CS 425 November 7, 2024

Chapter 10 – Dependable Systems

Ian Sommerville,

Software Engineering, 10th Edition

Pearson Education, Addison-Wesley

Note: These are a slightly modified version of Chapter 10 slides available from the author's site http://iansommerville.com/software-engineering-book/

Topics covered

- Dependability properties
- Sociotechnical systems

The following (in purple) are not required for the final exam:

- An example of STS: the cyber infrastructure of the NSF-funded Nevada Nexus project (past project with Dascalu co-PI)
- Redundancy and diversity
- Dependable processes
- Formal methods and dependability

System dependability

- For many computer-based systems, the most important system property is the dependability of the system

- The dependability of a system reflects the user’s degree of trust in that system. It reflects the extent of the user’s confidence that it will operate as the users expect and that it will not ‘fail’ in normal use.

- Dependability covers the related systems attributes of reliability, availability and security. These are all interdependent.

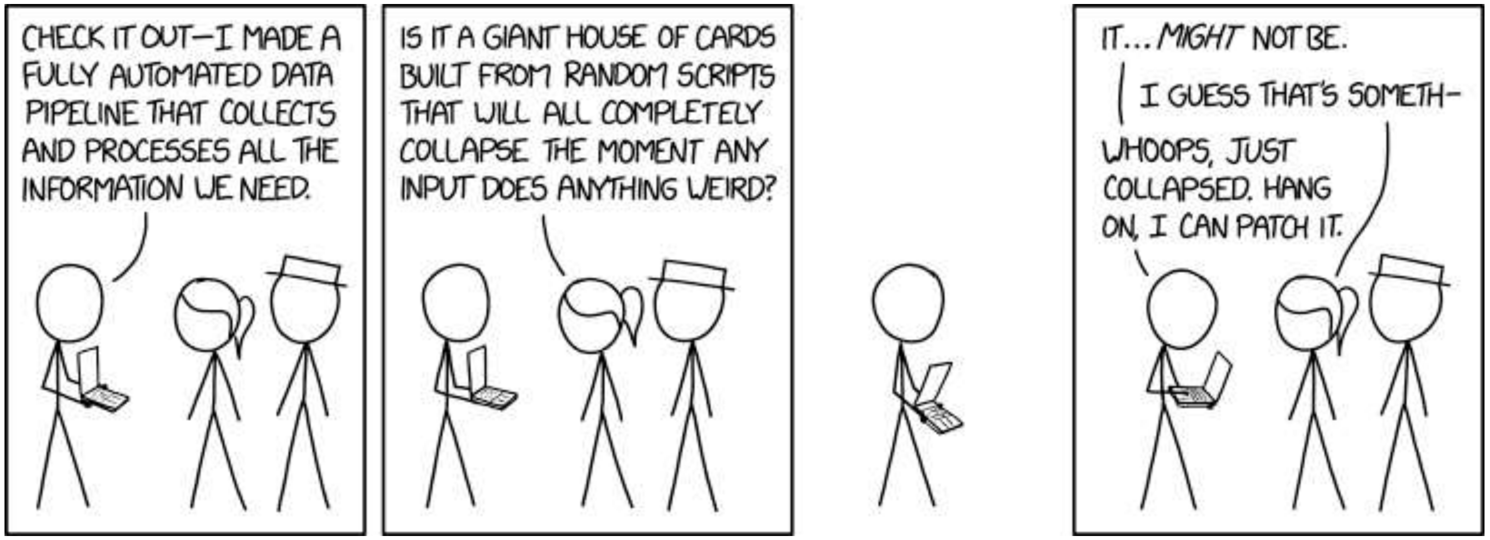
Importance of dependability

- System failures may have widespread effects with large numbers of people affected by the failure
- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users
- The costs of system failure may be very high if the failure leads to economic losses or physical damage
- Undependable systems may cause information loss with a high consequent recovery cost

Causes of failure

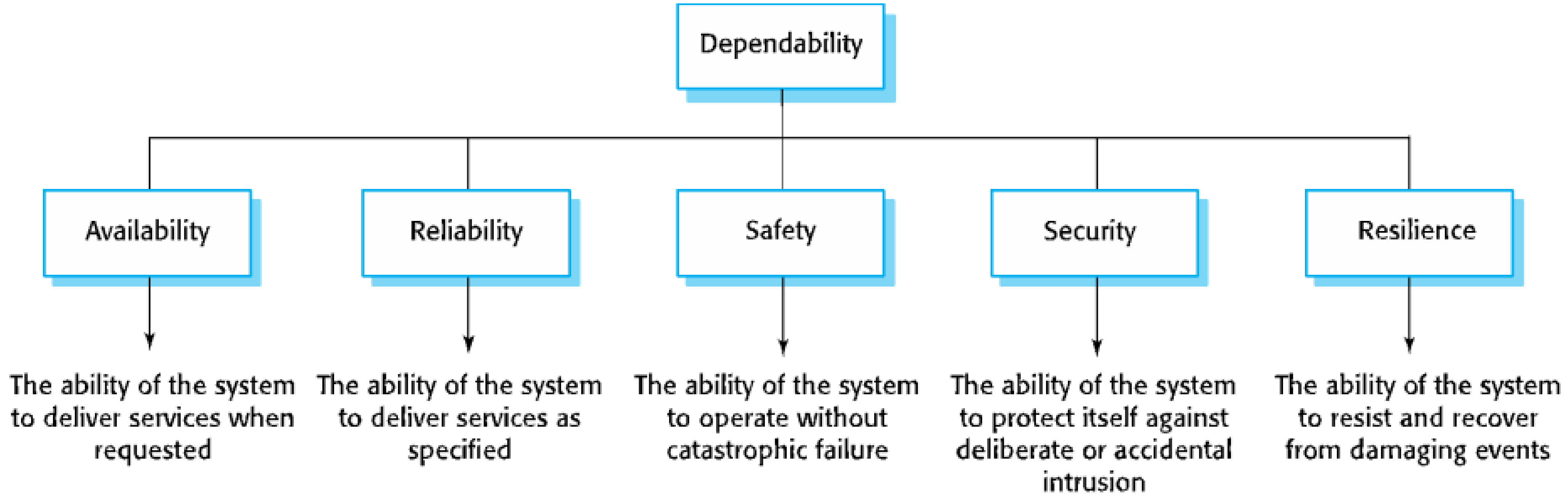
- Hardware failure
  - Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life
- Software failure
  - Software fails due to errors in its specification, design, or implementation
- Operational (human) failure
  - Human operators make mistakes. This is perhaps the largest single cause of system failures in socio-technical systems.

Software Engineering


CHECK IT OUT—I MADE A FULLY AUTOMATED DATA PIPELINE THAT COLLECTS AND PROCESSES ALL THE INFORMATION WE NEED.
IS IT A GIANT HOUSE OF CARDS BUILT FROM RANDOM SCRIPTS THAT WILL ALL COMPLETELY COLLAPSE THE MOMENT ANY INPUT DOES ANYTHING WEIRD?
IT... *MIGHT* NOT BE.
I GUESS THAT'S SOMETH—
WHOOPS, JUST COLLAPSED. HANG ON, I CAN PATCH IT.

Dependability properties

The principal dependability properties


Dependability
Availability
Reliability
Safety
Security
Resilience
The ability of the system to deliver services when requested
The ability of the system to deliver services as specified
The ability of the system to operate without catastrophic failure
The ability of the system to protect itself against deliberate or accidental intrusion
The ability of the system to resist and recover from damaging events

Principal properties

- Availability
  - The probability that the system will be up and running and able to deliver useful services to users
- Reliability
  - The probability that the system will correctly deliver services as expected by users
- Safety
  - A judgment of how likely it is that the system will cause damage to people or its environment

Principal properties

- Security
  - A judgment of how likely it is that the system can resist accidental or deliberate intrusions
- Resilience
  - A judgment of how well a system can maintain the continuity of its critical services in the presence of disruptive events such as equipment failure and cyberattacks

Other dependability properties

- Repairability
  - Reflects the extent to which the system can be repaired in the event of a failure
- Maintainability
  - Reflects the extent to which the system can be adapted to new requirements
- Error tolerance
  - Reflects the extent to which user input errors can be avoided and tolerated

Dependability attribute dependencies

- Safe system operation depends on the system being available and operating reliably
- A system may be unreliable because its data has been corrupted by an external attack
- Denial of service attacks on a system are intended to make it unavailable
- If a system is infected with a virus, you cannot be confident in its reliability or safety

Dependability achievement

- Avoid the introduction of accidental errors when developing the system
- Design V & V processes that are effective in discovering residual errors in the system
- Design systems to be fault tolerant so that they can continue in operation when faults occur
- Design protection mechanisms that guard against external attacks

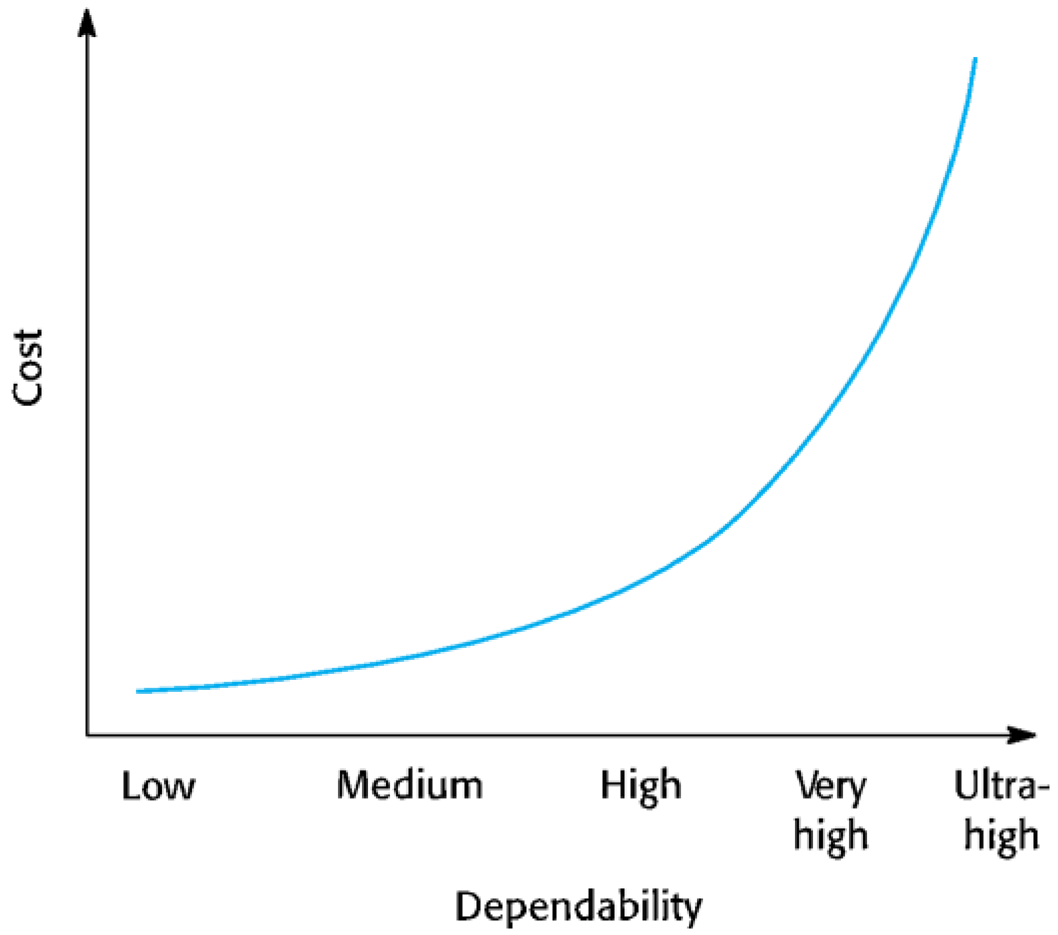
Dependability achievement

- Configure the system correctly for its operating environment
- Include system capabilities to recognize and resist cyberattacks
- Include recovery mechanisms to help restore normal system service after a failure

