Object Management Group

140 Kendrick Street
Building A Suite 300
Needham, MA 02494
USA

Telephone: +1-781-444-0404
Facsimile: +1-781-444-0320

A Foundation for the Agile Creation and Enactment of Software Engineering Methods

Request For Proposal

OMG Document: ad/2011-06-26

Letters of Intent due: November 22, 2011
Submissions due: February 22, 2012

Objective of this RFP

An agile approach to software engineering methods is one that supports practitioners of software engineering (architects, designers, developers, programmers, testers, deployers, analysts and project managers) in dynamically adapting and customizing their methods during the preparation and execution of a project, controlled through company specific governance, use of examples and other means. In contrast to previous approaches in this area, which have provided support mainly for process engineers, the goal here is to provide direct support for practitioners as the main target group.

The objective of this RFP is to obtain a foundation for the agile creation and enactment of software engineering methods (that themselves may be agile or more traditional) by development practitioners themselves. This foundation is to consist of a *kernel* of software engineering domain concepts and relationships that is extensible (scalable), flexible and easy to use, and a domain-specific modeling *language* that allows developers to describe the essentials of their current and future practices and methods.

These practices and methods can then be supported by tools based on this common foundation, and they can further be composed, simulated, applied, compared, enacted, tailored, used, adapted, evaluated and measured by practitioners as well as taught and researched by academic and research communities.

For further details see Chapter 6 of this document.

1.0 Introduction

1.1 Goals of OMG

The Object Management Group (OMG) is the world's largest software consortium with an international membership of vendors, developers, and end users. Established in 1989, its mission is to help computer users solve enterprise integration problems by supplying open, vendor-neutral portability, interoperability and reusability specifications based on Model Driven Architecture (MDA). MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform, and provides a set of guidelines for structuring specifications expressed as models. OMG has established numerous widely used standards such as OMG IDL [IDL], CORBA [CORBA], Realtime CORBA [CORBA], GIOP/IIOP [CORBA], UML [UML], MOF [MOF], XMI [XMI] and CWM [CWM] to name a few significant ones.

1.2 Organization of this document

The remainder of this document is organized as follows:

Chapter 2 – *Architectural Context* - background information on OMG's Model Driven Architecture.

Chapter 3 – *Adoption Process* - background information on the OMG specification adoption process.

Chapter 4 – *Instructions for Submitters* - explanation of how to make a submission to this RFP.

Chapter 5 – *General Requirements on Proposals* - requirements and evaluation criteria that apply to all proposals submitted to OMG.

Chapter 6 – *Specific Requirements on Proposals* - problem statement, scope of proposals sought, requirements and optional features, issues to be discussed, evaluation criteria, and timetable that apply specifically to this RFP.

Appendix A – *References and Glossary Specific to this RFP*

Appendix B – General References and Glossary

1.3 Conventions

The key words "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.4 Contact Information

Questions related to the OMG's technology adoption process may be directed to *omg-process@omg.org*. General questions about this RFP may be sent to *responses@omg.org*.

OMG documents (and information about the OMG in general) can be obtained from the OMG's web site *(http://www.omg.org/)*. OMG documents may also be obtained by contacting OMG at *documents@omg.org*. Templates for RFPs (like this document) and other standard OMG documents can be found at the OMG Template Downloads Page at *http://www.omg.org/technology/template_download.htm*

2.0 Architectural Context

MDA provides a set of guidelines for structuring specifications expressed as models and the mappings between those models. The MDA initiative and the standards that support it allow the same model specifying business system or application functionality and behavior to be realized on multiple platforms. MDA enables different applications to be integrated by explicitly relating their models; this facilitates integration and interoperability and supports system evolution (deployment choices) as platform technologies change. The three primary goals of MDA are portability, interoperability and reusability.

Portability of any subsystem is relative to the subsystems on which it depends. The collection of subsystems that a given subsystem depends upon is often loosely called the *platform,* which supports that subsystem. Portability – and reusability - of such a subsystem is enabled if all the subsystems that it depends upon use standardized interfaces (APIs) and usage patterns.

MDA provides a pattern comprising a portable subsystem that is able to use any one of multiple specific implementations of a platform. This pattern is repeatedly usable in the specification of systems. The five important concepts related to this pattern are:

1. *Model* – A model is a representation of a part of the function, structure and/or behavior of an application or system. A representation is said to be formal when it is based on a language that has a well-defined form ("syntax"), meaning ("semantics"), and possibly rules of analysis, inference, or proof for its constructs. The syntax may be graphical or textual. The semantics might be defined, more or less formally, in terms of things observed in the world being described (e.g. message

sends and replies, object states and state changes, etc.), or by translating higher-level language constructs into other constructs that have a well-defined meaning. The optional rules of inference define what unstated properties you can deduce from the explicit statements in the model. In MDA, a representation that is not formal in this sense is not a model. Thus, a diagram with boxes and lines and arrows that is not supported by a definition of the meaning of a box, and the meaning of a line and of an arrow is not a model—it is just an informal diagram.

