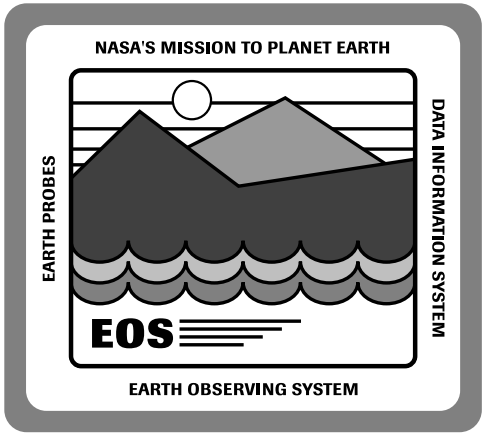
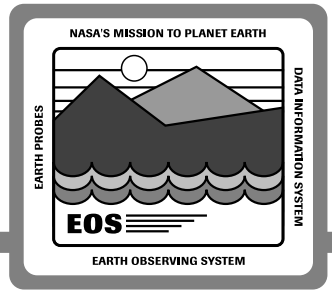


Evolution and Evolvability

George Percivall

13 - 14 December 1993

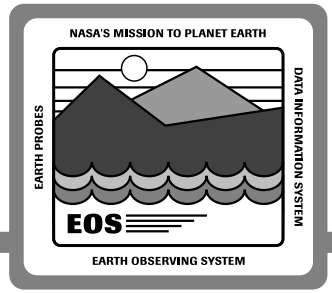
Evolution and Evolvability



Modifications to ECS Development Methodology for Evolution and Evolvability

- Emphasis was on incremental releases, prototyping for risk reduction
- Modifications in user involvement, collaborative prototyping, ongoing system architecture, prediction evolution
- Define ECS Processes using evolution and evolvability

Definitions



Evolutionary Development Process

Definition: development methodology that adapts throughout lifecycle using iteration, user feedback, and "in-line" prototyping