Dependability costs

- Dependability costs tend to increase exponentially as increasing levels of dependability are required
- There are two reasons for this:
  - The use of more expensive development techniques and hardware that are required to achieve the higher levels of dependability
  - The increased testing and system validation that is required to convince the system client and regulators that the required levels of dependability have been achieved

Cost/dependability curve



Dependability economics

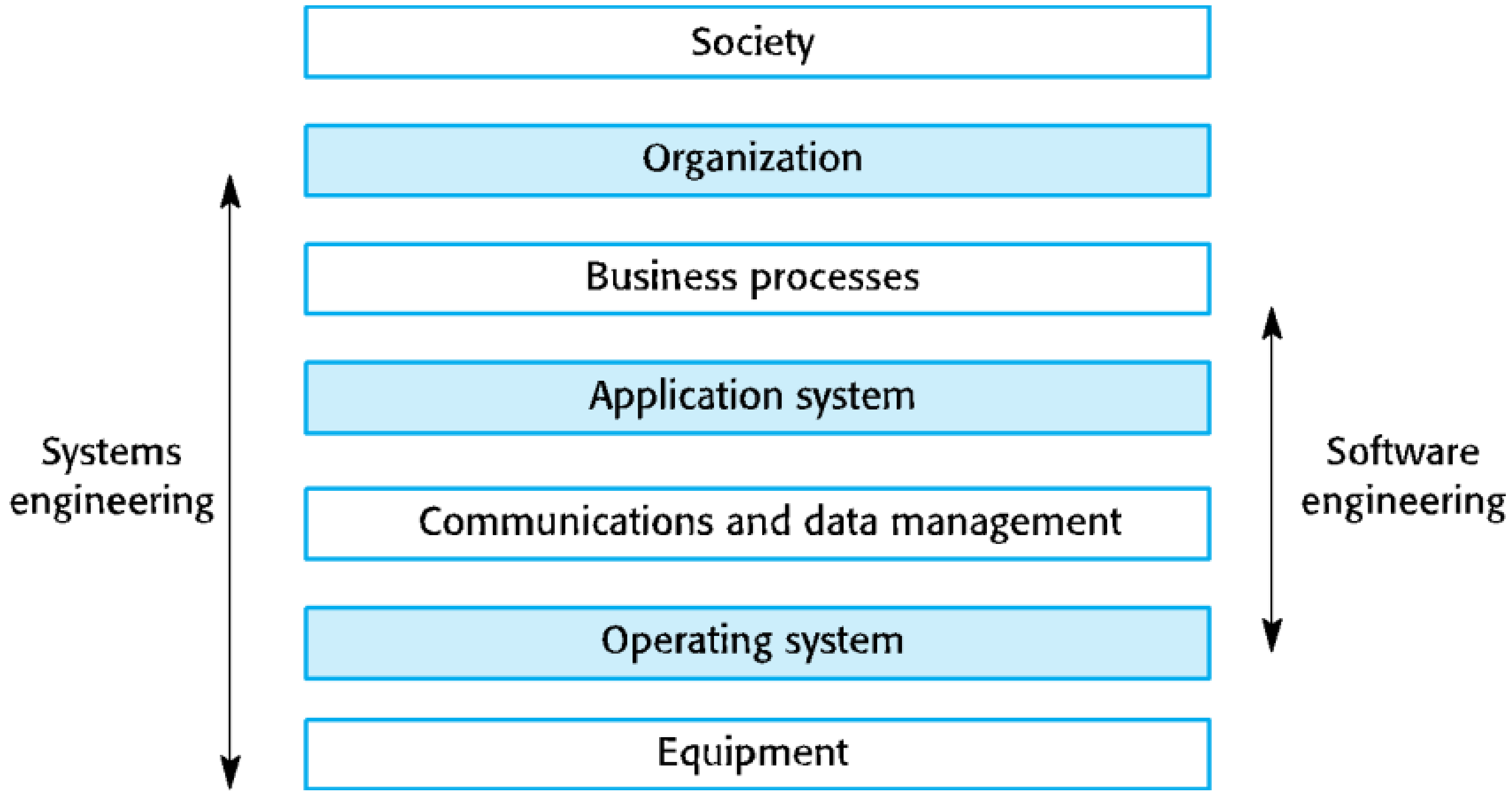
- Because of very high costs of dependability achievement, it may be more cost effective to accept untrustworthy systems and pay for failure costs
- However, this depends on social and political factors. A reputation for products that cannot be trusted may lose future business
- Depends on system type - for business systems in particular, modest levels of dependability may be adequate

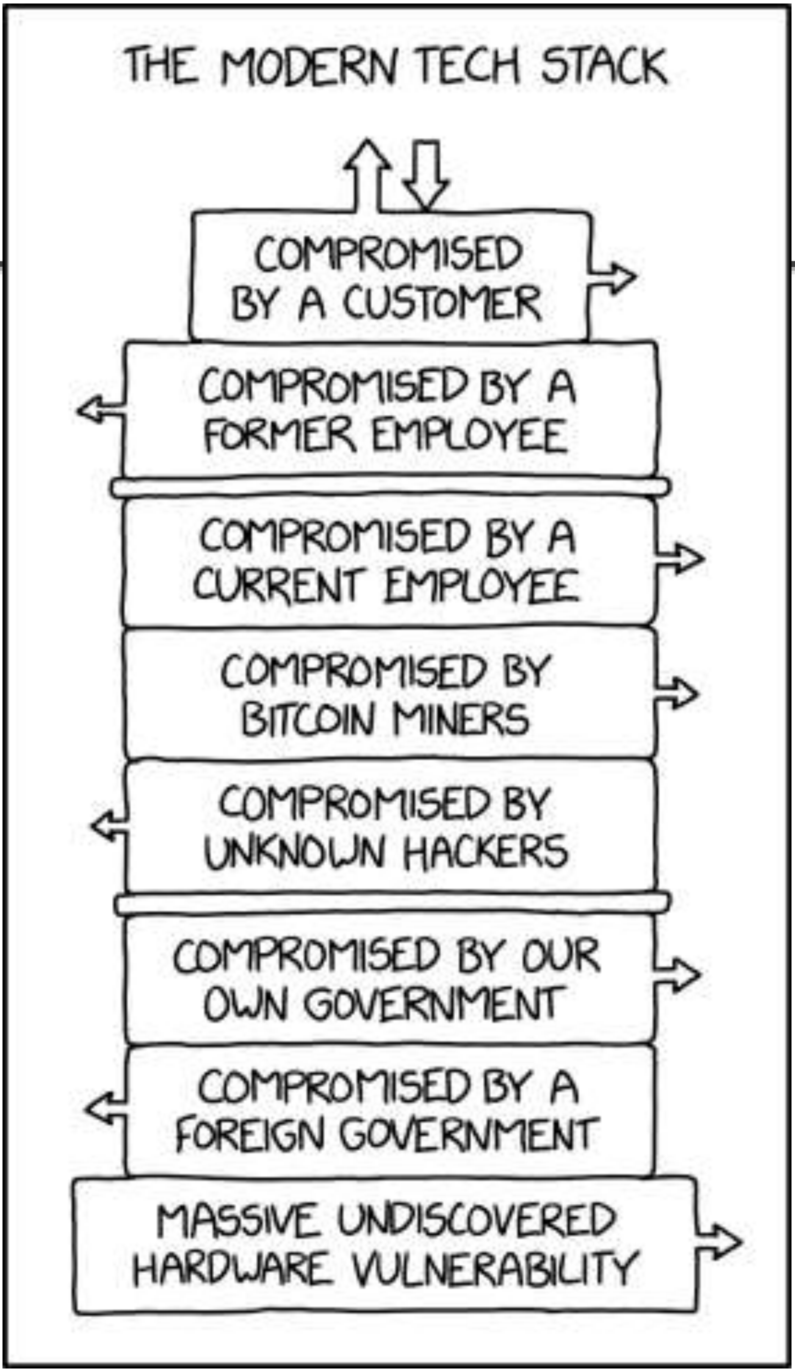
Sociotechnical systems (STS)

Systems and software

- Software engineering is not an isolated activity but is part of a broader systems engineering process
- Software systems are therefore not isolated systems but are essential components of broader systems that have a human, social or organizational purpose
- Example
    - The wilderness weather system is part of broader weather recording and forecasting systems
    - These include hardware and software, forecasting processes, system users, the organizations that depend on weather forecasts, etc.

The sociotechnical systems stack




THE MODERN TECH STACK
COMPROMISED BY A CUSTOMER
COMPROMISED BY A FORMER EMPLOYEE
COMPROMISED BY A CURRENT EMPLOYEE
COMPROMISED BY BITCOIN MINERS
COMPROMISED BY UNKNOWN HACKERS
COMPROMISED BY OUR OWN GOVERNMENT
COMPROMISED BY A FOREIGN GOVERNMENT
MASSIVE UNDISCOVERED HARDWARE VULNERABILITY

Layers in the STS stack

- Equipment
  - Hardware devices, some of which may be computers. Most devices will include an embedded system of some kind.
- Operating system
  - Provides a set of common facilities for higher levels in the system
- Communications and data management
  - Middleware that provides access to remote systems and databases
- Application systems
  - Specific functionality to meet some organization requirements

Layers in the STS stack

- Business processes
  - A set of processes involving people and computer systems that support the activities of the business
- Organizations
  - Higher level strategic business activities that affect the operation of the system
- Society
  - Laws, regulation and culture that affect the operation of the system

Holistic system design

- There are interactions and dependencies between the layers in a system and changes at one level ripple through the other levels
    - Example: Change in regulations (society) leads to changes in business processes and application software
- For dependability, a systems perspective is essential
    - Contain software failures within the enclosing layers of the STS stack
    - Understand how faults and failures in adjacent layers may affect the software in a system

