Design and Implementation of scalable GetFeasibility Service to enhance Sensor Planning Service for Earth Observation Task

Xiang Zhang, Nengcheng Chen
State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing
Wuhan University
Wuhan, China
zhangxiangsw@whu.edu.cn , cnc@whu.edu.cn

***Abstract*—Sensor Planning Service (SPS) is a critical web service in Sensor Web, given its complexity of feasibility check. This paper is dedicated to design a scalable GetFeasibility service outside of the SPS to determine the feasibility of a plan with a certain theme regarding an earth observation satellite sensor. To enables the scalability of GetFeasibility service, in the logic part of GetFeasibility, we construct the mapping between sensor capability and task. Time-window, intended applications and weather circumstance have been taken into consideration in the mapping to achieve a more reasonably result. In addition, the Simplified General Perturbation Model 4 (SGP4) is viewed as a virtual EO sensor named as sgp4sensor served by SPS. And we adopt Service-Oriented Architecture (SOA) based on Apache CXF which is an excellent Java Web Service technology to realize. To verify the effect of the feasibility check, one kind of emergency task in agriculture (flood monitoring) was submitted to SPS, and the SPS communicate with the GetFeasibility service using standard Web Service protocols, finally the SPS return a submit response message contained the report of feasibility. The results show that the proposed scalable GetFeasibility service cooperates well with the SPS, and more importantly, it can improve the reliability of SPS.**

Keywords—Sensor Planning Service; Geospatial Sensor Web; SGP4; Web Service; Flood Monitoring

I. INTRODUCTION

Geospatial Sensor Web has been an effective theory and method to model, mange, discovery and plan heterogeneous sensors and data in World Wide Web, which has been defined as the observation web that performs Earth Observation (EO) [1]. Its groundwork is the Sensor Web Enablement (SWE) initiative, conceived by OGC. The SWE provides a series of standard information encodings and service models, which is the key part of an integrated framework for discovering and interacting with Web-accessible sensors [2]. The Sensor Planning Service (SPS) is intended to provide a standard interface to task collection assets and to support systems that surround them [3]. SPS 2.0 standard defines five informational operations and six functional operations that can be requested by a client and performed by an SPS server. The informational operations include GetCapabilities, DescribeTasking, DescribeResultAccess, GetTask and GetStatus. And the functional operations are the GetFeasibility, Reserve, Confirm, Submit, Update and Cancel operations. To check the feasibility of a certain observation plan, the GetFeasibility interface will be called by SPS, which means through the feasibility response form the GetFeasibility operation, the SPS server will determine whether to accept the observation task submitted by a client. However, concerning the complexity of feasibility algorithm and various factors in planning especially in EO tasks, the design and implementation of GetFeasibility is still a big challenge for Geospatial Sensor Web applications.

The Heterogeneous Mission Accessibility Project (HMA) launched and overviewed by the Ground Segment Coordination Body (GSCB) has attempted to establish harmonized access to heterogeneous EO missions' data from multiple missions [4, 5]. The Feasibility Server built in HMA includes propagation algorithms for predicting satellite sensor earth track and functionalities for taking into account meteorological conditions [6]. GeoBliki EO-1 Sensor Web managed by the NASA Goddard Space Flight Center (GSFC) Information System Division examines the available sensors for the feasibility of tasking [7]. This includes the retrieval of data from ground sensors such as the Remote Automated Weather Stations (RAWS) [8]. The first two SPS implementations may take an advantage on reliability and accuracy, but they have not enough information for researchers to valid and innovate. Besides, no evidence shows that the GetFeasibility operation of them can be extended easily according to newly added feasibility rules. Open Sensor Web Architecture (OSWA) proposed by National Information and Communications Technology Centre of Australia (NICTA) also has an implementation of a set of Sensor Web services, which includes SPS. Inside the SPS, a rule engine is used to clarify the feasibility of the plan made by the user [9]. The 52North Sensor Web community has developed a SPS framework (52nSPS) for all kinds of sensor plugins. So the feasibility check is more about syntax check of the parameters submitted [10, 11]. But the OSWA and 52nSPS didn't support the EO sensor SPS at present. In short, in the Geospatial Sensor Web SPS, the existing GetFeasibility is not scalable and clear enough to enhance the EO SPS, so there is an urgent demand to redesign and realize an extensible EO SPS.