Goal: Develop a system which best meets users needs

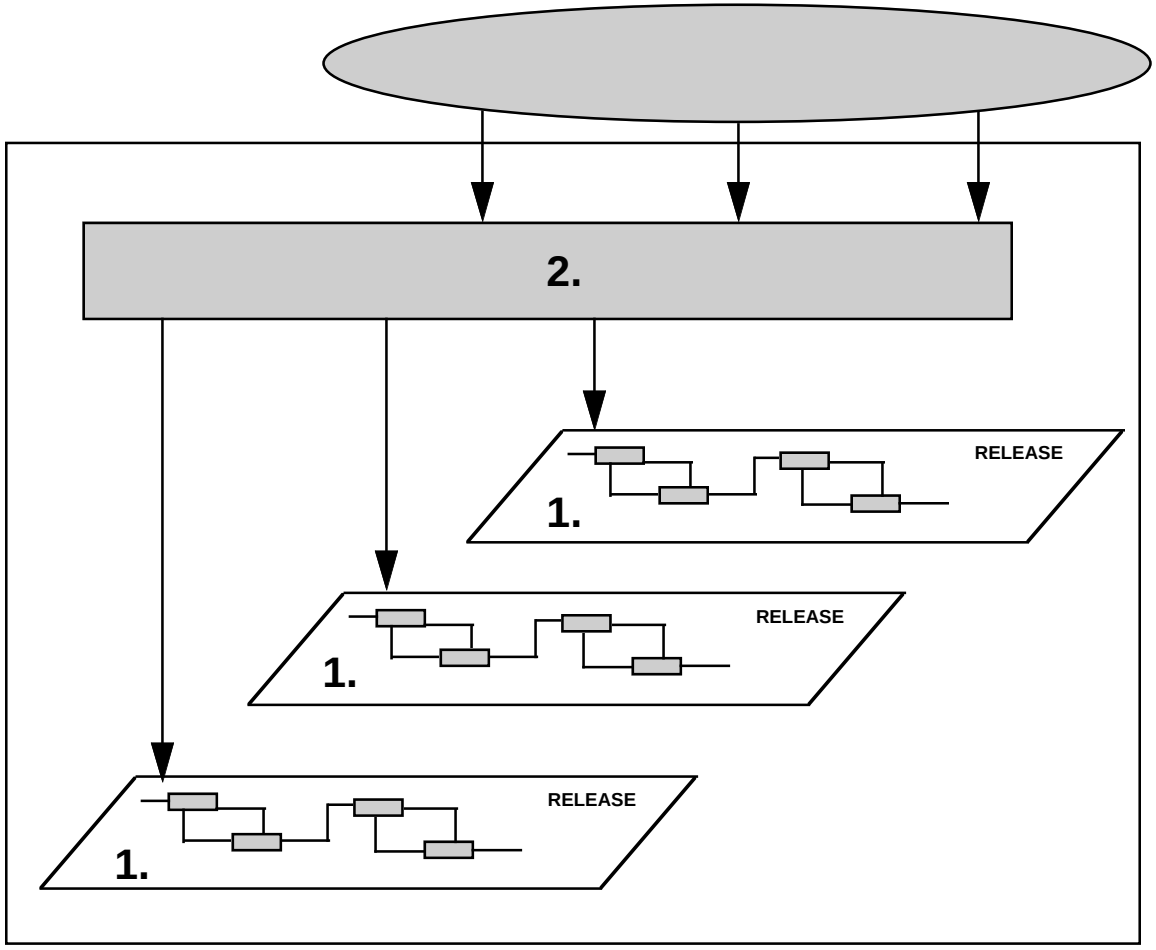
System Evolvability

Definition: the ability of a system to accommodate changes to the requirements, design or implementation in a timely and cost effective manner

Goal: Enable the system to be changed economically

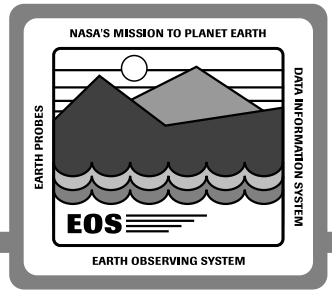
- **Isolate changes to "Affordable" component replacements or modifications.**
- **Separate fast changing items from slowly changing items**

Evolution Categories



1. **Refinement Through Iteration: convergence of requirements and design (e.g. review of interface mockup)**
2. **Planned Improvements: improving and adding functionality, technology insertion, system growth (e.g. upgrade of ECS using DAAC development)**
3. **Paradigm Changes: major changes in user paradigm change in system architecture (e.g. change to OO op. system)**

Evolutionary Development Process



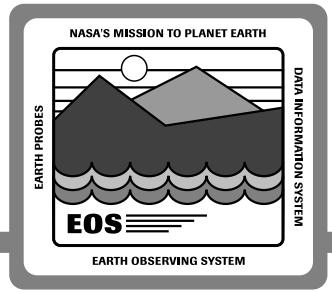
Key Elements of Evolutionary Development Process

- **User Involvement**
- **Prototypes**
- **Competition**
- **Technology Assessment**
- **Multi-Track Development**
- **Risk Management**

Each Key Element Aligned with Evolutionary Development

Evolutionary Development Process

User Involvement



Continuous, active user involvement in system decisions

<u>Evolution Category 1</u>:

- **Reviews, RRDB, focus teams**
- **ECS Tirekickers ala V0 Tirekickers**
- **Collaborative Prototypes**

<u>Evolution Category 2</u>:

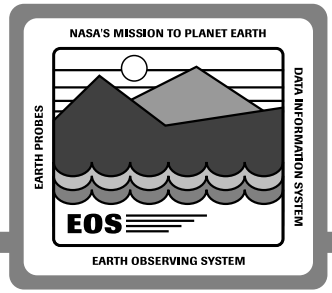
- **ECS upgrades based on user preferences at DAACs, SCFs**
- **User involvement in allocating functionality to releases**

<u>Evolution Category 3</u>:

- **ECS University Academic Outreach**
- **NASA Research Announcements**
- **Alternate Architecture Study**

Evolutionary Development Process

Prototypes



Prototype to support decisions about future steps in development process

Technology Analysis Prototypes

- **Supports a particular release (Evolution Category 1)**
- **Last step in the technology assessment process**
- **Examples: Evaluation of Archive Robotics Unit**

Engineering Prototypes

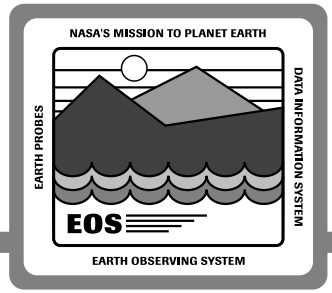
- **Iteration of requirements and design (Evolution Category 1)**
- **Release planning based on prototype results (Evolution Category 2)**
- **Example: Building an Expert Advisor/Decision Support for FOS**

