InSAR Scientific Computing Environment

Investigators: Paul A. Rosen[1] (PI), Eric Gurrola[1], Walter Szeliga[1], Giangi Sacco[1], Howard Zebker[2]

Consulting Collaborators: Mark Simons[3], Scott Hensley[1], David Sandwell[4]

Students: Piyush Shanker[2,3], Albert Chen[2], and Cody Wortham[2]

[1] Jet Propulsion Laboratory/California Institute of Technology
[2]Stanford University
[3]California Institute of Technology
[4]San Diego Institution of Oceanography

AIST-08-0023
ESTO Forum 2011

InSAR Scientific Computing Environment

The InSAR Scientific Computing Environment (ISCE) includes:

- Commitment to community needs and involvement

•Legacy (ROI_PAC) and New InSAR processing software based on SRTM to UAVSAR processors

- More accurate
- Better geolocation
- Faster

•A new computing environment that is easy to use, flexible, and extensible

- Canned Applications with defaults for quick and easy processing and Framework of Components from which the user can build his own Applications
- object-oriented Python Component wrappers that manage user interface, workflow, and the life cycle of processing components
- Fortran/C radar domain expert code left mostly in tact: preserve experience and testing
- Runtime polymorphism of components through plug-in architecture and factory pattern instantiation
- User configuration of Image/Meta data formats and I/O

InSARProc Workshops 2008 & 2011

- The goals of ISCE are a direct response to the priorities set by an international community of radar processor developers and users as determined by two NASA sponsored InSAR workshops, one convened in 2008 at Stanford University and the other in 2011 at the Scripps Institution of Oceanography.
- The goals of the 2008 workshop were to assess the strengths and weaknesses of the existing InSAR processing packages at the time, define the capabilities of the next-generation processors required by the user community, and set the standards and structure for new InSAR processor development.
- The goals of the 2011 workshop were to compare and validate the accuracy and performance of the various non-commercial InSAR processing packages, both legacy and new, and to generate feedback and suggestions for further developments.

Sponsorship: InSARProc Workshops were endorsed by NASA's DESDynI Steering Committee and sponsored jointly by NASA’s Earth Surface and Interior Program and by the Stanford Center for Computational Earth & Environmental Science (CEES) and the Scripps Institution of Oceanography.

InSARProc Workshop 2008 Recommendations

The high-level guidelines and recommendations coming out of InSARProc2008 were prioritized in two groups:

Highest priority:

•The next-generation software should be accurate in phase and location

•The package should be extensible, modular, and efficient

•The package should be well documented, supported, accessible to all users

Second priority:

•The software should be portable, thus with a small and light footprint

•The new codes should be open source in the sense that they should be available to anyone for inspection, use, modification, and redistribution.

•The code should be thoroughly tested, debugged, pass benchmarks, and verified.

•Results should be readily reproducible and repeatable.

•The package should follow well-defined, standardized products with clear coordinates.

InSARProc Workshop 2011

InSARProc Workshop 2011 compared the accuracy and performance of the various non-commercial InSAR processing packages. Four different InSAR packages were compared for several challenging data sets: (1) ROI_PAC (JPL Open Channel legacy code widely used by scientists); (2) ISCE; (3) the Stanford core processor STD_PROC contained in ISCE (as a separate package without the framework); and (4) GMT_SAR developed at Scripps.

•ISCE did well in terms of accuracy, speed, and ease of use.

•Some found it difficult to install. The use of Python and the gcc requires a self-consistency to the development environment that many users do not understand. One of the key actions for the ISCE team is to provide more complete descriptions of what constitutes a self-consistent environment, and provide the tools and information for a user to easily create one.