In order to address the challenge mentioned, our method is to construct a GetFeasibility Service using web service instead of embed in the SPS. In the algorithm of feasibility check, a scalable mapping between sensor capability and task was put forward.

The organization of this paper is as follows. Section II presents the architecture of GetFeasibility Service and the Mapping mechanism in it. Section III outlines the key method and technologies, including sgp4sensor, mapping, and CXF-based services. Section IV shows the flow path and result of the enhanced EO SPS regarding to a flood monitoring task in agriculture. Section V discusses the advantages of the proposed service, and Section VI summarizes and outlines further work.

II. Design

A. *Architecture and Interactions*

Our purpose of the GetFeasibility service outside of the SPS must take the followings into consideration.

- The GetFeasibility service must be loosely coupled with the standard SPS. If we want to do some changes to the feasibility check logic part, we can make it without changing the main framework of the SPS. Even more, if we want to remove the older GetFeasibility service entirely and connect to another newer GetFeasibility service, the change in SPS can be as small as possible. Conversely, the service can be used by other SPS as well.
- The logic part of GetFeasibility service must represent the unique about feasibility check for the EO task. In other words, the service must be called for enhancing EO SPS, especially in the task about satellite sensors. And the logic part should be extensible to meet different satellite sensors and needs.

To meet the above two goals, the GetFeasibility service must be scalable enough, and adopt the standard Java Web Service technology to construct SOA system. The architecture is proposed as in Fig. 1. There are three main layers in this architecture: Application Layer, Framework Layer, and Business Layer.

The Application Layer is a friendly presentation of SPS interface. Its main function is to provide a simple and readable way to construct and send a SPS request. When the responses of those requests come back, the result can be presented in the form of a XML document in this layer. At the same time, the Application layer can be based on diverse devices such as a personal computer, iPhone or PDA. As shown in the Fig. 1, the User Client send a submit request to the SPS that contain the tasking parameters to plan a sensor.

The Framework Layer is an all-purpose SPS framework that supports the SPS standard that can be used to receive request and send response to the other components (the User Client and GetFeasibility Service). And this layer must implement all the SPS operations except GetFeasibility. The important module in it is a sensor called sgp4sensor that can be viewed as a virtual sensor [12, 13]. The sgp4sensor was planned through the SPS framework according to its Meta-information, and it is a virtual collection of EO satellites, because it does not obtain real remote sensing data. All we want to know is the feasibility check about it. As shown in the Fig. 1, the SPS send a GetFeasibility request to the Business Layer to check the feasibility of a given EO task submitted by user above.

The Business Layer is the key role in the whole architecture, and the core is the design of GetFeasibility Service. In order to be loosely enough with SPS, this layer is design to use the standard Web Service technology to be extendable. A uniform WSDL service description is used to expose the service, bindings, port types and operations. After receiving the feasibility check request from the SPS according to the WSDL, the Mapping between the capability of satellite sensor and the task play a vital role in determining whether the EO task can be done or not. The Mapping has three mappings: Time-window is used to judge the pass-time of all EO satellites in the Sensor Database. Intended applications are a scientific Database that includes the potential application scenarios for a specific satellite sensor. The last one is a weather circumstance check, designed to meet the cloud demand for optical sensors. After the Mapping, the GetFeasibility Service return a GetFeasibility result to SPS, and the SPS will return the submit response to the user according to the GetFeasibility result.