Advanced Prototypes

- **Visionary investigations: new requirements, paradigms, or architectures for ECS**
- **ECS related, non-release specific (Evolution Category 3)**
- **Example: Alternate Architecture Study**

Evolutionary Development Process

Competition



Benefits of competitive approaches to evolve system

<u>Evolution Category 1</u>: Parallel prototypes

- **Identify functions and competitors, define requirements**
- **Method for selecting winner or merging of designs**

<u>Evolution Category 2</u>: Merging community solutions into baseline

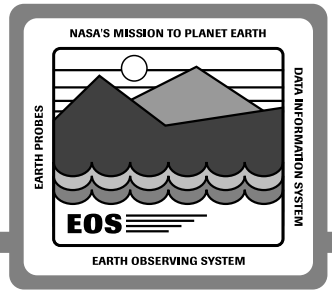
- **ECS infrastructure to enable, foster competition: e.g. APIs, providing source code**
- **ECS keep abreast of "non-ECS developed" solutions**
- **Merge some solutions into ECS baseline at later releases**

<u>Evolution Category 3</u>: Parallel system definitions

- **Visionary system concepts through advanced prototypes**
- **Study other systems implementing similar requirements**
- **Consideration of system architecture changes**

Evolutionary Development Process

Technology Assessment



Forward looking assessment and planning for evolution

Evolution Category 1: Current Technology

- **Current Performance**

  FDDI, PC's, X-Terms, Small Workstations, etc.

  Reduced Future Price

Evolution Category 2: Technology Insertion

- **Future Performance**

  Supercomputers, Compute Servers, Workstations, etc.

- **Future Technology Presently Planned for Use in ECS**

  3480 Optical Tape Drives and Media, DME, etc.

Evolution Category 3: Technology Evolution

- **Future Technology Under Study**

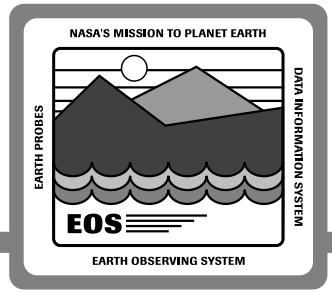
  MPP Proliferation, High Performance Fortran, Extended Relational and OO Databases, etc.

- **Standards**

  Open Systems Environment: Portable Systems Software, Portable Applications, Portable Users, Portable Data, Open Communications, etc.

Evolutionary Development Process

Multi-Track Development



Use development methodology appropriate to System/Segment priorities

Formal Release Development Track

Priorities: Requirements well understood, Support of launch or data storage

Process: Waterfall, Formal Reviews, Documentation

Examples: Data Ingest, Data Storage, Network Management, Flight Operations

Incremental Development Track

Priorities: Requirements change with design evaluation

Process: Iteration, Monthly Demonstrations, Informal Documentation, Merge with formal track for I&T

