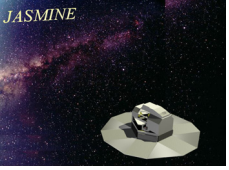


JASMINE Simulator

Yoshiyuki Yamada[a)], Naoteru Gouda[b)], Taihei Yano[b)], Yukiyasu Kobayashi[b)], Takuji Tsujimoto[b)], Masahiro Suganuma[b)], Yoshito Niwa[a,b)], Nobutada Sako[c)], Yoichi Hatsutori[c)], Takashi Tanaka[c)], and JASMINE Working group

a):*Department of Physics, Kyoto University*、 b):National Astronomical Observatory Japan、
c):(Department of Engineering, University of Tokyo)

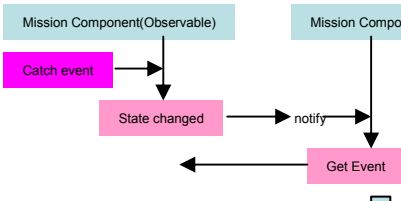
ABSTRACT

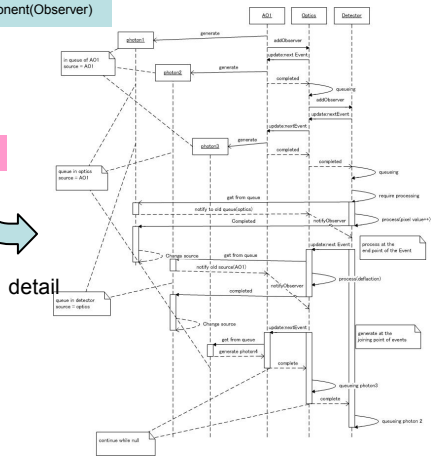
We explain simulation tools in JASMINE project(JASMINE simulator). The JASMINE project stands at the stage where its basic design will be determined in a few years. Then it is very important to simulate the data stream generated by astrometric fields at JASMINE in order to support investigations of error budgets, sampling strategy, data compression, data analysis, scientific performances, etc. Of course, component simulations are needed, but total end-to-end simulations which include all components from observation target to satellite system or scientific outputs are also very important. We find that new software technologies, such as Object Oriented(OO) methodologies are ideal tools for the simulation system of JASMINE(the JASMINE simulator).

The simulation system should include all objects in JASMINE such as observation techniques, models of instruments and a design of satellite bus system, orbit, data transfer, data analysis etc. in order to resolve all issues which can be expected beforehand and make it easy to cope with some unexpected problems which might occur during the mission of JASMINE. So, the JASMINE Simulator is designed as handling events such as photons from astronomical objects, control signals for devices, disturbances for satellite attitude, by instruments such as mirrors and detectors, successively. The simulator is also applied to the technical demonstration "Nano-JASMINE".

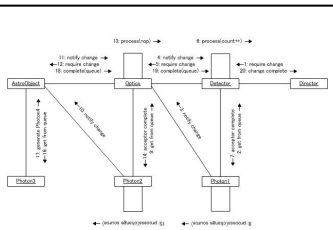
Simulator Framework

Mission component ⇒ apply Observer Pattern.
Telescope, Detector, OBC and etc. handle such events. These EventHandlers are implemented as both Observer and Observable. Event:s (Photon, Control signal, Disturbance) are passed from upsterm event handler to down.
Mission components know how to treat such events.
Satellite component ⇒ Façade (GetStatus class)
Mission components asks satellite position and attitude to Satellite components.
Satellite components compute attitude in complicated way.
Mission components should know only interfaces of GetStatus class.

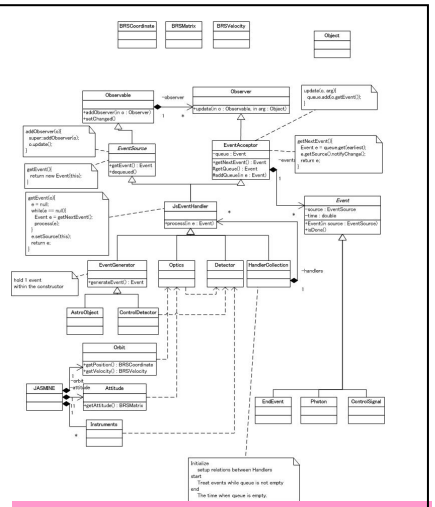




Sequence diagram

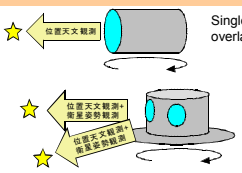


Collaboration diagram of mission component
UML collaboration diagram of the core framework. JASMINE components are placed within the chain. Events which the simulator treats are handed from the components at upstream to that at downstream.



Class diagram of mission component
UML class diagram of the framework. All JASMINE components are inherited from JSEventHandler class, which is a subclass of Observer and Observable.

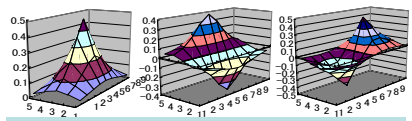
Observation method
Strategy of the methods are encapsulated into the class.



Single telescope with overlapping plate

Double telescope with great circle scan

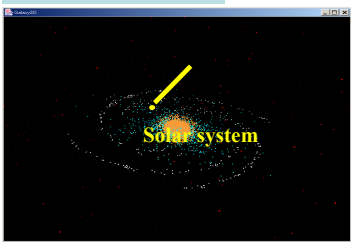
etc.



Requirement of data compression
High compression rates,
Enable with satellite CPU(80186 level @ 2006)
⇒
Solution: Combination of Karuhnen-Loeve transformation and Golomb-Rice code. About 60% compression rate is acchieved. Above are first three principal components.

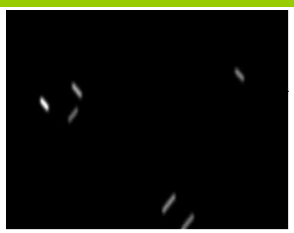
The requirement of attitude stability is very high. Use mission instrument as 'star tracker'. For achieving sub pixel accuracy of position detection, Karuhnen-Loeve transformation is also effective.

Galaxy model
Wainscoat et al(92), Cohen(94)

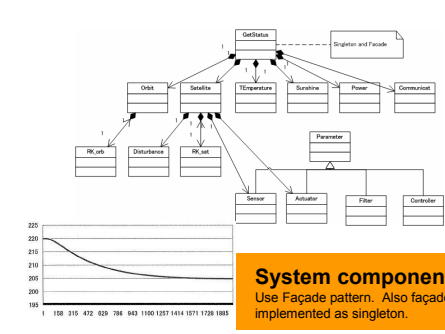


Core Framework

Focal plane simulation
CCD model⇒composite of pixels
Exposure are simulated in photon level.
PSF shape, attitude disturbance and attitude estimation error, error caused by CTI≠0, CR events etc. are simulated



Satellite attitude information in sub pixel accuracy are feeded back to the satellite attitude control system.



System component simulation
Use Façade pattern. Also façade class is implemented as singleton.

Reference to terms with specific meaning within a object-oriented programming context:
- **class** encapsulates common behavior of a group of objects;
- **attributes** data member of a class;
- **methods** functions which may be performed on instances of a class;
- **object** an instance of a class;
- **abstract class** a class from which no instances may be created;
- **inheritance** a way to form new classes or objects using predefined objects or classes where new ones simply take over old one's implementations and characteristics.