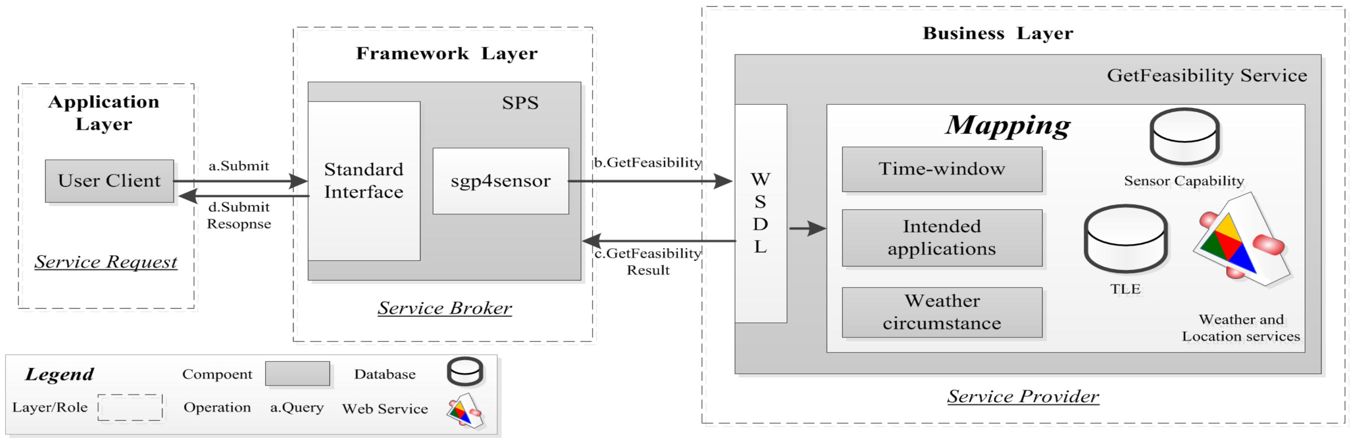


Fig. 1. Service-oriented architecture of scalable GetFeasibility service

B. The Mapping between Capability and Task

To address the feasibility check challenge, a Mapping between capability and task was proposed. While some basic principle has been explained in the Architecture part, the detail about the scalable mapping mechanism will be illustrated in this part.

A set of values for the tasking parameters constitutes a tasking request. Before the SPS can accept and carry out task, it has to check that task can be performed or not. That is called a feasibility check [3]. So the GetFeasibility Service shall proof the specific sensor is capable of executing the intended task. The aspects of check include five contents according to SPS standards: Syntax of tasking parameters, presence of mandatory parameters, validity of parameter configuration, asset availability and parameterization update according to current execution state [3]. The Mapping we proposed here is a specification of the standard as shown in Fig. 2.

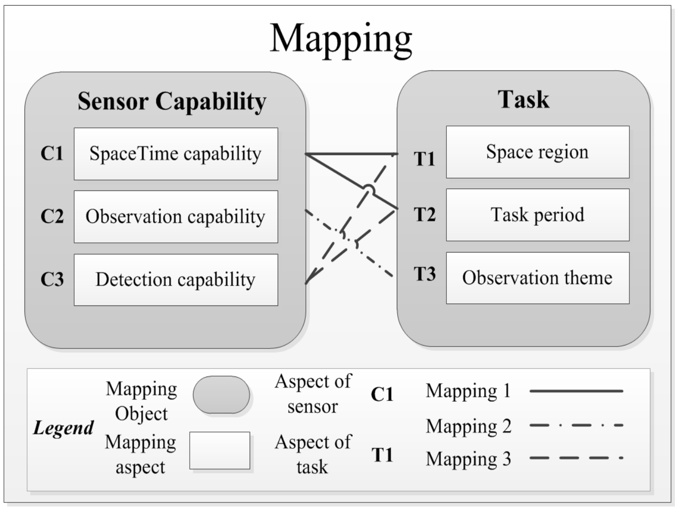


Fig. 2. Mapping mechanism

In the whole, two objects constitute the Mapping: sensor capability and task. Each of them is consist of three aspects. There are three mappings between the six aspects. For example, the Mapping 1 refers to two connections, that are C1-T1 and C1-T2, and the Mapping 2 refers to one connection, that is C2-T3, and the Mapping 3 refers to two connections that are C3-T1 and C3-T2. We define that, if the mapping 1 exists, and we called it Time-window in the architecture design, it means the satellite can pass the sky of tasking area, and it's the base of the observation. Similarly, if the mapping 2 exists, and we called it Intended applications in the architecture design, it means the satellite sensor can be used to observe the task with a certain theme (fire detection, flood monitoring etc.). If the mapping 3 exists, we called it Weather circumstance, it means the in the space region and period of the task, the weather have less adverse effect on the detection capability of the satellite sensor.