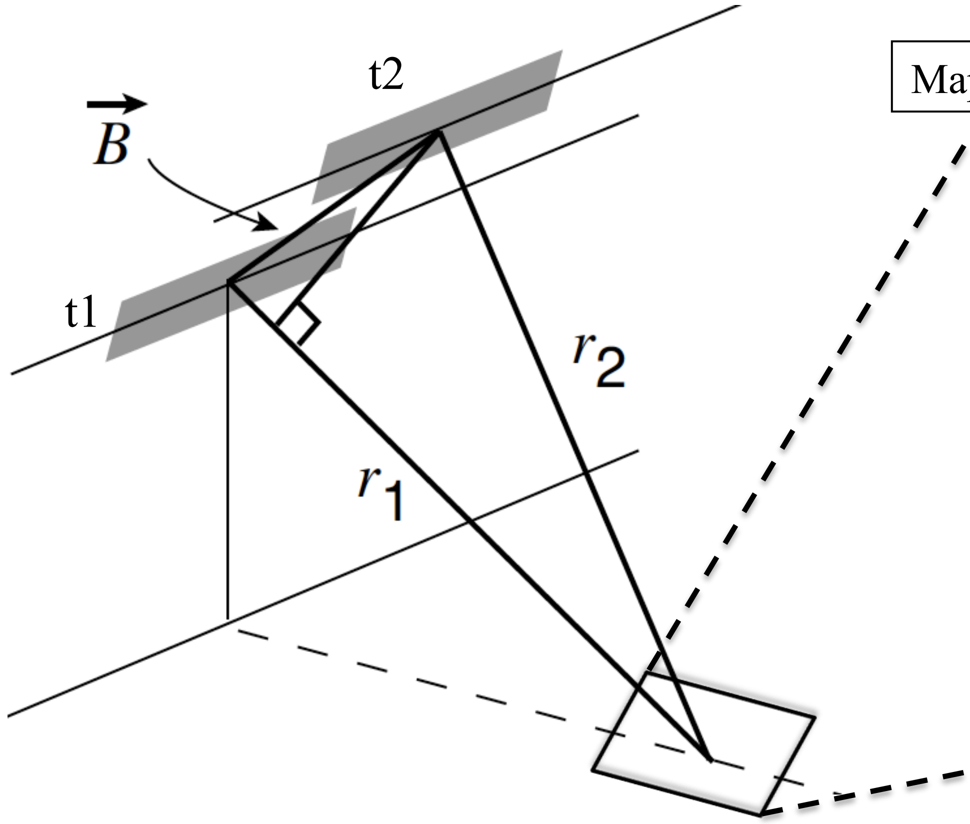
•Recommendations for further comparisons were made, including incorporation of a few commercial InSAR packages and simulated data to isolate differences in results found between the different packages.

•One of the top action items from the workshop was to make ISCE available for rapid and wide distribution, indicating that the development is of interest to the community.

