Service oriented architecture for scientific analysis at W7-X. An example of a field line tracer.

S.A. Bozhenkov*, J. Geiger, M. Grahl, J. Kißlinger, A. Werner, R.C. Wolf

Max-Planck-Institut für Plasmaphysik, EURATOM Association, Teilinstitut Greifswald, Wendelsteinstr. 1, 17491 Greifswald, Germany

Abstract

Service oriented architecture based on web-services is a universal method of combining software components. SOAP web-services chosen for W7-X are characterized by strong standards and readily available tools. In this paper the SOAP technology is explained and is illustrated with a new service for field line tracing.
The field line tracing package consists of a C++ library and a web-service interface. It features a flexible structure and can handle a realistic machine geometry. The following problems can be solved: getting a field line; making Poincaré maps; calculating flux surface characteristics; calculating heat fluxes to the wall; constructing magnetic coordinates, etc. The service is applied to estimate W7-X divetor loads with an 1/1 error field.

Keywords: SOA, SOAP, web-service, integrated modeling, field line tracing, island divertor, W7-X

1. Introduction

Wendelstein 7-X [1] is an optimized modular stellarator to come into operation by the end of 2014. Equipped with superconducting coils and an island divertor [2] it has to demonstrate quasi-stationary discharges of up to 30 min [3]. The machine complexity, pulse duration, high resolution diagnostics and more elaborated plasma models put new requirements on analysis tools. Many problems can be addressed only with multiple measurements and

*Corresponding author
Email address: boz@ipp.mpg.de (S.A. Bozhenkov)

models. A universal method of combining heterogeneous analysis software is, therefore, desirable. In addition, because of increasing collaborations, the analysis tools must be accessible remotely. Service oriented architecture based on web-services addresses the stated problems. In this paper the web-services concept is demonstrated with a new W7-X service for field line tracing.

Field line tracing is a basis for analysis of stellarator experiments. Therefore, it is one of the first W7-X services. The tracer solves several problems, apart from producing a line polygon: Poincaré maps, connection length, construction of magnetic coordinates, heat fluxes to the plasma facing components, spectra of error fields, etc. The problems are solved with a realistic machine geometry represented by a polygon mesh. To illustrate the service, calculations of W7-X divertor loads with an 1/1 error field are presented.

The idea of using web-services for scientific computing emerged a few years ago [4] and has been widely adopted [5, 6, 7, 8]. In fusion, SOAP web-services were proposed earlier: for Bayesian data analysis [9], for distributed applications [10], for W7-X data access [11], as a W7-X analysis framework [12]. Nevertheless, this approach is relatively new and a detailed discussion is missing.

Service oriented approach at W7-X has to provide a uniform access to analysis codes for building complex models. Critical parts of such models can be parallelized in a private Cloud. Web-services offer the following advantages: extensive standards; platform/language independence; off the shelf software; reusability of existing codes; loose coupling; individual services are developed by field experts; convenient usage.

An independent attempt of the fusion community to address the same problems is Integrated tokamak modeling (ITM). ITM is to produce a comprehensive simulation package for ITER [13] with both the hardware and the software, including the middleware like Consistent Physical Objects (CPO) and universal access layer (UAL) [14]. Integration of scientific models in ITM is centered around the Kepler workflow system [15]: an existing model is wrapped as a Kepler *actor*. A scientific workflow is a series of connected computational tasks composed together to achieve a final aim. Workflows are usually composed in a graphical form in an application called a work-

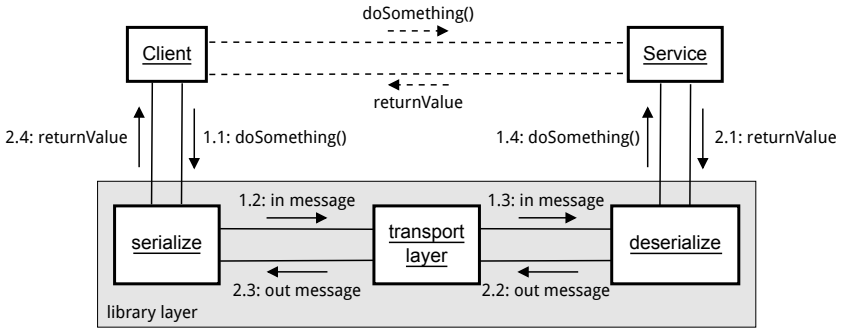


Figure 1: Client-server communication. The developers have an impression of a direct function call, dashed lines. In reality the call passes through the library layer, where it is serialized, transported and deserialized.

flow management system. In the Kepler system, computational blocks are known as actors. A workflow example is given later in figure 3. In contrast to ITM, W7-X does not focus on a particular workflow management and the middleware is commercially available. Here a model is made available as a service, which is accessible in a variety of ways, including the Kepler system.

The rest of the work is organized in the following way: section 2 explains the basis of the service oriented approach and of SOAP/WSDL web-services; section 3 gives details of the new W7-X field line tracer; in section 4 the tracer service is applied to estimate W7-X divertor loads with an 1/1 error field; and, finally, section 5 summarizes our results.

2. Web Services

Service oriented architecture (SOA) is a paradigm for building complex software by combining independent components - services. Services are invoked by sending language neutral, structured messages. Functions provided by a service are described in a language neutral interface document; and, hence, can be automatically discovered and used. Services may reside on different machines, which provides parallelization.

Modern technologies make applying SOA trivial. For example, a service call can be illustrated by figure 1. Given suitable tools, the client and the

service developers have an impression of a direct function call in their programming languages, as shown with dashed lines. In reality, thin libraries take care of forming and sending proper messages: the call in figure 1 follows path 1, and the response goes along path 2.

If services communicate via HTTP, the transport box in figure 1, they are called web-services. Having web-services simplifies the remote access via the Internet. There are several web-service standards: SOAP [16], REST [17], JSON-RPC [18], etc. If the requirement of a direct HTTP support is relaxed other solutions, e.g. Apache Thrift [19], can also be considered.

In REST (Representation State Transfer) [17] the HTTP vocabulary is used to perform a data (state) manipulation on the server and to receive back a representation, i.e. any data related information. The operations are limited to *GET, POST, PUT* and *DELETE*, while a problem is encoded in unified resource identifiers, e.g. `http://myserver/magneticField` for calculating magnetic field. Because of the lack of an interface description, this approach can be cumbersome for scientific interfaces [20, 21]. Besides, a data transfer format is a part of every interface.

