Module 1.
Introduction to Software Reuse

Dr. Shiping Chen
shipingchen2005@gmail.com

Module 1. Introduction to SR

- What is software reuse?
- Why software reuse?
- A software reuse example
- A systematic taxonomy of software reuse

Definition of Software Reuse

- Software Reuse (SR) is *a systematic practice* of developing software from a stock of *building blocks*, so that *similarities* in requirements and architecture between applications can be exploited to achieve *substantial benefits* in productivity, quality and business performance.

– From “Practical Software Reuse’ by Michel Ezran *et al*

Reuse is a Systematic Software Development Practice

- Understanding how software reuse can contribute to the business, i.e. reducing time to market
- Defining a technical and management strategy to achieve maximum value from reuse
- Integrating software reuse into the whole software development life cycle
- Ensuring all software developers have the necessary competence and motivation (team/company culture)
- Establishing appropriate organizational, technical and budgetary support
- Using appropriate measurement to control reuse performance, e.g., how many code reused? How much time saved? How much money saved?

Reuse Employs a Stock of Building Blocks

- Build blocks are **artifacts**, such as *classes, components, services*, which can be put together to make larger-scale artifacts
- It can be **not only** source code and binary library, **but also** design patterns, testing framework, documentation templates etc.
- The more general their properties, the more chances for reuse – called *Reusability*
- They should be stored in somewhere, which provides easy-use interface to easily discover – *Reuse Repository*

Reuse Exploits Similarities

- Reuse may be within an application domain, as well as across multiple application domains
- Requirements and architectures are two ultimate sources of similarities among applications
  - Requirements vs. Architecture
- Opportunities for reuse are like defects
  - The sooner they are found, the better
- But, how?
  - Systematic vs. Individual Heroics

Reuse offers Substantial Benefits

Aspects	Reasons	Benefits
Productivity	Writing less and faster	• Lower development cost • Shorter time to market
Quality	Reused software (usually) from successful projects with proven quality	• Less defects • Lower maintenance cost • High customer satisfaction • *But, with some risk!*
Business Performance	Improving predictability	• Enhanced Competitiveness • Profitability • More opportunities

Note: The above benefits can be direct and/or indirect.

Software Engineer Base Salary, US

Glassdoor Report: Software Engineer Base Salary Comparison by U.S. Metro	
U.S. Metro	**2012 Average Base Salary**
National Average	$92,648
Atlanta, GA	$76,821
Boston, MA	$93,403
Chicago, IL	$78,587
Dallas/Ft. Worth, TX	$80,947
Denver, CO	$83,164
Los Angeles, CA	$91,915
Minneapolis, MN	$75,032
New York, NY	$92,701
Philadelphia, PA	$80,270
Phoenix, AZ	$80,975
Portland, OR	$84,793
San Diego, CA	$93,529
San Francisco Bay Area, CA	$107,798
Seattle, WA	$102,006
Washington, DC	$85,404
Average base salaries based on at least 50 software engineer salary reports per U.S. metro for 2012 (10/8/11-10/7/12).	

Glassdoor Report: Software Engineer Base Salary Comparison	
Company	**2012 Average Base Salary**
National Average	$92,648
Amazon	$103,070
Apple	$114,413
Cisco	$101,909
eBay	$108,809
Facebook	$123,626
Google	$128,336
Hewlett-Packard	$95,567
IBM	$89,390
Intel	$92,194
Intuit	$103,284
Microsoft	$104,362
Oracle	$102,204
QUALCOMM	$98,964
Yahoo	$100,122
Zynga	$105,568
Average base salaries based on at least approximately 20 software engineer salary reports per company for 2012 (10/8/11-10/7/12).	