Examples: Data Access and Management, Toolkits

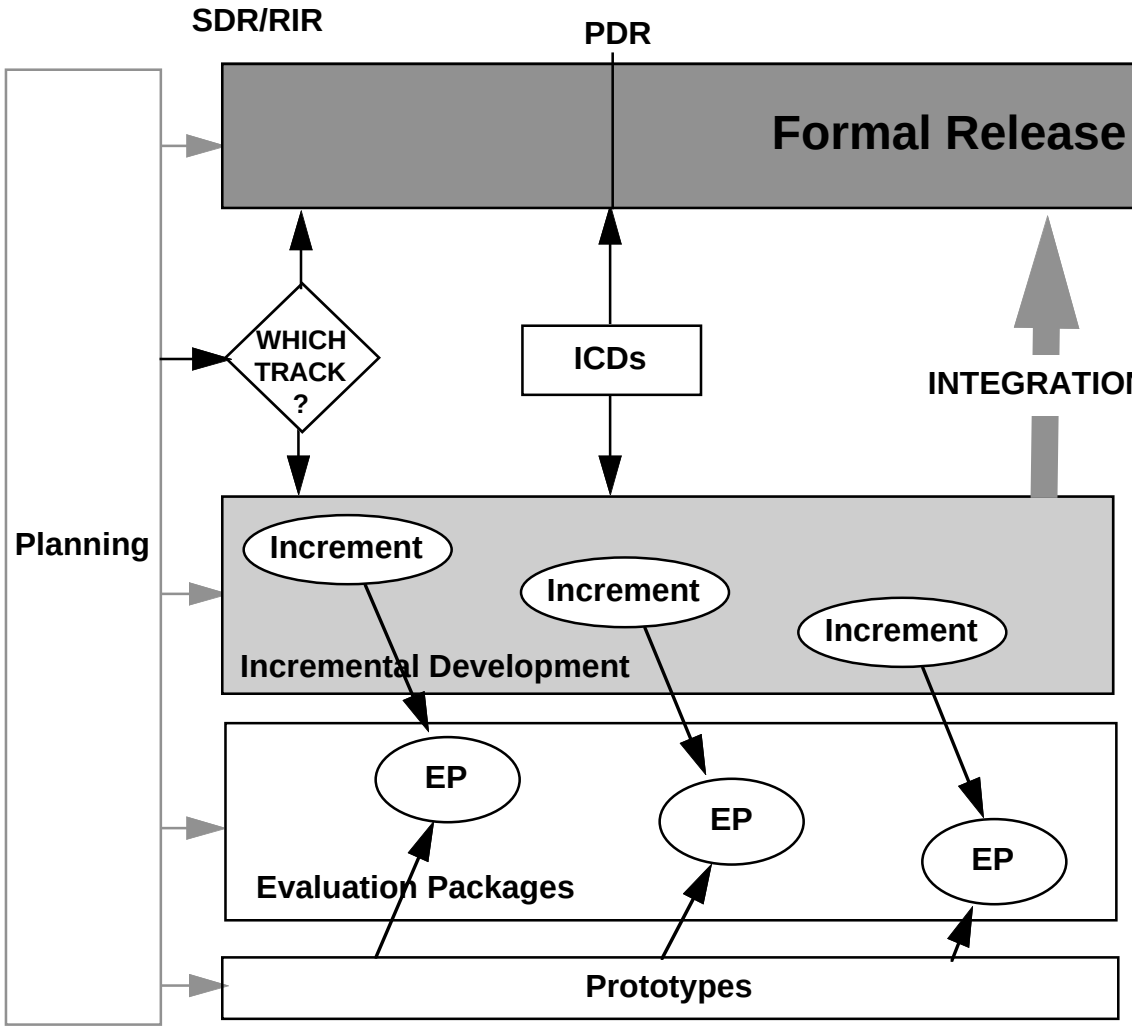
Evaluation Packages

Combination of prototypes and increments

Delivery Mechanism for User Feedback

Evolutionary Development Process

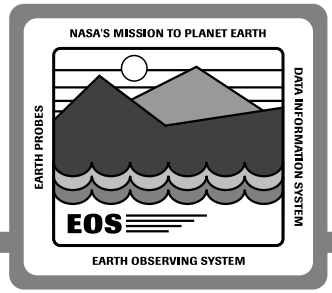
Multi-Track Integration



- **WHICH TRACK?**
  - SERVICES ALLOCATED TO A TRACK
  - BASED ON REQUIREMENTS STABILITY
- **ICDs**
  - INTERFACE CONTROL DOCUMENTS
  - FORMALLY DEFINED FOR PDR
- **INTEGRATION**
  - BEGINS IN INCREMENTAL
  - BASED ON AS-BUILT DOCUMENTATION
  - INTO FORMAL TRACK FOR TRR

Evolutionary Development Process

Risk Management



Design for Evolvability -- Implement to Requirements and Budget

Evolutionary Risks considered in Risk Management process

<u>Evolution Category 1</u>: Requirements Creep, System Reliability

- **Requirements creep -- Mitigation: Continuing system architecture activity.**
- **Lower reliability of iterated code -- Mitigation: Reliable integration of development tracks**

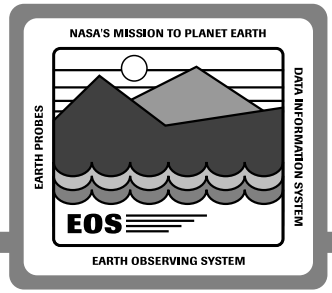
<u>Evolution Category 2</u>: Maintainability, Adaptive Flexibility.

