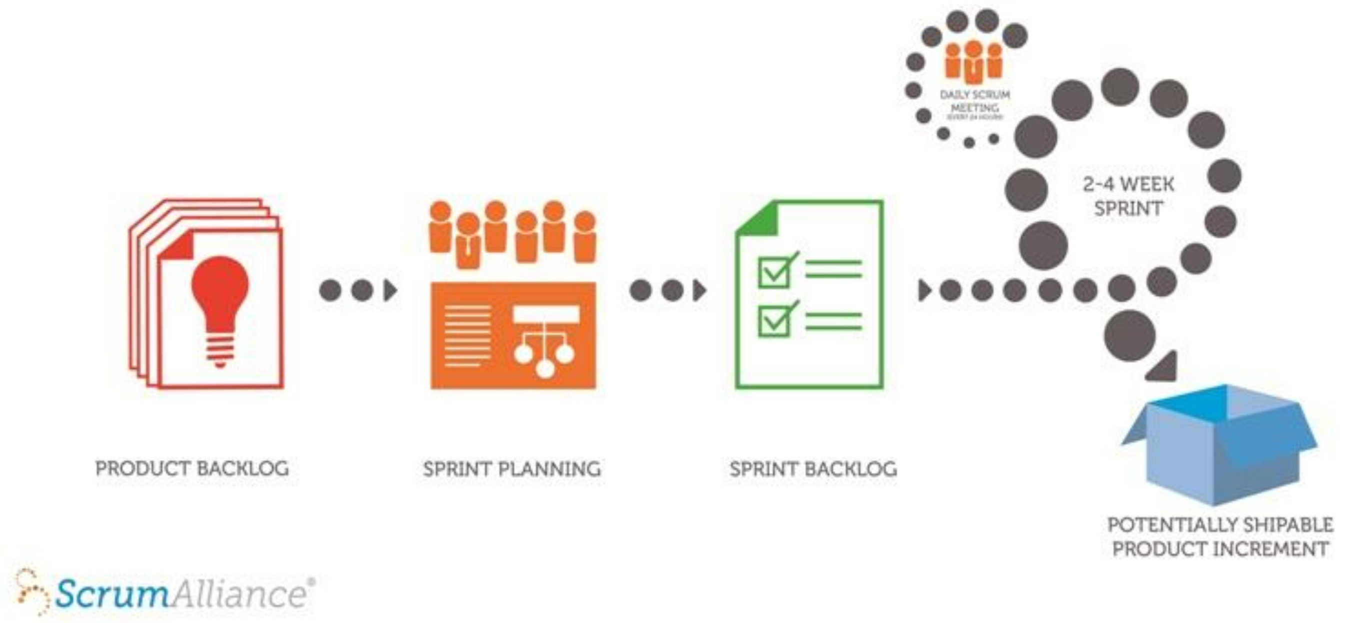
Sometimes the molehill IS a mountain

- Support for TI devices **HAD NOT** been pushed upstream and instead consisted of thousands of patches on old kernel revisions
- Moving these patches upstream while also developing support for new devices and IP was overwhelming
- We needed a way to keep track of the mountain but only worry about one molehill at a time
  - Currently our focus devices of AM335x, AM437x, OMAP5, and DRA7xx all boot directly from the mainline kernel with additional driver support being added



Mmhmm, you can fix it right?

Scrum, it's not as dirty as it sounds



- LCPD chose Scrum as the Agile process to help address our problems
- Having a shared backlog prioritized between customers allowed easier communication of trade-offs and visibility into the team shopload
- Giving developers focused time (a sprint) to work on items helped ease the chaos of fire fighting and priority churn
  - Reduced the shell-shock as well. Looking back we had moved the mountain one boulder at a time
- Making upstreaming part of the process kept focus on our charter