From: http://www.glassdoor.com/blog/15-tech-companies-software-engineer-salary-revealed-glassdoor-report

Downside of Software Reuse (1/2)

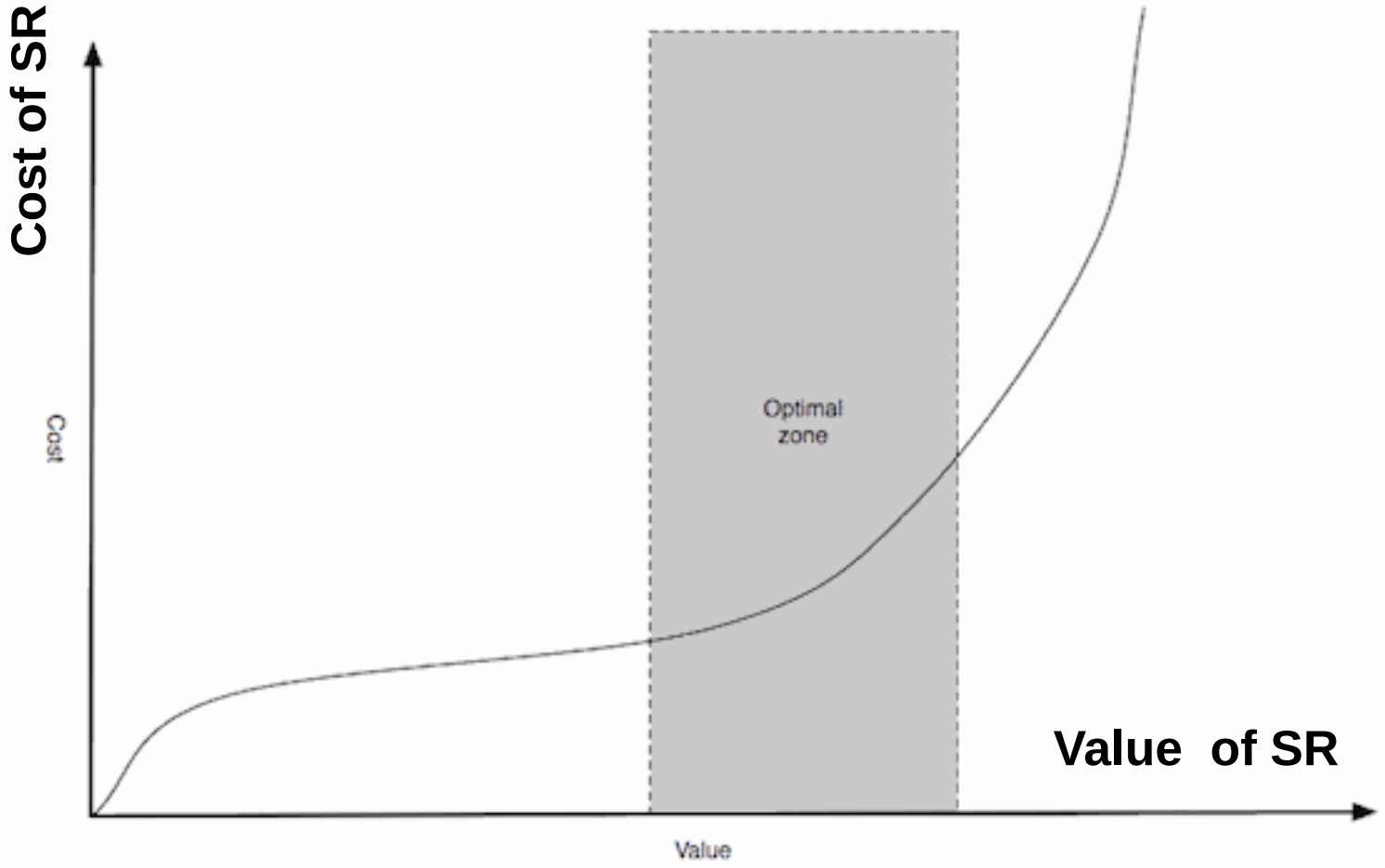
- For developing reusable software:
  - Developing software for reuse **is expensive**
  - Maintaining the reusable software assets requires **extra investment and efforts**

- Good practice and advices:
  - Don't go **too far**, esp.
    - Don't try to reuse for everything;
    - Don't try to make everything reusable;
  - Reusability comes **naturally** from refining after getting things working…

Downside of Software Reuse (2/2)

- For using reusable software:
  - Integrating reusable code may introduce an overhead in the software development process;
  - Validation/verification of reusable assets could be difficult and risky, especially:
    - when you don't have source code;
    - reuse in mission-critical systems.
  - Modifications may be not flexible, expensive, and even impossible!
  - IP issues may remain!