JSON-RPC [18] is a protocol based on JSON (JavaScript Object Notation) [22] for object serialization. Serialization is conversion of an object state into a form that can be transfered and used to recreate the state by deserialization. In JSON-RPC, a request message sent to a service contains a method name, list of input arguments and an id. The response consists of a result object, error message and the id. JSON combines a small, as compared to XML, message size, fast array processing, and human readability. Presently this standard lacks an industry support and there are no examples of complex scientific applications.

SOAP [16] is an industrial standard based on XML [23]. Considered broadly it includes at least two parts: (i) SOAP for the message; and (ii) WSDL for the interface description. It has readily available libraries, workflow and bus solutions; and an automated function discovery via WSDL. SOAP operations are remote procedure calls (RPC), i.e. they are similar to local functions but the actual processing is done on a server. For these reasons and because of its successes in scientific applications [5, 6, 7], SOAP is presently considered for W7-X. In the rest of this paper web-services are

understood as SOAP web-services.

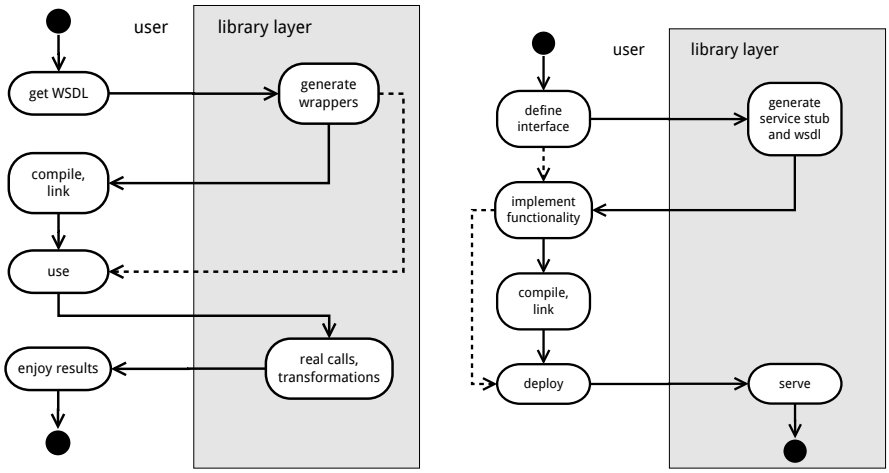
SOAP standard [16] describes request and response message structure. A valid message consists of an envelope, and a header and a body inside it. The body contains the data payload and may include a fault. The data are also passed as XML, which provides a flexibility for complex structures, but may have a performance impact [24, 25]. The standard does not define the payload, viz. a service specifies its data structures and operations via WSDL.

WSDL [26] standardizes service interface description and allows an automatic discovery and reuse. It is a WSDL document that is exchanged between a service developer and a user. There are five sections in a WSDL document, as of version 1.1: types - definition of data structures; messages - input and output messages for the operations; port type - exposed operations, where every operation is connected to its messages; binding - service transport protocol; and service - list of defined services. New data types are introduced in a XML Schema [27]. A XML Schema is a XML document describing structure of other documents. The SOAP data payload is described here as input and output messages. The common way of composing a message is document/literal, where all arguments are attached to a single structure that can be validated.

2.1. Using web-services

SOA supports a separation of responsibilities. A service is developed by an expert with his tools of choice. Since the standards are platform and language independent, later anyone can use this functionality from a programming environment or from a workflow management system.

A service use pattern from a programming environment is illustrated in figure 2a. Given a WSDL document, client tools generate wrappers, which are local functions handling service calls but appearing to be simple functions for the user. After compiling and linking, the client SOAP library performs message serialization, deserialization and passing. For dynamic languages the compilation and the linking are not required, as shown with a dashed line. Such a usage pattern is illustrated later in listing 4.



(a) Usage pattern. A WSDL document is supplied to generate call wrappers. Wrappers are used in a normal way.

(b) Development pattern. From the interface the tools generate service boilerplate code and a WSDL document. The developer fills in the logic.

Figure 2: Service development and usage. Dashed lines show shortcuts for dynamic languages.

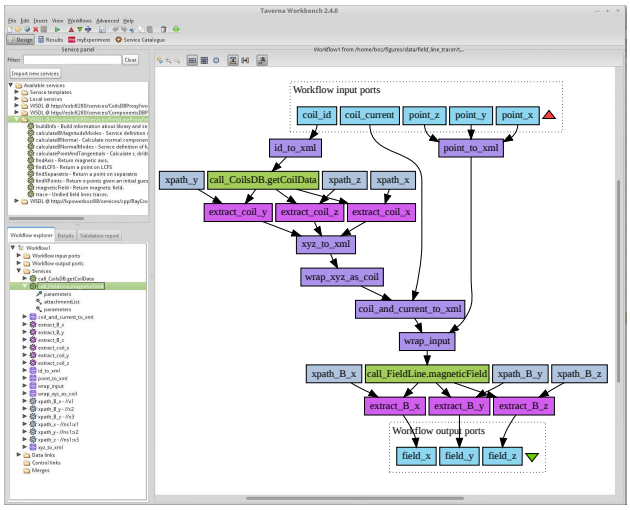


Figure 3: Using web-services in Taverna [28]. This example extracts the coil geometry from a database and calculates its field at a point. Two service calls are shown in green. The other operations are to form the input. The colours of the elements are arbitrary.

Listing 1: Web-service interface declaration with gSOAP.

```
class ns1__MagneticConfig
{
    public:
        std::vector<ns1__PolygonFilament > coils;
        std::vector<double > coilsCurrents;
};
//Return a point on LCFS
int ns1__findLCFS(double step, ns1__MagneticConfig *config, ns1__Machine *machine,
                  ns1__LCFSSettings *settings, ns1__LCFSResponse &response);
```

Web-services can also be used from a scientific workflow management system [29] like Kepler [15], Taverna [28], etc. In such a system a problem is solved in a graphical form, with operations being depicted as blocks. In figure 3 magnetic field at a point is calculated by using two web-services in Taverna. Coil geometry is first extracted from a database by *call_CoilsDB.getCoilData* and afterwards the field is calculated by *call_FieldLine.magneticField*. The input data include the coil id, coil current and the point. The output consists of the three field components. The operations in between the service calls are data transformations provided by Taverna. The service operations were automatically discovered by Taverna from the WSDL documents.