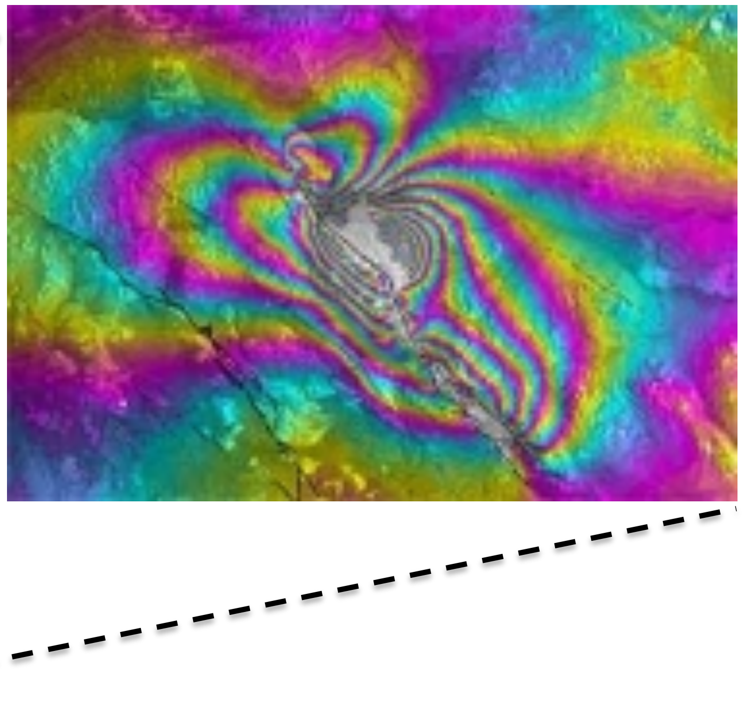
Earth Science Technology Office

Interferometric Synthetic Aperture Radar

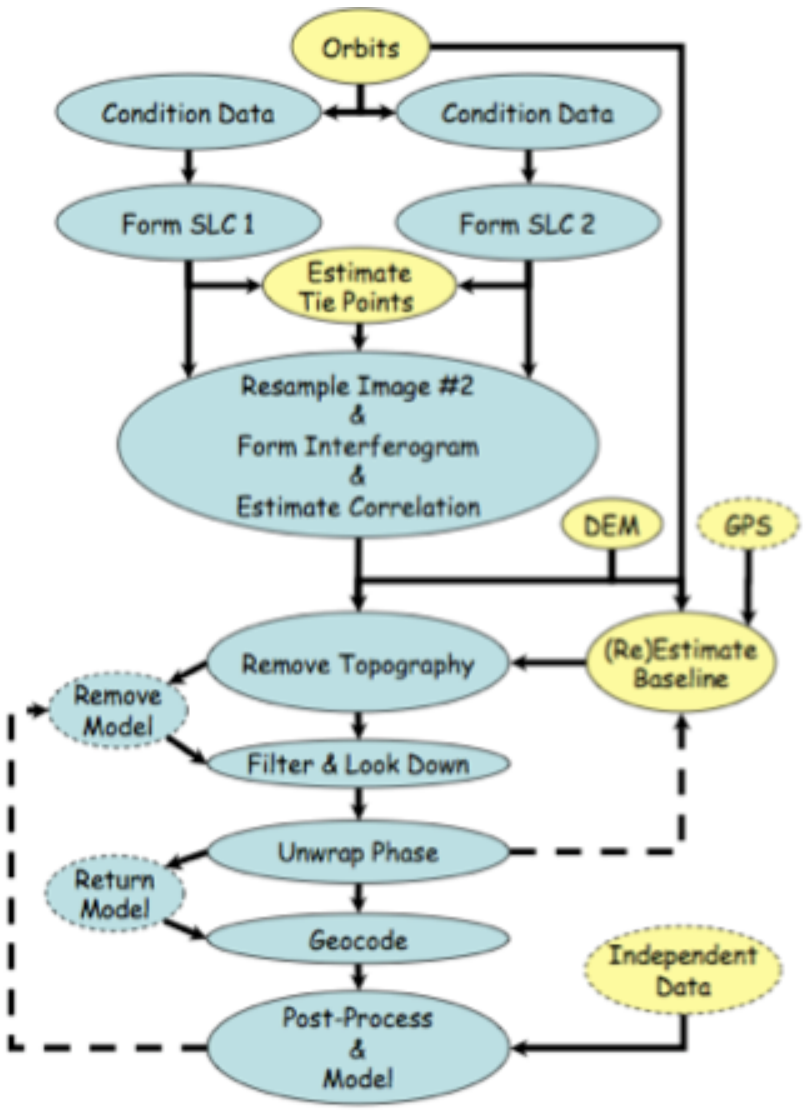
Radar data from two passes over a scene at different times from a variety of satellites (ERS, JERS, EnviSAT, ALOS, TerraSAR-X,…) are processed into images, interferograms, and geocoded topography and Earth displacement maps



Map of Earth Displacement between t1 and t2



ROI_PAC Legacy Processor: InSAR Processing Flow



Earth Science Technology Office

STD_PROC: Improved/Enhanced Processor for ISCE

• STD_PROC is a new InSAR processing package being developed at Stanford under the current AIST.

•STD_PROC overlaps much of the functionality of ROI_PAC but it will also extend the functionality to include ***time-series*** analysis methods for analyzing evolution of displacement fields over time from multiple passes over a scene and to include ***persistent scatterer*** methods to allow interferometric processing in the presence of low-correlation.