There must be a Trad-off



Obvious Reuse Examples

1. Copy & Paste – *Perhaps, the most reuse* ☺
2. stdio.h *et al* – subroutine libraries are bunches of reusable modules
3. IDE – a tool that generates GUI code automatically using built-in reusable modules

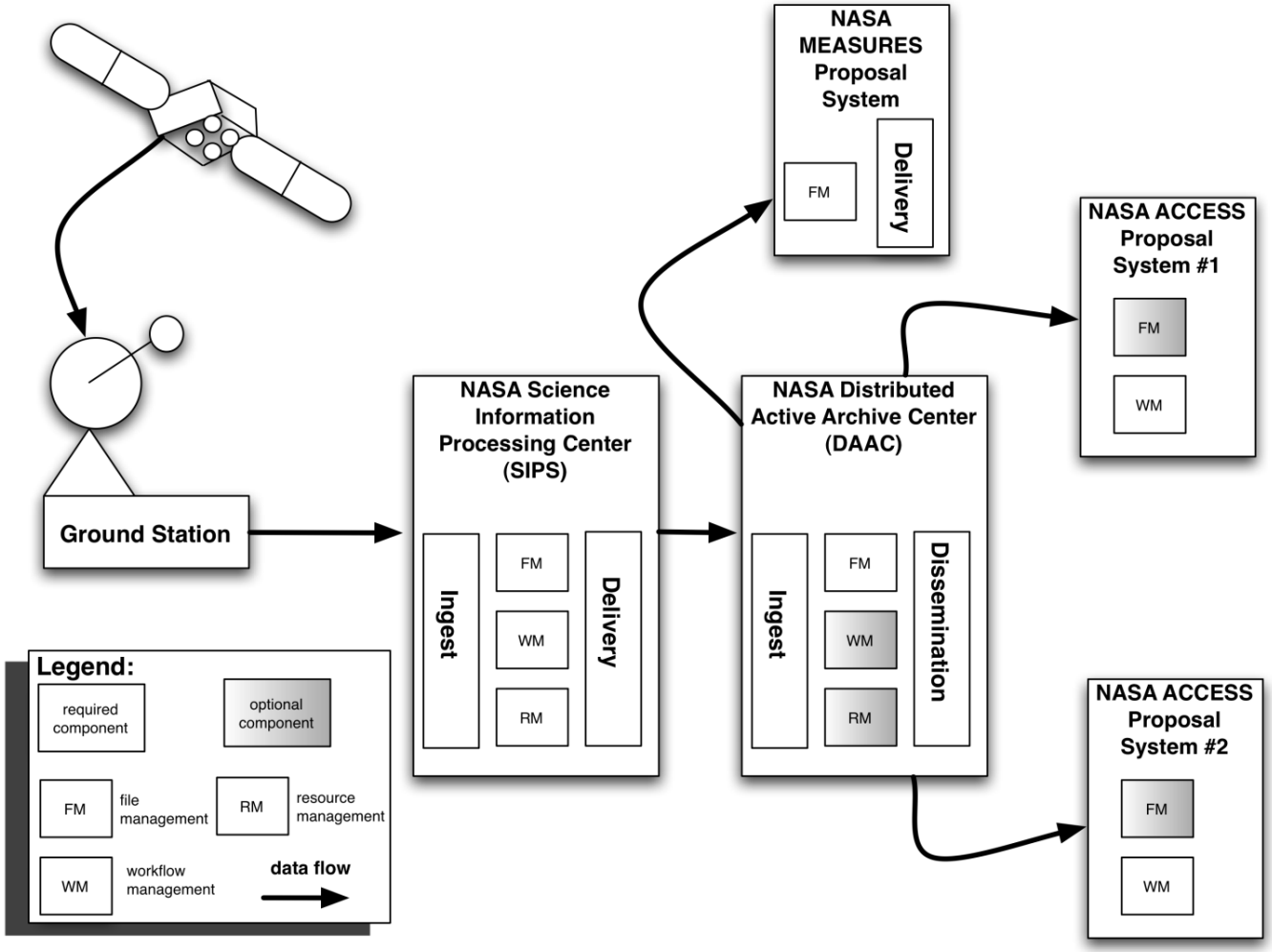
- But software reuse technologies are more than these…

