https://ntrs.nasa.gov/search.jsp?R=20180007632 2019-08-31T17:12:13+00:00Z

Mission Control Technologies

Empowering users

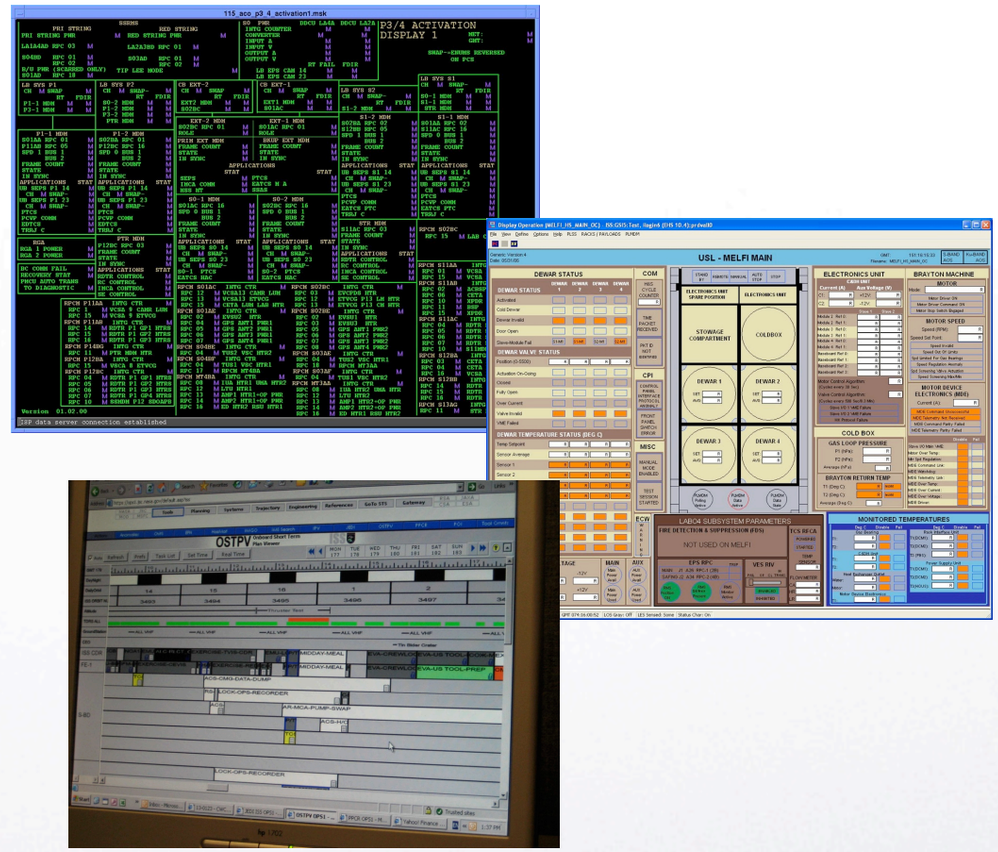
by Jay Trimble, NASA Ames Research Center
Tom Dayton, UARC at NASA Ames Research Center
Alan Crocker, NASA Johnson Space Center

Project Summary

- Purpose of the project
- Goals and Objectives
- Timeline

Project Summary

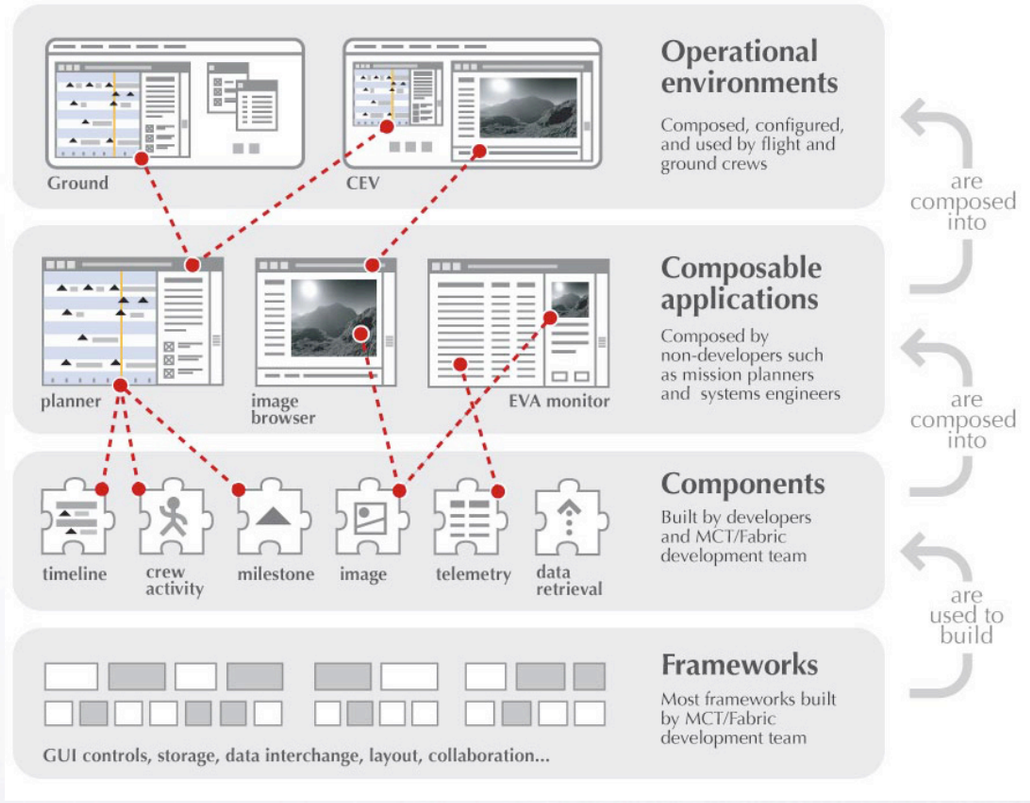
- The Core Issues
  - Mainstream software technology built and distributed as monolithic applications
  - For users, this creates artificial packaging of functionality
  - This adversely affects NASA mission operations, constraining flight controllers by technology limits, rather than operational tasks
  - It also creates heterogeneous environments that put the users in the role of software integrators