2. *Platform* – A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

3. *Platform Independent Model (PIM)* – A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

4. *Platform Specific Model (PSM)* – A model of a subsystem that includes information about the specific technology that is used in the realization of that subsystem on a specific platform, and hence possibly contains elements that are specific to the platform.

5. *Mapping* – Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel. A mapping may be expressed as associations, constraints, rules, templates with parameters that must be assigned during the mapping, or other forms yet to be determined.

For example, in case of CORBA the platform is specified by a set of interfaces and usage patterns that constitute the CORBA Core Specification [CORBA]. The CORBA platform is independent of operating systems and programming languages. The OMG Trading Object Service specification [TOS] (consisting of interface specifications in OMG Interface Definition Language (OMG IDL)) can be considered to be a PIM from the viewpoint of CORBA, because it is independent of operating systems and programming languages. When the IDL to C++ Language Mapping specification is applied to the Trading Service PIM, the C++-specific result can be considered to be a PSM for the Trading Service, where the platform is the C++ language and the C++ ORB implementation. Thus the IDL to C++ Language Mapping specification [IDLC++] determines the mapping from the Trading Service PIM to the Trading Service PSM.

Note that the Trading Service model expressed in IDL is a PSM relative to the CORBA platform too. This highlights the fact that platform-independence and platform-specificity are relative concepts.

The UML Profile for EDOC specification [EDOC] is another example of the application of various aspects of MDA. It defines a set of modeling constructs that are independent of middleware platforms such as EJB [EJB], CCM [CCM], MQSeries

[MQS], etc. A PIM based on the EDOC profile uses the middleware-independent constructs defined by the profile and thus is middleware-independent. In addition, the specification defines formal metamodels for some specific middleware platforms such as EJB, supplementing the already-existing OMG metamodel of CCM (CORBA Component Model). The specification also defines mappings from the EDOC profile to the middleware metamodels. For example, it defines a mapping from the EDOC profile to EJB. The mapping specifications facilitate the transformation of any EDOC-based PIM into a corresponding PSM for any of the specific platforms for which a mapping is specified.

Continuing with this example, one of the PSMs corresponding to the EDOC PIM could be for the CORBA platform. This PSM then potentially constitutes a PIM, corresponding to which there would be implementation language specific PSMs derived via the CORBA language mappings, thus illustrating recursive use of the Platform-PIM-PSM-Mapping pattern.

Note that the EDOC profile can also be considered to be a platform in its own right. Thus, a model expressed via the profile is a PSM relative to the EDOC platform.

An analogous set of concepts apply to Interoperability Protocols wherein there is a PIM of the payload data and a PIM of the interactions that cause the data to find its way from one place to another. These then are realized in specific ways for specific platforms in the corresponding PSMs.

Analogously, in case of databases there could be a PIM of the data (say using the Relational Data Model), and corresponding PSMs specifying how the data is actually represented on a storage medium based on some particular data storage paradigm etc., and a mapping from the PIM to each PSM.

OMG adopts standard specifications of models that exploit the MDA pattern to facilitate portability, interoperability and reusability, either through ab initio development of standards or by reference to existing standards. Some examples of OMG adopted specifications are:

1. *Languages* – e.g. IDL for interface specification, UML for model specification, OCL for constraint specification, etc.

2. Mappings – e.g. Mapping of OMG IDL to specific implementation languages (CORBA PIM to Implementation Language PSMs), UML Profile for EDOC (PIM) to CCM (CORBA PSM) and EJB (Java PSM), CORBA (PSM) to COM (PSM) etc.

3. *Services* – e.g. Naming Service [NS], Transaction Service [OTS], Security Service [SEC], Trading Object Service [TOS] etc.

4. *Platforms* – e.g. CORBA [CORBA].

5. *Protocols* – e.g. GIOP/IIOP [CORBA] (both structure and exchange protocol), XML Metadata Interchange [XMI] (structure specification usable as payload on multiple exchange protocols).

6. *Domain Specific Standards* – e.g. Data Acquisition from Industrial Systems (Manufacturing) [DAIS], General Ledger Specification (Finance) [GLS], Air Traffic Control (Transportation) [ATC], Gene Expression (Life Science Research) [GE], Personal Identification Service (Healthcare) [PIDS], etc.

For an introduction to MDA, see [MDAa]. For a discourse on the details of MDA please refer to [MDAc]. To see an example of the application of MDA see [MDAb]. For general information on MDA, see [MDAd].

