



National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Managing Complexity in Next Generation Robotic Spacecraft

From a software perspective..

Kirk Reinholtz
Jet Propulsion Laboratory
California Institute of Technology

JPL

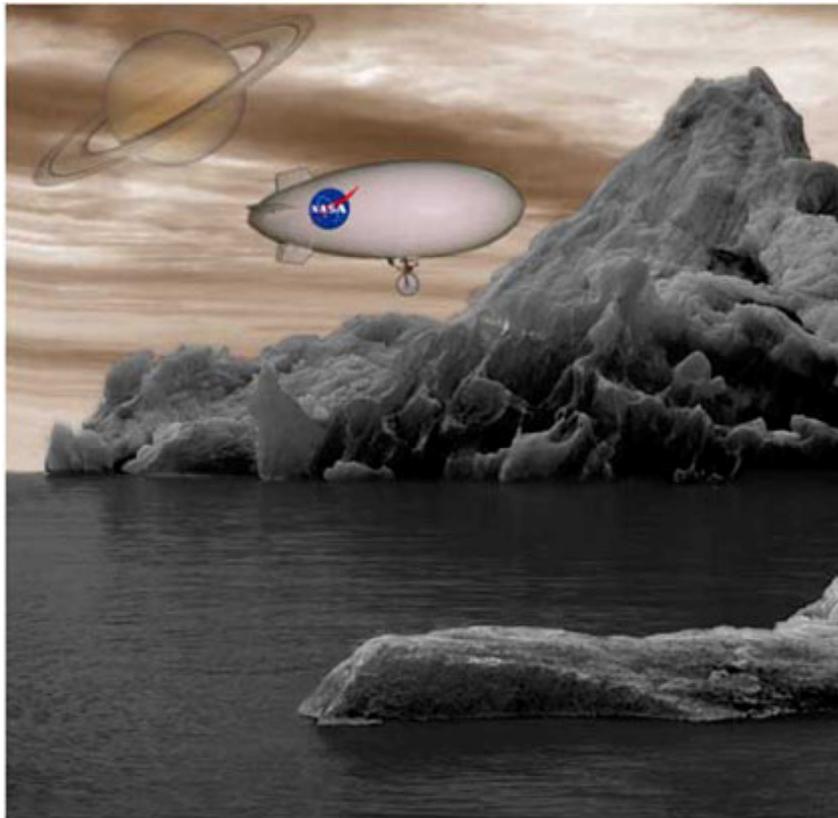




Our Future Undertakings



Titan Explorer



Hydrobot in Europa Ocean



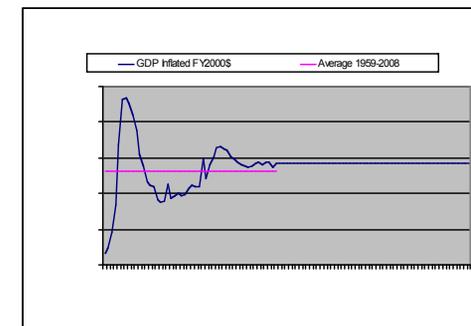
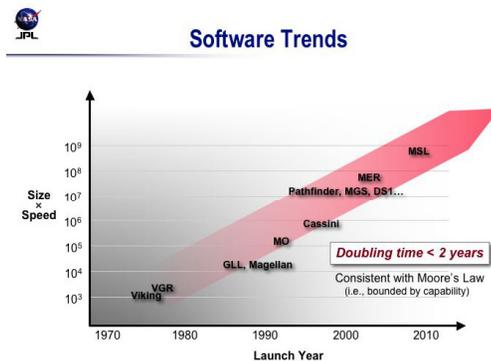


BIG Challenges



- **Much higher degree of autonomy required**
 - **Engineering - Execute mission without a lot of hand-holding**
 - Hand holding is expensive
 - Hand holding slows the pace of the mission (more \$\$ per unit science)
 - Dynamic environment -> Reaction time faster than light time allows
 - **Science - Find interesting targets, point, acquire data, condense, downlink**
- **Highly dynamic - Can't just stop when something goes wrong**
 - Fail-safe (the good old days) -> Fail-operational (future necessity)
- **More capability, mostly in the software, with same or higher reliability**
- **... and tight on the usual technical and programmatic constraints**
 - Limited power
 - Limited mass
 - Limited money

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.