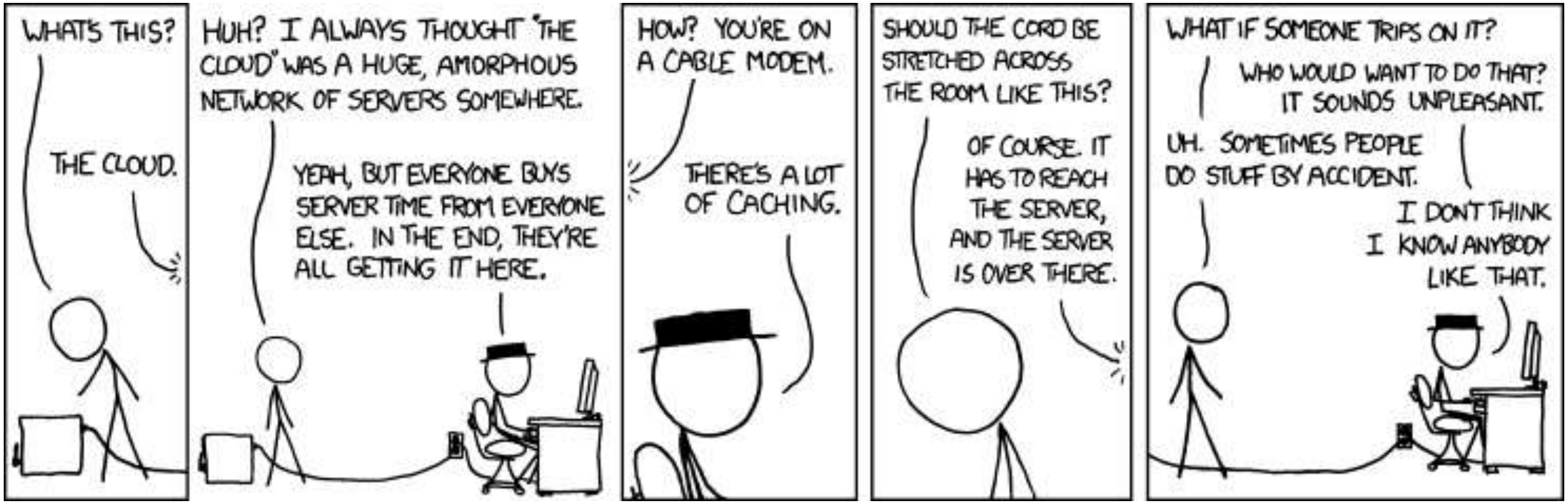
Regulation and compliance

- The general model of economic organization that is now almost universal in the world is that privately owned companies offer goods and services and make a profit on these
- To ensure the safety of their citizens, most governments regulate privately owned companies so that they must follow certain standards to ensure that their products are safe and secure

Regulated systems

- Many critical systems are *regulated systems*, which means that their use must be approved by an external regulator before the systems go into service
  - Nuclear systems
  - Air traffic control systems
  - Medical devices
- A safety and dependability case has to be approved by the regulator. Therefore, critical systems development has to create the evidence to convince a regulator that the system is dependable, safe and secure.

Software Engineering


WHAT'S THIS?
THE CLOUD.
HUH? I ALWAYS THOUGHT "THE CLOUD" WAS A HUGE, AMORPHOUS NETWORK OF SERVERS SOMEWHERE.
YEAH, BUT EVERYONE BUYS SERVER TIME FROM EVERYONE ELSE. IN THE END, THEY'RE ALL GETTING IT HERE.
HOW? YOU'RE ON A CABLE MODEM.
THERE'S A LOT OF CACHING.
SHOULD THE CORD BE STRETCHED ACROSS THE ROOM LIKE THIS?
OF COURSE. IT HAS TO REACH THE SERVER, AND THE SERVER IS OVER THERE.
WHAT IF SOMEONE TRIPS ON IT?
WHO WOULD WANT TO DO THAT? IT SOUNDS UNPLEASANT.
UH. SOMETIMES PEOPLE DO STUFF BY ACCIDENT.
I DON'T THINK I KNOW ANYBODY LIKE THAT.

Safety regulation

- Regulation and compliance (following the rules) applies to the sociotechnical system as a whole and not simply the software element of that system
- Safety-related systems may have to be certified as safe by the regulator
- To achieve certification, companies that are developing safety-critical systems have to produce an extensive safety case that shows that rules and regulations have been followed
- It can be as expensive to develop the documentation for certification as it is to develop the system itself

An example of STS: the cyber infrastructure (CI) of the NSF-funded Nevada Nexus project

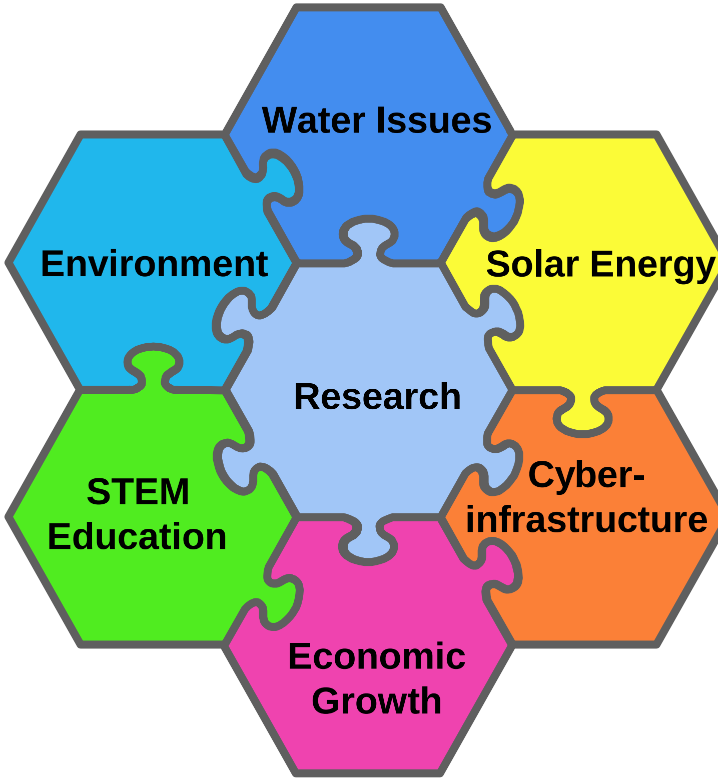
The Nexus project: *overview*

- Project funded by the National Science Foundation (NSF) as an EPSCoR RII Track 1 project:
  - 2013-2023
  - About $28.0 million in funding
- Project Director: Dr. Gayle Dana (DRI), until 2019, then Dr. Fred Harris (UNR)
- Participants: over 40 faculty members (of which 5 co-Principal Investigators – PIs), 30 students, and 20 technicians from several main research institutions in the state, including UNR, UNLV, and DRI

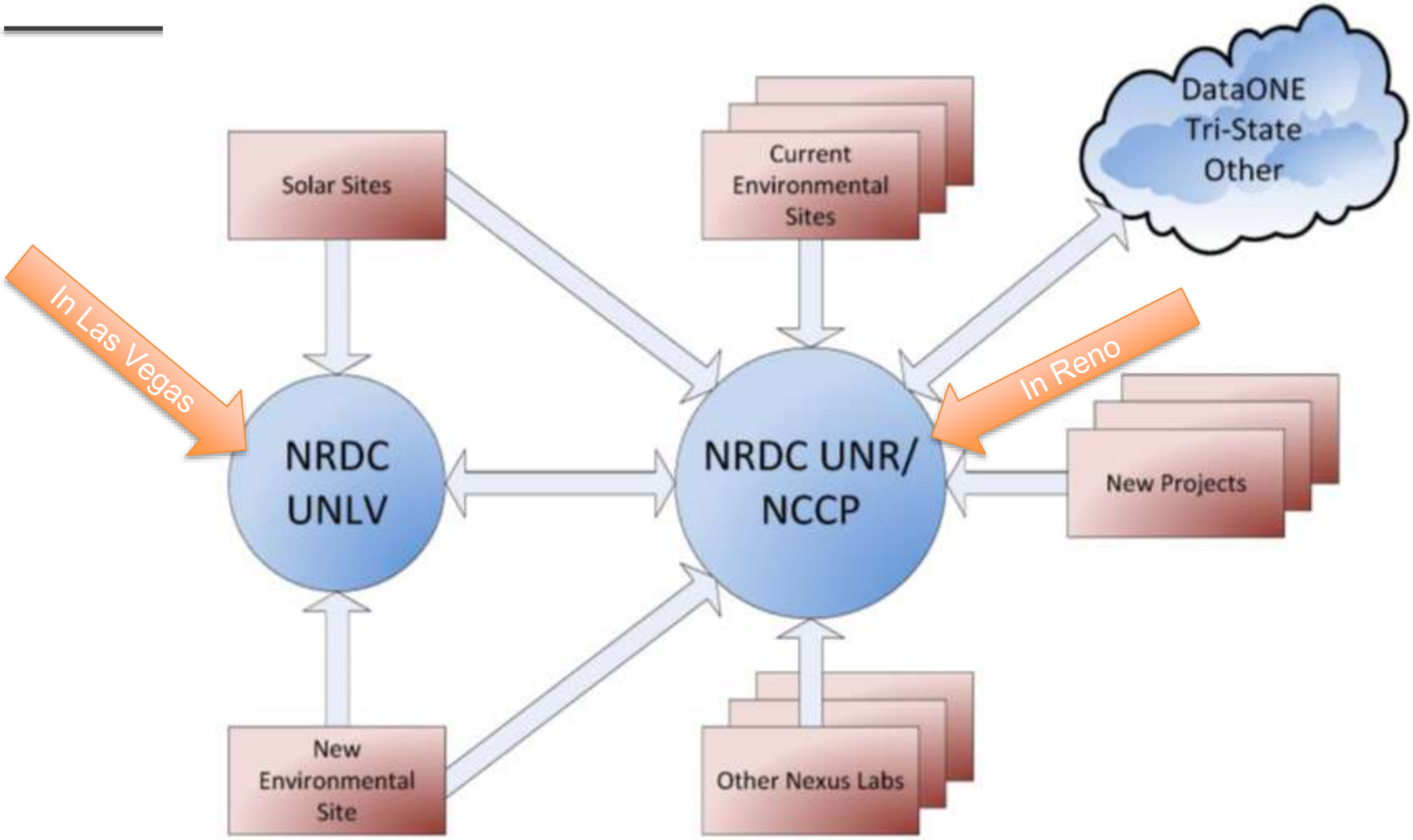
The Nexus project: *mission*

Our mission is to advance knowledge and discovery through research on solar energy generation technology, its environmental impacts and associated water issues, and accelerate this research by developing new capabilities in cyberinfrastructure.

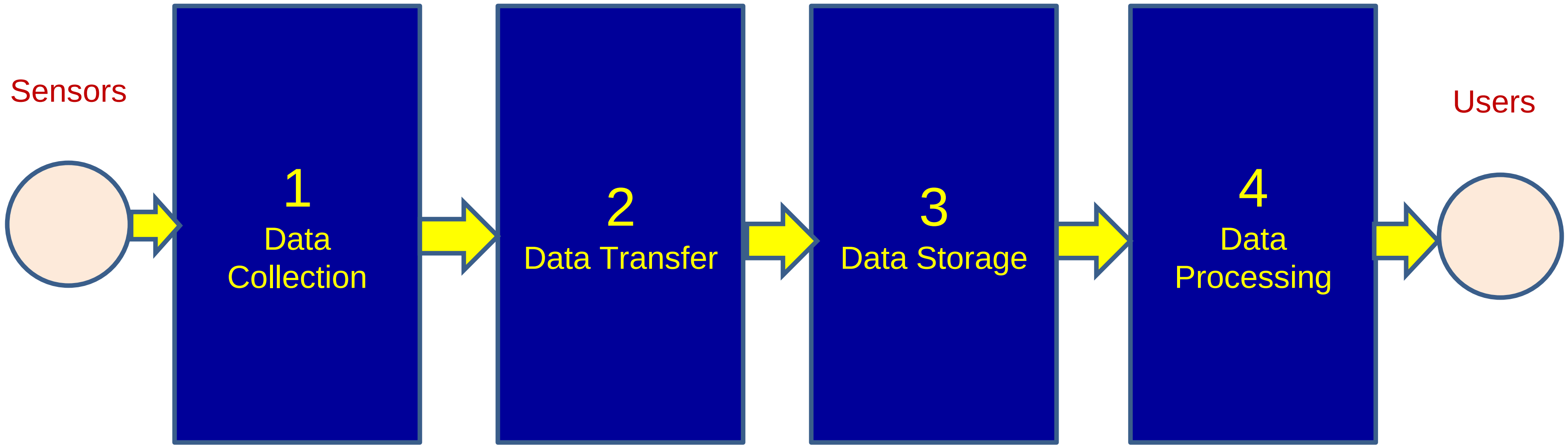
Benefits include diversifying Nevada's economy, building its workforce, and developing innovative approaches to STEM education.