Object Management Architecture (OMA) is a distributed object computing platform architecture within MDA that is related to ISO's Reference Model of Open Distributed Processing RM-ODP [RM-ODP]. CORBA and any extensions to it are based on OMA. For information on OMA see [OMA].

3.0 Adoption Process

3.1 Introduction

OMG adopts specifications by explicit vote on a technology-by-technology basis. The specifications selected each satisfy the architectural vision of MDA. OMG bases its decisions on both business and technical considerations. Once a specification adoption is finalized by OMG, it is made available for use by both OMG members and non-members alike.

Request for Proposals (RFP) are issued by a *Technology Committee* (TC), typically upon the recommendation of a *Task Force* (TF) and duly endorsed by the *Architecture Board* (AB).

Submissions to RFPs are evaluated by the TF that initiated the RFP. Selected specifications are *recommended* to the parent TC after being *reviewed* for technical merit and consistency with MDA and other adopted specifications and *endorsed* by the AB. The parent TC of the initiating TF then votes to *recommend adoption* to the OMG Board of Directors (BoD). The BoD acts on the recommendation to complete the adoption process.

For more detailed information on the adoption process see the *Policies and Procedures of the OMG Technical Process* [P&P] and the *OMG Hitchhiker's Guide* [Guide]. In case of any inconsistency between this document and the [P&P] in all cases the [P&P] shall prevail.

3.2 Steps in the Adoption Process

A TF, its parent TC, the AB and the Board of Directors participate in a collaborative process, which typically takes the following form:

- *Development and Issuance of RFP*

  RFPs are drafted by one or more OMG members who are interested in the adoption of a standard in some specific area. The draft RFP is presented to an appropriate TF, based on its subject area, for approval and recommendation to issue. The TF and the AB provide guidance to the drafters of the RFP. When the TF and the AB are satisfied that the RFP is appropriate and ready for issuance, the TF recommends issuance to its parent TC, and the AB endorses the recommendation. The TC then acts on the recommendation and issues the RFP.

- *Letter of Intent (LOI)*

  A Letter of Intent (LOI) must be submitted to the OMG signed by an officer of the member organization which intends to respond to the RFP, confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements. (See section 4.3 for more information.). In order to respond to an RFP the organization must be a member of the TC that issued the RFP.

- *Voter Registration*

  Interested OMG members, other than Trial, Press and Analyst members, may participate in specification selection votes in the TF for an RFP. They may need to register to do so, if so stated in the RFP. Registration ends on a specified date, 6 or more weeks after the announcement of the registration period. The registration closure date is typically around the time of initial submissions. Member organizations that have submitted an LOI are automatically registered to vote.

- *Initial Submissions*

  Initial Submissions are due by a specified deadline. Submitters normally present their proposals at the first meeting of the TF after the deadline. Initial Submissions are expected to be complete enough to provide insight on the technical directions and content of the proposals.

- *Revision Phase*

  During this time submitters have the opportunity to revise their Submissions, if they so choose.

- *Revised Submissions*

Revised Submissions are due by a specified deadline. Submitters again normally present their proposals at the next meeting of the TF after the deadline. (Note that there may be more than one Revised Submission deadline. The decision to set new Revised Submission deadlines is made by the registered voters for that RFP.)

- *Selection Votes*

  When the registered voters for the RFP believe that they sufficiently understand the relative merits of the Revised Submissions, a selection vote is taken. The result of this selection vote is a recommendation for adoption to the TC. The AB reviews the proposal for MDA compliance and technical merit. An endorsement from the AB moves the voting process into the issuing Technology Committee. An eight-week voting period ensues in which the TC votes to recommend adoption to the OMG Board of Directors (BoD). The final vote, the vote to adopt, is taken by the BoD and is based on technical merit as well as business qualifications. The resulting draft standard is called the *Alpha Specification*.

- *Business Committee Questionnaire*

  The submitting members whose proposal is recommended for adoption need to submit their response to the BoD Business Committee Questionnaire [BCQ] detailing how they plan to make use of and/or make the resulting standard available in products. If no organization commits to make use of the standard, then the BoD will typically not act on the recommendation to adopt the standard - so it is very important to fulfill this requirement.

- *Finalization*

  A Finalization Task Force (FTF) is chartered by the TC that issued the RFP, to prepare an Alpha submission for publishing as a Formal (i.e. publicly available) specification, by fixing any problems that are reported by early users of the specification. Upon completion of its activity the FTF recommends adoption of the resulting Beta (draft) specification. The parent TC acts on the recommendation and recommends adoption to the BoD. OMG Technical Editors produce the Formal Specification document based on this Beta Specification.

- *Revision*

  A Revision Task Force (RTF) is normally chartered by a TC, after the FTF completes its work, to manage issues filed against the Formal Specification by implementers and users. The output of the RTF is a Beta specification reflecting minor technical changes, which the TC and Board will usually approve for adoption as the next version of the Formal Specification.