Real Benefits from Software Reuse

Company	Benefits from Software Reuse
DEC	• Cycle time: 67~80% lower (reuse level 50~80%)
Fujitsu	• Projects on schedule: increased from 20% to 70% • Effort: reduced from 600 person-days to 4 person-days
HP	• Defects: 24~76% lower using 2 projects for example • Productivity: 40~57% higher • Time-to-Market: 42% lower
NEC	• Productivity: 6.7 times higher • Quality: 2.8 times better
Toshiba	• Defeats: 20~30% lower (reuse level 60%)
Raytheon	• Productivity: 50% higher (reuse level 60%)

– From "Practical Software Reuse' by Michel Ezran *et al*

A Software Reuse Initiative in NASA



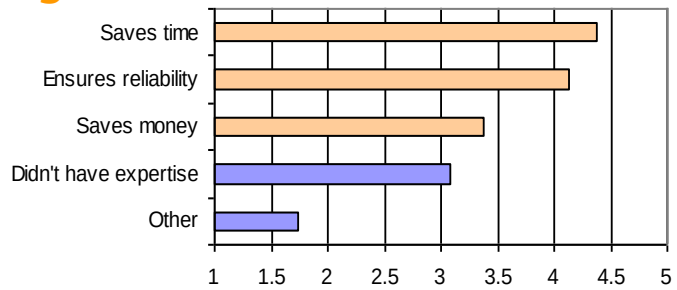
- From http://www.grss-ieee.org/ieee-grss-newsletter-2011

NASA Earth Science Data Systems (ESDS) Software Reuse Working Group (WG)

- The WG was started in 2004 to facilitate reuse of software assets within the NASA Earth science community.
- Membership is limited to NASA-funded projects and investigators, though there have been many contributions from the general Earth science community.
- The WG has been working to establish a “marketplace” for reusable Earth science software artifacts by working to increase the supply and availability of reusable assets.
- Also, the WG has worked to increase the community capacity and desire for reuse by demonstrating the feasibility and value of reuse

Software Reuse Surveys by ESDS

- A survey on the reuse practices of the Earth science community was conducted in 2004 and repeated in 2005
- Both surveys show the same basic results:
  - Developers need to be able to easily locate and evaluate available reusable artifacts.
  - Top three motivations for reuse match the WG goals:
    - Saving time
    - Saving money
    - Ensuring reliability
  - Top three factors to increase reuse:
    - Earth science catalog/repository of reusable assets
    - Greater use of open source licensing
    - More education and guidance on reuse
  - Top two barriers to reuse:
    - Did not know reusable assets existed
    - Did not know where to look for reusable assets



*** Source: Marshall, J.J.; Olding, S.W.; Wolfe, R.E.; Delnore, V.E., "Software Reuse Within the Earth Science Community," *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*, pp.2880–2883, July 31, 2006 – Aug. 4, 2006.**

Reuse WG Activities

Reuse Implementation Projects
Efforts that result in the publication or use of a reusable component

Support/Enablement Activities
Efforts that provide tools and mechanisms to enable reuse

Outreach and Education Activities
Efforts that increase community awareness & understanding of benefits, best practices, etc.

