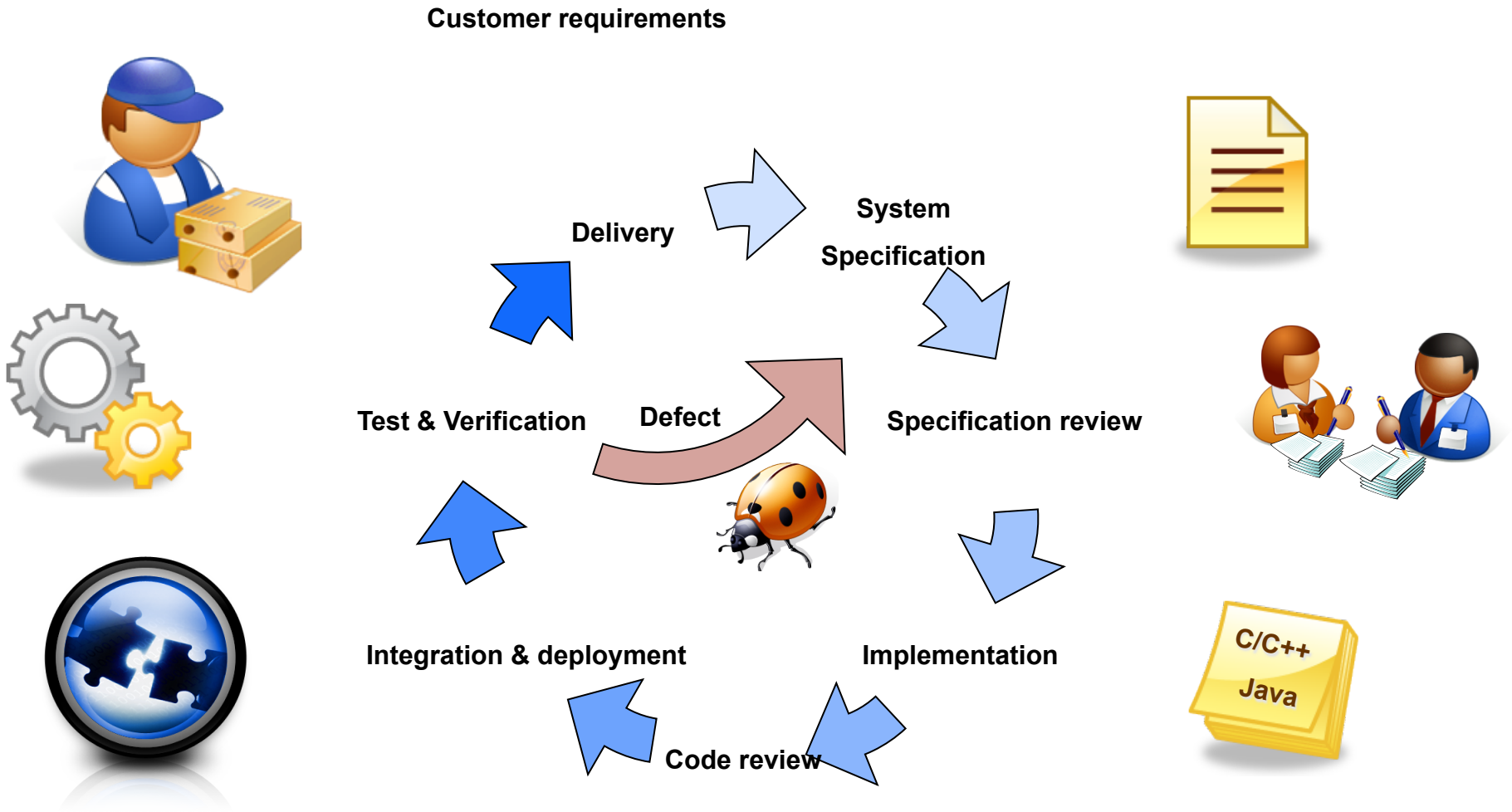
Traditional Development Process

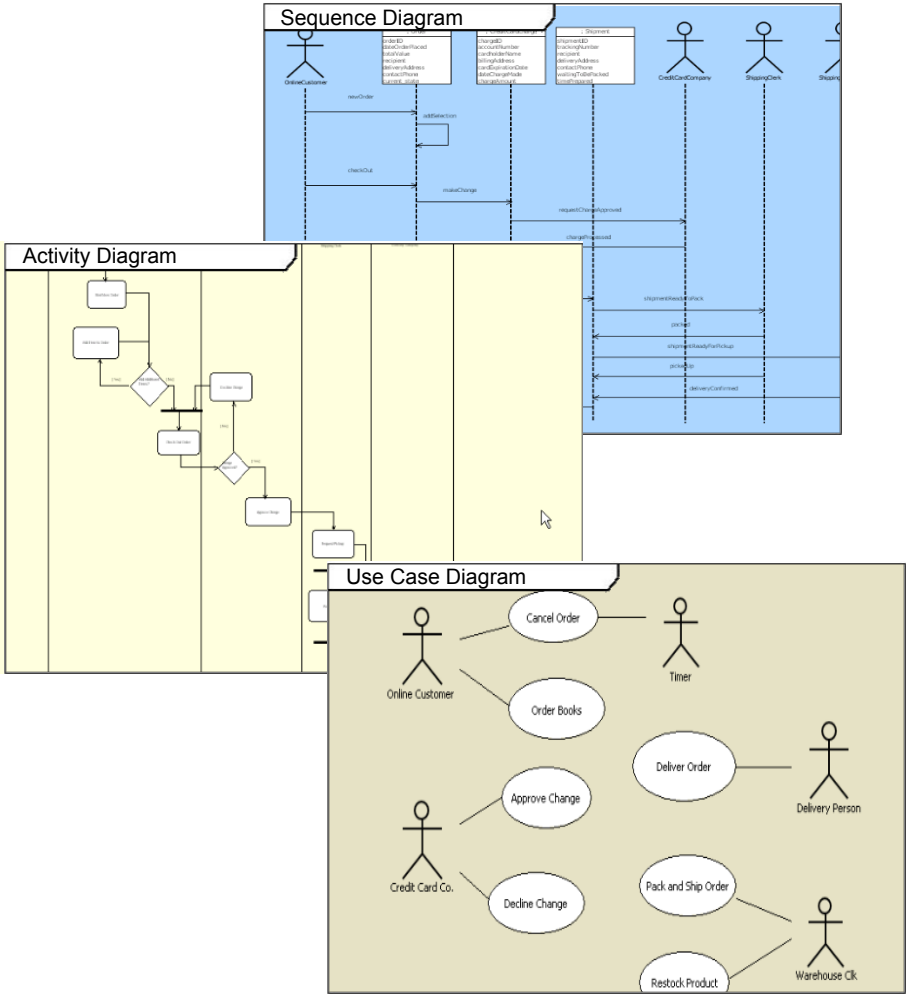


Levels of Commitment

1. **Natural language and informal diagrams**
   - Use case, sequence diagrams etc.
2. **Structural models**
   - Interconnected components
   - Interfaces
   - Data types
3. **Passive class models**
   - Classes
4. **Behavioral models**
   - State machines
   - Active ports