Still another option for using web-services is to employ the industry workflow standard BPEL [30], which brings off the shelf software like GUI designers and run-time engines, but has shortcomings concerning dynamic resource, long runs and large data packages [8].

2.2. Developing web-services

There are two approaches in developing a web-service. The first path is a WSDL centric one: a WSDL is created ab initio and is used to produce a boilerplate code for the service. In the second one the interface is defined in the developer's programming language. It is the latter that is illustrated here, since it does not require understanding XML or WSDL.

Service development pattern is illustrated in figure 2b. A developer defines the interface in his programming, or very close to that, language. From these definitions library tools generate boilerplate code for the service and its WSDL document. Afterwards, the developer has to fill the auto-generated code with the logic and to install the service. The SOAP library takes care

Listing 2: WSDL corresponding to listing 1

```
1  <complexType name="MagneticConfig">
2    <sequence>
3      <element name="coils" type="ns1:PolygonFilament" minOccurs="0" maxOccurs="unbounded"/>
4      <element name="coilsCurrents" type="xsd:double" minOccurs="0" maxOccurs="unbounded"/>
5    </sequence>
6  </complexType>
7  <element name="findLCFS">
8    <complexType> <sequence>
9        <element name="step" type="xsd:double"/>
10       <element name="config" type="ns1:MagneticConfig" nillable="true"/>
11       <element name="machine" type="ns1:Machine" nillable="true"/>
12       <element name="settings" type="ns1:LCFSSettings" nillable="true"/>
13     </sequence> </complexType>
14 </element>
15 <message name="findLCFS">
16   <part name="parameters" element="ns1:findLCFS"/>
17 </message>
18 <operation name="findLCFS">
19   <documentation>Return a point on LCFS</documentation>
20   <input message="tns:findLCFS"/>
21   <output message="tns:LCFSResponse"/>
22 </operation>
```

of calling a requested function and of serializing/deserializing the messages. The service logic usually consists of calls to an underlying library.

An example of the discussed pattern is given in listings 1, 2, 3 with an interface definition, WSDL and an implementation for gSOAP C++ library [31]. In listing 1 a magnetic configuration class and a method for identifying the last closed flux surface are declared. A configuration consists of a number of filaments and their currents. The method *findLCFS* accepts an integration step, configuration, machine geometry, settings and an output object. The declarations are written with C++ syntax, with *ns1__* being an XML namespace. An extract of the auto-generated WSDL document is presented in listing 2. Lines 1 – 6 correspond to the magnetic configuration class, lines 7 – 17 concern the input message of the operation *findLCFS*, and the operation itself is introduced in lines 18 – 22. gSOAP tools also generated boilerplate code for the service, whereto the logic was introduced, as shown in listing 3. Most of the *findLCFS* function deals with reforming the data, lines 5-9 and from 13, while the processing is implemented by a library call on line 10.

A web-service is to be installed behind an HTTP server either on a usual workstation or on a more sophisticated system like a cluster, Grid or a Cloud. The latter cases allow parallelization with a commercial software like an enterprise service bus (ESB). An enterprise service bus provides a single access

Listing 3: Web-service method implementation.

```
int srv::FieldLineService::findLCFS(double step, ns1__MagneticConfig *config,
    ns1__Machine *machine, ns1__LCFSSettings *settings, ns1__LCFSResponse &response)
{
    try{
        FieldFactory field_factory(config); //field configuration parser
        SceneFactory scene_factory(machine); //scene parser
        const AbstractField &field = field_factory();
        Scene *scene = scene_factory();
        ...
        ::findLCFS(left, right, settings->LCFSNumPoints, step, field, scene,
                settings->LCFSAccuracy, settings->LCFSThreshold);
        response.result.x = left[0];
        ...
    } catch (std::exception &e) { ... }
    return SOAP_OK;
}
```

point to all services, thus reducing the system complexity. It can perform message transformation and routing, load balancing between servers, etc.

3. Field line tracing

Field line tracing is a basic task for stellarator experiments; and, therefore, it is the first tool implemented as a W7-X web-service. Up to now such a functionality was only available in expert codes like GOURDON [32]. The web-service interface makes it available to a larger community. The new package includes a C++ library and a web-service interface. The web-service layer was developed with gSOAP library [31].

3.1. Library

The library integrates the field line equation:

$$\frac{d\vec{r}(s)}{ds} = \vec{F}(s,\vec{r}) = \frac{\vec{B}}{|\vec{B}|} \tag{1}$$

with the initial condition $\vec{r}(0) = \vec{r}_0$. To provide flexibility and extendability the integration is organized in a way presented in figure 4. Equation 1 is solved by a tracer, from Adams or Runge-Kutta family, that requires the following parameters: *point* - the initial condition; *step* - an integration step; *field* - the right hand side of the equation; *stat* - an object collecting statistics along the line and defining a stop criterion; *modify* - a field line modifier,

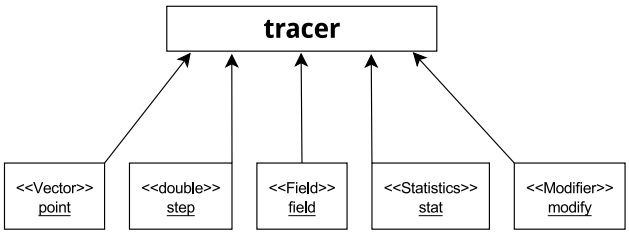


Figure 4: Scheme of the field line tracing library. A tracer solves the field line equation by engaging abstract interfaces of objects *field*, *stat* and *modify*. The supplied objects define the problem to be solved.

e.g. diffusion. The objects for field, statistics and modification are used via abstract interfaces, i.e. virtual C++ functions. Presently two implementations of the magnetic field interface are available. The field can be either computed by Biot-Savart law for filaments or interpolated with a tri-linear scheme, e.g. for an equilibrium.

A statistics object defines a problem to be solved. The tracer calls this object with the current location and expects *true* to continue the integration or *false* to stop. The following problems were implemented: field line polygon; Poincaré map in a $\varphi = const$ plane; Poincaré map in a plane given by normal and point; flux surface values like effective radius r_{eff} and rotational transform ι; connection length; heat fluxes to the wall; construction of magnetic coordinates; finding magnetic axis, separatrix, LCFS. Most of these problems are straightforward. To build a Poincaré map the field line is tested for a plane intersection. The rotational transform ι is calculated from the ratio of travelled angles (see also section 3.4). The magnetic axis is identified as a point that returns to itself after one period; similarly, X-points are found by minimizing the distance after m toroidal turns, where m is the poloidal mode number. But heat fluxes to the wall and magnetic coordinates are more involved and, hence, are explained in detail in sections 3.3 and 3.4. Besides the problems solved during the tracing, there are convenience methods to calculate a flux surface, normal field B_n, and its modes (section 4).