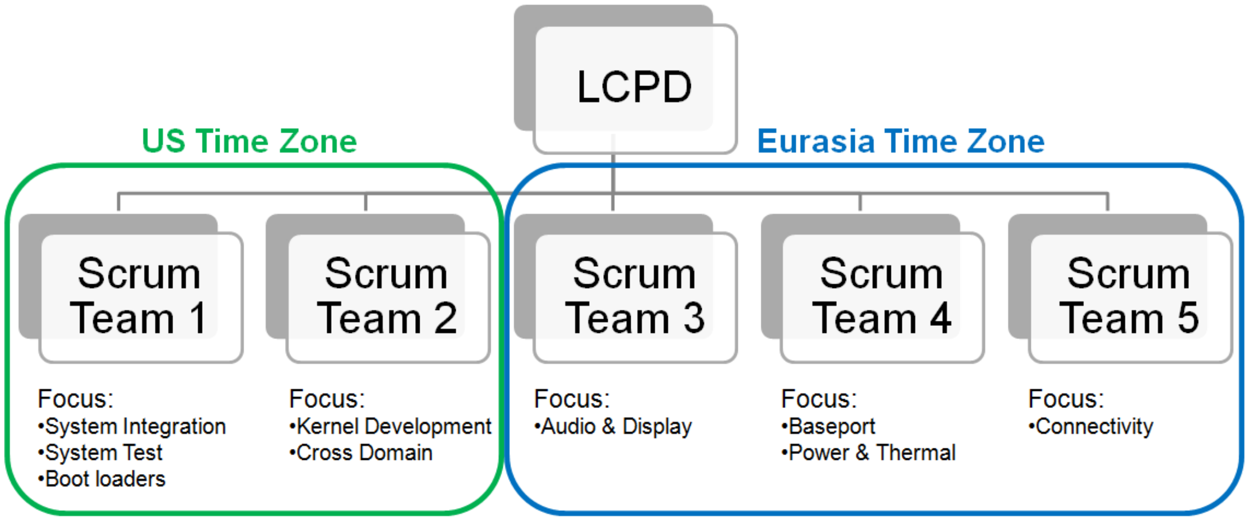
Make sure you have the right tool

- Needed an online tool which can be accessed both inside and outside of our firewall
  - This is particularly helpful for our remote/home based developers
- Needed a tool that allows all of LCPD to share a backlog while still grouping development tasks for functional teams
- Needed a tool that does release planning, sprint tracking, etc all from one tool
- Needed something that integrates with bug trackers like CQ to allow us to track bugs in a unified backlog
- Wanted to give visibility to our customers of our backlog, priorities, and progress
  - This allows for them to pull information, rather than us having to push contant updates when requested
- LCPD chose VersionOne (V1), an Agile SW development management tool
- NOTE: There are many other good tools available to chose from, this was just the one we picked

Let’s make sausage

Sometimes I feel like you are a world away

- As mentioned in the LCPD introduction our team is scattered around the world
- Furthermore, the members of the different functional teams are scattered (limited co-location)
- There is very little time overlap to allow for scrum meetings at a functional team level
- Scrum teams are organized first by time zone, then by functional area



- Backlog refinement meetings are held weekly at the functional team level
  – The functional team reviews that domains backlog at that time
  – People align on which team members plan to take which backlog item

It’s done when WE say it’s done

- LCPD shares a definition of when something is done, which reduces confusion
- A development item is done when:
  - The code has been written
  - The code has been validated (system test or developer)
  - Where appropriate the patches have been submitted upstream for review
    - In this manner the upstreaming of work is part of our development flow
- A defect item is done when:
  - The code has been written
  - The code has been merged into the production tree
  - Where appropriate the patches have been submitted upstream for review
  - System test has validated the fix in the production tree
- The main difference is that system test operates against the production tree. Defects found there are checked for applicability to the latest mainline and if so fixed for mainline and then backported to production tree

I want it NOW

- Support escalations can happen at any time
- Customers generally don't care if you are in the middle of a sprint
- How do you plan a sprint for two weeks and still be responsive to customers?
  - Many scrum practitioners face this same problem so no need to invent anything new
- Allocate overhead in each sprint for the typical customer support load
  - Usually about 25%
  - This time lets customers see progress being made
  - For simple issues this is likely enough
  - For complex issues this is enough to replicate the problem and plan more time in the next sprint
- The Kernel Community is treated as a critical "customer". This gives us time to respond to feedback
- If no customer support comes in we can opportunistically work on something else from the backlog, assist other team members, or do code clean-up, etc

How long will it take to upstream this?