Policy Change Activities
Efforts to reduce policy barriers to reuse

Reuse Incentive Activities
Awards and structural changes that directly or indirectly encourage reuse

- Examples of work in some of these areas:
  - Recommending that NASA create a Reuse Enablement System; developing Reuse Readiness Levels; developing guideline documents
  - Creating a web site to promote and provide information about reuse; participating in social network groups
  - Developing a reuse peer-recognition award
- Dozens of WG members have contributed to this work.

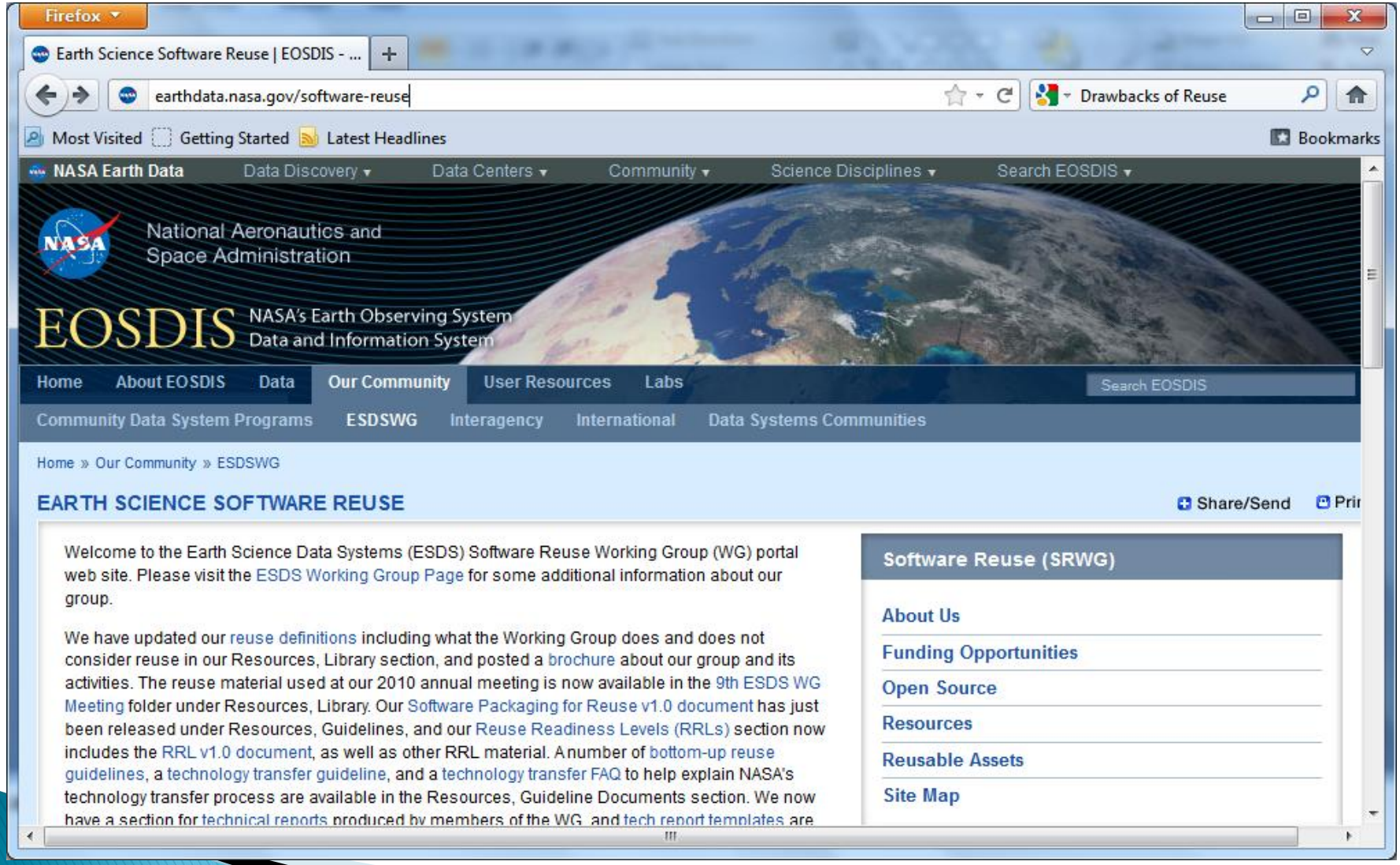
Reuse Readiness Levels (RRLs)