So, if all the three mappings do exist, we consider the task is feasible to be performed by satellite sensors. If any one of them failed, the task is not feasible. The real world is dynamic, and the satellite sensors and tasks are dynamic too, so the mapping result is a dynamic result as well. That is the key mechanism of the whole feasibility check in the GetFeasibility service we proposed.

Above all, the new aspects of sensor and task can be added to the mapping system, such as horizontal spatial resolution and spectrum resolution. Therefore, the new mapping, such as Mapping 4…Mapping n can be established to enhance the SPS further. So the capability of the GetFeasibility service itself can be promoted, and that is what we called- scalable.

III. IMPLEMENTATION

The key part of the implementation is the mapping and the GetFeasibility service, so the Framework Layer–SPS adopts 52nSPS. The 52nSPS accomplished by the 52North Sensor Web community is an excellent implementation of SPS according to standards 1.0 at present, and the main characteristic of 52nSPS is that it's flexible enough for the developer to develop new sensor plugins [11]. So the sgp4sensor was created to be a plugin planned in the SPS, and the user client is the 52nSPS user client.

A. Sgp4sensor

The sgp4sensor is a virtual sensor to represent all EO sensors on the satellites in the world. According to the guideline of 52nSPS, sgp4sensor must be registered into the Asset Manager (AM). The register document is shown in a website (http://202.114.114.26:8080/GetFeasibility-XML/register.xml) on the basis of the schema of "amRegister" in 52nSPS.

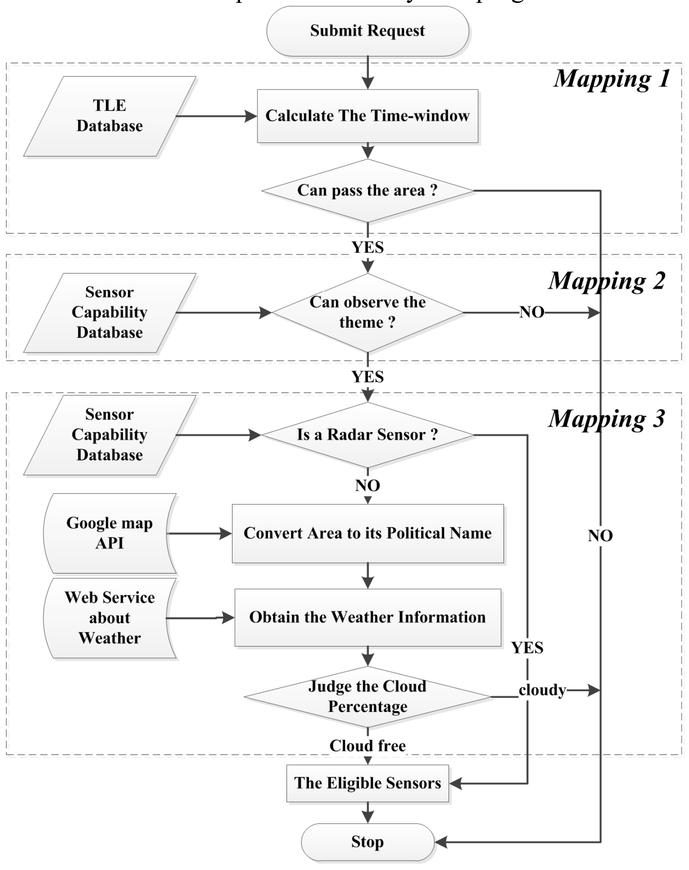
The register request will be sent to SPS, and SPS will control the sensor. In the register document, the name of sgp-4sensor is named as sgp4sensor1, and three parameters (space, starttime, endtime and theme) are needed to send a task-ing request. The submit request is shown in a website (http://202.114.114.26:8080/GetFeasibility-XML/submit.xml).

B. Mapping

According to the mapping mechanism described in the above, we have implemented it as shown in Fig. 3.