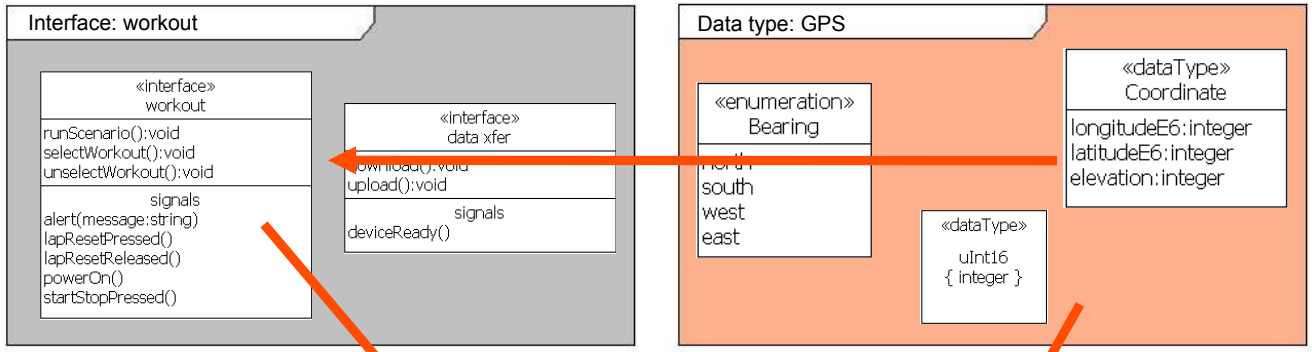
Natural Language and Informal Diagrams

- Common tool environment
- Well known concepts
- Report generation



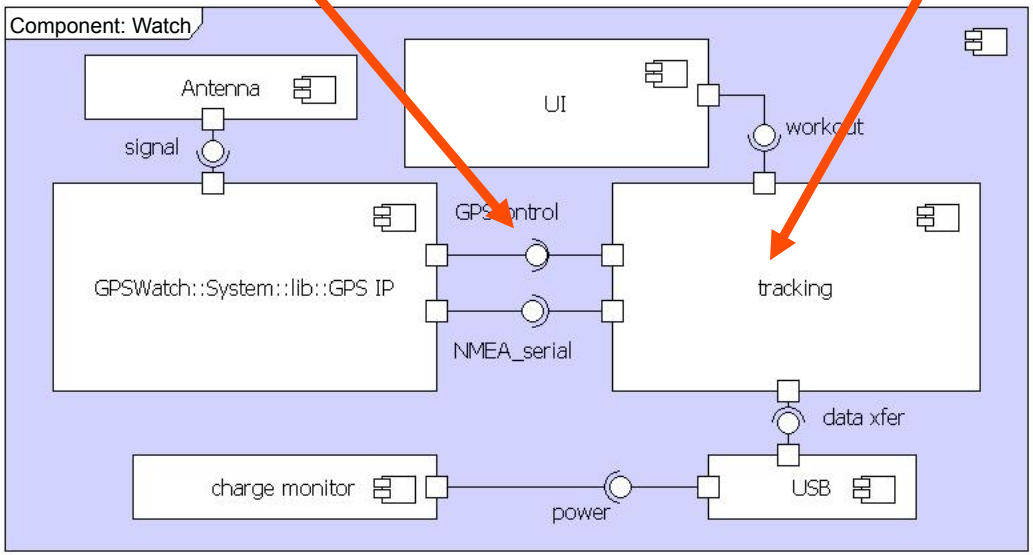
Structural Models

System level interfaces declare messages that carry data



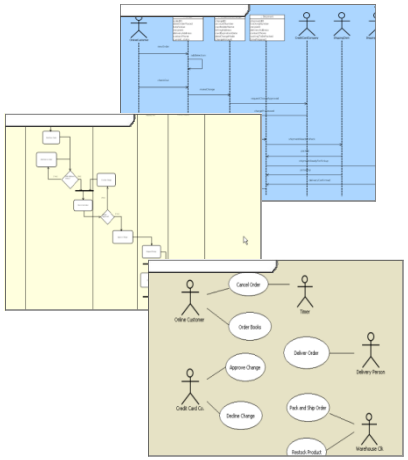
System level data types

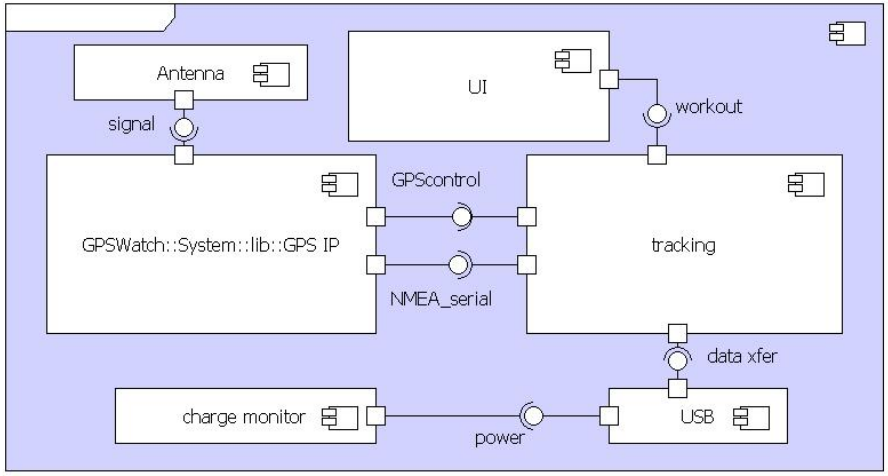
Interfaces are provided and required by components