The library supports a realistic stellarator geometry by using polygon meshes. At every integration step the field line can be tested for an intersection with the wall. To accelerate the testing the space is partitioned into a

regular grid, with every grid cell keeping a list of enclosed polygons. During the testing only the polygons in the cells covered by the line segment are considered. As a result, the intersection testing has a negligible slow down, i.e. well below 10 %, even for meshes with millions of triangles.

3.2. Web services

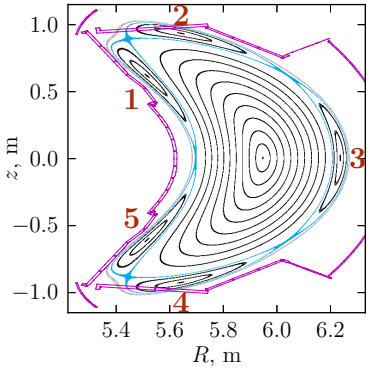
The web-service interface exposes the library functionality. Starting from the WSDL, this functionality is accessible from multiple languages and platforms. Some details of the service implementation were given in section 2.2. All the results shown hereafter were obtained by using the service.

In addition to the field line tracer a few complementary services were made available. They include: *CoilsDB* - a database for coil geometries; *ComponentsDB* - a database for wall meshes, like baffles, targets, etc.; *Biot-Savart* - calculation of magnetic field by Biot-Savart; *MeshProcessor* - ray and plane crossing with polygon meshes; and *FieldLine* - field line tracer. The database services *CoilsDB* and *ComponentsDB* are to relieve an end user from manipulating machine details for every call: the other services can accept database references. For example, the W7-X coil system consists of 50 non-planar coils, 20 planar coils, 10 control coils and 5 trim coils, where each coil has own unique errors. Presently the both databases are implemented in Python with ZODB [33] and soaplib [34].

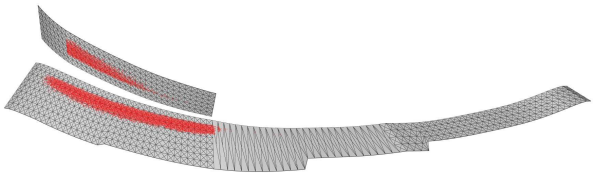
The web-services are installed in a virtual environment in a private Cloud. Presently they are CGI applications behind an HTTP-server. The test Cloud totals to 48 CPU cores and 1 TB of memory. The services in the Cloud are accessed via an ESB, which performs load-balancing, parallelization and caching.

3.3. Heat fluxes to plasma facing components

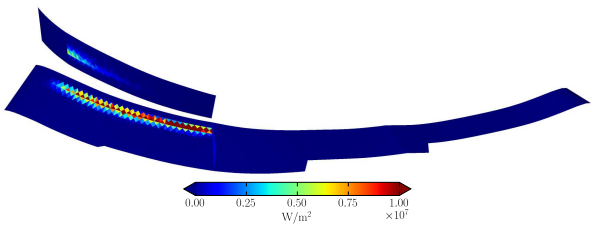
To estimate heat fluxes to the plasma facing components a Monte-Carlo diffusion model proposed in [2] was realized. In this model field lines from random points from inside the separatrix are followed and once in a while random steps are performed to simulate the perpendicular transport. This process is continued until a field line terminates on the wall or a length limit is reached. The resulting hit points represent the strike lines and from the



(a) Poincaré map and the first wall in $\varphi = 0$ plane. The 5/5 island chain is marked.



(b) Hit points, in red, on the target. The black lines demonstrate the used triangulation.



(c) Divertor power loads in false colours.

Figure 5: Divertor loads for the W7-X standard configuration. Only parts of one of 10 identical divertor units are shown.

number of events per polygon the power loads can be estimated.

The field line diffusion corresponds to the modification object in figure 4. The parameters to describe the process are: perpendicular diffusion coefficient $D_\perp$, mean free path λ, and velocity v. The "transport" steps are performed after a random length x with the distribution:

$$p(x) = \frac{1}{\lambda} \exp\left(-x/\lambda\right) \tag{2}$$

The direction of the steps is uniform in the plane perpendicular to the field, while the step size r is uniform in the interval:

$$r \in \left[0, \sqrt{\frac{12 D_\perp \lambda}{v}}\right], \tag{3}$$

so as to average to the correct 2d diffusion. The diffusion operator does not reverse the parallel direction; consequently, the calculations have to be repeated with the inverse magnetic field.

An example of heat fluxes to the W7-X divertor is shown in figure 5. Figure 5a presents a Poincaré map for the used magnetic configuration in plane $\varphi = 0$. In this configuration a 5/5 island chain is used for the divertor. The target plates are in contact with two islands at the top and two islands at the bottom, and are combined into 10 identical divertor units [35]. Since for the ideal case all 10 units are loaded symmetrically, figure 5 contains only parts of a single unit. Hit points are shown in red in figure 5b, on the left hand side both on the horizontal and the vertical targets. These hit points correspond to the power loads in figure 5c, under the assumption of 10 MW power going through the separatrix and negligible plasma radiation. The highest power is concentrated on the horizontal target, with the maximal value being of the order of 10 MW/m^2. The fragmentation of the loads is due to triangulation. The presented results are for a perpendicular diffusion coefficient of 1 m^2/s, and a free length and velocity corresponding to electrons at 10 eV.

Inspite of neglecting drift effects and radiation losses, the model is a basis for the design and a preliminary analysis of the W7-X divertor [2]. The importance of the other effects can be deduced by comparison with the experiment. Finally, this model can be parallelized in the W7-X Cloud.