3.3 Goals of the evaluation

The primary goals of the TF evaluation are to:

- Provide a fair and open process
- Facilitate critical review of the submissions by members of OMG
- Provide feedback to submitters enabling them to address concerns in their revised submissions
- Build consensus on acceptable solutions
- Enable voting members to make an informed selection decision
- Submitters are expected to actively contribute to the evaluation process.

4.0 Instructions for Submitters

4.1 OMG Membership

To submit to an RFP issued by the Platform Technology Committee the submitter or submitters must be either Platform or Contributing members on the date of the submission deadline, while for Domain Technology RFPs the submitter or submitters must be either Contributing or Domain members. Submitters sometimes choose to name other organizations that support a submission in some way; however, this has no formal status within the OMG process, and for OMG's purposes confers neither duties nor privileges on the organizations thus named.

4.2 Submission Effort

An RFP submission may require significant effort in terms of document preparation, presentations to the issuing TF, and participation in the TF evaluation process. Several staff months of effort might be necessary. OMG is unable to reimburse submitters for any costs in conjunction with their submissions to this RFP.

4.3 Letter of Intent

A Letter of Intent (LOI) must be submitted to the OMG Business Committee signed by an officer of the submitting organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements. These terms, conditions, and requirements are defined in the *Business Committee RFP Attachment* and are reproduced verbatim in section 4.4 below.

The LOI should designate a single contact point within the submitting organization for receipt of all subsequent information regarding this RFP and the submission. The name of this contact will be made available to all OMG members. The LOI is typically due 60 days before the deadline for initial submissions. LOIs must be sent by fax or paper mail to the "RFP Submissions Desk" at the main OMG address shown on the first page of this RFP.

Here is a suggested template for the Letter of Intent:

This letter confirms the intent of <organization required> (the organization) to submit a response to the OMG <RFP name required> RFP. We will grant OMG and its members the right to copy our response for review purposes as specified in section 4.7 of the RFP. Should our response be adopted by OMG we will comply with the OMG Business Committee terms set out in section 4.4 of the RFP and in document omg/06-03-02.

<contact name and details required> will be responsible for liaison with OMG regarding this RFP response.

The signatory below is an officer of the organization and has the approval and authority to make this commitment on behalf of the organization.

<signature required>

4.4 Business Committee RFP Attachment

This section contains the text of the Business Committee RFP attachment concerning commercial availability requirements placed on submissions. This attachment is available separately as an OMG document omg/06-03-02.

Commercial considerations in OMG technology adoption

A1 Introduction

OMG wishes to encourage rapid commercial adoption of the specifications it publishes. To this end, there must be neither technical, legal nor commercial obstacles to their implementation. Freedom from the first is largely judged through technical review by the relevant OMG Technology Committees; the second two are the responsibility of the OMG Business Committee. The BC also looks for evidence of a commitment by a submitter to the commercial success of products based on the submission.

A2 Business Committee evaluation criteria

A2.1 Viable to implement across platforms

While it is understood that final candidate OMG submissions often combine technologies before they have all been implemented in one system, the Business Committee nevertheless wishes to see evidence that each major feature has been implemented, preferably more than once, and by separate organisations. Pre-product implementations are acceptable. Since use of OMG specifications should not be dependant on any one platform, cross-platform availability and interoperability of implementations should be also be demonstrated.

A2.2 Commercial availability

In addition to demonstrating the existence of implementations of the specification, the submitter must also show that products based on the specification are commercially available, or will be within 12 months of the date when the specification was recommended for adoption by the appropriate Task Force. Proof of intent to ship product within 12 months might include:

1) *A public product announcement with a shipping date within the time limit.*
2) *Demonstration of a prototype implementation and accompanying draft user documentation.*

Alternatively, and at the Business Committee's discretion, submissions may be adopted where the submitter is not a commercial software provider, and therefore will not make implementations commercially available. However, in this case the BC will require concrete evidence of two or more independent implementations of the specification being used by end-user organisations as part of their businesses. Regardless of which requirement is in use, the submitter must inform the OMG of completion of the implementations when commercially available.

A2.3 Access to Intellectual Property Rights

OMG will not adopt a specification if OMG is aware of any submitter, member or third party which holds a patent, copyright or other intellectual property right (collectively referred to in this policy statement as "IPR") which might be infringed by implementation or recommendation of such specification, unless OMG believes that such IPR owner will grant a license to organisations (whether OMG members or not) on non-discriminatory and commercially reasonable terms which wish to make use of the specification. Accordingly, the submitter must certify that it is not aware of any claim that the specification infringes any IPR of a third party or that it is aware and believes that an appropriate non-discriminatory license is available from that third party. Except for this certification, the submitter will not be required to make any other warranty, and specifications will be offered by OMG for use "as is". If the submitter owns IPR to which an use of a specification based upon its submission would necessarily be subject, it must certify to the Business Committee that it will make a suitable license available to any user on non- discriminatory and commercially reasonable terms, to permit development and commercialisation of an implementation that includes such IPR.