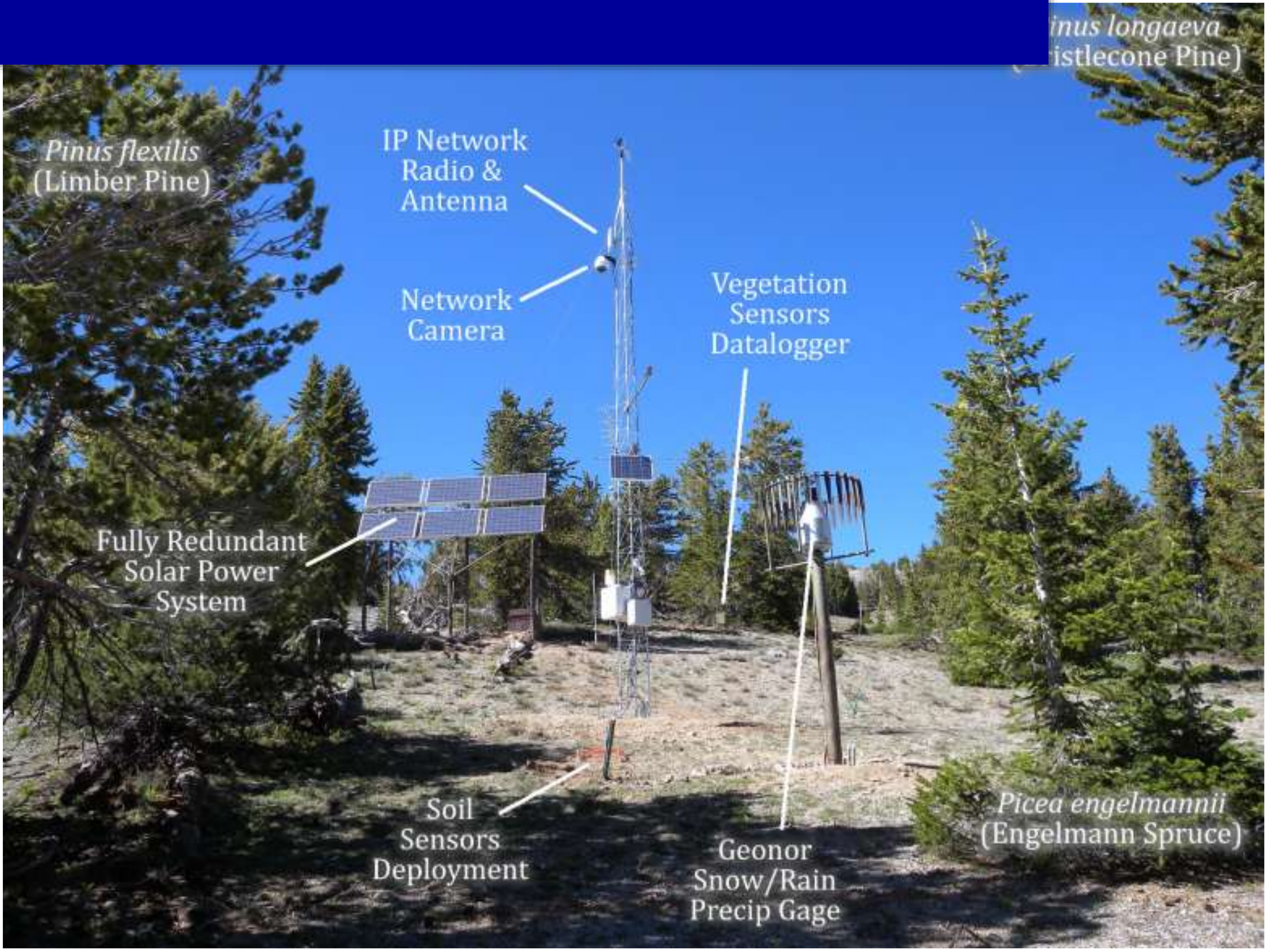
The Nexus project: *CI synopsis*



The end-to-end CI system



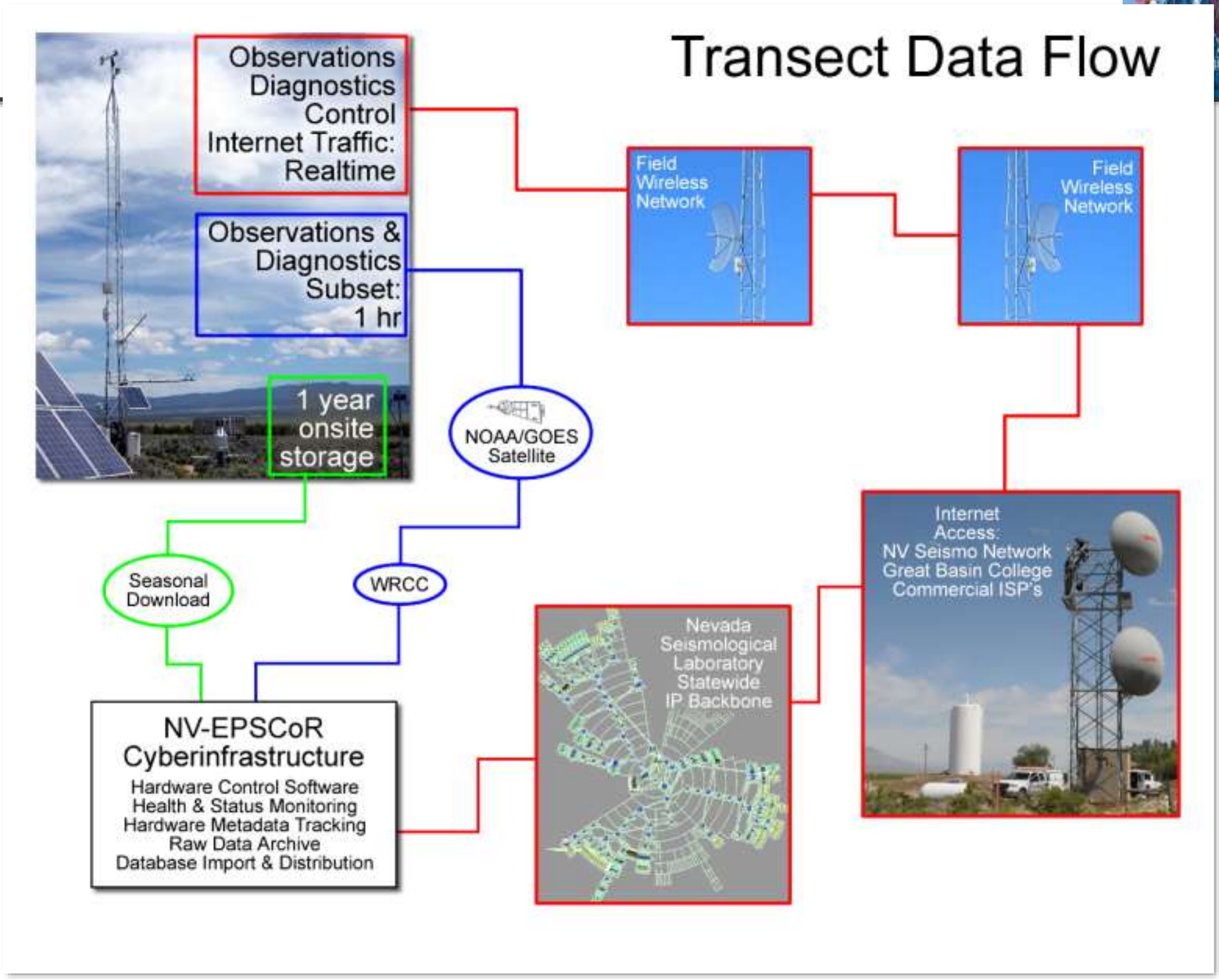
The CI system: 1 - data collection



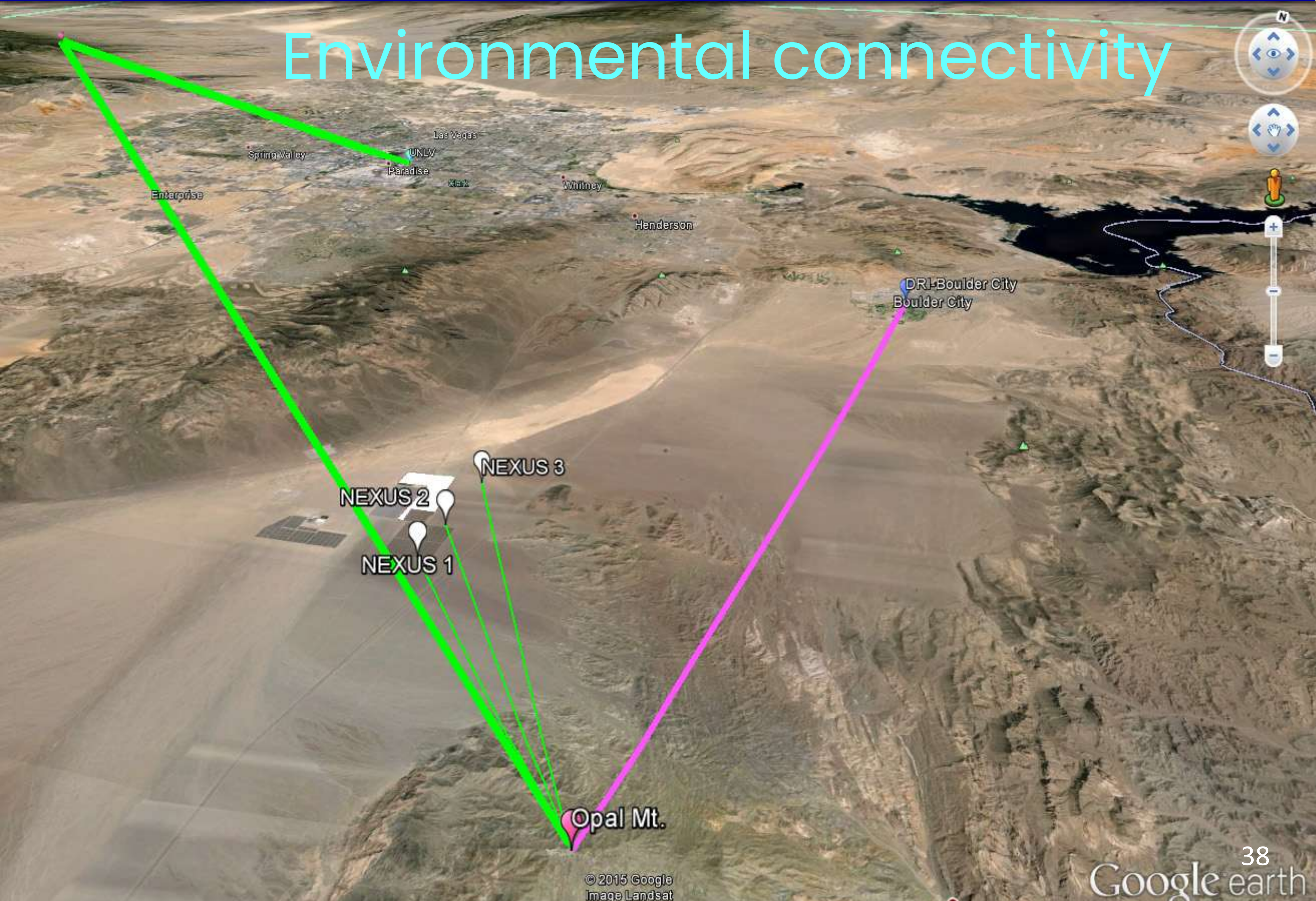
The CI system: 1- data collection

The CI system: 1- data collection

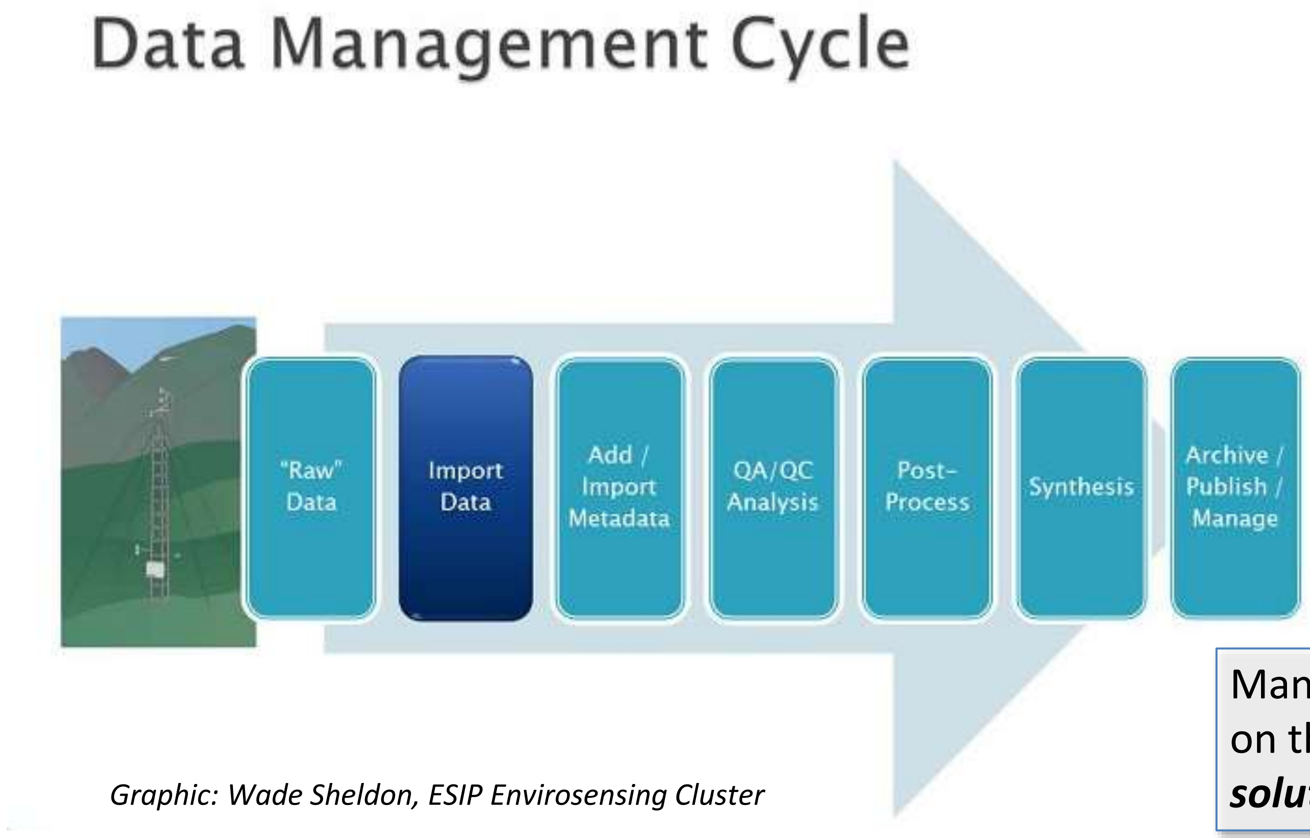
The CI system: 2 - data transfer



The CI system: 2 - data transfer


Environmental connectivity