Data types are used in component design

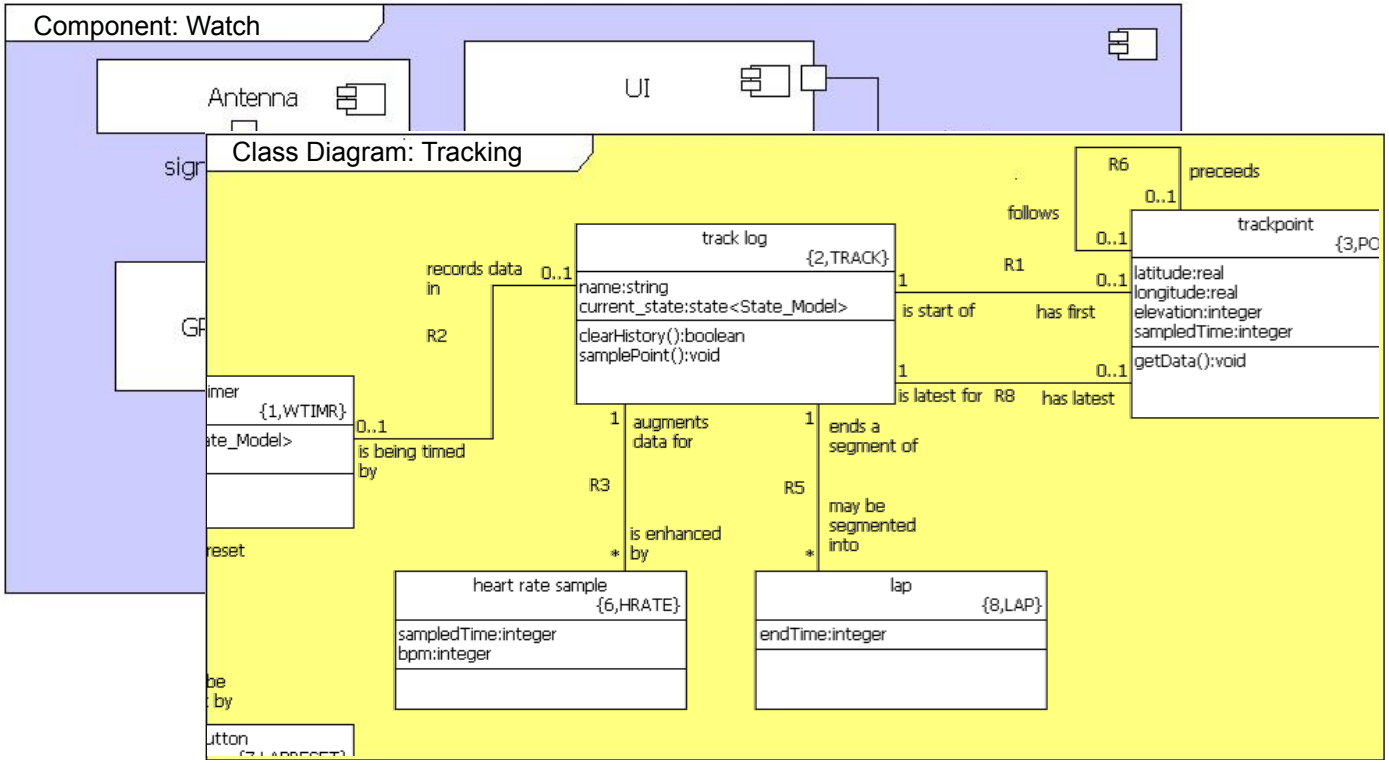
Structural Models



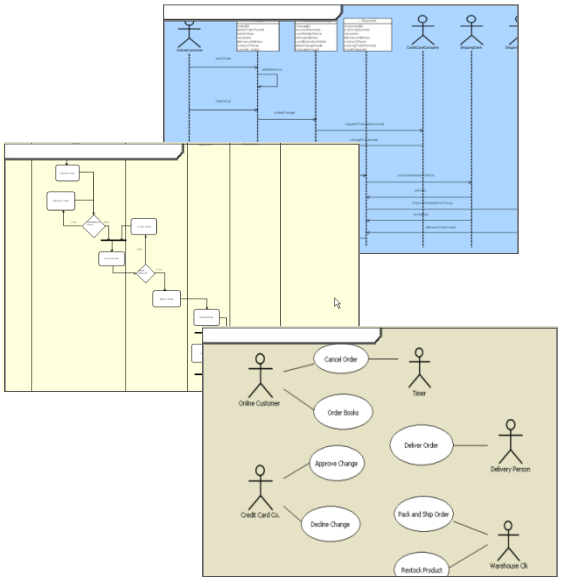


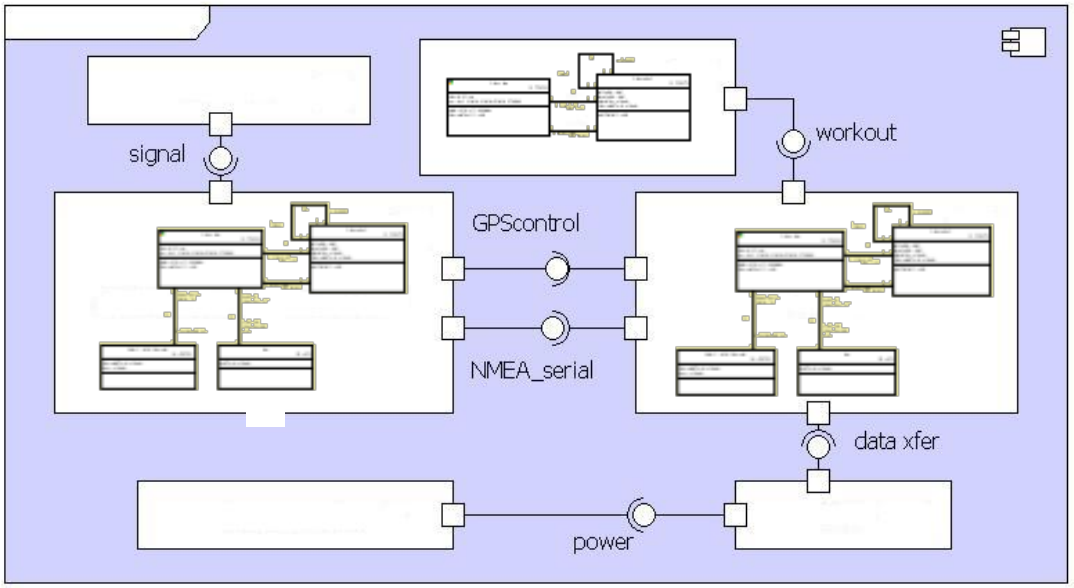
- Work that is already being done
- No inconsistencies – one source of information
- Reuse of components, interfaces and data types
- Model diff reveals communication changes
- Allow informal models to be formalized