• STD_PROC borrows from and builds upon the improved processing algorithms developed for SRTM and UAVSAR InSAR processors.

• STD_PROC applies a motion compensation algorithm to produce images in a common geographical coordinate system from the start to facilitate time-series analysis of interferograms formed from multiple pairings of radar images.

• STD_PROC is more efficient and much faster through the use of improved algorithms and the use of OpenMP

STD_PROC Motion Compensation Algorithm

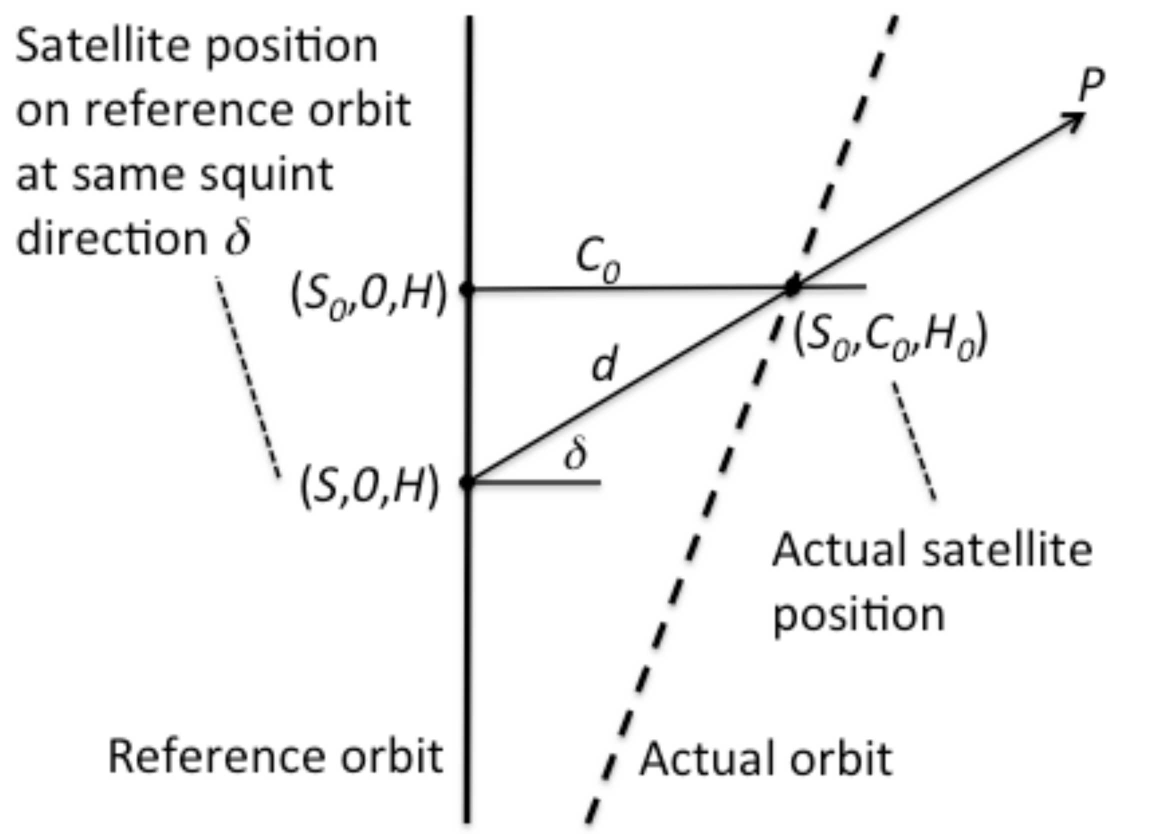
• STD_PROC, at the core of ISCE, preserves accuracy of its data products by taking advantage of the improved accuracy of orbit determination now available and implementing all of the code in a uniform geometric framework (Zebker *et al*., 2010).

•This approach is based on well-known motion compensation techniques and facilitates analysis of a time series of many observations of a particular location by use of a motion-compensated geodetic coordinate system rather than the traditional range/Doppler coordinate system specific to a given observation (used by for example ROI_PAC).