The CI system: 3 - data management

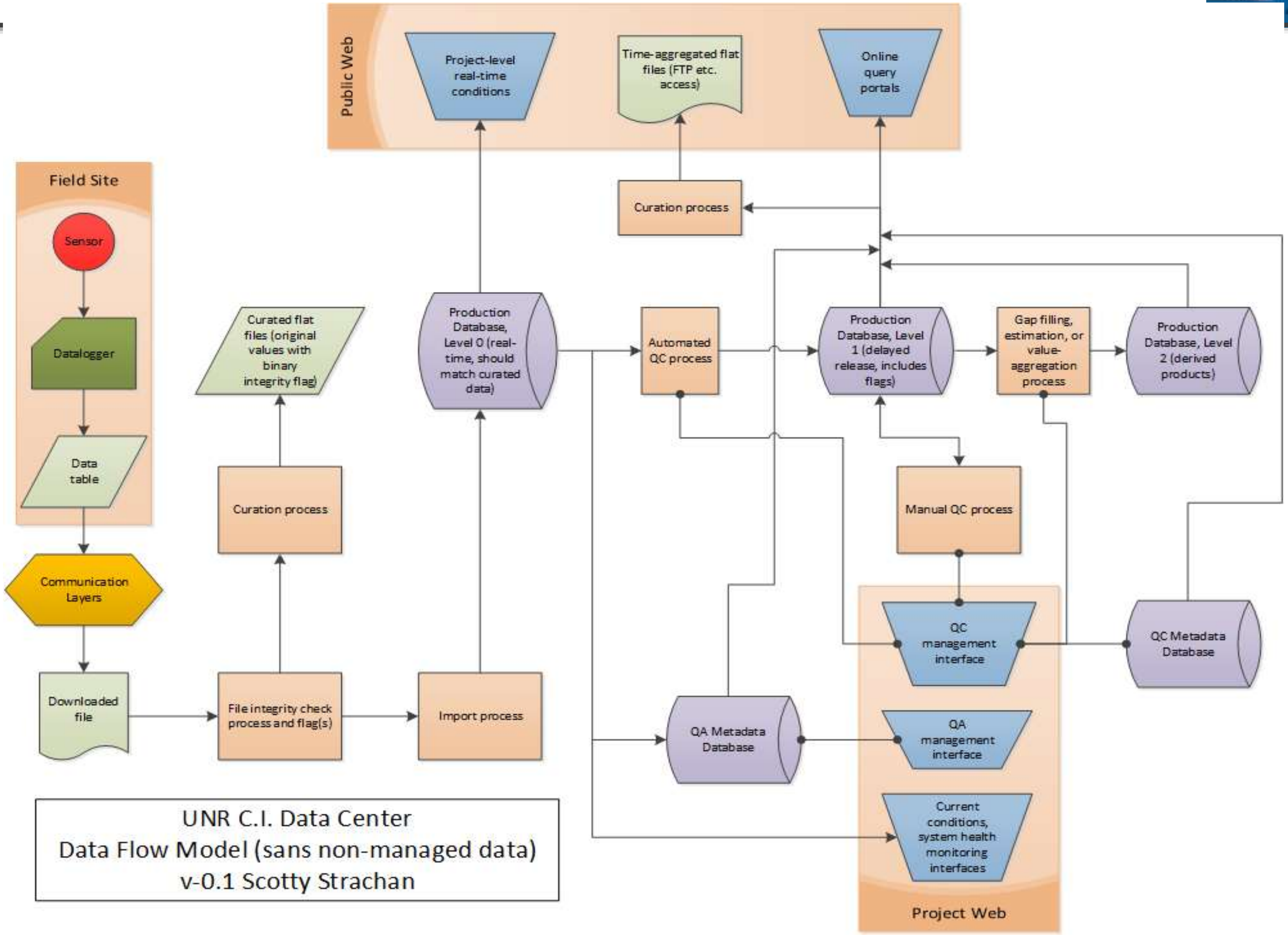


Graphic: Wade Sheldon, ESIP Envirosensing Cluster

This management ***problem is universal*** for observational science, particularly the earth sciences.

Many organizations are working on this problem, but ***no "one" solution exists*** yet.

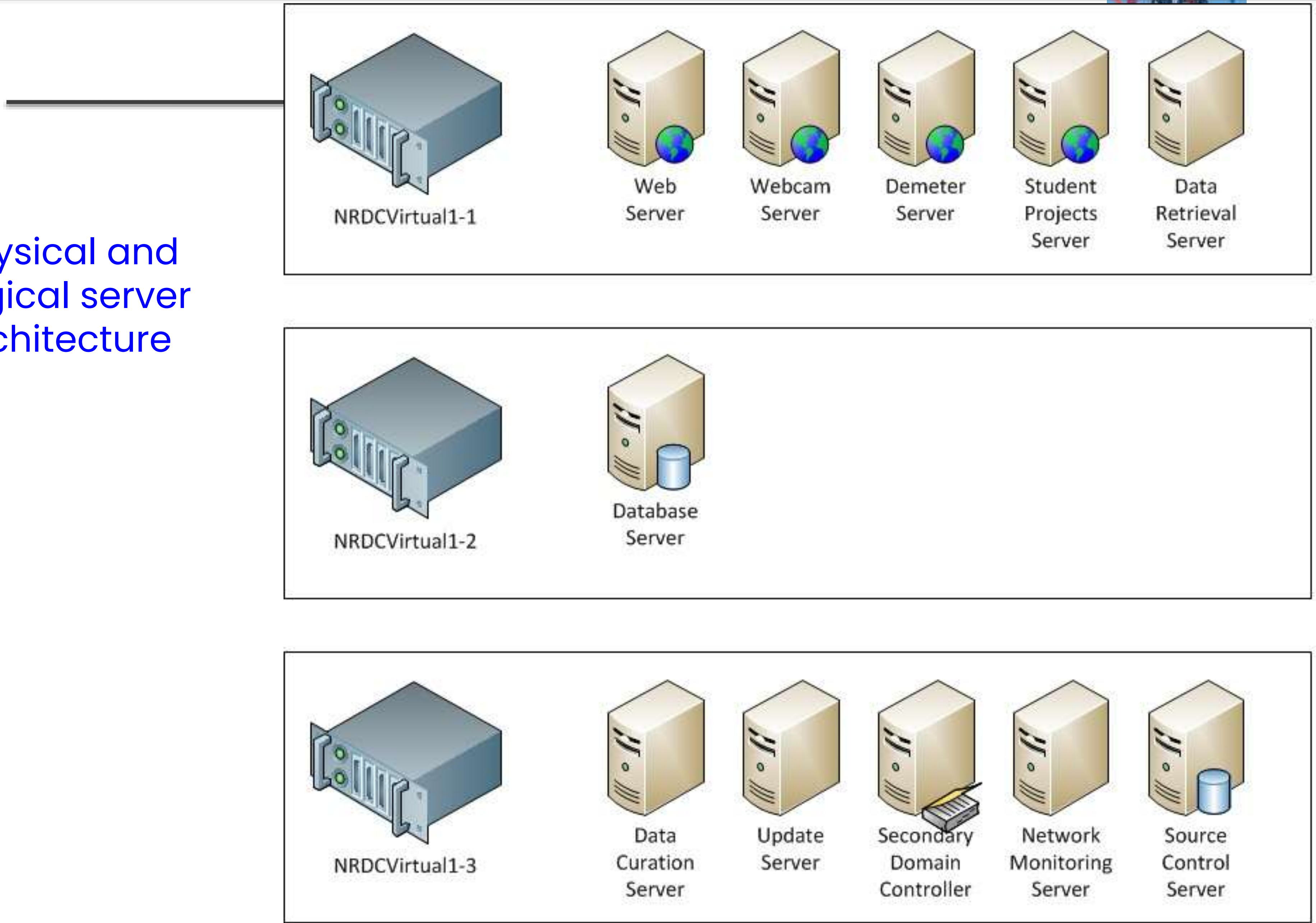
The CI system: 3 - data flow & management