- **Maintainability of iterated code -- mitigation: Reliable integration of development tracks, Software development practices, Adherence to standards and architecture constraints**
- **Adaptive flexibility to changes -- mitigation: System Evolvability**

<u>Evolution Category 3</u>: Unanticipated Large Changes.

- **Large late changes of ECS -- mitigation: Evolutionary Development Process, System Evolvability**

System Evolvability - Agenda



Active process of projecting present system into the future and evaluating resulting changes

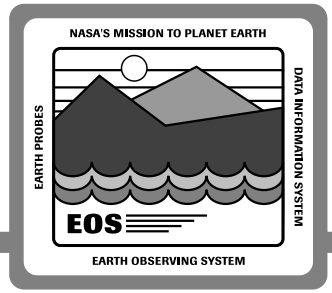
Predicting Evolution

- **User/Data Model**
- **Evolvability Design Criteria**
- **Evolvability Requirements**
- **Evolvability Tests**

System Architecture

- **Elements of a System Architecture**
- **Architecture for Evolvability**
- **Architecture Change Analysis**

Evolvability Design Criteria



Adaptive Flexibility
Prepare for Expansions, Improvements, Extensions

Interoperability Features
Transparency of Access, Location, Migration, etc.

Distributed Design Features
Distributed Control, Autonomy, Variety of Users, etc.

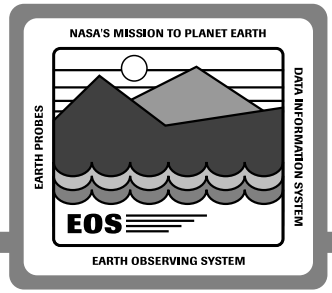
Standard Design Techniques

Heuristics Used in Trade Studies and Evaluation of Architecture and Designs

Criteria to be Listed in Specifications

Validated by Inspection at SDR, PDR/IDRs, CDRs

Evolvability Requirements



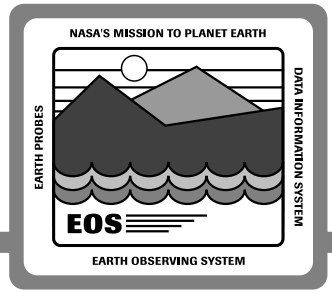
Definition of Evolvability Requirements

- Requirements for changes that are likely
- Contained in specifications, Validated by analysis at PDR/IDR
- Present specification contains growth requirements

Existing Level 3 Evolvability Requirements - Paraphrased

- Accommodate growth in all functions and addition of new functions (EOSD0545)
- Support additional spacecraft (FOS-0050)
- DAAC growth without changes to architecture or design (DAAC0290)
- ESN capacity, performance extensions to meet ECS functions (ESN-1207)

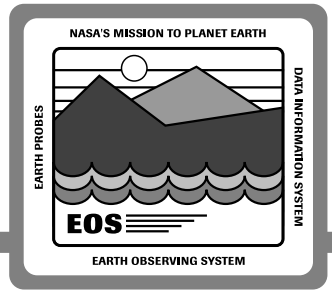
Evolvability Requirements
(Continued)



Potential Level 3 Evolvability Requirements

- Provide APIs and infrastructure for science user extensions and direct access to data
- Evolve to be an apparent federated unit within GCDIS, e.g. GCDIS data centers should not have to negotiate different interfaces with each DAAC
- Facilitate sharing of ECS-developed software with other GCDIS agencies and science users
- Extended provider support, e.g. client can access data/services at SCF and DAAC interoperably
- Enable addition of significantly different data types with minimal changes
- Enable addition of ICCs, ISTs
- Enable adding a DAAC, service providers
- Enable expansion to GByte Network

Evolvability Tests



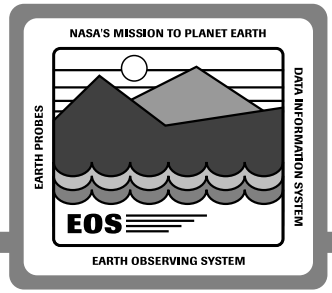
Definition of Evolvability Tests

- User paradigm shifts or technology evolution which ECS would need to accommodate
- Future requirements, Not in specifications
- Basis for Architecture Change Analysis