- Upstreaming is a process that takes time.
- It is not a process that can always be predicted
- So how do you handle upstreaming in Scrum with fixed time boxes and an indeterminite process?
- Back to LCPD definition of Done we consider an item “done” when we have submitted it upstream for review
  - Small feedback goes into the “customer support” overhead bucket
  - Significant feedback gets a new story allocated to address the feedback and a new submission. This is given critical priority
- This cycle iterates until the work is upstream
- If we expect feedback on a series we plan for it in the next sprint. i.e. an RFC will likely have feedback that needs to be addressed

It’s bigger than just you

- As active community developers some LCPD team members also have maintainership responsibilities in the broader community
- In our Scrum implementation we handle this by creating recurring stories representing the maintainership time and tasks
- The maintainers pull these stories into every sprint, ensuring that they have enough time reserved to take care of not just TI, but their community responsibilities as well

What’s your plan?

- Agile development doesn’t mean no planning
- The product owners plan the major deliverables as epics and let the teams break them down
  - This is what management uses for customer commitments and tracking
- What we don’t do is plan every minute detail, as that is likely wrong
  - Instead we plan the broad goals and when we think we can accomplish them and let the details evolve over time
- There is a difference between when code is available and when code is upstream. You can plan for available

Texas Instruments

Given what you know now, would you do it again?

Heck yeah!!!

- LCPD has been able to make significant progress pushing support for our devices upstream
- We productized and released our SDK based on the then latest stable kernel, boot loader, and Yocto releases with an eye towards LTS
- We have been able to balance customer support escalations and still provide proper developer focus for upstreaming
- The team feedback is that Scrum has provided the desired focus and minimized distractions

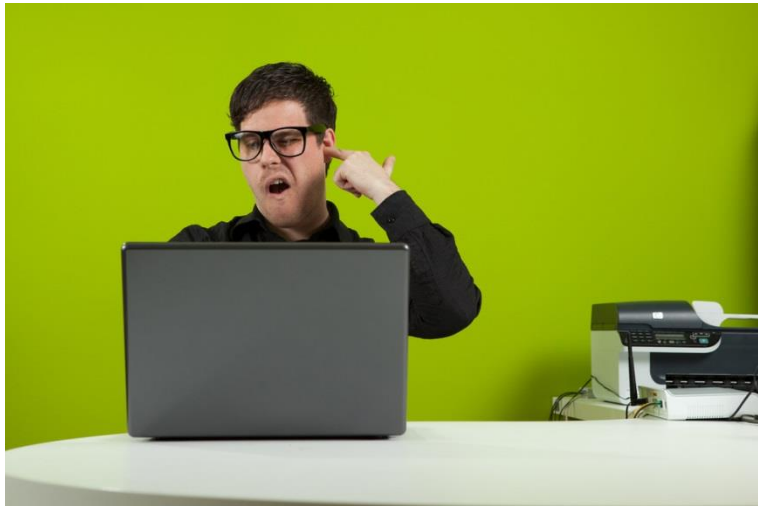
Continuous improvement

If you develop it, they will come.. for support

- Customer support needs to be planned for
  - Whether internal support or community
- You can't just wait until the next sprint to address issues
  - Been there, tried that, no one was happy
- Instead, leave enough time for the basics and plan the bigger items
  - It usually takes a while just to replicate the issue and realize the issue is big enough to need more dedicated time

Time drags when you're planning dumb

- With scrum teams full of specialists we often found that planning part two was tedious
  - This is where stories are broken down into tasks
- What we did find useful was:
  - Reviewing the steps required to complete a story. This allowed others to learn about the pieces of a story and the approach to solving the problem
  - The estimate of time required to complete the story. This gave us a way as a team to sanity check the commitment
  - Areas where people could help each other, such as reviewing documentation or performing testing
  - Allowing people to bring their own experiences into the story such as planning missing tasks based on similar experiences
- However, doing this breakdown online if front of everyone was painful
- So instead we decided to introduce a break between planning part one and planning part two to allow people to do an initial breakdown, then review with the team
- Planning went much faster and we found more attention was paid to what was being done and more team interaction



Nobody is perfect….

- ….And neither is your scrum implementation, backlog, etc
- Waiting for perfection or until you have defined every last part of your agile process is the antithesis of agile
  - That's not to say you shouldn't have you basic framework in place
- Just remember to be willing to adapt, learn and get moving. The rest will come over time as you find what works for you
  - Just keep an eye on the benefits of each step and find how you can get that benefit in the way least painful for you
- The only real requirement is participation

What would you change?

- We do not have enough cross-training. Back to having too many experts
  - Grouping by functional area helps but I would like more cross-training
- PO roles should be more official and dedicated
  - Should have one per team and not split among teams
- Have test team resources around the world to be able to embed testers in each scrum team
  - Current test team is US based

Q&A