Fundamental Limits



- The fundamental limit to what we can accomplish is how much thinking is required per unit of accomplishment
- WE build our systems to do OUR bidding. We need to understand what we want to accomplish, and how to accomplish it, and know that it'll really work when we field the system
- This is usually called “complexity” - The amount of wetware effort required to understand, control, and predict the behavior of a system (be it hardware, software, the spacecraft, or the project as a whole). The parts we don't understand become residual risk.
- Complexity is difficult to manage because it does not modularize well - Over-simplification of one subsystem (say the command structure) over-complicates another system (say the operations of the system) It's all about the complexity of the elements and their interactions...and understanding “simple with respect to ?”

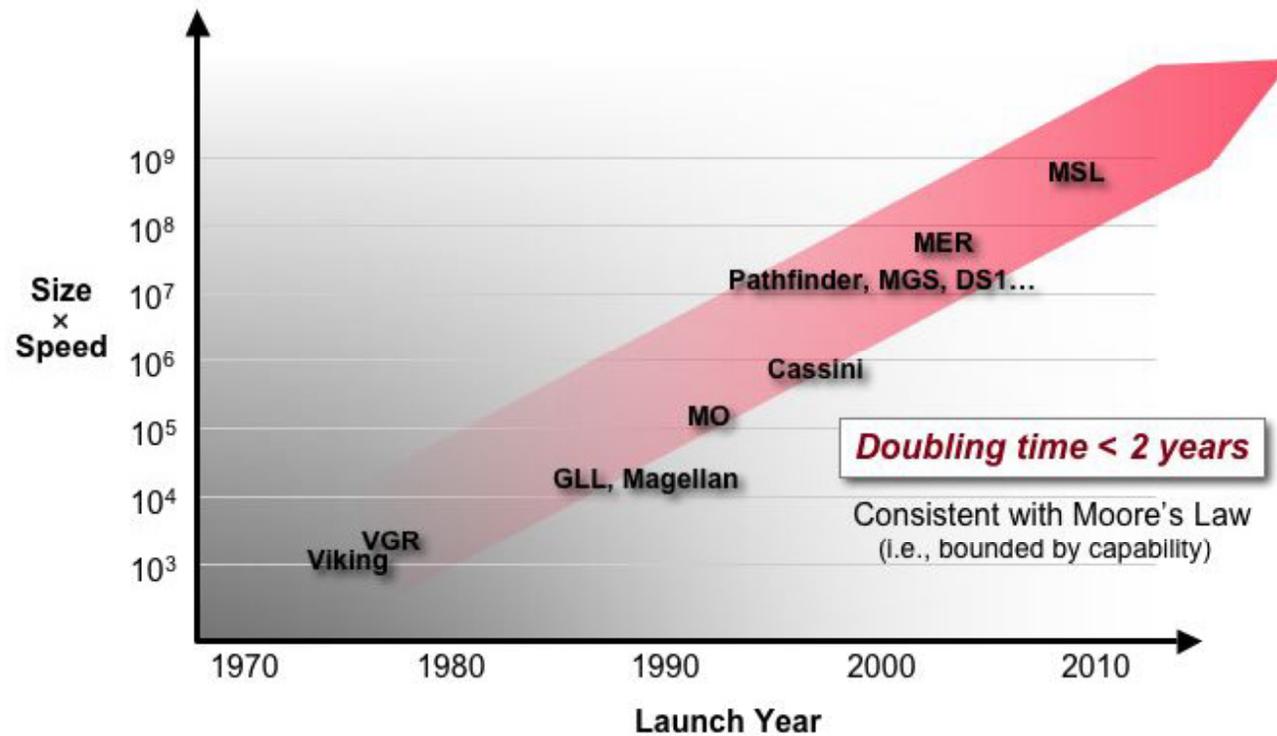
The Apollo I hatch is an example of the difficulty of applying the notion of simplicity. The hatch was very simple in one sense, as it consisted of a fairly simple structure that was simply bolted in place to seal the capsule. This is a classic example of design simplicity, as defined in the Constellation Architecture Requirements Document (“CARD”): “*Design Simplicity: Minimize the number of moving parts and the amount of interdependence on other systems, ease of operations and maintenance by the crew and ground personnel. Simple systems require less operations attention, necessitate less training, impose fewer operator constraints, enhance reliability for long duration missions, and streamline ground turnaround activities for reusable flight elements*”. Unfortunately this was later found to be the wrong type of simplicity to apply in this case. The hatch was difficult to open quickly, and so astronauts were killed when they could not get out of the capsule when it caught fire during a test. The lesson was learned, and a new hatch was designed and used for the remainder of the Apollo program. The new hatch had over 400 parts and was certainly a complex (in the sense of “lots of parts”) design, but it could be opened, simply and reliably and under stressful conditions, in about ten seconds. What was viewed as complex was suddenly viewed as simple.