Public Web
Project-level real-time conditions
Time-aggregated flat files (FTP etc. access)
Online query portals
Curation process
Field Site
Sensor
Datalogger
Data table
Communication Layers
Downloaded file
Curated flat files (original values with binary integrity flag)
Curation process
File integrity check process and flag(s)
Import process
Production Database, Level 0 (real-time, should match curated data)
Automated QC process
Production Database, Level 1 (delayed release, includes flags)
Gap filling, estimation, or value-aggregation process
Production Database, Level 2 (derived products)
Manual QC process
QC management interface
QC Metadata Database
QA Metadata Database
QA management interface
Current conditions, system health monitoring interfaces
Project Web

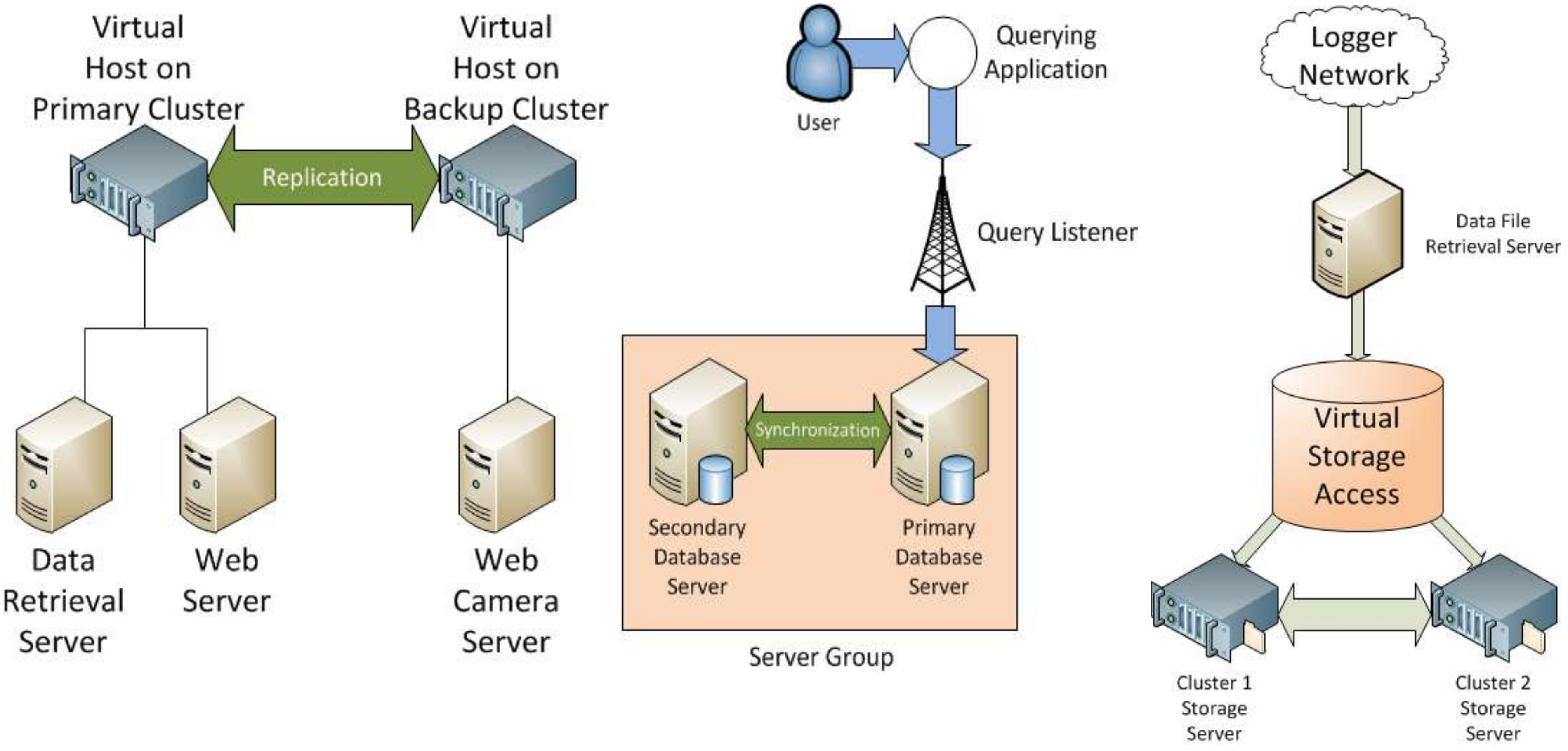
UNR C.I. Data Center
Data Flow Model (sans non-managed data)
v-0.1 Scotty Strachan

The CI system: 3 - data center architecture

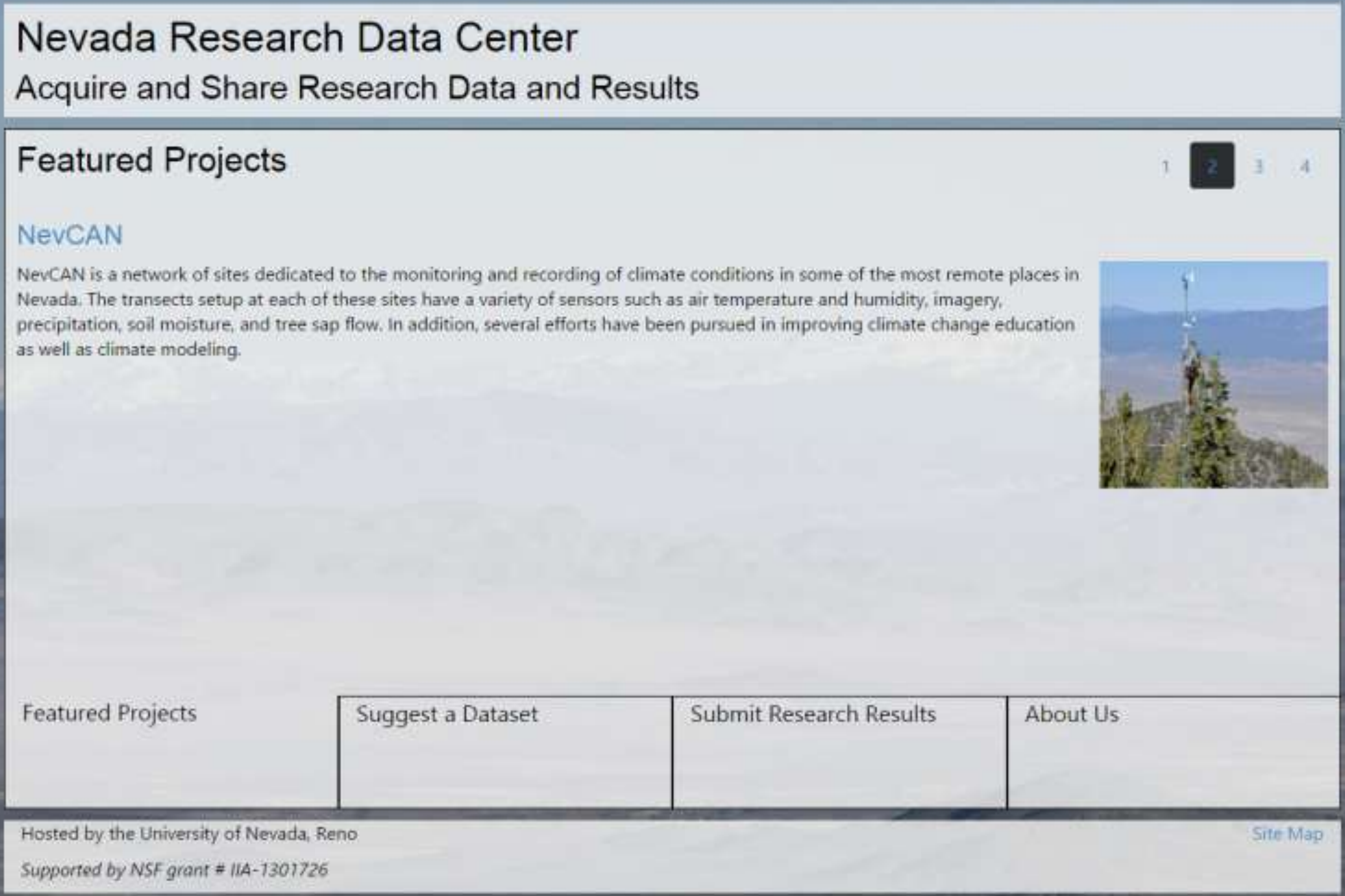
Physical and logical server architecture



The CI system: 3 - data storage



The CI system: 4 - data processing



Nevada Research Data Center

Acquire and Share Research Data and Results

Featured Projects

1 2 3 4

NevCAN

NevCAN is a network of sites dedicated to the monitoring and recording of climate conditions in some of the most remote places in Nevada. The transects setup at each of these sites have a variety of sensors such as air temperature and humidity, imagery, precipitation, soil moisture, and tree sap flow. In addition, several efforts have been pursued in improving climate change education as well as climate modeling.