It is the goal of the OMG to make all of its technology available with as few impediments and disincentives to adoption as possible, and therefore OMG strongly encourages the submission of technology as to which royalty-free licenses will be available. However, in all events, the submitter shall also certify that any necessary licence will be made available on commercially reasonable, non-discriminatory terms. The submitter is responsible for disclosing in detail all known restrictions, placed either by the submitter or, if known, others, on technology necessary for any use of the specification.

A2.4 Publication of the specification

Should the submission be adopted, the submitter must grant OMG (and its sublicensees) a world- wide, royalty-free licence to edit, store, duplicate and distribute both the specification and works derived from it (such as revisions and teaching materials). This requirement applies only to the written specification, not to any implementation of it.

A2.5 Continuing support

The submitter must show a commitment to continue supporting the technology underlying the specification after OMG adoption, for instance by showing the BC development plans for future revisions, enhancement or maintenance.

4.5 Responding to RFP items

4.5.1 Complete proposals

A submission must propose full specifications for all of the relevant requirements detailed in Chapter 6 of this RFP. Submissions that do not present complete proposals may be at a disadvantage.

Submitters are highly encouraged to propose solutions to any optional requirements enumerated in Chapter 6.

4.5.2 Additional specifications

Submissions may include additional specifications for items not covered by the RFP that they believe to be necessary and integral to their proposal. Information on these additional items should be clearly distinguished.

Submitters must give a detailed rationale as to why these specifications should also be considered for adoption. However submitters should note that a TF is unlikely to consider additional items that are already on the roadmap of an OMG TF, since this would pre-empt the normal adoption process.

4.5.3 Alternative approaches

Submitters may provide alternative RFP item definitions, categorizations, and groupings so long as the rationale for doing so is clearly stated. Equally, submitters may provide alternative models for how items are provided if there are compelling technological reasons for a different approach.

4.6 Confidential and Proprietary Information

The OMG specification adoption process is an open process. Responses to this RFP become public documents of the OMG and are available to members and non-members alike for perusal. No confidential or proprietary information of any kind will be accepted in a submission to this RFP.

4.7 Copyright Waiver

Every submission document must contain: (i) a waiver of copyright for unlimited duplication by the OMG, and (ii) a limited waiver of copyright that allows each OMG member to make up to fifty (50) copies of the document for review purposes only. See Section 4.9.2 for recommended language.

4.8 Proof of Concept

Submissions must include a "proof of concept" statement, explaining how the submitted specifications have been demonstrated to be technically viable. The technical viability has to do with the state of development and maturity of the technology on which a submission is based. This is not the same as commercial availability. Proof of concept statements can contain any information deemed relevant by the submitter; for example:

"This specification has completed the design phase and is in the process of being prototyped."

"An implementation of this specification has been in beta-test for 4 months."

"A named product (with a specified customer base) is a realization of this specification."

It is incumbent upon submitters to demonstrate the technical viability of their proposal to the satisfaction of the TF managing the evaluation process. OMG will favor proposals based on technology for which sufficient relevant experience has been gained.

4.9 Format of RFP Submissions

This section presents the structure of a submission in response to an RFP. *All submissions* must contain the elements itemized in section 4.9.2 below before they can

be accepted as a valid response for evaluation or a vote can be taken to recommend for adoption.

4.9.1 General

- Submissions that are concise and easy to read will inevitably receive more consideration.
- Submitted documentation should be confined to that directly relevant to the items requested in the RFP. If this is not practical, submitters must make clear what portion of the documentation pertains directly to the RFP and what portion does not.
- The key words **"must"**, **"must not"**, **"required"**, **"shall"**, **"shall not"**, **"should"**, **"should not"**, **"recommended"**, **"may"**, and **"optional"** shall be used in the submissions with the meanings as described in RFC 2119 [RFC2119].

4.9.2 Required Outline

A three-part structure for submissions is required. Part I is non-normative, providing information relevant to the evaluation of the proposed specification. Part II is normative, representing the proposed specification. Specific sections like Appendices may be explicitly identified as non-normative in Part II. Part III is normative specifying changes that must be made to previously adopted specifications in order to be able to implement the specification proposed in Part II.

PART I

- A cover page carrying the following information (a template for this is available [Inventory]):

  The full name of the submission

  The primary contact for the submission

  The acronym proposed for the specification (e.g. UML, CORBA)

  The name and document number of the RFP to which this is a response

  The document number of the main submission document

  An inventory of all accompanying documents, with OMG document number, short description, a URL where appropriate, and whether they are normative.
- List of OMG members making the submission (see 4.1) listing exactly which members are making the submission, so that submitters can be matched with LOI responders and their current eligibility can be verified.
- Copyright waiver (see 4.7), in a form acceptable to the OMG.