In the flowchart of mapping algorithm, the Mapping 1 is implemented by the Simplified General Perturbation Model 4 (SGP4). The TLE (Two Line Element) database is a live database from North American Aerospace Defense Command (NORAD) website: http://celestrak.com/NORAD/elements/, and the TLEs will be send into the SGP4.We select all Weather & Earth Resources Satellites and Scientific Satellites in it whose running cycle is less than 225 minutes to calculate more than 304 satellites' latitude and longitude of sub-satellite point. If the latitude and longitude of sensors in the task period are in the area's minimum bounding rectangle, it means the sensors on board can pass the area. So the sensor has SpaceTime capability (designed in Fig. 2), that is the Mapping 1 (C1-T1 and C1-T2).

The Mapping 2 is implemented mainly by a Sensor Capability Database based on the CEOS Database maintained by ESA (the European Space Agency) and WMO Database maintained by WMO (World Meteorological Organization) [14, 15]. The two databases contain extensive information on the capabilities of satellite, and relate them in some detail to the requirements of key user programs.



Submit Request
Mapping 1
TLE Database
Calculate The Time-window
Can pass the area ?
YES
Mapping 2
Sensor Capability Database
Can observe the theme ?
NO
YES
Mapping 3
Sensor Capability Database
Is a Radar Sensor ?
NO
Google map API
Convert Area to its Political Name
NO
Web Service about Weather
Obtain the Weather Information
YES
Judge the Cloud Percentage
cloudy
Cloud free
The Eligible Sensors
Stop

Fig. 3. The flowchart of mapping algorithm

So we built a Sensor Capability Database about the observation capability concluded by them. The database is composed of most of the EO satellite sensors encoding in SensorML [16]. According the SensorML 1.0.1, we put the observation capability information in the <sml:classifier> node. Website (http://202.114.114.26:8080/GetFeasibility-XML/MODIS_AQUA.xml) is an instance of "MODIS_AQUA.xml" encoding in SensorML. Form the four nodes <classifier name="intendedApplication">, we can conclude that if the theme of task is in five listed intended applications, MODIS will observe the theme in good quality, then the mapping 2 exists. Conversely, if the theme of task is "Soil moisture at surface", MODIS will have no good observation capability (designed in Fig. 2) in that task, then the mapping 2 disappears. That is what we called Mapping 2.

Mapping 3 is mainly about the weather impact on optical satellite sensor. Of course, Mapping 3 for radar is always exists. It is known that, the quality of remote sensing image will decrease when the cloud percentage increase. So in the feasibility check, it is necessary to predict and check the weather in the task period, and that is detection capability (designed in Fig. 2). Firstly, check the sensor type depended on the sensor capability database described in the Mapping 2. Secondly, we utilize the Google map API to convert task area (encoding in latitude and longitude) to its political name for weather predict in the next step. The Google map Geocoding API support reverses geocoding directly using the "latlng" parameter. For example, if we send a request like: "http://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&sensor=true_or_false", we will get the political name, which is Brooklyn, New York [17]. Thirdly, we take advantage of a web service about weather predict supported by China Meteorological Administration. The WSDL of the web service is: "http://webservice.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl" and we use the operation - "getWeatherbyCityName" by sending a SOAP request. The parameter of city name is the political name that is the result of second step. Because the weather predict is only about the following 3 days, so we only support the cloud estimate in the following 3 days at present. If the weather predict about the task region in the following 3 days contains "sunny" or "partly cloudy", then it means optical satellite sensor can detect the target in higher-quality, so the Mapping 3 exists; Conversely, If the weather predict result contains "cloudy", "rainy" or "snowy", then it means the cloud percentage is too high to get a high quality image, so the Mapping 3 disappears.

The three mappings consist of key of the feasibility check in the GetFeasibility service. After the mapping, the eligible sensors will be the output.