http://youtu.be/_etDYWy9v2s

Project Summary

- To address the problems we built a new software framework, Mission Control Technologies (MCT)
  - MCT provides users with an environment populated with "live" composable user-objects, from which they may assemble their own software
- Eliminate artificial software boundaries that put users in the role of software integrators



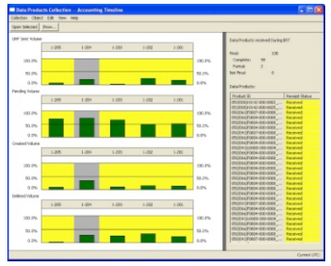
Compositions Instead of Applications

Timeline

2001 — The idea - Mars Rover Training, users request interoperability not possible with monolithic apps

2003 — Exploratory workshop with industry, including OpenDoc and CUA user experience leads/ architects

2005 — First prototype component model demonstrated

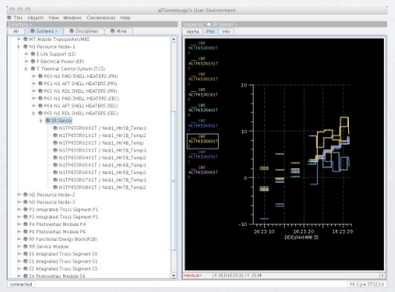


2007 — First design sessions with users, prototype test in mission control test facility, user surveys

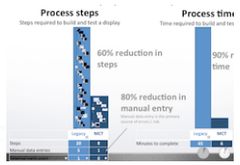
2008 — Project go-ahead

2009 — First Compositions/User Feedback

2010 — 1st user-ready delivery/ongoing iterative design with users

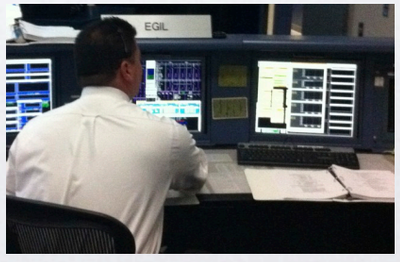


2011 — First user measurements with product in operational environment



2012 — Operational Certification

Formal Testing



Stakeholders

Director, Mission Operations
NASA Johnson Space Center
Safe, successful, cost effective
Mission operations

Mission Operations Chief Engineer
NASA Johnson Space Center
Safe, successful, cost effective
Mission operations

Space Station Mission Controllers
NASA Johnson Space Center
Minimal operational disruptions
Safe missions
Some want new software, some don't

Mission Control Facilities Team
Working mission control center for mission controllers, IT security and operations

Principle Investigator
NASA Ames
Replace monolithic apps with user composable objects, update mission control

User Experience Team
NASA Ames
Make object oriented composable GUI style mainstream, make great software for the users

Project Requirements

- Business Marketing
- Salability
- Functional
- Technical
- Organizational
- Customer
- Usability
- Others

Business/Marketing

- Replace existing multitude of heterogeneous apps
- Save software sustaining costs
- Believed by users, but not quantifiable, that significant benefits to operations emerge with use

http://youtu.be/s3nb7Opjzsg

What Sold the Project

- Management
  - Reduced sustaining costs, software modernization, inter-center NASA collaboration
  - A fundamental change in how operational displays are certified for use. Current practice requires every display to be certified. There is no object reuse. The MCT object model will allow certification of objects, which need only be done once. Objects may then be reused.
  - Management believed that the power for users to compose their own software had significant potential benefits. However, since those were not quantifiable they were not part of the business case
- Users
  - The power to compose your own software, empowerment, build your own displays

Organizational

- The customer must be able to take over maintenance of the software
- Do not incur new significant training requirements on users
- Do not force re-authoring of already existing content
- Do not incur new risks on existing host facilities

Customer

- Provide a user interface that conforms to customer's cultural norms but is significantly more usable than current tools.
  - Same look and feel
  - More bells and whistles
- Do not incur new risks on existing host facilities

The Conundrum

- The customer expects a new product with new capability.
- Yet they want it to look just like their old capability. The first action is often to rebuild exactly what they already had in their legacy products
- Over time, and with ongoing use, they become more open to new features.