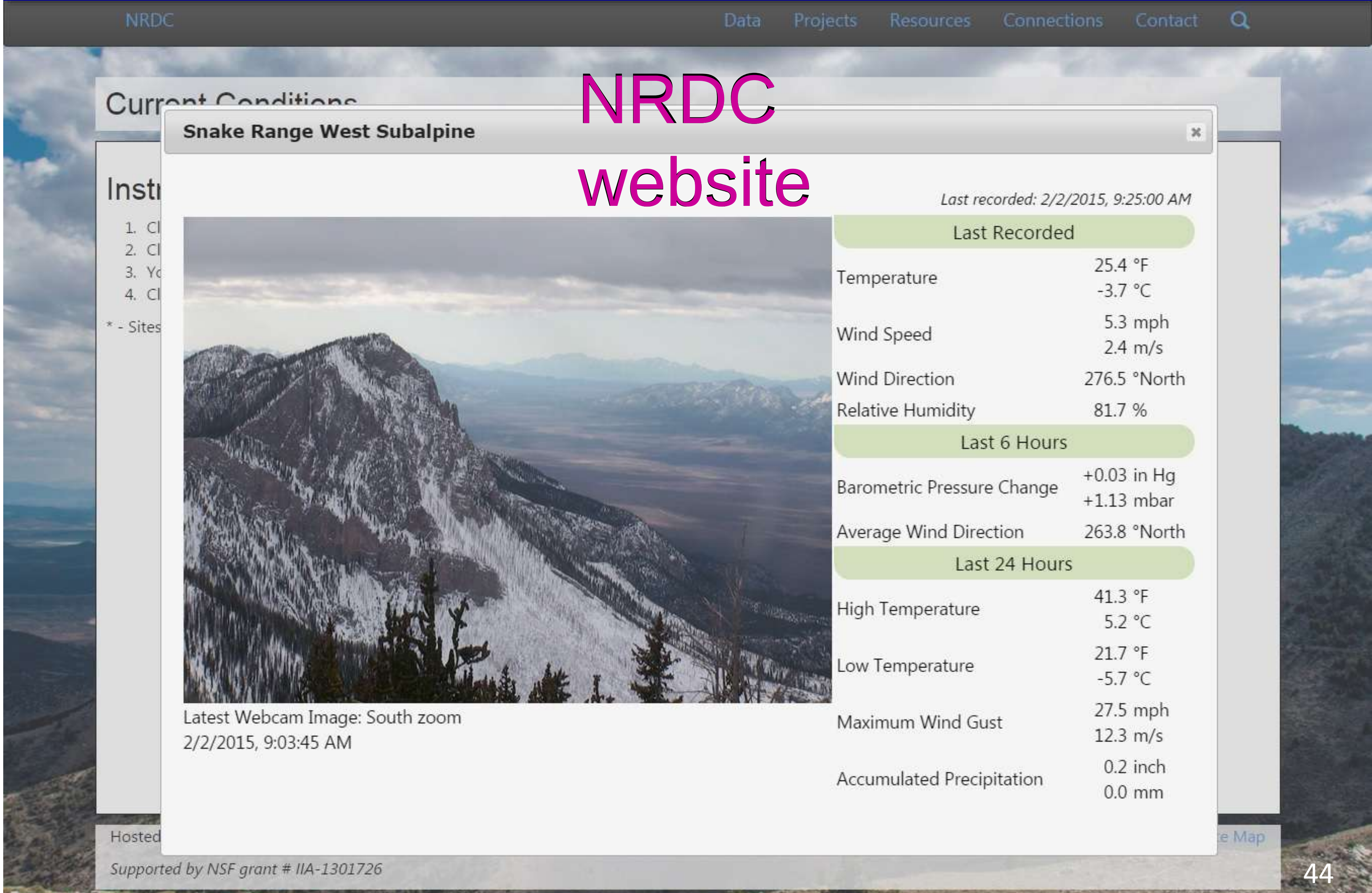
Featured Projects | Suggest a Dataset | Submit Research Results | About Us

Hosted by the University of Nevada, Reno

Site Map

Supported by NSF grant # IIA-1301726

The CI system: 4 - data processing



NRDC website

NRDC | Data | Projects | Resources | Connections | Contact

Current Conditions

Instr

1. Cl
2. Cl
3. Yo
4. Cl

* - Sites

Snake Range West Subalpine

Last recorded: 2/2/2015, 9:25:00 AM

Last Recorded	
Temperature	25.4 °F -3.7 °C
Wind Speed	5.3 mph 2.4 m/s
Wind Direction	276.5 °North
Relative Humidity	81.7 %
Last 6 Hours	
Barometric Pressure Change	+0.03 in Hg +1.13 mbar
Average Wind Direction	263.8 °North
Last 24 Hours	
High Temperature	41.3 °F 5.2 °C
Low Temperature	21.7 °F -5.7 °C
Maximum Wind Gust	27.5 mph 12.3 m/s
Accumulated Precipitation	0.2 inch 0.0 mm

Latest Webcam Image: South zoom
2/2/2015, 9:03:45 AM

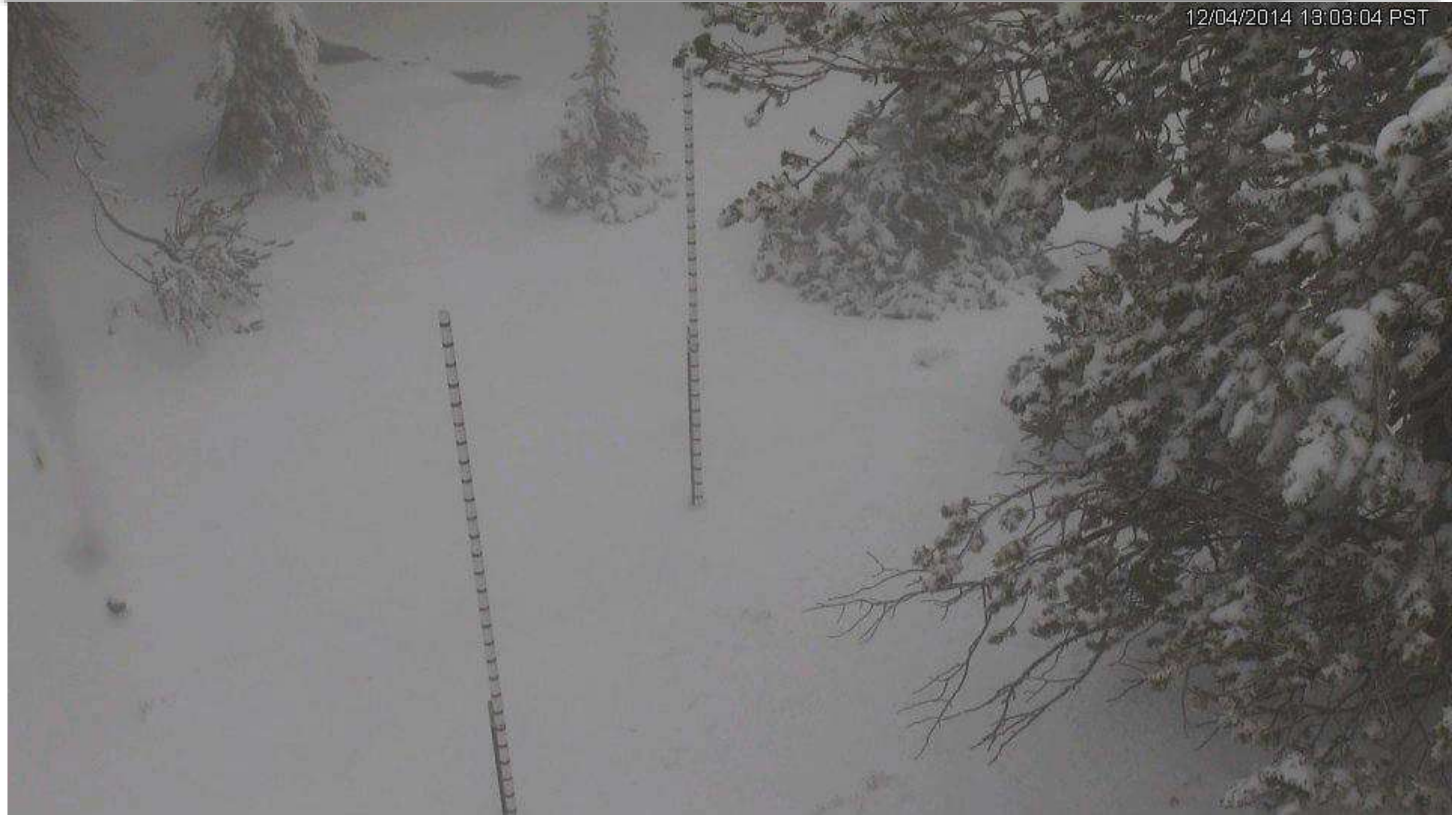
Hosted

e Map

Supported by NSF grant # IIA-1301726

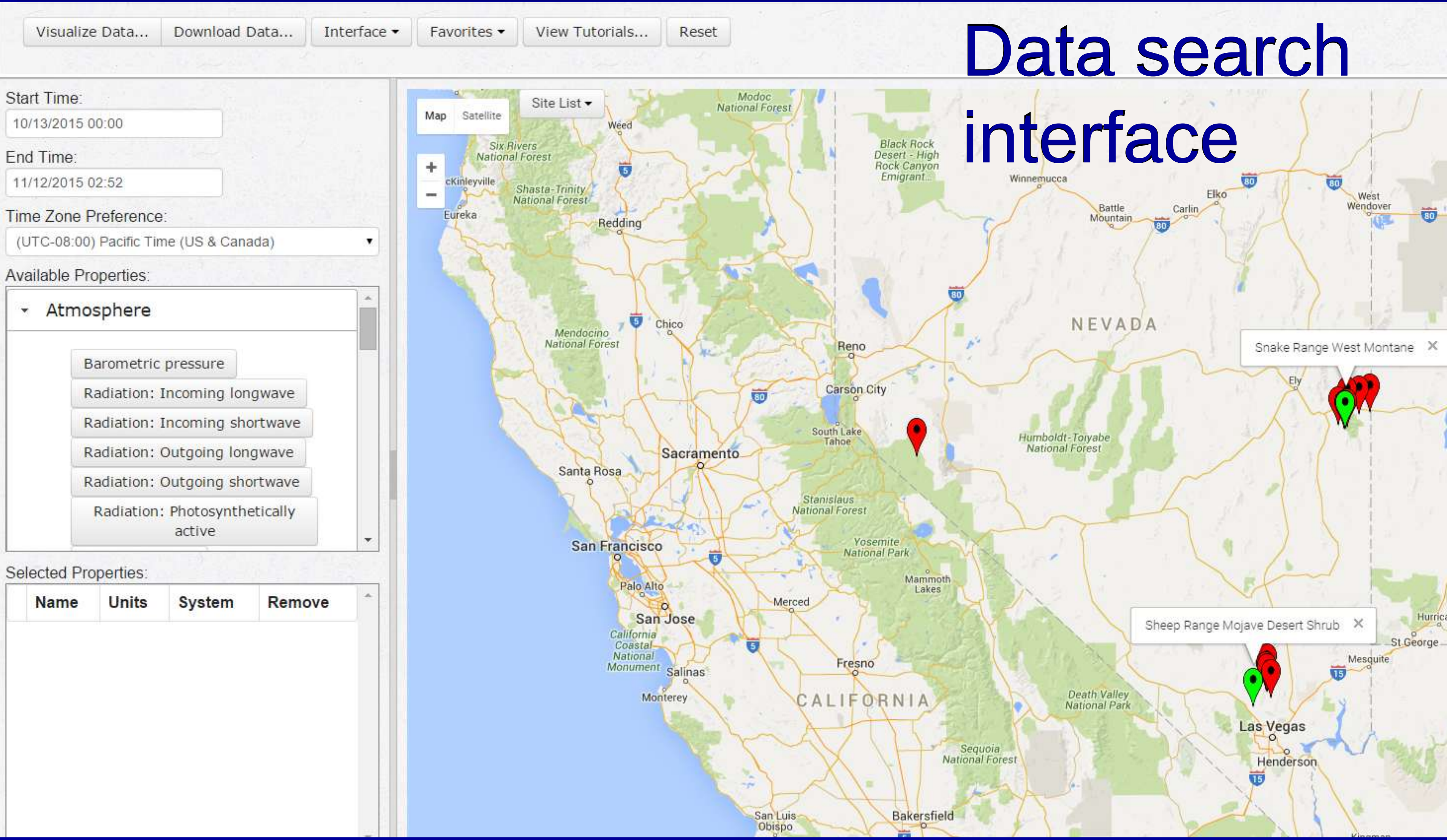
The CI system: 4 - data processing

Observation of ongoing phenomena



The CI system: 4 - data processing

Data search interface



Visualize Data... | Download Data... | Interface ▾ | Favorites ▾ | View Tutorials... | Reset