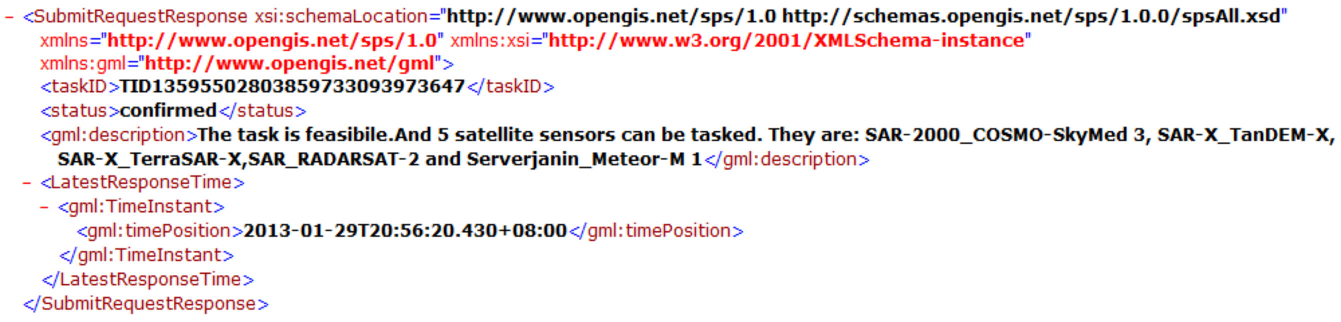
C. *GetFeasibility Service*

The GetFeasibility service is set up by Apache CXF that is an open-Source Service Framework. CXF helps us build and develop services using frontend programming APIs, like JAX-WS and JAX-RS. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI [18]. So we adopt Apache CXF Library 2.4.10 and build the service. The WSDL of GetFeasibility service is shown in a website (http://202.114.114.26:8080/GetFeasibility-XML/GetFeasibility.xml).

From the WSDL, the service name, parameters, bindings, port types and operations are exposed to client i.e. the SPS. We can notice that the SPS will send 19 parameters to GetFeasibility service and get a response encoding in linkedList. This is how we implement Business Layer in the architecture design.

IV. CASE STUDIES

Flood is one of the most horrific disasters in agriculture every year in China and East Asia. It is reported that, more than 480 km^2 crops had been hit by heavy rains only in late July 2012. Geospatial Sensor Web can and has to be a good support to forecast, detect, monitor, and evaluate flood. So it's critical to study the application of SPS for flood monitoring in agriculture. Therefore the case we used is



```
- <SubmitRequestResponse xsi:schemaLocation="http://www.opengis.net/sps/1.0 http://schemas.opengis.net/sps/1.0.0/spsAll.xsd"
    xmlns="http://www.opengis.net/sps/1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gml="http://www.opengis.net/gml">
    <taskID>TID1359550280385973309397364 7</taskID>
    <status>confirmed</status>
    <gml:description>The task is feasibile.And 5 satellite sensors can be tasked. They are: SAR-2000_COSMO-SkyMed 3, SAR-X_TanDEM-X,
      SAR-X_TerraSAR-X,SAR_RADARSAT-2 and Serverjanin_Meteor-M 1</gml:description>
  - <LatestResponseTime>
    - <gml:TimeInstant>
        <gml:timePosition>2013-01-29T20:56:20.430+08:00</gml:timePosition>
      </gml:TimeInstant>
    </LatestResponseTime>
  </SubmitRequestResponse>
```

Fig. 4. Submit response from SPS

flood monitoring to valid the scalable GetFeasibility service for enhancing the EO SPS.

A. *Experiment Design*

It is estimated that Wu Han city, China will suffer a long term rain from 1/29/2013. In order to monitor the potential flood to avoid the loss of agricultural in Wu Han area, we use SPS to submit an EO task. According to the design and implementation discussed before, we will submit a planning request to 52nSPS as shown in part B of Section III. The "space" parameter in request is (29, 112, 33, 116.6), (29,112) is the latitude and longitude of lower corner about the tasking area, and (33, 116.6) represents upper corner. The "starttime" parameter is "2013-01-30T00:00:00+08:00" and "endtime" is "2013-01-30T23:59:00+08:00". "+08:00" is an offset compared with Universal Time Coordinated (UTC). And the "theme" parameter is "flood monitoring".