HCI Best Practice Solution

- Project Narrative

Methods used and how

- Research
- Design
- Development
- Deployment
- Product lifecycle

Research

- Research ethnography provided an overview of mission control center disciplines and issues
- The data helped to select focus areas for software design and development

http://youtu.be/lEmPFx_mpaY

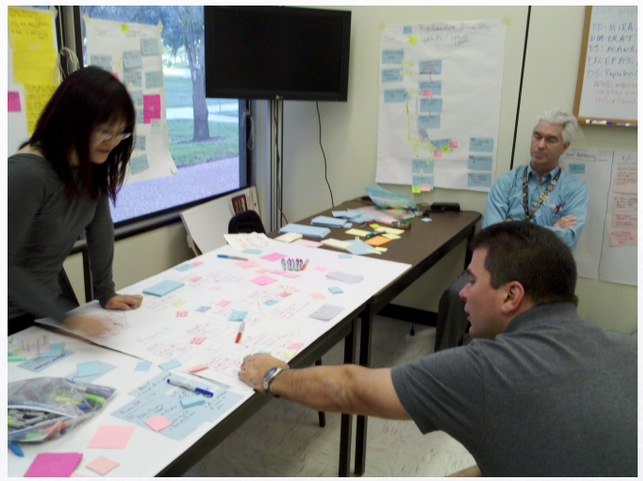
Participatory Analysis, Design & Assessment

- Customers are part of the design team
- Use customer domain expertise
- Shared ownership of the design

http://youtu.be/Oe5rpE2mA6I

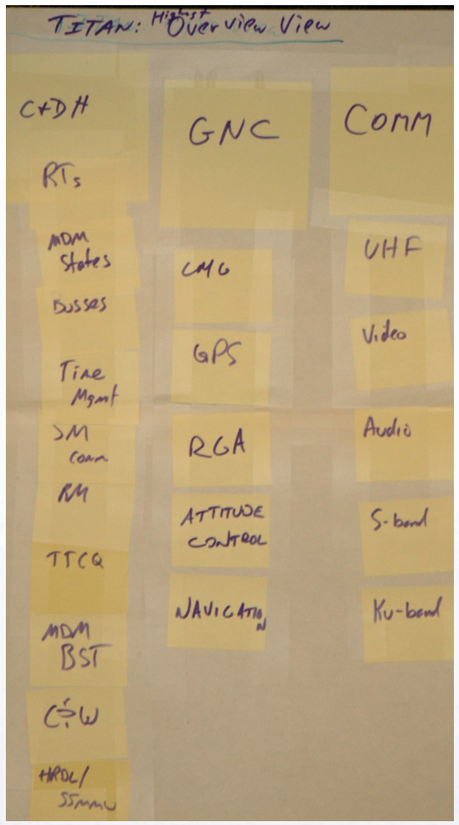
Two Teams Become One

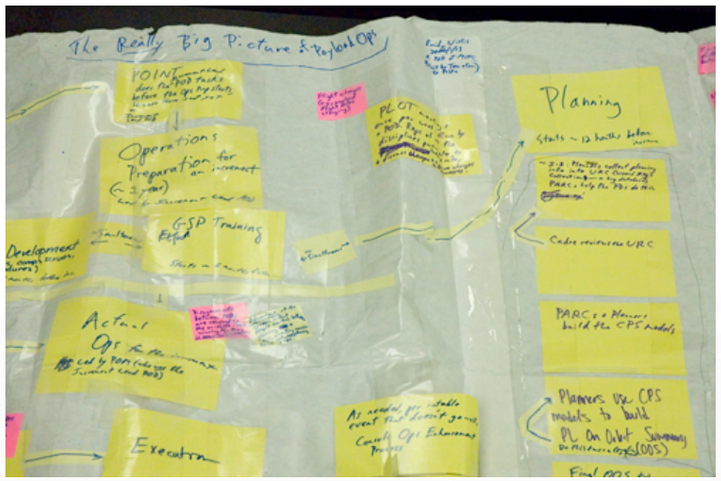
- The tangible output from participatory design is a series of artifacts
- Perhaps as important, the method built a joint team out of what began as two separate teams
- We developed a shared mental model and a common language
- For the users, who were experts at performing their tasks, but not creating explicit representations, they saw their job in a new way, with new possibilities