Bigger Systems

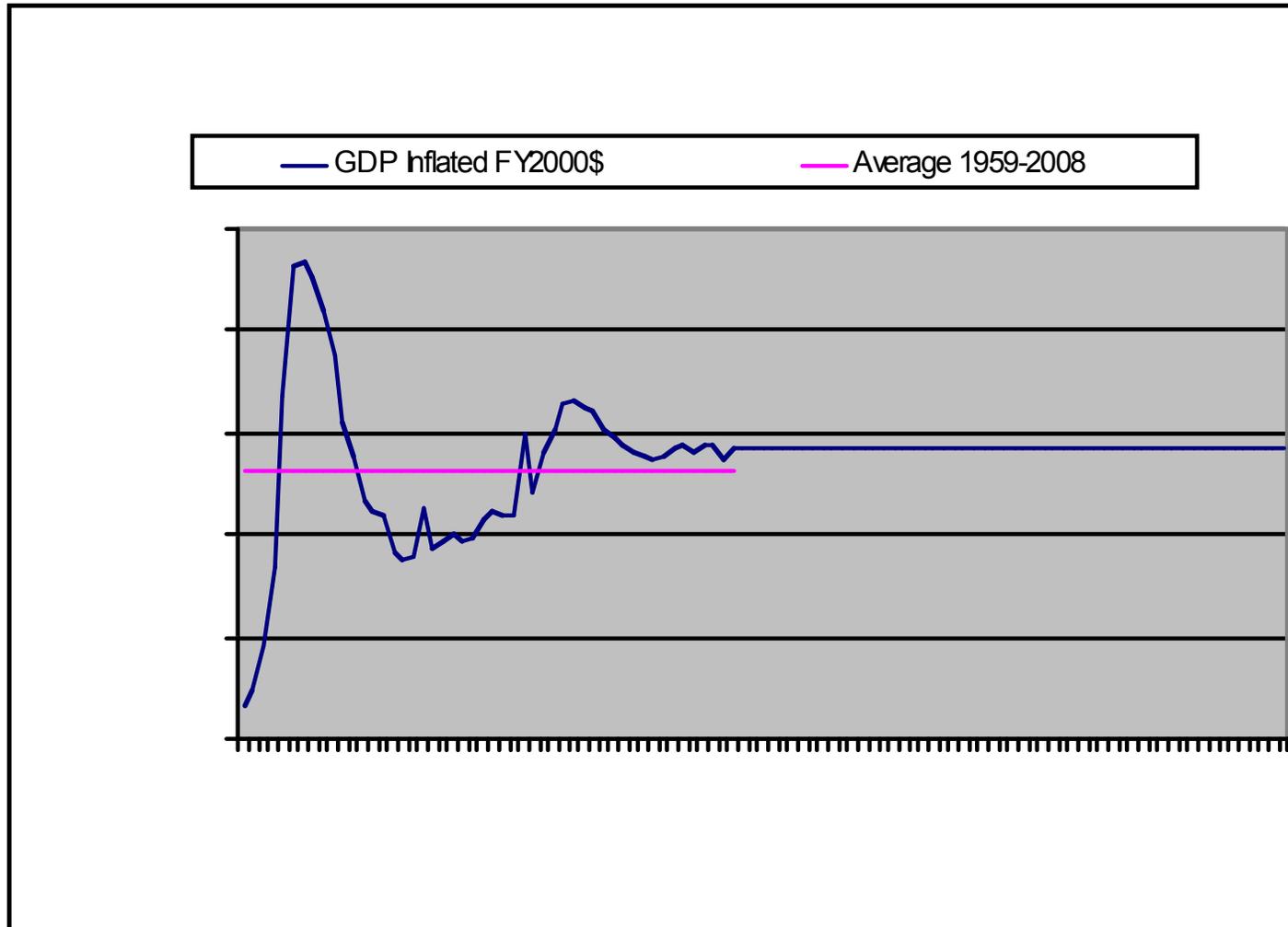


Software Trends





Same Money





Complexity Wall



378

D.A. Bearden / Acta Astronautica 52 (2003) 371–379