3.4. Magnetic coordinates

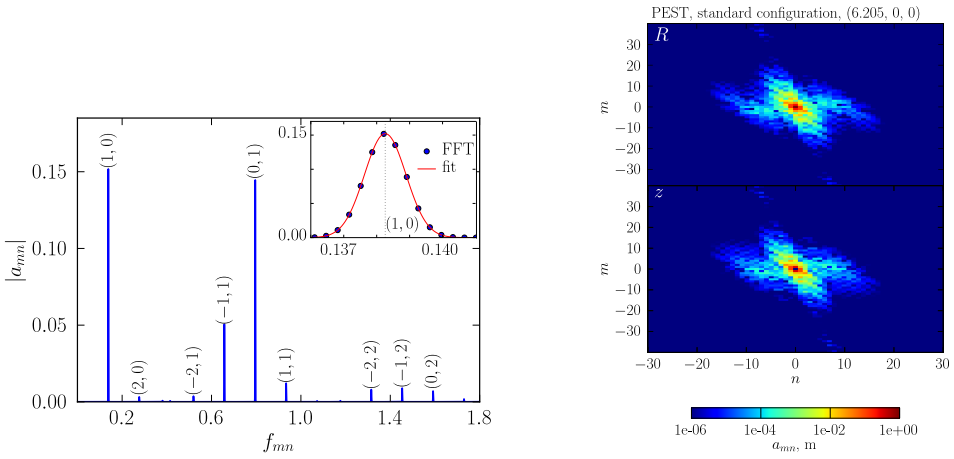
Magnetic coordinates describe a whole flux surface and can be applied for visualization, field line penetration studies, calculating error field spectra, etc. They are curvilinear coordinates (ρ, θ, φ) with straight field lines [36, par. 6.1]:

$$\vec{B} = \psi' \cdot \vec{\nabla}\rho \times \vec{\nabla}\theta + \iota\psi' \cdot \vec{\nabla}\varphi \times \vec{\nabla}\rho, \tag{4}$$

where ι is rotational transform and ψ' is the radial derivative of the normalized toroidal flux $\psi = \Psi/2\pi$. Such a system is not unique and the library constructs three common ones: PEST [37], Hamada [38] and Boozer [39]. The problem is to find expressions for poloidal and toroidal angles θ and φ.

To construct a magnetic system, cylindrical coordinates on a flux surface are decomposed into double Fourier series $\exp i(m\theta + n\phi)$, which along a field line $\theta = \iota\varphi$ reduces to:

$$\{R, z\} \quad = \quad \sum_{mn} \{R_{mn}, z_{mn}\} e^{i(m\iota + \mathfrak{s}n)\varphi} \tag{5}$$



(a) 1d power spectrum of z coordinate in the PEST system. The modes (m, n) are identified by frequency $f_{mn} = (m\iota + \mathfrak{s}n)/2\pi$. The insert shows an enlarged view of the $(1, 0)$ peak.

(b) Power spectra of R_{mn} (top) and z_{mn} (bottom) modes for the PEST coordinate system. m - poloidal mode number, n - toroidal mode.

Figure 6: Construction of magnetic coordinates.

$$\varphi_g = \varphi + \sum_{mn} \varphi_{mn} e^{i(m\iota + \mathfrak{s}n)\varphi}, \tag{6}$$

where $\mathfrak{s}$ is device symmetry, 5 for W7-X. The mode amplitudes can be reconstructed during the field line tracing from FFT [40], if φ is known. For the PEST system φ is the geometrical angle φ_g, and for two other systems it is found as an integral [41, eq. 35], [40, eq. 6]:

$$\varphi = 2\pi \frac{d\Psi}{dV} \int \frac{dl}{|B|} \tag{7}$$

$$m\theta + n\mathfrak{s}\varphi = \frac{m\iota + n\mathfrak{s}}{I_{pol} + \iota I_{tor}} \cdot \int |B| dl \tag{8}$$

The variables to be transformed in equations 5, 6 are usually not periodic along a field line. To overcome this and to have an error estimation a Gaussian windowing $1/\sqrt{\pi}\sigma \cdot \exp\left(-\left(\varphi - \bar{\varphi}\right)^2/\sigma^2\right)$ is used [42]. Here $\bar{\varphi} = max(\varphi)/2$ is the average angle of a traced line and the width σ is found as $\bar{\varphi}/K$, with K being a free parameter about 4 in practice. The line length

$max(\varphi)$ is chosen so as to limit the cross-contributions of neigbouring peaks in the frequency space below ε:

$$max(\varphi) \geq \frac{2K}{\pi min(\delta f_{mn})} \cdot \sqrt{\log \frac{\varepsilon}{2}} \tag{9}$$

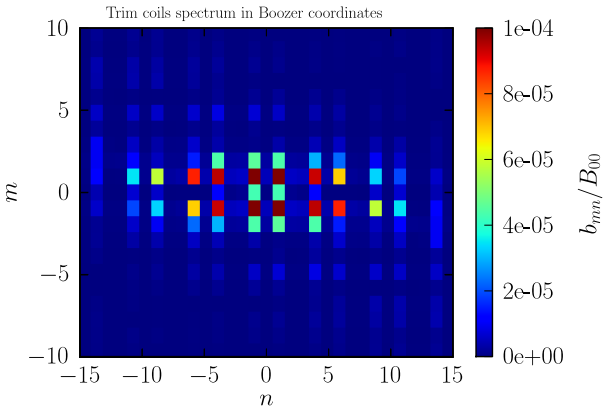
The limit ε is set to the error from cutting the Gaussian at the edges, which appears because the area under such a Gaussian is different from 1:

$$\varepsilon = \frac{1}{\sqrt{\pi}K} e^{-K^2} \tag{10}$$

The amplitudes of the Gaussians in the frequency space are equal to the desired mode amplitudes and can be determined by fitting.

Mode identification for all three coordinate systems assumes knowledge of the rotational transform ι. In addition, Hamada and Boozer systems require $d\Psi/dV$ and I_{pol}, I_{tor} correspondingly. These values are determined as $(0,0)$ modes of: derivative of geometrical angles $(d\theta_g/d\phi_g)_{00}$ for ι [43]; derivative of specific volume $U = \int dl/B$ over toroidal angle $2\pi\,(dU/d\varphi_g)_{00}$ for $dV/d\Psi$ [44]; covariant field component $(B_\theta)_{00}$ for the toroidal current inside the surface I_{tor}; and covariant field component $(B_\varphi)_{00}$ for the poloidal current outside of the surface I_{pol}. The currents are found from the covariant components in the PEST system.

Construction of PEST coordinates is illustrated in figure 6. Figure 6a presents the power spectrum of z-coordinate FFT along a field line, with marked modes (m,n), where m is the poloidal and n is the toroidal mode. Given the rotational transform ι, the modes were identified by their frequencies $f_{mn} = (m\iota + \mathfrak{s}n)/2\pi$. An insert demonstrates a single peak $(1,0)$, which exhibits Gaussian shape due to the windowing. Mode amplitudes are measured by fitting such peaks. The result of the mode finding is demonstrated in figure 6b as a 2d power spectrum for both R_{mn} and z_{mn}. The poloidal mode number m changes along the vertical axis, while the toroidal mode number n along the horizontal one. This example is for a flux surface close to the separatrix.