•The equations implemented in the processor are simplified by use of a local spherical earth approximation and a corresponding circular approximation of the platform orbit. The spherical coordinate system, referred to as SCH (in which S is the local along track direction, C is the cross track direction, and H is the height above the approximating sphere.

•In this approach, the direction from the satellite to a point on the Earth is determined through the estimated Doppler centroid. The position of the satellite is compensated along this direction back to the reference track as a range correction. A corresponding phase correction is also applied. The equations encoded in STD_PROC and hence ISCE are developed in detail in Zebker *et al.* (2010) and Gurrola *et al* (2010).

ISCE Motion Compensation Geometry


Satellite position on reference orbit at same squint direction δ
P
C_0
$(S_0,0,H)$
(S_0,C_0,H_0)
d
δ
$(S,0,H)$
Actual satellite position
Reference orbit
Actual orbit

The motion compensation geometry, showing the correspondence of an image pixel at P, actual satellite position (S_0 ,C_0, H_0) and the assigned reference position on the circular orbit ($S,0,H$).

Key Drivers of the ISCE Architecture

Key drivers of the ISCE architecture:

1. Preserve the vast expertise and testing currently encoded in Legacy Software including both ROI_PAC and STD_PROC.

2. Make that Legacy Software more lean in terms of the number of auxiliary tasks it needs to do (such as self configuration and I/O configuration).

3. Build modern object oriented structures around and behind the legacy code to manage that code and push rather than pull user configuration onto that code before execution.

4. Implement common functions and services such as I/O through APIs to allow their implementations to change and to allow for user configuration and selection of those functions at run-time.

5. Build in polymorphism mechanisms to allow user selections to alter the implementations of major processing steps and common functions. Also allows just-in-time insertion of alternative functions and major components

ISCE Component Architecture

Componentization of a legacy program:

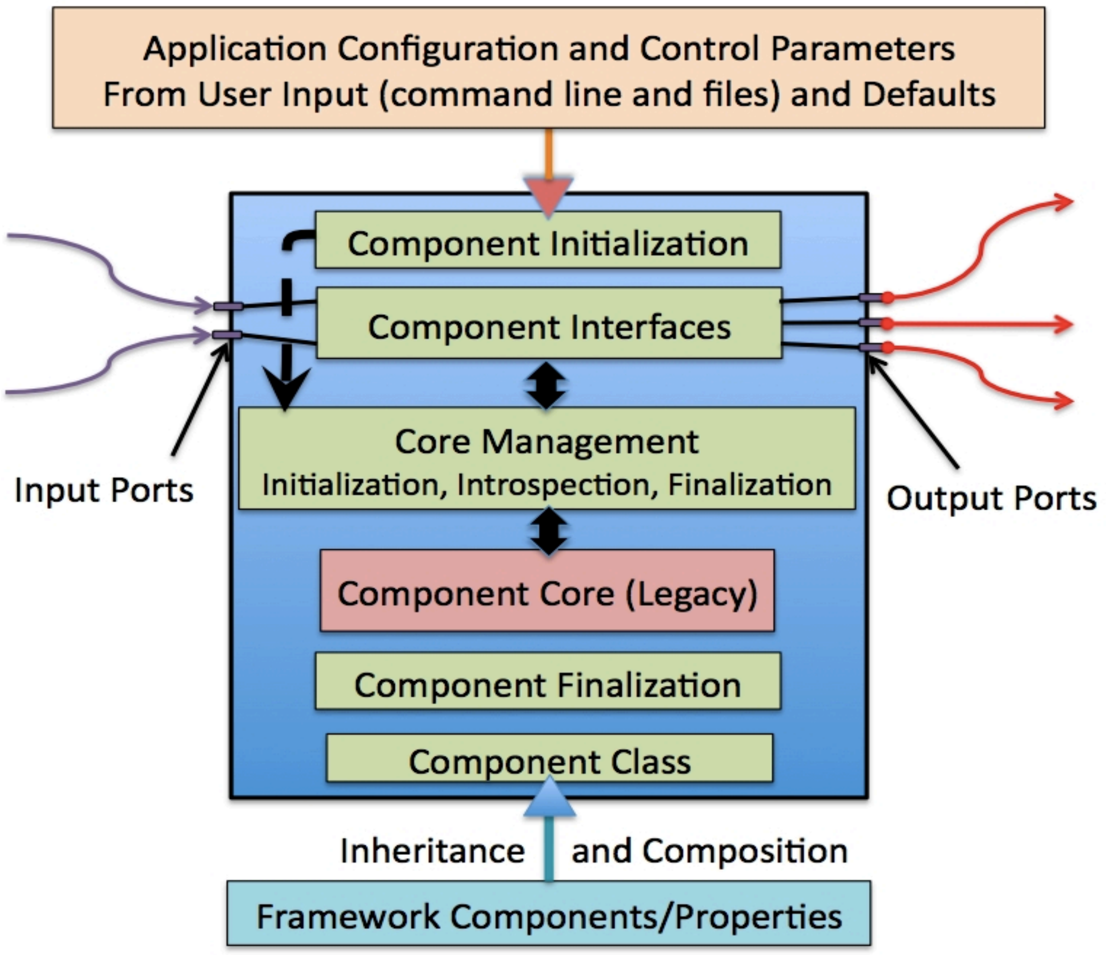
(a) Embed the legacy program at the core of an object-oriented component written in Python.