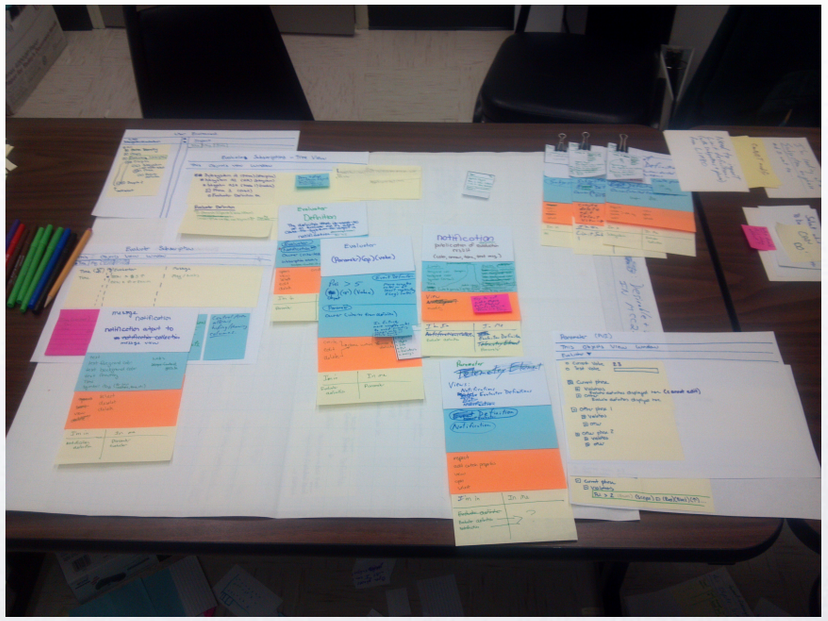


Design Artifacts

- Triggers
- Task Flows
  - Blue sky
  - Real world

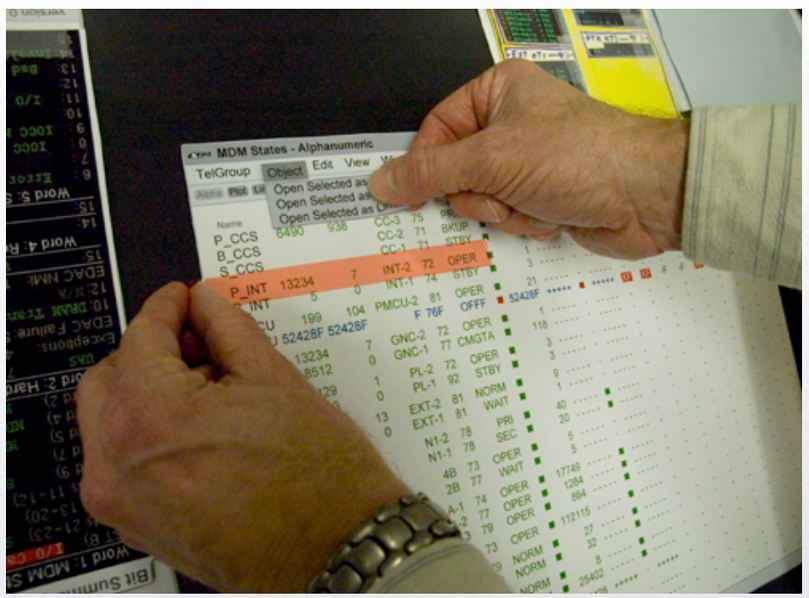






Design Artifacts

- Task Objects
- User Objects
- Windows

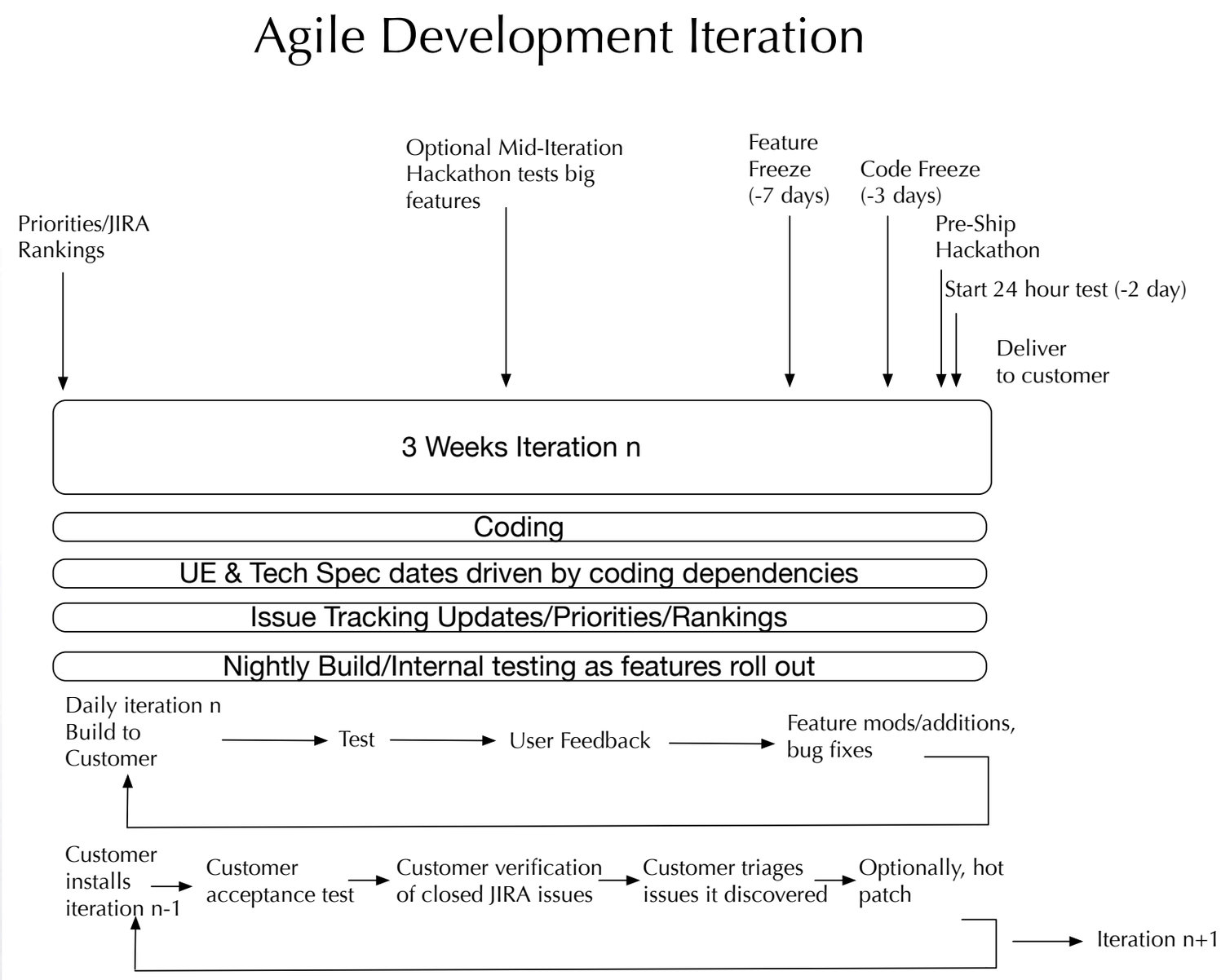


Design is Not Enough

- Designing a great product with users is meaningless if it cannot be built
- We developed a three-week agile user-centered delivery cycle, with our customer being a key part of the flow
- This enabled our small team to focus on the highest priorities, and to quickly react to customer inputs
- Minimal lag between design and customer hands on experience with the deployed features