Start Time: 10/13/2015 00:00

End Time: 11/12/2015 02:52

Time Zone Preference: (UTC-08:00) Pacific Time (US & Canada)

Available Properties:

- Atmosphere
  - Barometric pressure
  - Radiation: Incoming longwave
  - Radiation: Incoming shortwave
  - Radiation: Outgoing longwave
  - Radiation: Outgoing shortwave
  - Radiation: Photosynthetically active

Selected Properties:

Name	Units	System	Remove

The CI system: 4 - data processing

[Scotty Strachan, Geography, UNR – *Sub-alpine mountain sublimation*]

Redundancy and diversity

Redundancy and diversity

- Redundancy
  - Keep more than a single version of critical components so that if one fails then a backup is available
- Diversity
  - Provide the same functionality in different ways in different components so that they will not fail in the same way
- Redundant and diverse components should be independent so that they will not suffer from 'common-mode' failures
  - For example, components implemented in different programming languages means that a compiler fault will not affect all of them

Diversity and redundancy examples

- Redundancy. Where availability is critical (e.g., in e-commerce systems), companies normally keep backup servers and switch to these automatically if failure occurs
- Diversity. To provide resilience against external attacks, different servers may be implemented using different operating systems (e.g., Windows and Linux)

Process diversity and redundancy

- Process activities, such as validation, should not depend on a single approach, such as testing, to validate the system
- Redundant and diverse process activities are important especially for verification and validation
- Multiple, different process activities that complement each other and allow for cross-checking help to avoid process errors, which may lead to errors in the software

Problems with redundancy and diversity

- Adding diversity and redundancy to a system increases the system complexity
- This can increase the chances of error because of unanticipated interactions and dependencies between the redundant system components
- Some engineers therefore advocate simplicity and extensive V & V as a more effective route to software dependability
- Airbus FCS architecture is redundant/diverse; Boeing 777 FCS architecture has no software diversity

Dependable processes

Dependable processes

- To ensure a minimal number of software faults, it is important to have a well-defined, repeatable software process
- A well-defined repeatable process is one that does not depend entirely on individual skills; rather, it can be enacted by different people
- Regulators use information about the process to check if good software engineering practice has been used
- For fault detection, it is clear that the process activities should include significant effort devoted to verification and validation

Dependable process characteristics

- Explicitly defined
  - A process that has a defined process model that is used to drive the software production process. Data must be collected during the process that proves that the development team has followed the process as defined in the process model.
- Repeatable
  - A process that does not rely on individual interpretation and judgment. The process can be repeated across projects and with different team members, irrespective of who is involved in the development.

Attributes of dependable processes

Process characteristic	Description
Auditable	The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement.
Diverse	The process should include redundant and diverse verification and validation activities.
Documentable	The process should have a defined process model that sets out the activities in the process and the documentation that is to be produced during these activities.
Robust	The process should be able to recover from failures of individual process activities.
Standardized	A comprehensive set of software development standards covering software production and documentation should be available.

Dependable process activities

- Requirements reviews to check that the requirements are, as far as possible, complete and consistent

- Requirements management to ensure that changes to the requirements are controlled and that the impact of proposed requirements changes is understood.

- Formal specification, where a mathematical model of the software is created and analyzed

- System modeling, where the software design is explicitly documented as a set of graphical models, and the links between the requirements and these models are documented

Dependable process activities

- Design and program inspections, where the different descriptions of the system are inspected and checked by different people.
- Static analysis, where automated checks are carried out on the source code of the program.
- Test planning and management, where a comprehensive set of system tests is designed.
  - The testing process has to be carefully managed to demonstrate that these tests provide coverage of the system requirements and have been correctly applied in the testing process.

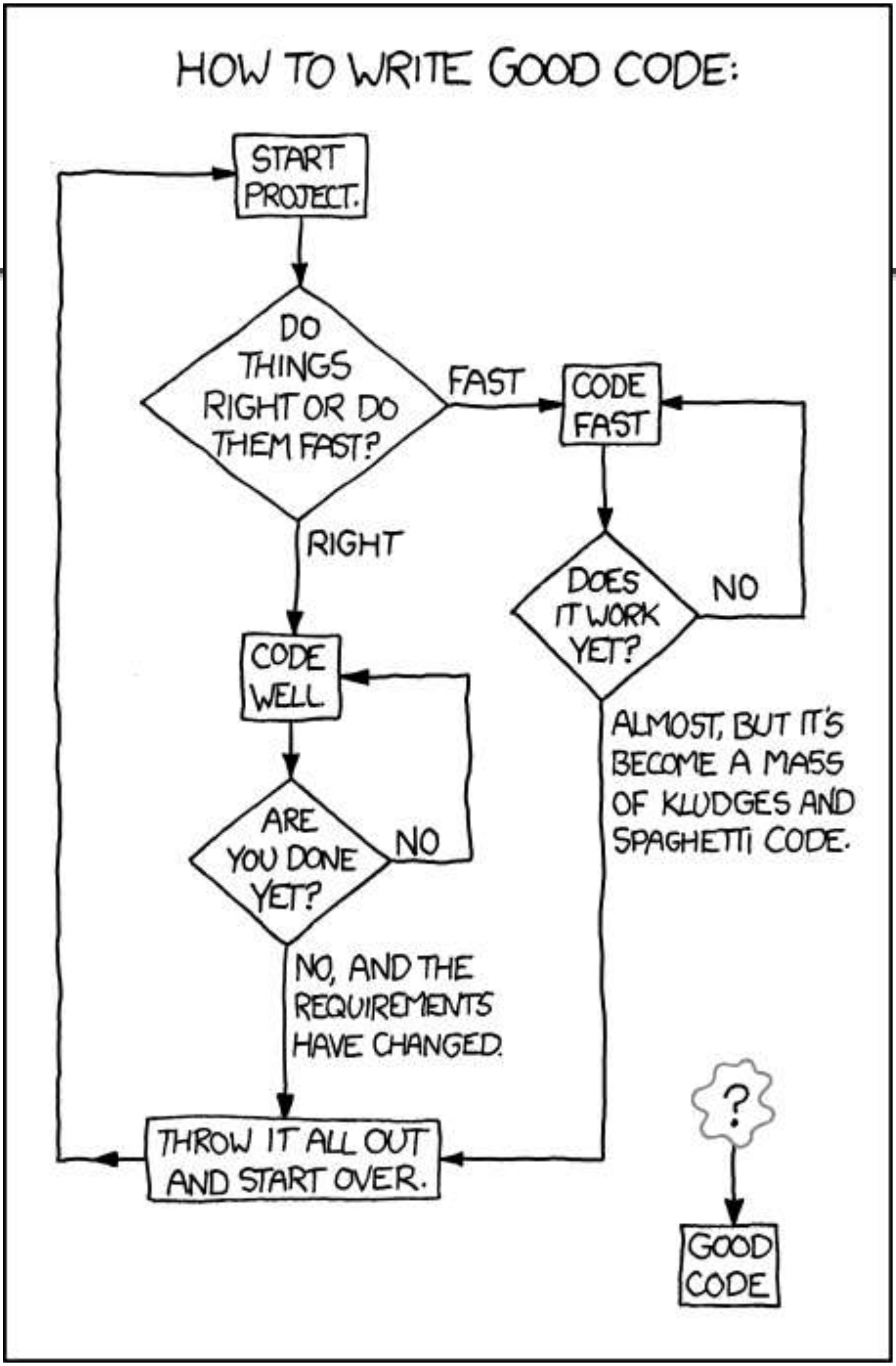
Dependable processes and agility

- Dependable software often requires certification so both process and product documentation has to be produced
- Up-front requirements analysis is also essential to discover requirements and requirements conflicts that may compromise the safety and security of the system
- These conflict with the general approach in agile development of co-development of the requirements and the system and minimizing documentation.

Dependable processes and agility

- An agile process may be defined that incorporates techniques such as iterative development, test-first development and user involvement in the development team
- So long as the team follows that process and documents their actions, agile methods can be used
- However, additional documentation and planning is essential so 'pure agile' is impractical for dependable systems engineering

Software Engineering


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

Formal methods and dependability

Formal specification

- Formal methods are approaches to software development that are based on mathematical representation and analysis of software
- Formal methods include:
  - Formal specification
  - Specification analysis and proof
  - Transformational development
  - Program verification
- Formal methods significantly reduce some types of programming errors and can be cost-effective for dependable systems engineering

Formal approaches

- Verification-based approaches
  - Different representations of a software system such as a specification and a program implementing that specification are proved to be equivalent
  - This demonstrates the absence of implementation errors
- Refinement-based approaches
  - A representation of a system is systematically transformed into another, lower-level representation, e.g., a specification is transformed automatically into an implementation
  - This means that, if the transformation is correct, the representations are equivalent

Use of formal methods

- The principal benefits of formal methods are in reducing the number of faults in systems
- Consequently, their main area of applicability is in dependable systems engineering. There have been several successful projects where formal methods have been used in this area
- In this area, the use of formal methods is most likely to be cost-effective because high system failure costs must be avoided

Classes of error

- Specification and design errors and omissions
    - Developing and analyzing a formal model of the software may reveal errors and omissions in the software requirements. If the model is generated automatically or systematically from source code, analysis using model checking can find undesirable states that may occur such as a deadlock in a concurrent system.
- Inconsistencies between a specification and a program
    - If a refinement method is used, mistakes made by developers that make the software inconsistent with the specification are avoided. Program proving discovers inconsistencies between a program and its specification.

Benefits of formal specification

- Developing a formal specification requires the system requirements to be analyzed in detail. This helps to detect problems, inconsistencies and incompleteness in the requirements.
- As the specification is expressed in a formal language, it can be automatically analyzed to discover inconsistencies and incompleteness
- If you use a formal method such as the B method, you can transform the formal specification into a 'correct' program
- Program testing costs may be reduced if the program is formally verified against its specification

Acceptance of formal methods

- Formal methods have had limited impact on practical software development:
  - Problem owners cannot understand a formal specification and so cannot assess if it is an accurate representation of their requirements
  - It is easy to assess the costs of developing a formal specification but harder to assess the benefits. Managers may therefore be unwilling to invest in formal methods.
  - Software engineers are unfamiliar with this approach and are therefore reluctant to propose the use of FM
  - Formal methods are still hard to scale up to large systems
  - Formal specifications are not really compatible with agile development methods

Key points

- System dependability is important because failure of critical systems can lead to economic losses, information loss, physical damage or threats to human life.
- The dependability of a computer system is a system property that reflects the user's degree of trust in the system. The most important dimensions of dependability are availability, reliability, safety, security, and resilience.
- Sociotechnical systems include computer hardware, software and people, and are situated within an organization. They are designed to support organizational or business goals and objectives.

Key points

- The use of a dependable, repeatable process is essential if faults in a system are to be minimized. The process should include verification and validation activities at all stages, from requirements definition through to system implementation.
- The use of redundancy and diversity in hardware, software processes and software systems is essential to the development of dependable systems.
- Formal methods, where a formal model of a system is used as a basis for development help reduce the number of specification and implementation errors in a system.