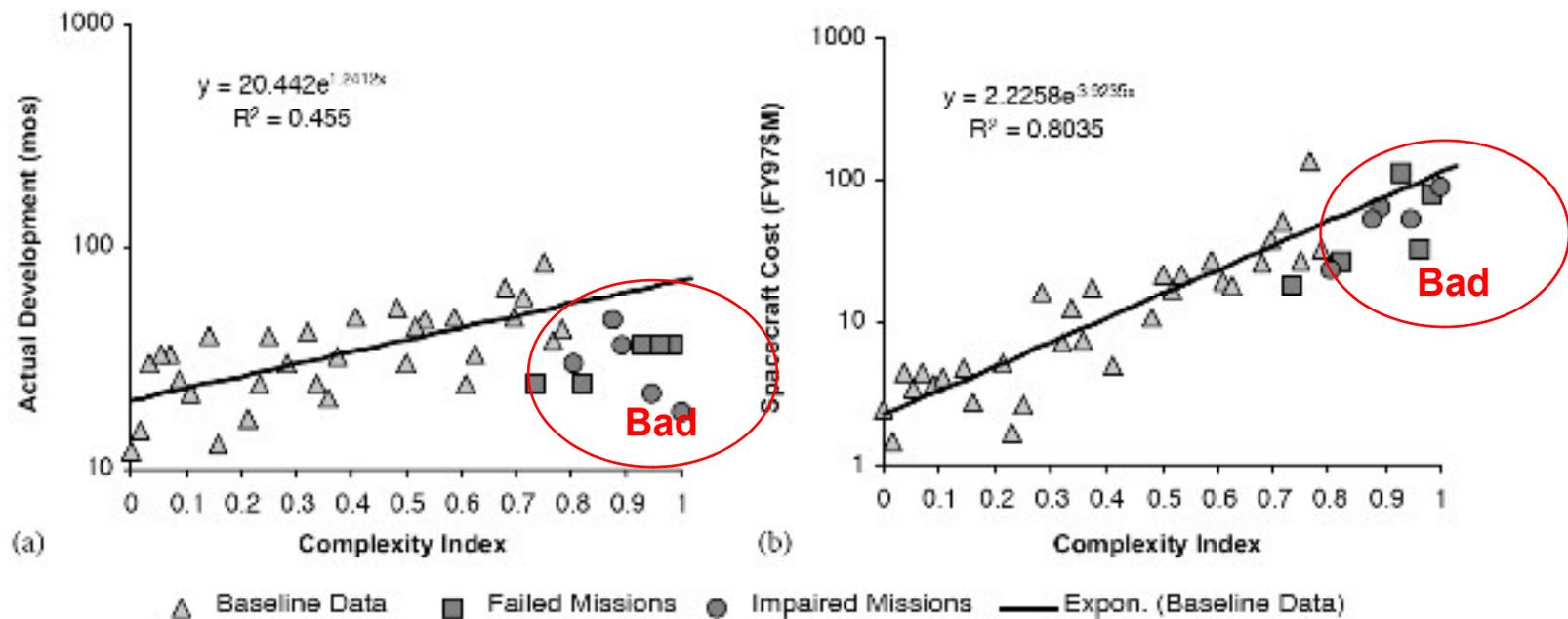


Fig. 1. Cost and schedule as a function of complexity. (a) Schedule as function of complexity, (b) Spacecraft cost as function of complexity.