  One acceptable form is:

"Each of the entities listed above: (i) grants to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version, and (ii) grants to each member of the OMG a nonexclusive, royalty-free, paid up, worldwide license to make up to fifty (50) copies of this document for internal review purposes only and not for distribution, and (iii) has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used any OMG specification that may be based hereon or having conformed any computer software to such specification."

If you wish to use some other form you must get it approved by the OMG legal counsel before using it in a submission.

- For each member making the submission, an individual contact point who is authorized by the member to officially state the member's position relative to the submission, including matters related to copyright ownership, etc. (see 4.3)
- Overview or guide to the material in the submission
- Overall design rationale (if appropriate)
- Statement of proof of concept (see 4.8)
- Resolution of RFP requirements and requests

Explain how the proposal satisfies the specific requirements and (if applicable) requests stated in Chapter 6. References to supporting material in Part II should be given.

In addition, if the proposal does not satisfy any of the general requirements stated in Chapter 5, provide a detailed rationale.

- Responses to RFP issues to be discussed

  Discuss each of the "Issues To Be Discussed" identified in Chapter 6.

PART II

The contents of this part should be structured based on the template found in [FORMS] and should contain the following elements as per the instructions in the template document cited above:

- Scope of the proposed specification
- Proposed conformance criteria

  Submissions should propose appropriate conformance criteria for implementations.
- Proposed normative references

Submissions should provide a list of the normative references that are used by the proposed specification.

- Proposed list of terms and definitions

  Submissions should provide a list of terms that are used in the proposed specification with their definitions.

- Proposed list of symbols

  Submissions should provide a list of special symbols that are used in the proposed specification together with their significance.

- Proposed specification

PART III

- Changes or extensions required to existing OMG specifications

  Submissions must include a full specification of any changes or extensions required to existing OMG specifications. This should be in a form that enables "mechanical" section-by-section revision of the existing specification.

4.10 How to Submit

Submitters should send an electronic version of their submission to the *RFP Submissions Desk* (omg-documents@omg.org) at OMG Headquarters by 5:00 PM U.S. Eastern Standard Time (22:00 GMT) on the day of the Initial and Revised Submission deadlines. Acceptable formats are Adobe FrameMaker source, ODF (ISO/IEC 26300), OASIS Darwin Information Typing Architecture (DITA) or OASIS DocBook 4.x (or later).

Submitters should make sure they receive electronic or voice confirmation of the successful receipt of their submission. Submitters should be prepared to send a single hardcopy version of their submission, if requested by OMG staff, to the attention of the "RFP Submissions Desk" at the main OMG address shown on the first page of this RFP.

5.0 General Requirements on Proposals

5.1 Requirements

5.1.1 Submitters are encouraged to express models using OMG modeling languages such as UML, MOF, CWM and SPEM (subject to any further constraints on the types of the models and modeling technologies specified in Chapter 6 of this RFP). Submissions containing models expressed via OMG modeling languages shall be accompanied by an OMG XMI [XMI] representation of the models (including a machine-readable copy). A

best effort should be made to provide an OMG XMI representation even in those cases where models are expressed via non-OMG modeling languages.

5.1.2 Chapter 6 of this RFP specifies whether PIM(s), PSM(s), or both are being solicited. If proposals specify a PIM and corresponding PSM(s), then the rules specifying the mapping(s) between the PIM and PSM(s) shall either be identified by reference to a standard mapping or specified in the proposal. In order to allow possible inconsistencies in a proposal to be resolved later, proposals shall identify whether the mapping technique or the resulting PSM(s) are to be considered normative.

5.1.3 Proposals shall be *precise* and *functionally complete*. All relevant assumptions and context required for implementing the specification shall be provided.

5.1.4 Proposals shall specify *conformance criteria* that clearly state what features all implementations must support and which features (if any) may *optionally* be supported.

5.1.5 Proposals shall *reuse* existing OMG and other standard specifications in preference to defining new models to specify similar functionality.

5.1.6 Proposals shall justify and fully specify any *changes or extensions* required to existing OMG specifications. In general, OMG favors proposals that are *upwards compatible* with existing standards and that minimize changes and extensions to existing specifications.

5.1.7 Proposals shall factor out functionality that could be used in different contexts and specify their models, interfaces, etc. separately. Such *minimalism* fosters re-use and avoids functional duplication.

5.1.8 Proposals shall use or depend on other specifications only where it is actually necessary. While re-use of existing specifications to avoid duplication will be encouraged, proposals should avoid gratuitous use.