Agile User-Centered Development Process

- Continuous customer feedback/ nightly build
- Just in time user experience specs
- QA verification with feature rollout
- Feature closure upon customer acceptance



Participatory Design + Agile

- Agile and participatory design work together
- Strategically plan the design cycles so that design specifications are ready when they are needed by developers - this means planning months ahead
- Design ahead - use “gap” times, such as engineering focused iterations, to begin long lead design cycles
- We abandoned big design specs. The designers linked individual specifications to developer issues in the team issue tracking system
- The developers set due dates for design specs - a design spec was due when the developer needed it for start of coding

Participatory Design + Agile

- Participatory Design and Agile both facilitated close customer participation, but on different timescales
- Through agile, and the availability of the nightly build, our users would run new versions of the software almost every day
  - The customer used the nightly builds to provide constant feedback, thus they were a core part of feature development, validation and design
  - Feedback from the nightly build was often spontaneous - the customer would call when they had an input, or the designers and developers would initiate calls when they had questions

Participatory Design + Agile

- Participatory Design, like agile, incorporated the customer as part of the design team
- Unlike agile, which had us talking to the customer daily, through an ongoing feedback loop that was initiated by either party spontaneously, the design cycles were planned based on the strategic road map
- Design sessions typically lasted several days and focused on a small number of features in detail

Participatory Design + Agile

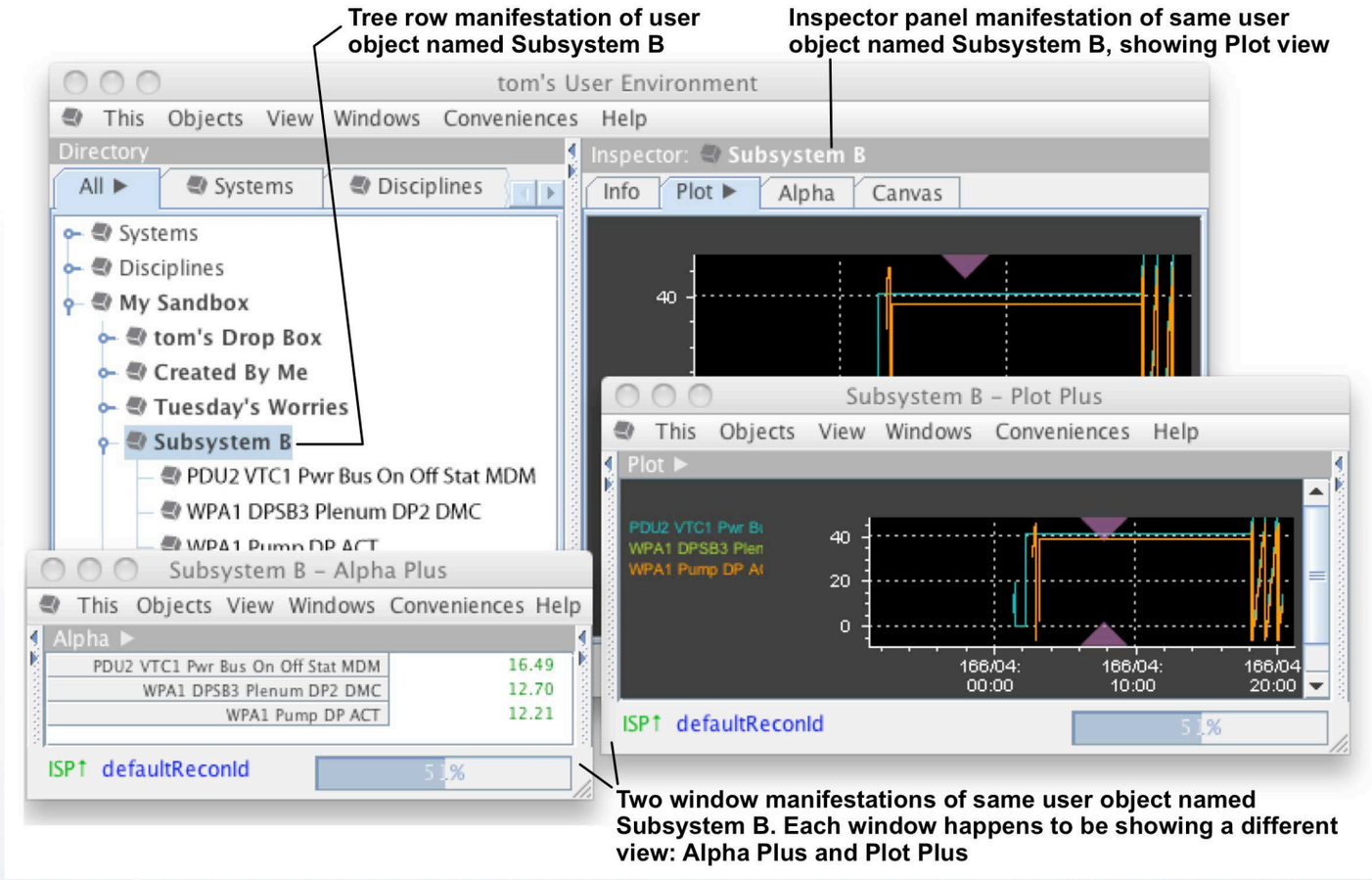
- The design sessions set joint expectations among customer and developer for what we would see when each feature rolled out
- Customers first saw features in the nightly builds
- By the time a feature was officially delivered, in an iteration or a release, the customer was familiar with it both from their participation in design sessions and the nightly builds