Fundamental Limits



- ... We will always push limits of performance per dollar
- That's our business!



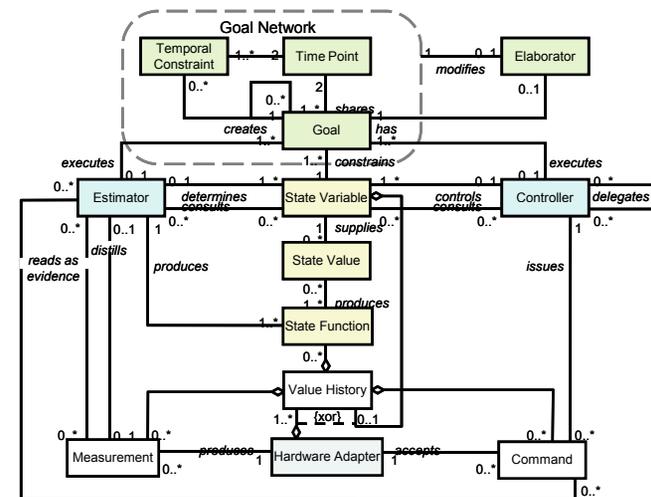
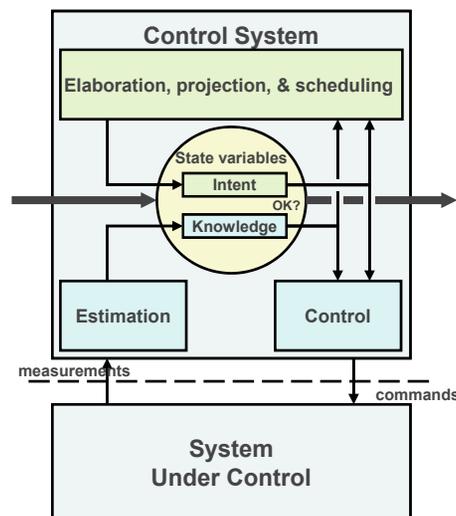
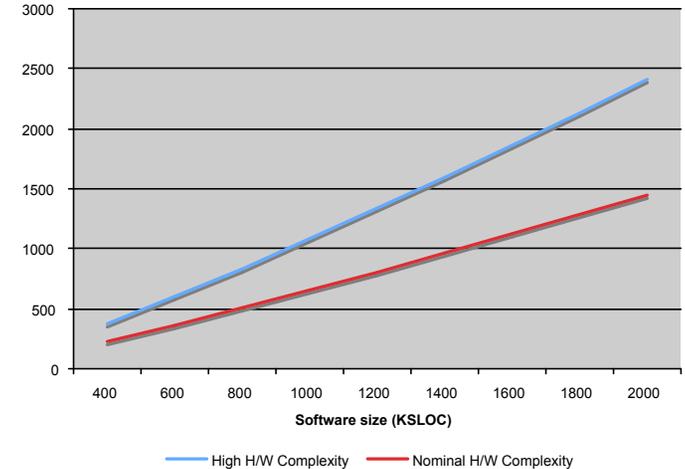


Managing Complexity



- Separate concerns - modularize
- Hide concerns - Abstraction, Layering
- Re-baseline the measure of complexity - Personal and institutional education, experience, commonality, standards, engineering handbooks.