5.1.9 Proposals shall be *compatible* with and *usable* with existing specifications from OMG and other standards bodies, as appropriate. Separate specifications offering distinct functionality should be usable together where it makes sense to do so.

5.1.10 Proposals shall preserve maximum *implementation flexibility*. Implementation descriptions should not be included and proposals shall not constrain implementations any more than is necessary to promote interoperability.

5.1.11 Proposals shall allow *independent implementations* that are *substitutable* and *interoperable*. An implementation should be replaceable by an alternative implementation without requiring changes to any client.

5.1.12 Proposals shall be compatible with the architecture for system distribution defined in ISO's Reference Model of Open Distributed Processing [RM-ODP]. Where such compatibility is not achieved, or is not appropriate, the response to the RFP must

include reasons why compatibility is not appropriate and an outline of any plans to achieve such compatibility in the future.

5.1.13 In order to demonstrate that the specification proposed in response to this RFP can be made secure in environments requiring security, answers to the following questions shall be provided:

- What, if any, are the security sensitive elements that are introduced by the proposal?
- Which accesses to security-sensitive elements must be subject to security policy control?
- Does the proposed service or facility need to be security aware?
- What default policies (e.g., for authentication, audit, authorization, message protection etc.) should be applied to the security sensitive elements introduced by the proposal? Of what security considerations must the implementers of your proposal be aware?

The OMG has adopted several specifications, which cover different aspects of security and provide useful resources in formulating responses. [CSIV2] [SEC] [RAD].

5.1.14 Proposals shall specify the degree of internationalization support that they provide. The degrees of support are as follows:

a) Uncategorized: Internationalization has not been considered.

b) Specific to <region name>: The proposal supports the customs of the specified region only, and is not guaranteed to support the customs of any other region. Any fault or error caused by requesting the services outside of a context in which the customs of the specified region are being consistently followed is the responsibility of the requester.

c) Specific to <multiple region names>: The proposal supports the customs of the specified regions only, and is not guaranteed to support the customs of any other regions. Any fault or error caused by requesting the services outside of a context in which the customs of at least one of the specified regions are being consistently followed is the responsibility of the requester.

d) Explicitly not specific to <region(s) name>: The proposal does not support the customs of the specified region(s). Any fault or error caused by requesting the services in a context in which the customs of the specified region(s) are being followed is the responsibility of the requester.

5.2 Evaluation criteria

Although the OMG adopts model-based specifications and not implementations of those specifications, the technical viability of implementations will be taken into account during the evaluation process. The following criteria will be used:

5.2.1 Performance

Potential implementation trade-offs for performance will be considered.

5.2.2 Portability

The ease of implementation on a variety of systems and software platforms will be considered.

5.2.3 Securability

The answer to questions in section 5.1.13 shall be taken into consideration to ascertain that an implementation of the proposal is securable in an environment requiring security.

5.2.4 Conformance: Inspectability and Testability

The adequacy of proposed specifications for the purposes of conformance inspection and testing will be considered. Specifications should provide sufficient constraints on interfaces and implementation characteristics to ensure that conformance can be unambiguously assessed through both manual inspection and automated testing.

5.2.5 Standardized Metadata

Where proposals incorporate metadata specifications, usage of OMG standard XMI metadata [XMI] representations must be provided as this allows specifications to be easily interchanged between XMI compliant tools and applications. Since use of XML (including XMI and XML/Value [XML/Value]) is evolving rapidly, the use of industry specific XML vocabularies (which may not be XMI compliant) is acceptable where justified.

6.0 Specific Requirements on Proposals

6.1 Problem Statement

It is widely recognized that the successful development of significant software systems benefits from the application of effective methods and well-defined processes. However, even after more than forty years of work in software engineering, methods are still too often applied haphazardly by development teams.

The traditional prescription for resolving such problems is to carefully define the methods to be used in a software development effort and then provide the development team with the tools to enact the methods so defined. But such software method engineering approaches are often considered by development teams as being too heavyweight and inflexible. Software developers today are used to agility and flexibility in carrying out their projects. However, software methods defined by separate method engineers typically do not leave enough flexibility for a development team themselves, without a method engineering specialist, to customize, tailor and adapt the process they use, not just at the beginning, but *continuously* as necessary over the course of a development effort.

A similar concern has sometimes been raised against model-driven approaches in general, based on the perception that they require detailed modeling to be done "all up front", often by modelers who are not developers. On the other hand, many development teams do use UML or other less formal notations all the time to sketch out designs *amongst the developers themselves.* And the most effective model-driven efforts integrate "agile modeling" by regular developers as a normal practice within their development effort rather than as a heavyweight "additional thing".