B. *Experiment Operation Flow and Result*

After the user send the submit request to 52nSPS by using the default client, 52nSPS will judge the type of request and dispatch different requests to different handlers. Then SPS will parse all parameters described above and does some syntax check. After that, SPS will send a GetFeasibility request contained all parameters to GetFeasibility service according to the WSDL described in part C of Section III. The mapping will start to work after receiving needed parameters.

The sectional result of Mapping 1 is like "sensor_satellite" : "AMR_JASON 2, JMR_ JASON 2, Aquarius_SAC-D, HSC_SAC-D, KMSS_Meteor-M 1, MSU-MR_ Meteor-M 1, MTVZA_Meteor-M 1, NIRST_SAC-D, SAR-2000_COSMO-SkyMed 3, SAR-X_TanDEM-X, SAR-X_TerraSAR-X, SAR_RADARSAT-2, Serverjanin_Meteor-M 1, VIRS_TRMM, ASTER_TERRA, ETM+_Landsat-7, HiRI_Pleiades 1A, KMSS_Meteor-M 1, MODIS_TERRA, MRI_NigeriaSat-X, MSC_KOMPSAT-2, MSU-MR_Meteor-M 1, MTVZA_Meteor-M, SAR-2000_COSMO-SkyMed4, Serverjanin_Meteor-M 1 …", and it means in the period of task, all of the above sensors can pass the target area.

Then Mapping 2 will determine how many sensors among them can be tasked to observe "flood monitoring" with the help of Sensor Capability Database. The sectional result is "AMR_JASON 2, JMR_ JASON 2, Aquarius_SAC-D, HSC_SAC-D, KMSS_Meteor-M 1, MSU-MR_ Meteor-M 1, MTVZA_Meteor-M 1, NIRST_SAC-D, SAR-2000_COSMO-SkyMed 3, SAR- X_TanDEM-X, SAR-X_TerraSAR-X, SAR_RADARSAT-2, Serverjanin_Meteor-M 1, VIRS_TRMM, ASTER_TERRA, ETM+_Landsat-7 …".

Mapping 3 is based on the result of Mapping 2. Mapping 3 will check the type of sensors, so radar sensors will be the eligible sensors, while optical sensors will be examined in the next step. The first thing is to convert the target area encoding in latitude and longitude to its political name. The main result is "Wu Han city, Hu Bei province, China". Then we get a weather report in "Wu Han", and the result fragment extracted is "30/1 cloudy to showers; 31/1 moderate rain to light rain; 1/2 overcast to cloudy". So all optical satellite sensor were filtered out because they do not have a good detection capability about capturing a qualified image during the task period. At last, mapping finished, and the GetFeasibility response contained the eligible sensors was send to SPS. Finally, the submit response was send to user client, which is shown in Fig. 4.

V. DISCUSSION

A. *Scalable to Be Extended*

Compared to the existing GetFeasibility operation, our service is flexible to be extended to meet more needs in EO task.

The Mapping can be extended. As shown in Section III, three mappings are used, while we can add more mappings to satisfy varied requirements. For example, we can add a mapping named Mapping 4 in the above GetFeasibility: horizontal spatial resolution must be fine enough to have a clear look at farmland. Then Serverjanin sensor on Meteor-M-N1 will be filtered out because its resolution is 400-500m or 700-1000m based on two operation modes. We can add similar mappings easily in various agriculture application scenarios.

The key is that, we treat GetFeasibility as a service cooperates with SPS instead of an operation inside SPS. By releasing development limitations imposed by SPS framework, we take it out as an independent service component. So the independent service approach we take make sure feasibility check can be more easily to be promoted according to changing needs.

B. Reliable to Enhance SPS in EO Tasks

What we do in this paper is to emphasize the GetFeasibility service in EO tasks. Planning satellite sensors is much more complex and necessary unlike in-suit sensors. We have to check every factor in tasking period which may has a significant impact on the quality of captured images. Therefore, we have considered the unique part in EO tasks. SpaceTime capability, Observation capability and Detection capability demonstrate what we have to concern in EO tasks which have been defined in part B Section II. So it is reliable to provide feasibility check to enhance SPS in EO tasks.