Software Cost Driven by Hardware Complexity



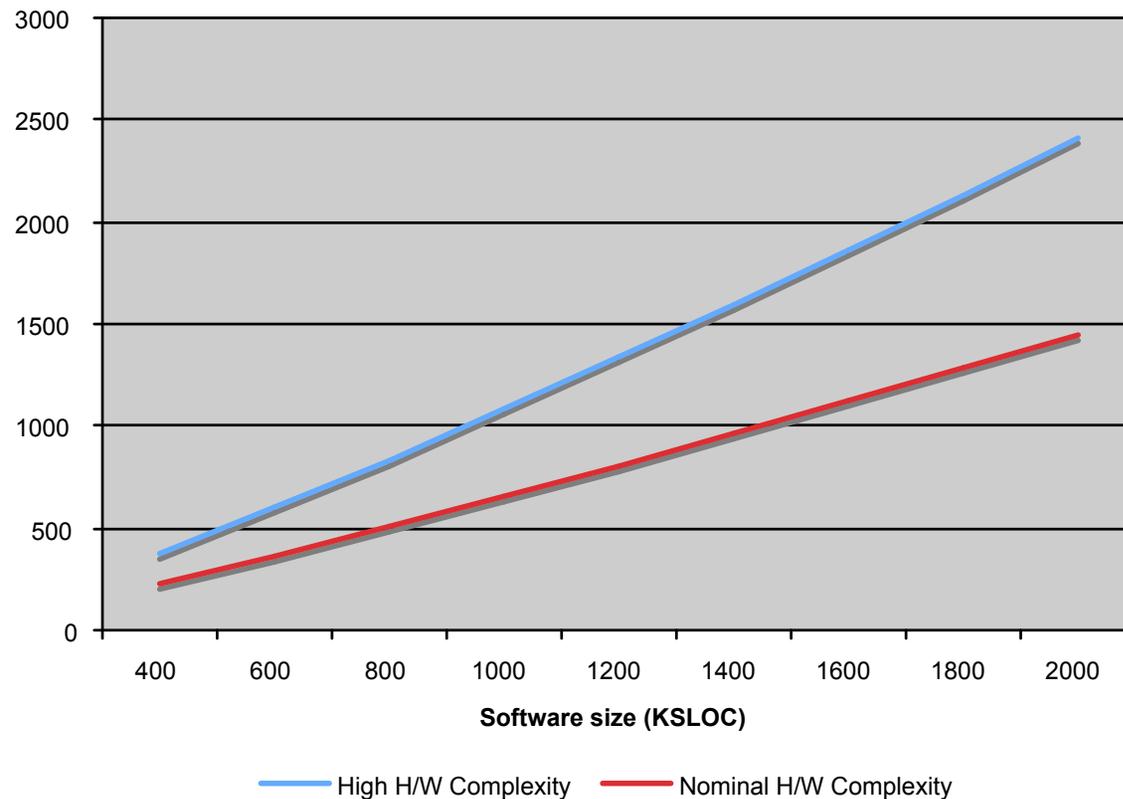


Managing Hardware Complexity



- **Hardware complexity is EXPENSIVE to deal with**
 - **Would rather spend this on capability**

Software Cost Driven by Hardware Complexity





Managing Hardware Complexity



- **Multi-threaded/Multi-processor programming model**
 - Application programmer using locks of any type **BAD BAD BAD**
 - Transactional memory, wait-free shared structures **GOOD**
 - MP/MT libraries (e.g. math) **GOOD** (as long as I can manage resources)
- **High interrupt rates, lots of kinds of interrupts **BAD****
- **Corner cases **BAD****
 - Long errata sheet very expensive
- **Imbalanced system **BAD****
 - Not enough cache
 - Highly NUMA
 - Insufficient RAM
 - Slow interconnect fabric
- **Same data at same time to all strings **GOOD****
- **Binary output compare **GOOD**. MVS etc **BAD****
- **Can't virtualize the CPU **BAD****
- **Inappropriate fault containment regions **BAD****
- **Can set initial hardware state **GOOD****
- **Repeatable execution state trajectory **GOOD****
- **No mechanisms for atomic operations (CAS, LL/SC) **BAD****
 - **BONUS POINTS: LL/SC atomic on two separate addresses **VERY GOOD****



Managing Hardware Complexity



- **Difficult I/O model BAD. Specifically...**
- **Double buffering GOOD**
- **Buffers that require very fast handling to avoid under/overflow BAD**
- **Inability to detect under/overflow BAD**
- **Registers that read back what was last written GOOD**
- **Atomic operations GOOD**
- **Atomic value split across registers with race condition BAD**
- **Any undetected race condition BAD**



What could we do with all that hardware???