(a) Power spectrum of the error field. m - poloidal mode number, n - toroidal mode number.

b_{11}/B_{00}	$2.3 \cdot 10^{-4}$
b_{22}/B_{00}	$5 \cdot 10^{-6}$
b_{33}/B_{00}	$8 \cdot 10^{-7}$

(b) Resonant modes.

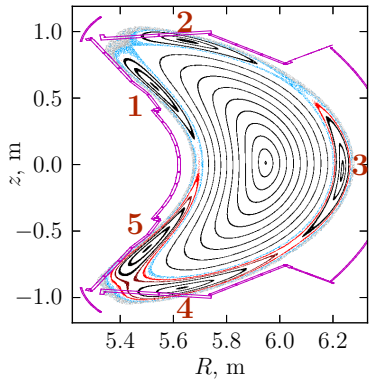
Figure 7: Error field due the chosen trim coil configuration. The spectrum is in Boozer coordinates, for a flux surface close to the separatrix. The amplitude of the current distribution in trim coils is 10 kA.

4. Application example: divertor overloading by 1/1 error field

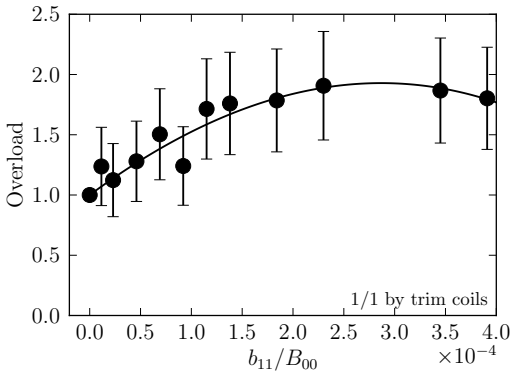
The W7-X divertor exploits an intrinsic island chain with fivefold symmetry [35]. In most cases, the rotational transform at the plasma edge is close to 1, which makes the divertor sensitive to symmetry breaking perturbations. Such perturbations appear because of manufacturing imperfections [45]. In this section the field line tracing service is applied to analyze modification of the divertor loads by an 1/1 perturbation.

In order to compensate possible field errors, five trim coils will be installed at W7-X [46]. These coils are designed to minimize the low order resonant modes from 1/1 to 4/4. In such an operational mode, the coils are fed so as to counteract to the existing perturbations. However, if the trim coil field is superimposed on the ideal configuration, the influence of a selected perturbation can be studied systematically. It is this latter option that was used here; furthermore, the currents in the trim coils were tailored for the 1/1 mode.

Figure 7a presents a power spectrum of the dedicated trim coil arrangement. The spectrum was calculated by a service method *FieldLine.calculateBNormalModes*



(a) Poincaré map in $\varphi = 0$ plane. A 1/1 structure, colour, appears at the edge.



(b) Divertor overloading as the ratio of the maximum heat flux with the 1/1 mode to that without. The errors are statistical. The fitting is to guide the eye only.

Figure 8: Influence of the 1/1 error field.

in Boozer coordinates for the normal field component B_n. The method algorithm is the following one. Given the Boozer coordinates, equations 5 and 6, a standard 2d FFT on a uniform grid in magnetic angles (θ, φ) is performed. The component B_n is determined by first finding the full $\vec{B}$ vector for the point (θ_i, φ_j) and a subsequent dot product with the surface normal $\vec{n} \sim \vec{e}_\theta \times \vec{e}_\varphi$. The contravariant basis vectors $\vec{e}_\theta$ and $\vec{e}_\varphi$ are found by taking derivatives of the coordinate Fourier series.

The trim coils produce a wide mode spectrum, but the relevant modes are those resonant to $\iota = 1$ (table 7b). For the case considered the 1/1 mode is dominant and dictates the edge modification (figure 8a). The induced 1/1 structure makes the divertor loads asymmetric; namely, some targets receive a load below the ideal case, while the others are overloaded. Figure 8b shows results of the field line diffusion simulations with a different amplitude of the perturbation. Shown is the ratio of the maximum heat flux with the 1/1 error field to that without; the error bars are statistical. Already a rather small 1/1 field of about $2 \cdot 10^{-4}$ results in a two times overloading of the target. This demonstrates the importance of minimizing the resonant error fields and also shows that the trim coils are a valuable tool.

Listing 4: Calling the web-services in Python. Poincaré map example.

```
1   from osa import Client
2   tracer = Client("http://.../FieldLine?wsdl")
3   compdb = Client("http://.../ComponentsDB?wsdl")
4   mesher = Client("http://.../MeshProcessor?wsdl")
5   points = tracer.types.Points3D() #start points
6   task = tracer.types.Task() # statistics object
7   config = tracer.types.MagneticConfig()
8   machine = tracer.types.Machine()
9   ...
10  machine.meshedModels = compdb.service.getComponentData([0,1])
11  poincare = tracer.service.trace(points, config, task, machine)
12  wall_cut = mesher.service.intersectPolygonPhiPlane(0, machine)
13  for surface in poincare.surfs:
14      x = array(surface.points.x1)
15      ...
16      plot(sqrt(x**2+y**2), z, ',k')
17  for sec in wall_cut.intersections:
18      x = array(sec.vertices.x1)
19      ...
20      plot(sqrt(x**2+y**2), z, '-m')
```

The analysis in this section was performed by using the developed services in Python with *osa* library [47]. For example, the Poincaré map in figure 8a was produced with a script shown schematically in listing 4. In line 1 the SOAP library is imported. The library creates call wrappers for the tracing service, components database service and the polygon intersection service in lines 2–4. In lines 5–8 the objects required for tracing are constructed. Further details of setting up the input are omitted here. In lines 10–12 the web-services are engaged. First the machine geometry is extracted and then it is used during the tracing and to get a cross section in $\varphi = 0$ plane. Finally, in lines 13 – 16 the Poincaré map is plotted and in the remaining part the machine cross section is displayed.

5. Conclusions

Both the fusion experiments and the analysis tools become increasingly complex. Nowadays many problems are addressed with multiple measurements and models. A universal method of combining heterogeneous analysis software is, therefore, desirable. In addition, because the experiments become increasingly collaborative, the analysis tools must be accessible remotely. In this article service oriented architecture based on SOAP web-services was discussed as a possible solution to these problems.