At the same time, the communication between GetFeasibility and SPS is through standard web service, which means as long as the service is ready, users from all over the world can make use of it in the same and universal way and form SOA architecture in future.

VI. Conclusion And Future Work

To design and implement a scalable GetFeasibility service to enhance EO SPS is a great challenge. Although the standard of SPS 2.0 tells us more about how to do a feasibility check compared 1.0, we still need an innovative and feasible instance. This paper proposes a novel mapping mechanism based on standard web service to achieve a scalable GetFeasibility service. The service architecture, especially the mapping and the implementation is illustrated in detail in Section II and III to get a more scientific and logical GetFeasibility result in EO tasks. In brief, the proposed GetFeasibility service is a brand new approach to enhance SPS for EO task.

The next step is to study how to get a real-time weather predict to support real-time SPS and a more precise and complete satellite sensor database is needed to be established.

Acknowledgment

This work has been supported in part by the National Basic Research Program of China under Grant 2011CB707101, by the National Natural Science Foundation of China under Grant 41171315, by the program for New Century Excellent Talents in University under Grant NCET-11-0394, and by National High Technology Research and Development Program of China under Grant 2013AA01A608.

References

[1] L. Di, “Geospatial Sensor Web and Self-adaptive Earth Predictive Systems(SEPS),” ESTO/AIST Sensor Web PI Meeting, San Diego, California,USA, Febrary 13-14, 2007.

[2] M. Botts, G. Percivall, C. Reed, and J. Davidson, “OGC sensor web enablement: overview and high level architecture (OGC 07-165)”, Open Geospatial Consortium Inc., 2007.

[3] I. Simonis and P. Dibner, “OpenGIS sensor planning service implementation specification (OGC 07-014r3)”, Open Geospatial Consortium Inc., 2011.

[4] T. Usländer, Y. Coene and P. G. Marchetti, Heterogeneous Missions Accessibility, ESA Communications: European Space Agency, April, 2012.

[5] B. Lawrence, M. Pritchard and A. Woolf, “Review of the heterogeneous mission accessibility Project”, [Online] Available: http://home.badc.rl.ac.uk/lawrence/blog/2007/03/14/review_of_the_esa_hma_project.

[6] HMA Architecture Website [Online]. Available: http://earth.esa.int/hma/architecture.html.

[7] EO-1 SPS Website [Online]. Available: http://eo1.geobliki.com/.

[8] Mandl, D. et al., “Sensor Webs with a Service-Oriented Architecture for On-demand Science Products” SPIE, San Diego, CA, August 26-30, 2007.

[9] X. Chu, T. Kobialk, B. Durnota, and R. Buyya, “Open sensor Web architecture: Core services,” in Proc. 4th Int. Conf. Intelligent Sens.Inf. Process., 2006, pp. 98–103.

[10] WIKI 52nSPS sensor plugins Website [Online]. Available: https://wiki.52north.org/bin/view/SensorWeb/SpsPlugins.

[11] H. Bredel. “Sensor web enablement howTo for developing plugins for a 52°North SPS version 1-00-00”.

[12] Z. Chen, N. Chen, L. Di, J. Gong, “A flexible data and sensor planning service for virtual sensors based on web service”. IEEE Sens J. vol. 11, pp. 1429–1439, June 2011.

[13] B. Zhang, L. Di, G. Yu, H, Wang, “Implementation of a prototype system for data and sensor planning service” 17th International Conference on Geoinformatics, Fairfax, USA, August 12-14, 2009.

[14] CEOS EO handbook Website [Online]. Available: http://database.eohandbook.com.

[15] WMO database Website [Online]. Available: http://www.wmo-sat.info/oscar/.

[16] M. Botts and A. Robin, “OpenGIS Sensor Model Language (SensorML) implementation specification,” Open Geospatial Consortium (OGC), Wayland, MA, OGC Document, 2007, pp. 1–180.

[17] Google Map API Website [Online]. Available: https://developers.google.com/maps/documentation/geocoding/?hl=en.

[18] Apache CXF WebSite [Online]. Available: http://cxf.apache.org/.