Participatory Design + Agile

- Constantly showing the product, every day, and always using the product hands on, drives everyone to improve it
  - Agile facilitates fast reaction to customer feedback
- The customer saw our product on three timescales
  - Nightly Build
  - Three-week iterations
  - Twelve-week Release
- These multi-faceted interactions with different cycles, each with it's own set of expectations, forced all of us to make the product better

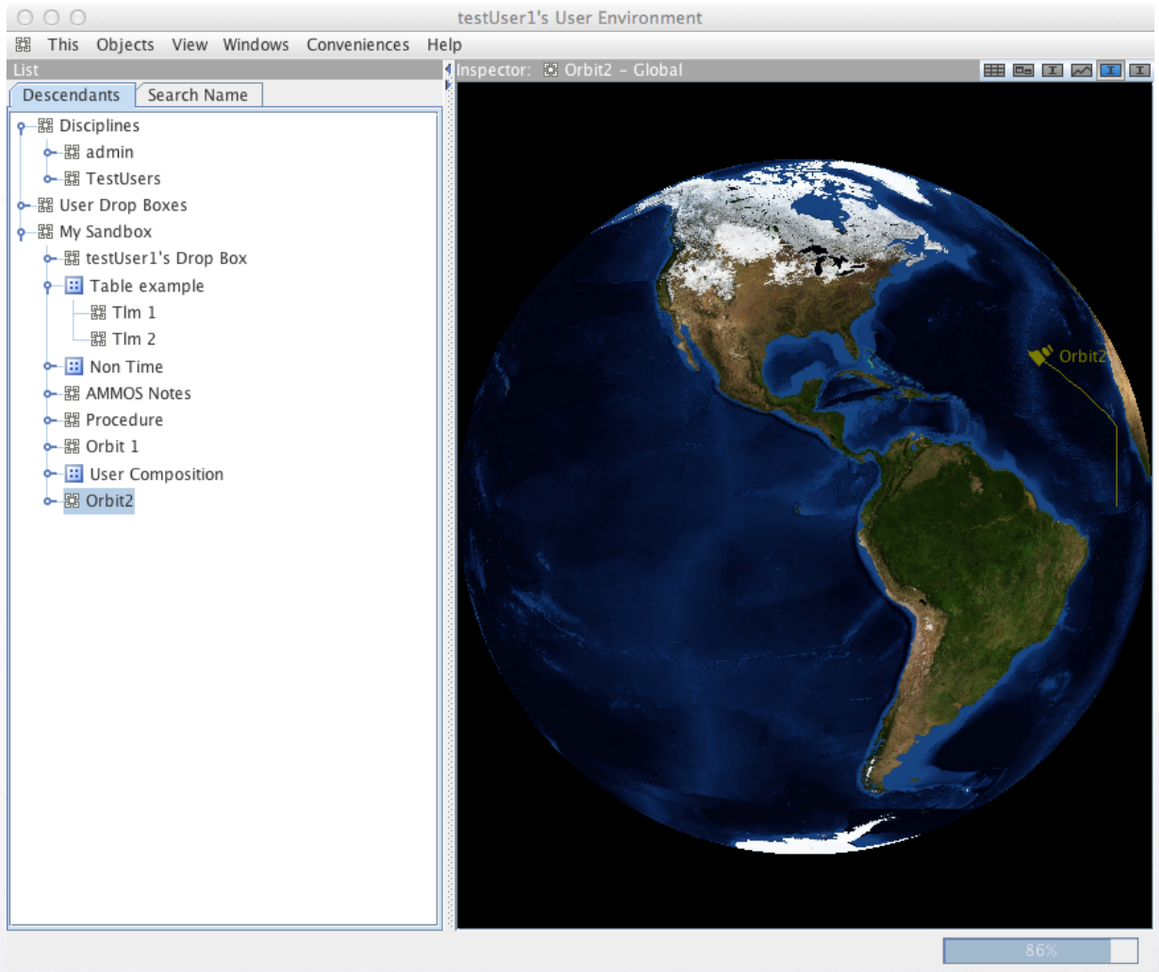
User Objects not Widgets

- User Objects
  - Representations of real-world domain objects
  - View the same thing in different ways
  - Shareable
  - Composable entities
  - “Live”
  - Consistent behavior