Passive Class Models



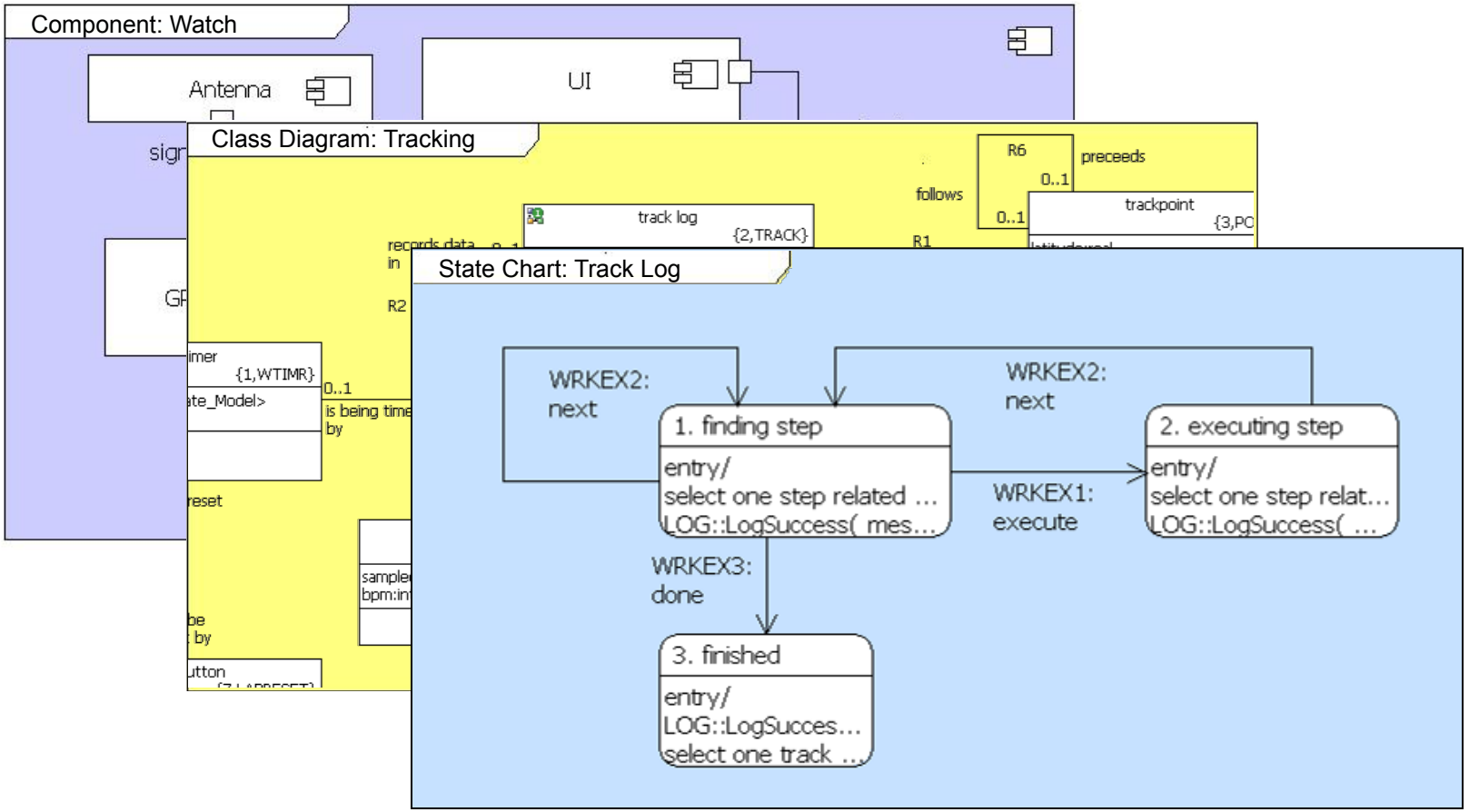
Passive Class Models



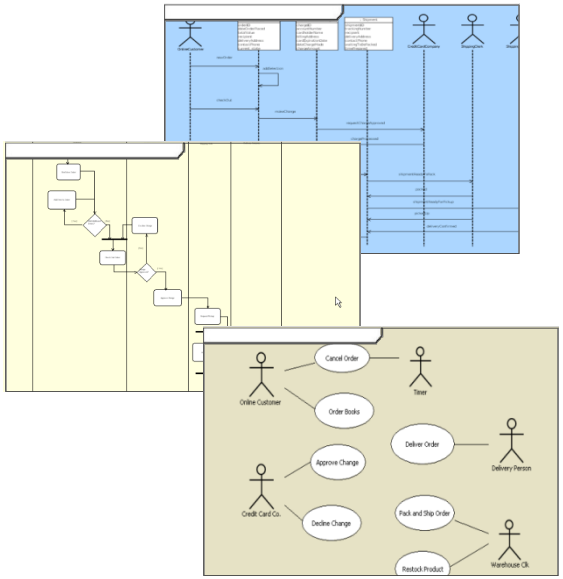