Level	Summary	Description
RRL 1	Limited reusability; the software is not recommended for reuse.	Little is provided beyond limited source code or pre-compiled, executable binaries. There is no support, contact information for developers or rights for reuse specified, the software is not extensible, and there is inadequate or no documentation.
RRL 2	Initial reusability; software reuse is not practical.	Some source code, documentation, and contact information are provided, but these are still very limited. Initial testing has been done, but reuse rights are still unclear. Reuse would be challenging and cost-prohibitive.
RRL 3	Basic reusability; the software might be reusable by skilled users at substantial effort, cost, and risk.	Software has some modularity and standards compliance, some support is provided by developers, and detailed installation instructions are available, but rights are unspecified. An expert may be able to reuse the software, but general users would not.
RRL 4	Reuse is possible; the software might be reused by most users with some effort, cost, and risk.	Software and documentation are complete and understandable. Software has been demonstrated in a lab on one or more specific platforms, infrequent patches are available, and intellectual property issues would need to be negotiated. Reuse is possible, but may be difficult.
RRL 5	Reuse is practical; the software could be reused by most users with reasonable cost and risk.	Software is moderately portable, modular, extendable, and configurable, has low-fidelity standards compliance, a user manual, and has been tested in a lab. A user community exists, but may be a small community of experts. Developers may be contacted to request limited rights for reuse.
RRL 6	Software is reusable; the software can be reused by most users although there may be some cost and risk.	Software has been designed for extensibility, modularity, and portability, but software and documentation may still have limited applicability. Tutorials are available, and the software has been demonstrated in a relevant context. Developers may be contacted to obtain formal statements on restricted rights or to negotiate additional rights.
RRL 7	Software is highly reusable; the software can be reused by most users with minimum cost and risk.	Software is highly portable and modular, has high-fidelity standards compliance, provides auto-build installation, and has been tested in a relevant context. Support is developer-organized, and an interface guide is available. Software and documentation are applicable for most systems. Brief statements are available describing limited rights for reuse and developers may be contacted to negotiate additional rights.
RRL 8	Demonstrated local reusability; the software has been reused by multiple users.	Software has been shown to be extensible, and has been qualified through test and demonstration. An extension guide and organization-provided support are available. Brief statements are available describing unrestricted rights for reuse and developers may be contacted to obtain formal rights statements.
RRL 9	Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems.	Software is fully portable and modular, with all appropriate documentation and standards compliance, encapsulated packaging, a GUI installer, and a large support community that provides patches. Software has been tested and validated through successful use of application output. Multiple statements describing unrestricted rights for reuse and the recommended citation are embedded into the product.