The Product

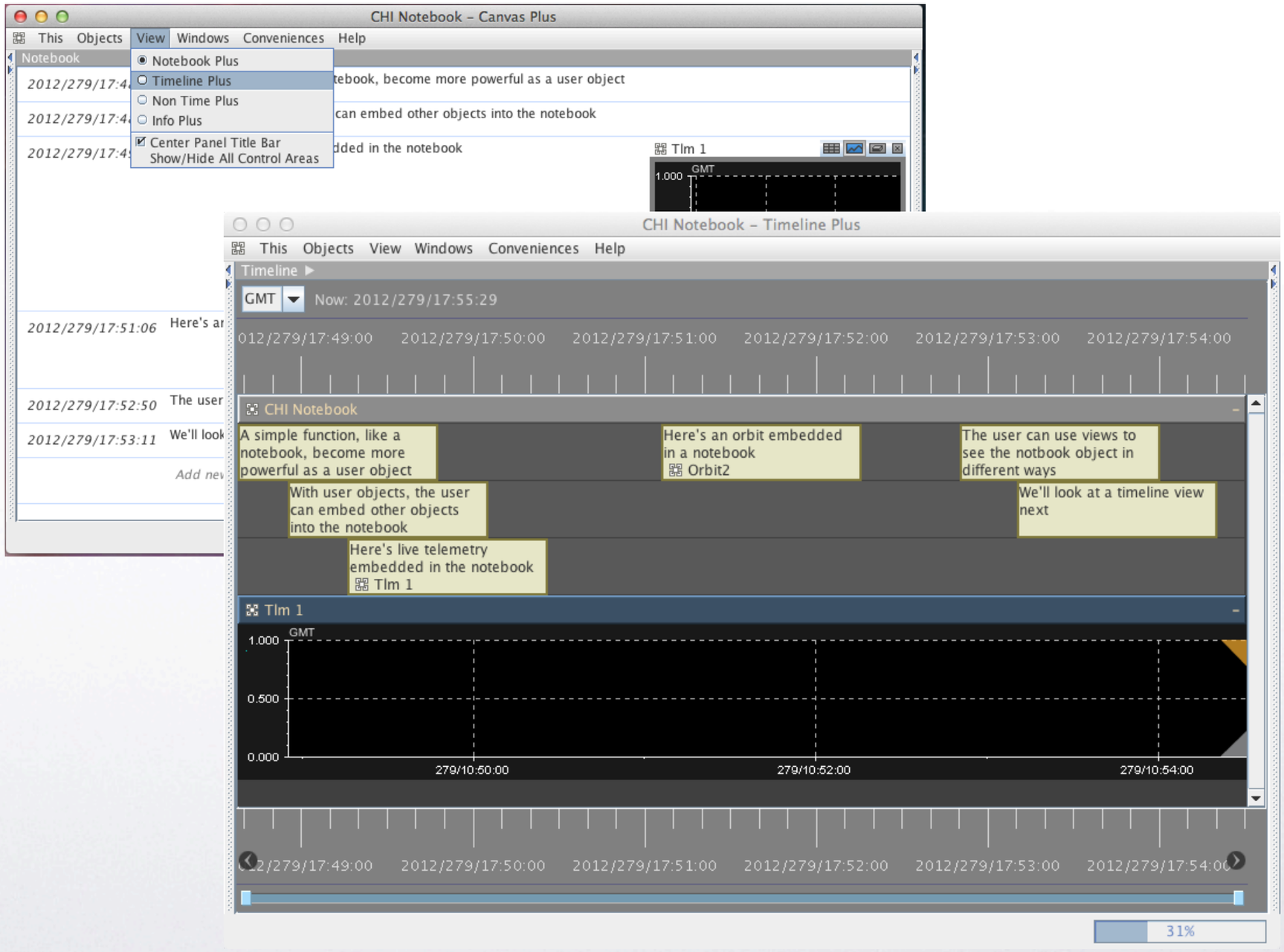
- Everything is a user-object
- Objects may be groups into collections
- Collections are user-objects



http://youtu.be/9YxOqIw2NME

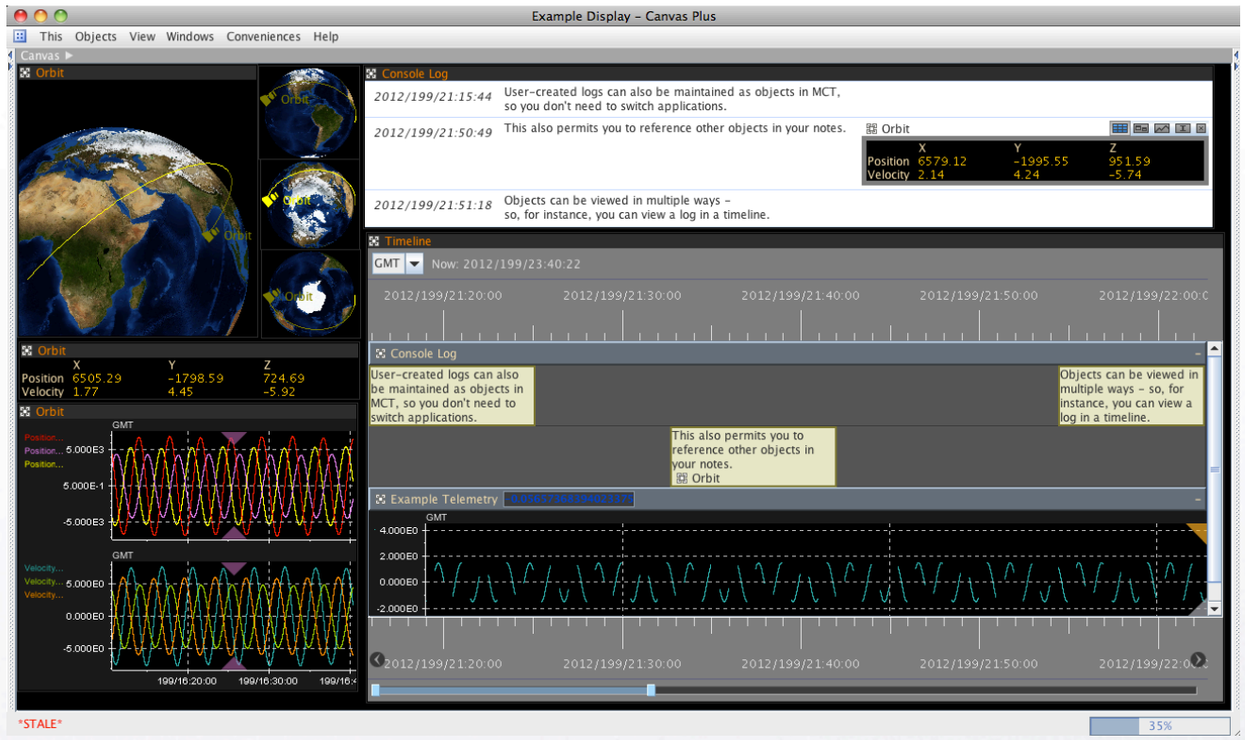
The Power of Objects

- This notebook is a user-object, with embedded text and telemetry objects
- The same thing is shown in two views - notebook and timeline