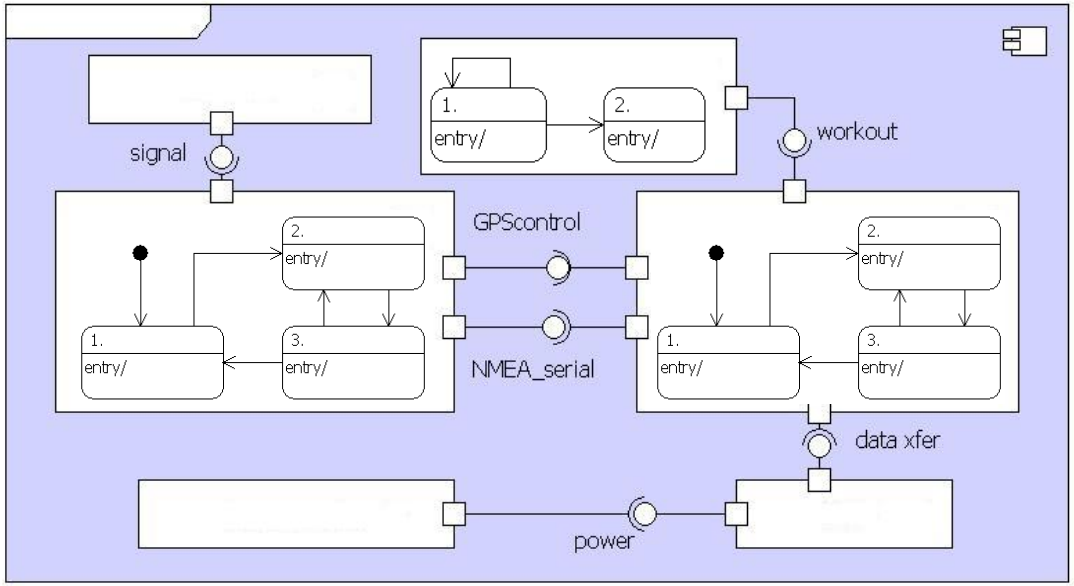
- **Significantly improves impact analysis**
- **Class model reuse**
- **Reachable with no risk and little effort**