(b) Provide complete management of the core component through its life-cycle from initialization through proper finalization. Provide previously unavailable introspection capability

(c) Provide input and output ports with well-defined interfaces

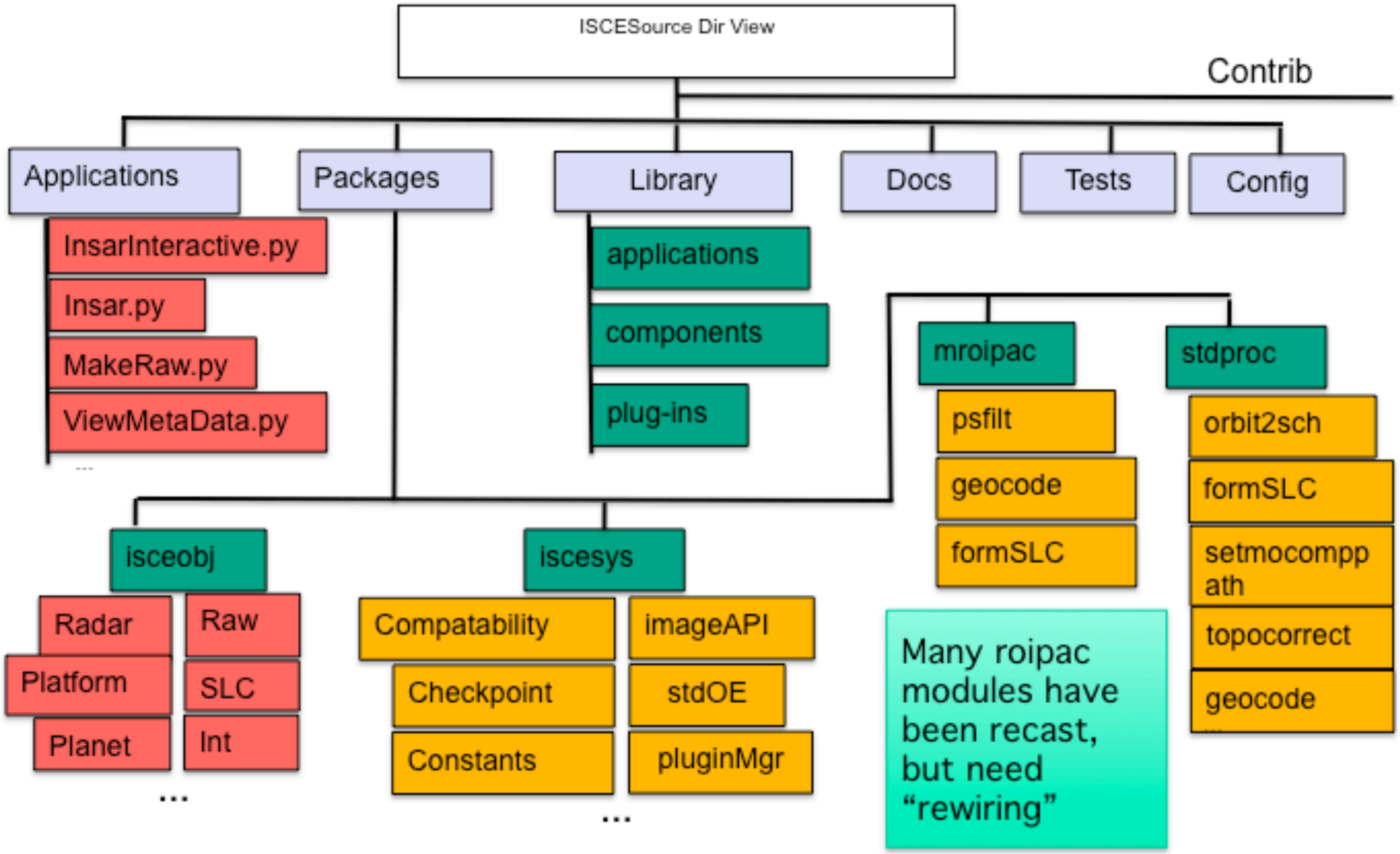
(d) Provide well-defined interface for flow of control parameters from the user through controlling applications to the component.