SOAP web-services rely on industry standards and have ready to use li-

braries and tools, which makes both development and use of services fast and flexible. Web-services are platform/language independent, allow reuse of existing codes, introduce loose coupling, make remote access transparent, and make parallelization in a Grid or in a Cloud possible.

In the SOAP scheme a service and its WSDL interface description are made available on the network. To use a service a user has to supply its WSDL to a client SOAP library or a workflow system to generate wrappers. The wrappers compose suitable SOAP messages, submit the messages to the service and receive the response. For the user calling the service is reduced to calling a local wrapping function.

Developing a web-service usually includes two steps: (i) developing a simulation library; and (ii) providing a service interface. In such a way the service interface remains decoupled. The service interface is usually defined in the developer's programming language. The developer's SOAP tools generate a WSDL document and boilerplate code for the server implementation. The generated code has to be filled with the desired logic. At the run time the SOAP library calls a requested method and performs the necessary conversions.

Service development was illustrated with a new W7-X service for field line tracing. The tracing package consists of a C++ library and a gSOAP C++ [31] service interface. It has a flexible structure and can handle a realistic machine geometry in a meshed form. The following problems can be solved: getting field line; making a Poincaré maps; finding flux surface characteristics; finding connection length; calculating heat fluxes to the plasma facing components with field line diffusion; constructing PEST, Hamada, or Boozer magnetic coordinates; calculating surface, normal field component B_n, and its Fourier modes; finding magnetic axis, separatrix, LCFS.

The tracer service was applied to estimate the W7-X target loads with a 1/1 perturbation produced by the W7-X trim coils [46]. A scaling of the divertor overload with the 1/1 amplitude was obtained. For an error field of about $2 \cdot 10^{-4}$ the maximal divertor load increased by a factor of about 2. This demonstrates the importance of minimizing the resonant error fields and also shows that the trim coils are a valuable tool to ensure the divertor operation.

The implementation of the tracer service also indicated where the SOAP performance can be improved. In scientific applications large datasets are usually generated and transfered in a service call [8]. For example, the tracer can accept a magnetic field array from an equilibrium solver or a large machine model. Large datasets cause CPU and memory penalties. For the client, the XML serialization of arrays has a large memory overhead and takes relatively long time. Under a load the central node like ESB may become a bottleneck [48].

To overcome the serialization problem, large data arrays can be transfered outside of the XML structure, while the SOAP message is still used to pass short parameters and function names. Arrays can either be stored externally [8], in a proxy or in a database, or can be represented in a binary form [24, 25]. In the former case the datasets are labeled with a unique identifier, which also helps to unload the central node [8, 49]. The disadvantage of these methods is that they are not standardized; hence, a proper serialization is to be implemented on the client and the server side. HTTP chunking could also be employed to reduce the memory footprint and the latency [24, 25]. A detailed performance analysis of the available W7-X services will show if any of the discussed measures are necessary.

6. Acknowledgements

The authors would like to thank Drs. T. Andreeva and M. Endler for the fruitful discussions of the paper and the help with the W7-X coils data.

References

[1] G. Grieger, C. Beidler, E. Harmeyer, J. Junker, J. Kißlinger, W. Lotz, P. Merkel, A. Montvai, J. Nührenberg, F. Rau, A. Schlüter, H. Wobig, R. Zille, Physics studies for helical-axis advanced stellarators, in: Plasma Physics and Controlled Nuclear Fusion Research, Proceedings of the 12th International Conference, Nice, 1988, volume 2, IAEA, Vienna, 1989, pp. 369 – 387.

[2] J. Kißlinger, C. D. Beidler, E. Harmeyer, F. Rau, H. Renner, H. Wobig, Island divertor for the stellarator Wendelstein 7-X, in: Controlled Fusion

and Plasma Physisc, Proceedings of 21st European Conference, Montpellier, 1994, volume 18B, part 1, European Physical Society, Geneva, 1994, pp. 368 – 371.

[3] H. Renner, J. Boscary, V. Erckmann, H. Greuner, H. Grote, J. Sapper, E. Speth, F. Wesner, M. Wanner, The capabilities of steady state operation at the stellarator W7-X with emphasis on divertor design, Nuclear Fusion 40 (2000) 1083 – 1093.

[4] I. Foster, Service-oriented science, Science 308 (2005) 814 – 817.

[5] E. Zudilova-Seinstram, N. Yang, L. Axner, A. Wibisono, D. Vasunin, Service-oriented visualization applied to medical data analysis, Service oriented computing and applications 2 (2008) 187 – 201.

[6] S. W. Berrick, G. Leptoukh, J. D. Farley, H. Rui, Giovanni: A web service workflow-based data visualization and analysis system, IEEE Transactions on Geoscience and Remote Sensing 47 (2009) 106 – 113.

[7] R. Mak, J. Walton, L. Keely, D. Heher, L. Chan, A reliable service-oriented architecture for NASA's Mars exploration rover mission, in: IEEE 2005 Aerospace Conference, Big Sky, MT, 2005, pp. 1 – 14.

[8] A. Bosin, N. Dessi, B. Pes, Extending the SOA paradigm to e-Science environments, Future Generation Computer Systems 27 (2011) 29 – 31.

[9] J. Svensson, A. Dinklage, J. Geiger, A. Werner, R. Fischer, Integrating diagnostic data analysis for W7-AS using Bayesian graphical models, Review of Scientific Instruments 75 (2004) 4219–4221.

[10] V. Pais, V. Stancalie, Using web services for remote data access and distributed applications, Fusion Engineering and Design 81 (2006) 2013 – 2017. 5th IAEA TM on Control, Data Acquisition, and Remote Participation for Fusion Research, 5th IAEA TM.