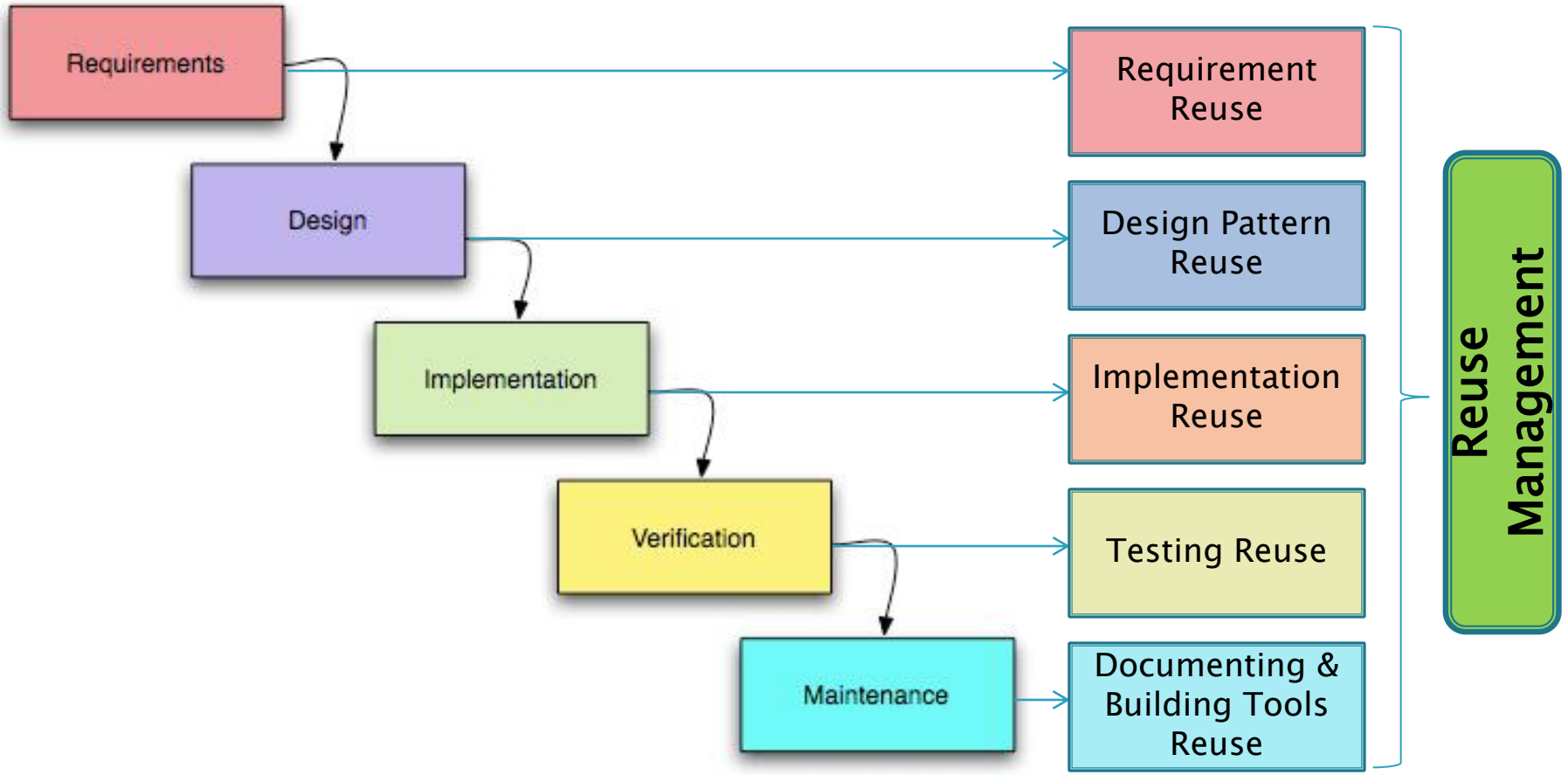
Software Reuse Award

- The WG developed a Peer-Recognition Software Reuse Award to recognize those people whose efforts and projects contribute to the practice of software reuse in the Earth science community.
- There are three categories of the Reuse Award:
  - *Contribution*, for the creation and offering of reusable software assets
  - *Utilization*, for the reuse of existing software assets in new projects and the demonstration of reuse for the benefit of the community
  - *Peer Education*, for providing guidance to others on preparing or adopting software for reuse
- http://www.esdswg.com/softwarereuse/Resources/awards/ for additional details

http://earthdata.nasa.gov/software-reuse



A Systematic Reuse Taxonomy



Part of this figure is from: http://en.wikipedia.org/wiki/File:Waterfall_model.png

Summary of Module 1

- Definition of Software Reuse
- Motivations and Benefits of Software Reuse
- Downside of Software Reuse
- A Software Reuse Case Study from NASA
- A Taxonomy of Software Reuse in SDLC

Further Readings

- "Practical Software Reuse' by Michel Ezran *et al*
- NASA Software Reuse Case: http://www.grss-ieee.org/ieee-grss-newsletter-2011/
- Art of Software Reuse (Blog): http://artofsoftwarereuse.com/