(e) Provide Framework Components and Properties for common object definitions and common tasks.



Application Configuration and Control Parameters From User Input (command line and files) and Defaults
Component Initialization
Component Interfaces
Core Management
Initialization, Introspection, Finalization
Component Core (Legacy)
Component Finalization
Component Class
Input Ports
Output Ports
Inheritance
and Composition
Framework Components/Properties

ISCE Architecture



Conclusion

•The ISCE AIST project has met all of its goals to date and is currently in its planned final year of testing and documentation.

•A beta version of ISCE has been licensed to a few users for testing and feedback. We are currently working to find a proper licensing procedure for wider release of the software to the user community.

•ISCE is now mature enough that other customers are beginning to adopt it for future development.

- ➢ ISCE is being baselined as the core engine for a new internal development task being formulated at JPL and Caltech for rapid response to earthquakes and other natural disasters. The JPL/Caltech team plans to write substantial proposals to agencies interested in scientifically based rapid response capabilities
- ➢ ISCE is generating considerable interest in the traditional ROI_PAC community. Community ROI_PAC developers are now investing their time in ISCE rather than updating ROI_PAC
- ➢ Keck Institute of Space Science Earth Change Project at Caltech baselining ISCE
- ➢ Several new ROSES proposals (Applications for Geodetic Imaging) are baselining ISCE

Acknowledgements

•We thank Albert Chen and Cody Wortham, both PhD candidates at Stanford University, and Piyush Shanker, PostDoc at Caltech, formerly PhD candidate student at Stanford, for their contributions to the Stanford legacy code at the core of ISCE.

•The authors would like to thank the Earth Science Technology Office at NASA for support. This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under a contract with NASA, and at Stanford University under a contract with JPL.

References

•Gurrola, E., P. Rosen, G. Sacco, W. Szeliga, H. Zebker, M. Simons, D. Sandwell, P. Shanker, C. Wortham, and A. Chen (2010). "InSAR Scientific Computing Environment". 2010 American Geophysical Union Meeting.

• Rosen, P. *et al.* (2009). "InSAR Scientific Computing Environment". 2009 American Geophysical Union Meeting. Also presented a summary of InSAR SCE and progress at the WinSAR meeting at Fall AGU 2009

•Zebker, H., S. Hensley, P. Shanker, C. Wortham (2010). Geodetically Accurate InSAR Data Processor. *IEEE Trans. On Geoscience and Remote Sensing*, 48(12).

•Rosen, P. A., S. Hensley, and G. Peltzer (2004), Updated Repeat Orbit Interferometry Package released, Eos Trans. AGU, 85(5).