[11] T. Bluhm, S. Jacob, A. Werner, P. Heimann, C. Hennig, G. Khner, H. Kroiss, H. Laqua, M. Lewerentz, J. Maier, H. Riemann, J. Schacht, A. Spring, M. Zilker, A service based interface for scientific data retrieval, Fusion Engineering and Design 85 (2010) 579 – 582. Proceedings of the 7th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[12] A. Werner, J. Svensson, G. Khner, T. Bluhm, P. Heimann, S. Jakob, C. Hennig, H. Kroiss, H. Laqua, M. Lewerentz, H. Riemann, J. Schacht, A. Spring, M. Zilker, J. Maier, Scientific component framework for W7-X using service oriented grid middleware, Fusion Engineering and Design 85 (2010) 394 – 398. Proceedings of the 7th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[13] B. Guillerminet, M. Airaj, P. Huynh, G. Huysmans, F. Iannone, F. Imbeaux, J. Lister, G. Manduchi, P. Strand, Integrated tokamak modelling: Infrastructure and software integration project, Fusion Engineering and Design 83 (2008) 442 – 447. Proceedings of the 6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[14] F. Imbeaux, J. Lister, G. Huysmans, W. Zwingmann, M. Airaj, L. Appel, V. Basiuk, D. Coster, L.-G. Eriksson, B. Guillerminet, D. Kalupin, C. Konz, G. Manduchi, M. Ottaviani, G. Pereverzev, Y. Peysson, O. Sauter, J. Signoret, P. Strand, A generic data structure for integrated modelling of tokamak physics and subsystems, Computer Physics Communications 181 (2010) 987 – 998.

[15] Kepler, `https://kepler-project.org/`, accessed in April 2012.

[16] W3C, SOAP 1.1 standard, `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`, accessed in April 2012.

[17] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. thesis, University of California, Irvine, 2000.

[18] JSON RPC specification, `http://json-rpc.org/`, accessed in April 2012.

[19] Apache Thrift, `http://thrift.apache.org/`, accessed in November 2012.

[20] M. Hartikainen, M. Laitkorpi, A. Ruokonen, T. Systä, How to drill down to REST APIs: Resource harvesting with a pattern tool, in: 13th IEEE International Symposium on Web Systems Evolution (WSE 2011), Williamsburg, VA, USA, September 2011.

[21] F. Belqasmi, R. Glitho, C. Fu, RESTful web services for service provisioning in next-generation networks: a survey, IEEE Communications Magazine 49 (2011) 66 – 73.

[22] JSON specification, http://www.json.org/, accessed in April 2012.

[23] W3C, XML 1.0 standard, http://www.w3.org/TR/2008/REC-xml-20081126/, accessed in April 2012.

[24] R. A. V. Engelen, Pushing the SOAP envelope with web services for scientific computing, in: Proceedings of the International Conference on Web Services (ICWS), Las Vegas, 2003, pp. 346 – 352.

[25] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. v. Engelen, M. J. Lewis, Toward characterizing the performance of SOAP toolkits, in: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Washington, DC, USA, 2004, pp. 365–372.

[26] WSDL 1.1 standard, http://www.w3.org/TR/wsdl, accessed in April 2012.

[27] W3C, XML Schema 1.0 standard, http://www.w3.org/TR/xmlschema-0/, accessed in May 2013.

[28] Taverna, http://www.taverna.org.uk/, accessed in April 2012.

[29] A. Barker, J. van Hemert, Scientific Workflow: A Survey and Research Directions, in: R. Wyrzykowski, et al. (Eds.), Seventh International Conference on Parallel Processing and Applied Mathematics, PPAM 2007, Gdansk, Poland, September 9 – 12, 2007, Revised Selected Papers, LNCS, volume 4967, Springer, 2008, pp. 746–753.

[30] WSBPEL, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, accessed in April 2012.

[31] gSOAP library, http://www.cs.fsu.edu/~engelen/soap.html, accessed in April 2012.

[32] C. Gourdon, Programme optimise de calculs numeriques dans le configurations magnetique toroidales, CEN Fontenay aux Roses, 1970.

[33] ZODB database, http://www.zodb.org, accessed in April 2012.

[34] soaplib, `https://github.com/arskom/soaplib`, accessed in April 2012.

[35] R. Stadler, A. Vorköper, J. Boscary, A. Cardella, F. Hurd, C. Li, B. Mendelevitch, A. Peacock, H. Pirsch, The in-vessel components of the experiment Wendelstein 7-X, Fusion Engineering and Design 84 (2009) 305 – 308. Proceeding of the 25th Symposium on Fusion Technology (SOFT-25).

[36] W. D'haeseleer, W. N. G. Hitchon, J. D. Callen, J. Shohet, Flux Coordinates and Magnetic Field Structure, Springer-Verlag Berlin, 1991.

[37] R. Grimm, R. Dewar, J. Manickam, Ideal MHD stability calculations in axisymmetric toroidal coordinate systems, Journal of Computational Physics 49 (1983) 94 – 117.

[38] S. Hamada, Hydromagnetic equilibria and their proper coordinates, Nuclear Fusion 2 (1962) 23.

[39] A. H. Boozer, Guiding center drift equations, Physics of Fluids 23 (1980) 904–908.

[40] A. H. Boozer, Establishment of magnetic coordinates for a given magnetic field, Physics of Fluids 25 (1982) 520–521.

[41] N. Nakajima, J. Todoroki, M. Okamoto, On relation between Hamada and Boozer magnetic coordinate system, Technical Report 173, NIFS, 1992.

[42] A. Reiman, H. Greenside, Numerical solution of three-dimensional magnetic diferential equations, Journal of Computational Physics 75 (1988) 423–443.

[43] J. Todoroki, Calculating rotational transform following field lines, Journal of Plasma Fusion Research 79 (2003) 321–322.

[44] L. S. Solov'ev, V. D. Shafranov, Plasma Confinement in Closed Magnetic Systems, volume 5 of *Reviews of Plasma Physics*, Consultants Bureau, New-York - London, p. 1.

[45] T. Andreeva, T. Bräuer, M. Endler, J. Kißlinger, U. Toussaint, Influence of construction errors on Wendelstein 7-X magnetic configurations,

Fusion Engineering and Design 84 (2009) 408 – 412. Proceeding of the 25th Symposium on Fusion Technology (SOFT-25).

[46] T. Rummel, K. Rißе, J. Kißlinger, M. Koppen, F. Fullenbach, H. Neilson, T. Brown, S. Ramakrishnan, The trim coils for the Wendelstein 7-X magnet system, Applied Superconductivity, IEEE Transactions on 22 (2012) 4201704.

[47] osa Python library, `https://bitbucket.org/sboz/osa/overview`, accessed in April 2012.

[48] A. Barker, J. B. Weissman, J. I. van Hemert, Reducing data transfer in service-oriented architectures: The circulate approach, IEEE Transactions on Services Computing 99 (2011).

[49] S. Koulouzis, R. Cushing, K. A. Karasavvas, A. Belloum, M. Bubak, Enabling web services to consume and produce large datasets, IEEE Internet Computing 16 (2012) 52–60.