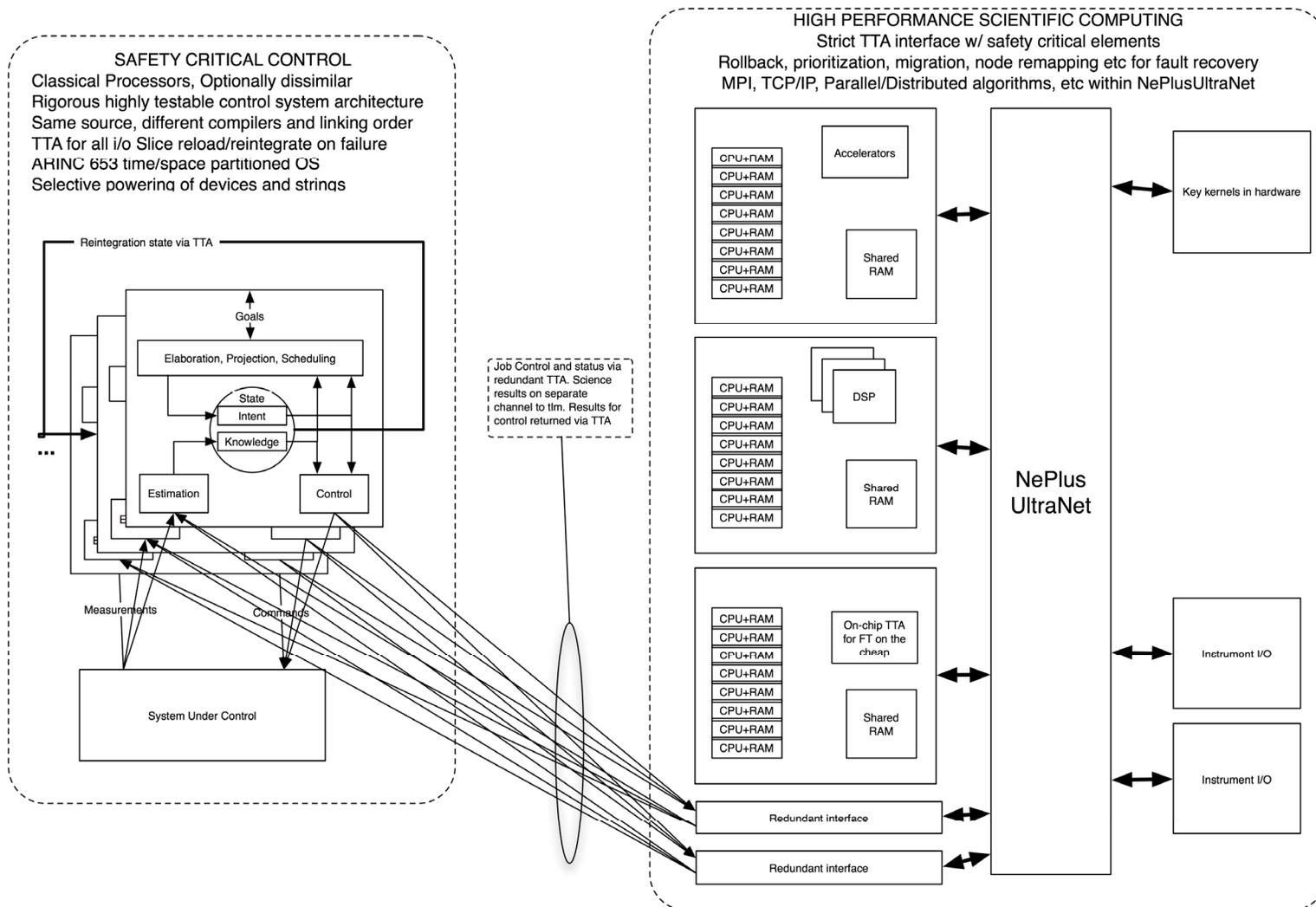
- OK, so we avoid the needless complexity of the previous slides
- What can we do with all those gates to make a leap in control capability and science return?
- Some speculations follow...