Behavioral Models



Behavioral Models





- Forced to deal with the hard questions up front
- Execution reveals defects early
- Can be introduced in small portions
- Reuse behavioral models in design phase

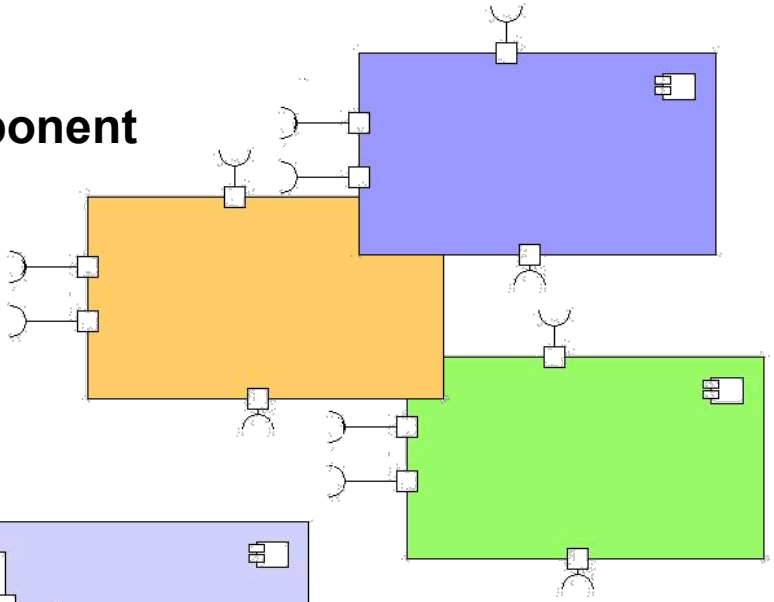
Sumo Robot Competition – Round 3

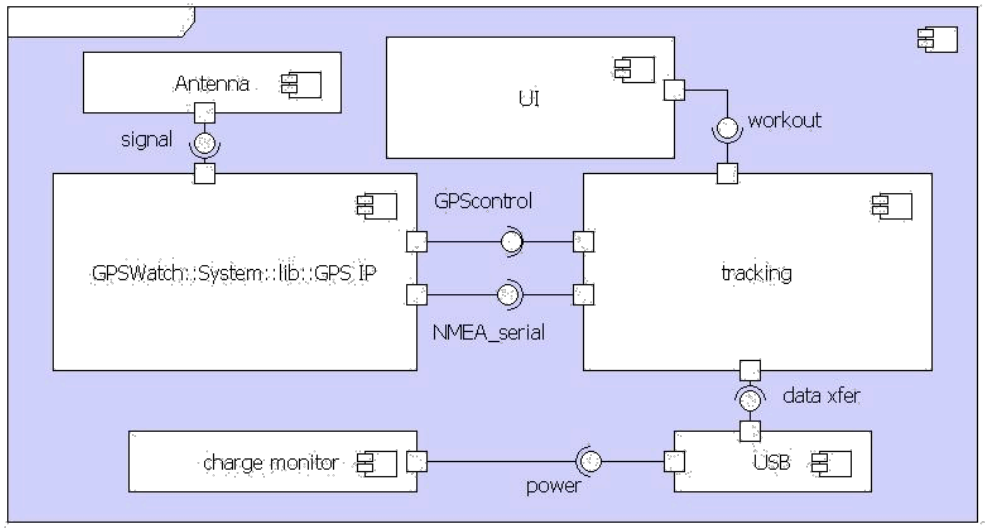
UML 2.0 Component Definition

- A modular part of a system design that hides its implementation behind a set of external interfaces
- Within a system, components satisfying the same set of interfaces may be substituted freely