Examples

- Multi-media collaborative environments, videoconferencing.
- Heterogeneous (e.g. non-Posix compliant) operating system environment
- Dynamically adding or dropping a DAAC, ADC or ODC
- Advances in extended relational and OO Databases
- High Performance Computing (MPP proliferation and workstation farms)
- Networked *in situ* sensors: deployed sensor arrays with realtime data acquisitions

Elements of a System Architecture



Architecture: Modules, Interfaces, Design Constraints

Provides structure, reduces complexity to allow engineering analysis for system design

Developing and adhering to architecture is key to evolvability

<u>Modules</u>

- **System partitioned into modules: e.g. subsystems, components**
- **Defined by allocation of services: e.g. advertising service**

<u>Interfaces</u>

- **Greatest leverage in architecting is at interfaces**
- **Data flow, control flows**
- **Standard interfaces for plug and play in "Earth Science Web"**

<u>Design Constraints</u>

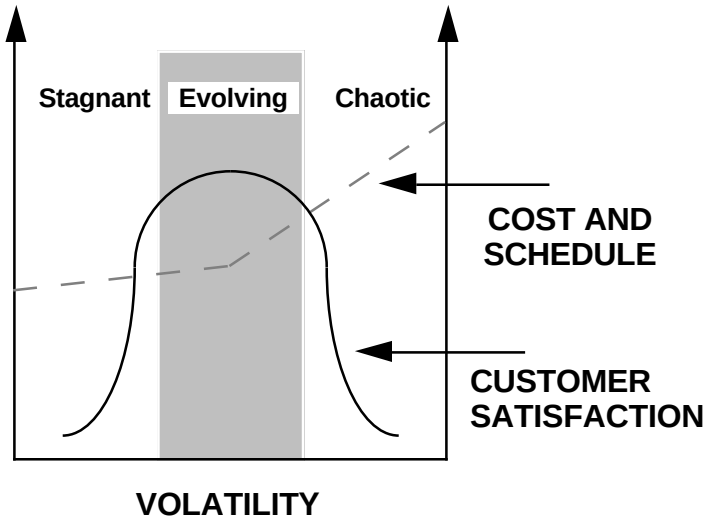
- **Reference Model and Applicable Standards**
- **System interconnection structure**

System Architecture For Evolvability

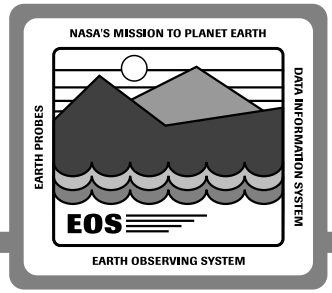
Traits of a System for Optimal Evolution

Based on studies of multiple types of complex systems

- System complexity increases with time
- Small reusable components
  - Change at the component level
- Mixture of Substructure Dynamics
  - Stable intermediate forms
  - Changing components for evolvability
- Boundary layer of volatility
  - Analogy to phase transition



Architecture Change Analysis



Process for looking "near-term" and "far-term" simultaneously and iteratively

Analyze changes in architecture to address an Evolvability Test

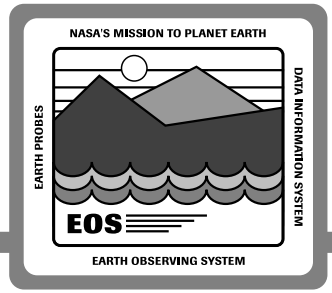
Architecture Change Analysis part of each cycle, evolvability considered at each review:

- **Start with present architecture**
- **Identify applicable Evolvability Tests**
- **Analyze change in architecture to accommodate test**
- **Modify architecture to minimize cost of transition**

Present examples

- **GCDIS/UserDIS Study**
- **Parallel and Distributed Testbed**

ECS Evolutionary Development Summary



To be successful, ECS must evolve

Evolutionary Development Process

- **Develop the System interactively with users**
- **Embed evolution in all processes: Prototyping, Technology Assessment , Competition, Multi-Track Development, Risk Management**
- **Prepare for all sizes of changes: iteration, upgrades, paradigm shift**

System Evolvability

- **Deliver a System which can change cost-effectively**
- **Predict System Changes: User/Data Model, Evolvability Criteria, Evolvability Requirements, Evolvability Tests**
- **System architecture to prepare for evolution**