End User Composition

- This multi-domain composition contains multiple user-object types
- Each object may be viewed and manipulated independently within the composition



http://youtu.be/yBGhOh_MTME

Data

- A core project assertion that was not quantifiable at inception, was that we could significantly reduce the time it took flight controllers to build displays, or to modify existing displays
- Upon delivery of a usable product we were able to measure the number of steps and time to build displays with users in context in their work environment

Data Display Build Steps and Time

Process steps

Steps required to build and test a display

60% reduction in steps

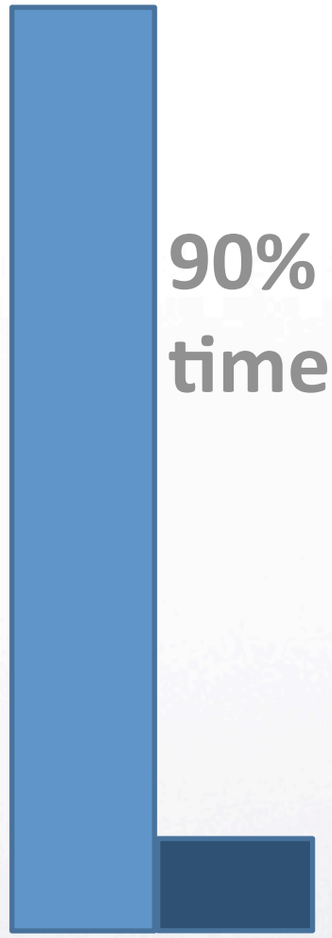
	Legacy	MCT
Steps	20	8
Manual data entries	5	1
External tools used	1	0

80% reduction in manual entry

Manual data entry is the primary source of errors / risk.

Process time

Time required to build and test a display



90% reduction in time

	Legacy	MCT
Minutes to complete	65	6


Challenges

- Technical
- Team
- Stakeholder
- Other constraints

Technical Challenges

- The component model to facilitate end user composability was too complex for developers and used non-standard technology
  - Addressed by simplification of the component model
  - Focus on core requirements, eliminate unneeded features
  - Required use of standard technologies, Java/Swing

Technical Challenges

- Initial product performance was unacceptable, too many features added too fast, buggy
  - Developed an agile user-centered development process
  - We found that process dictums can go unheeded. To succeed a new process required either automation or socialization within the team

Team Challenges

- The designers and developers were not communicating effectively, causing development effort to be spent building the wrong things
  - Addressed by: Shared lab space, emphasis on social skills and team compatibility in hiring
  - Daily communication with agile development cycles
  - Developers must be part of the participatory design process, even if it seems that resources don't permit it, you'll pay more later if developers are not part of design from inception

Stakeholder Challenges

- Many of our stakeholders did not understand software design and development.
- A significant number of users resented the idea of replacing their existing applications
- Users were concerned about losing the functionality of their existing applications

Other Constraints

- There was an ongoing mismatch between project budget and project scope
- The constant threat of cancellation often resulted in mitigation strategies rather than development strategies
- The stakeholder requirements changed over time, causing delays

Results of your teams efforts: The Bad

- The tightly integrated developer/customer team exacerbated a pre-existing polarization that pitted those who wanted new software "against" those who did not.
- Our deploy early and often model was incompatible with the broader user groups mental model of users not seeing the software until the final product.

Results of your teams efforts: The Bad

- Ongoing difficulties in educating the community of stakeholders who participated occasionally and who's pre-existing beliefs influenced their perception as much or more than the real product

Results of your teams efforts: The Good

- A breakthrough product providing end user empowerment
- Democratization of end user software
- The potential to change the relationship between users and IT by allowing IT to provide a certified environment that users may configure without IT support

Results of your teams efforts: The Good

- Through participatory design and agile development we built a unified team composed of the designers, developers and users.
- A user object model in which objects behave as consistent representations of their real world domain object counterparts - these are not widgets
- We successfully built a modular user-composable software architecture that was certified for operations for the International Space Station Mission Control Center

References

- Participatory GUI Design From Task Models, Tom Dayton, Joseph Kramer, Al McFarland, Monica Heidelberg, CHI 96
- Bridging User Needs to OO GUI Prototype Via Task Object Design, Tom Dayton, Al McFarland, Joseph Kramer, User Interface Design: Bridging the Gap from Requirements to Design, CRC Press, 1998
- Agile Development Methods for Space Operations, Jay Trimble, Chris Webster, Spaceops 2012, American Institute of Aeronautics and Astronautics
- From Traditional, to Lean, to Agile Development: Finding the Optimal Software Engineering Cycle, Jay Trimble, Chris Webster, Hawaii International Conference on System Sciences, January 2013
- UARC Frontiers of Science, http://128.114.198.239/packages/5