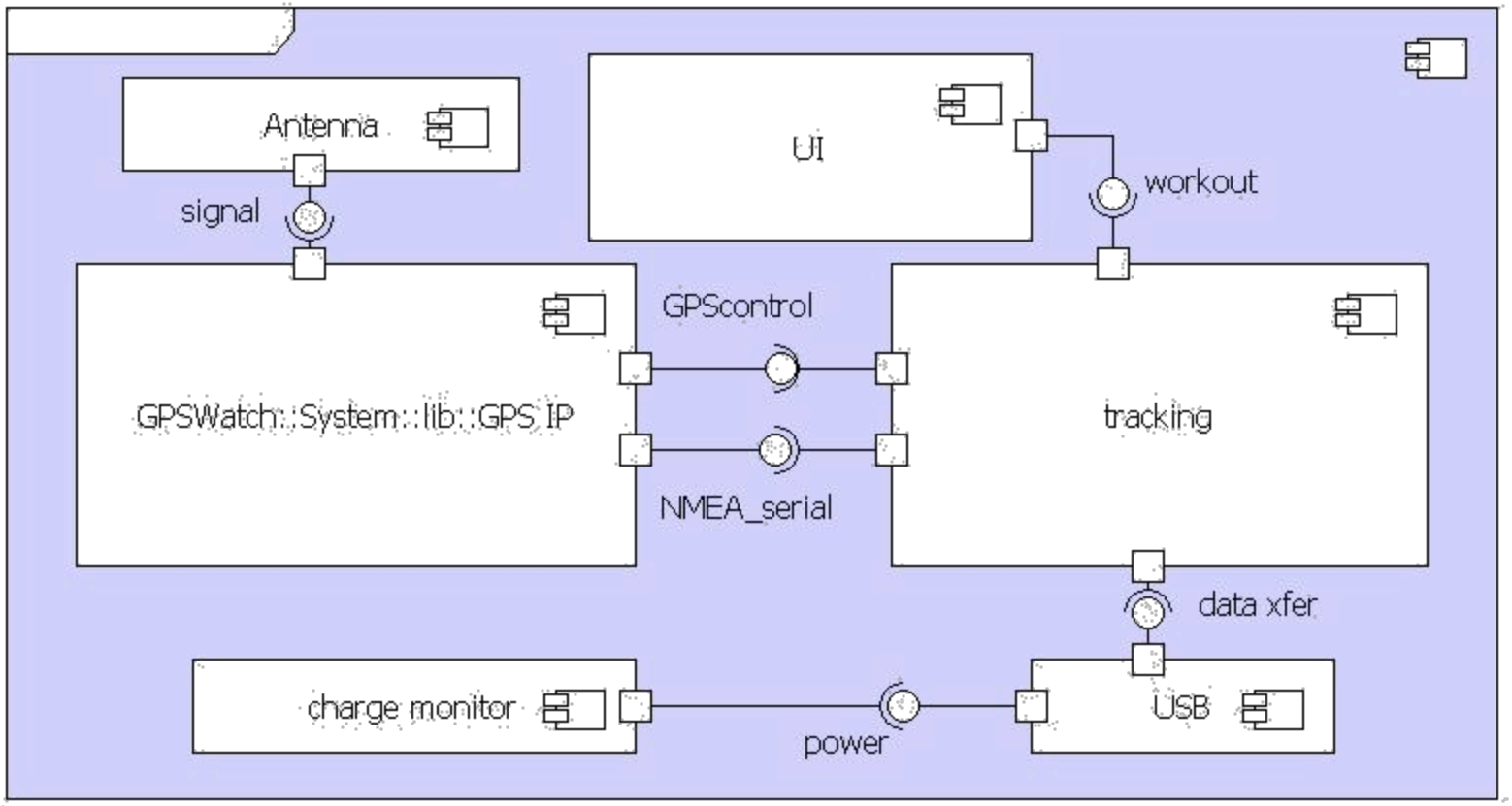
Components and Substitution

- **A component may be:**
  - **Behavioral system level component**
  - **Implementation component**
  - **Test stub**
  - **External code**
  - **Others…**





Refining the model for code generation

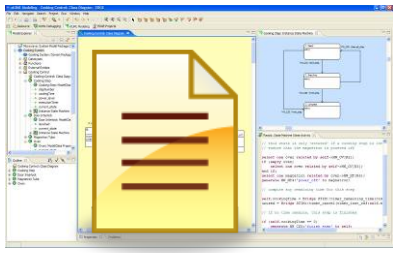


System spec

Implementation

Test stub

Improved Development Process



Customer requirement

Application Specification

Spec. review

Implementation

Code review

Integration & deployment

Test & Verification

Defect removal

Delivery

Conclusions

- Executable specs can be introduced in small increments in existing modeled or non-modeled systems
- Verification takes place early when recovery is cheap
  - Increase productivity and improve quality
- Executable specs are unambiguous
- System level models may be reused in design phase
  - Minimal handover
  - Bridge the gap between system and design