Putting it together - Conservative



- Extreme control reliability and extreme science compute capability



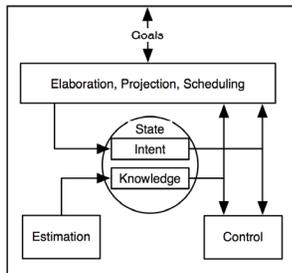


Putting it together - A bit radical

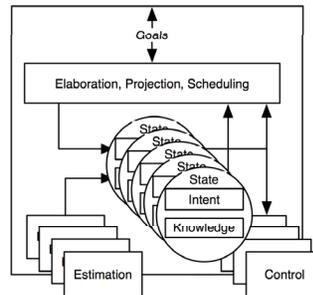


- **Extreme control reliability AND control compute capability**
 - (Science as on previous page)

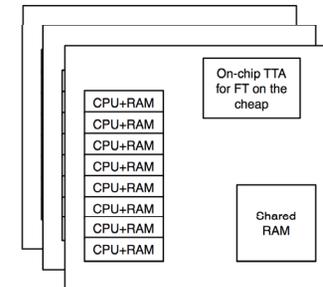
Centralized control system



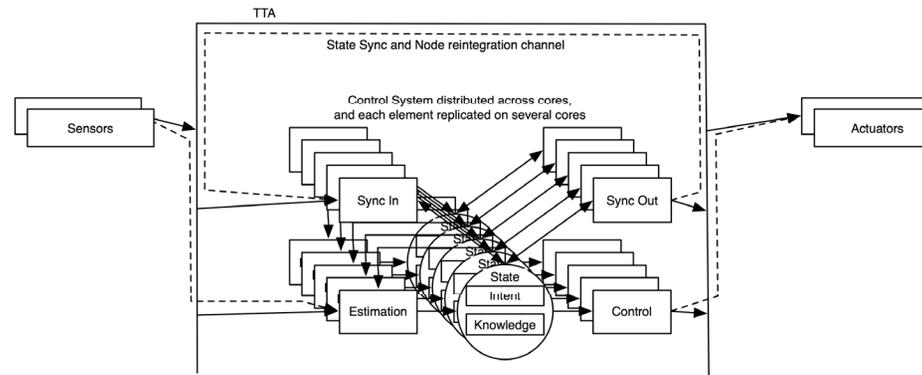
Easily made into distributed control system



Distribute and replicate on many cores



Combine with on-chip and off-chip TTA
yields extremely high computation capability
and extremely high fault tolerance for both
permanent and transient faults

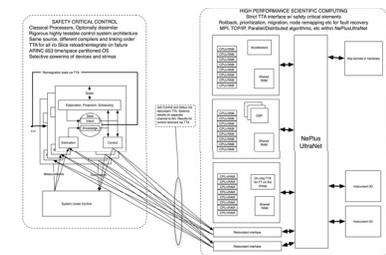
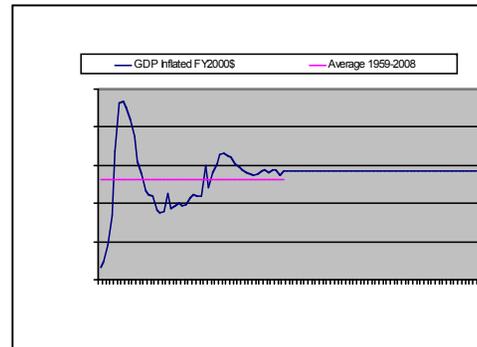
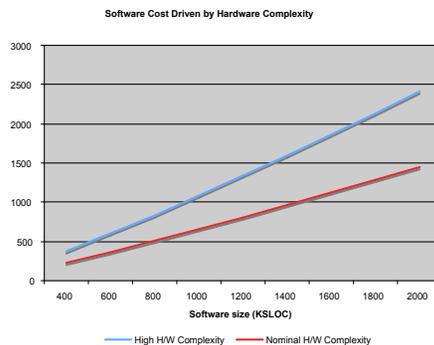




Wrap up



- Our job, basically, is to
- ... Deliver ever more capability
- ... At very high reliability
- ... Without more time
- ... And Without more money
- Complexity is the fundamental limit
- We must spend our complexity wisely
 - ... High capability (as viewed by customer) per unit complexity GOOD
 - ... complexity that bogs us down BAD
- Ever more gates is the basis of more capability
- ... BUT only if we can use them effectively and reliably
- Examples given show this is a reasonable objective





Future Work



- **Develop programming models for the new hardware**
- **Integrate software architecture for safety critical uses with TTA**
- **Provide read reserved/store conditional or CAS that operates on two separate addresses simultaneously**