Similarly, current software method and process modeling approaches are perceived to require methods to be modeled "all up front", by method engineers who are not developers. This disconnects "method engineering" from the day-to-day work of development practitioners and requires an investment in additional staff and infrastructure that can be hard to justify except in other than large companies working on large projects. Projects come in all different sizes and there is a huge amount of software development that occurs in small to mid-sized companies that do not and realistically will not have method engineers at all.

On the other hand, developers pragmatically talk all the time about what is and is not working in their development method and how to adjust it. But, there has been far less acceptance of any consistent notation/language (such as SPEM) to use in this discussion than, say, the acceptance of UML for modeling the system itself. Further, developers do not even have any good codification of the software engineering practices on which effective methods should be based.

As a result, the method discussions within development teams often end up being either overly influenced by the visibility of the latest "hot" approaches or are limited to tailoring around the edges of some method that has been dictated to them. This seriously limits the ability of a team to be effective and scalable while remaining flexible and agile.

6.2 Scope of Proposals Sought

The use of an application framework of some sort is generally recognized these days as an effective way to enhance the productivity of a development team, particularly in conjunction with an agile development process. Such frameworks provide both a toolkit of components and an easy-to-use scripting language for flexibly composing the components.

In order to address the issues discussed in Section 6.1, development teams should similarly have available to them a framework that allows for the rapid construction of software methods for the team's own use. Such a framework would then allow the team to agilely tailor their methods as they see fit during the course of a development effort. The crucial intent is to support the needs of the software development practitioners themselves, as they see them. (It should also be noted that, while having agility in tailoring methods in critical, the software development methods that result are, by intent, of whatever kind is most appropriate for a specific development effort and team, whether those be more traditional prescriptive methods or themselves agile methods.)

Rather than standardizing on one such framework, though, it would be more desirable to allow industry to develop various frameworks that may be useful in various different contexts. But, to avoid "silos" of non-interoperable frameworks (which is one downside of the current plethora of frameworks in the area of application development), and to promote the adoption of sound software engineering best practices, all these frameworks should be based on a common, standardized foundation. That is what is being requested in response to this RFP.

6.2.1 Methods and Practices

A *method* may be defined as a systematic way of doing things in a particular discipline. For the purpose of this RFP, the relevant discipline is software engineering. Software engineering methods support tasks such as the development of a new software system, the maintenance of an existing system or even the integration of an entire enterprise system architecture.

(Note that the terms "method" and "process" are often used interchangeably in discussions of software engineering. However, there is actually no clear consensus in the community on whether these are really the same thing, or whether methods contain processes or processes contain methods. For the present discussion, these two terms can be considered to be largely synonymous, though in the remainder of the RFP the term

"method", in the sense that it is defined in this section, will be consistently used in preference to "process".)

Methods at this level may be considered as composed from well-defined *practices.* A practice is a general, repeatable approach to doing something with a specific purpose in mind, providing a systematic and verifiable way of addressing a particular aspect of the work at hand. It should have a clear goal expressed in terms of the results its use will achieve and provide guidance on what is to be done to achieve the goal and to verify that it has been achieved. Such practices may include specific approaches for software design, coding, testing at various levels, integration, organizing and managing the development team, etc. Examples of practices are Iterative Development, Use Case Driven Development and Test Driven Development.

Note that the definition of a practice is intentionally similar to that of a method. Indeed, practices at various levels may be composed from lower-level practices, and a method may be considered to be simply a composite practice targeted at the level of support of an entire discipline. This also allows for the further composition of methods at even higher levels within and across disciplines.

6.2.1.1 Enactment of Methods

A further distinction, however, is that a method must be *enactable,* while a practice in isolation will in general not be. In the context of this RFP, the enactment of a method can be defined as the carrying out of that method in the context of a specific project effort. Within this context, the practices within the method may be considered use cases for the work that must be carried out to achieve the project objectives, with each practice providing a specific aspect of the overall method.

Enacting a method includes using the method to create elements such as tasks and work products during the software endeavor, focusing on *what* to produce and *how* to produce it. Since software development is a collaborative team-based effort that involves knowledge workers, enactment of methods should support monitoring and progressing the software endeavor through human agents foremost and automation secondly.

Enactment is always done by practitioners and involves members of a team collaborating on decision-making, planning and execution. Enactment may be partially supported by method repositories and process engines that are linked to tools such as project management and issue tracking systems.

6.2.1.2 Composition of Practices

The composition of a method from practices is more than just a juxtaposition or grouping of a chosen set of practices. While each practice must be individually definable, a practice will of necessity have expectations on work done outside its scope that must be met by other practices within a complete method. For example, a testing practice will place certain expectations on the concept of a *unit* that is central to a

coding practice and the concept of a *testable requirement* generated by an analysis practice. A good method framework must provide effective, easy-to-use means for resolving the expectations across a consistent set of practices and weaving them together into a complete, enactable method.

6.2.1.3 Practice Infrastructure

In this light, this RFP requests the foundation for a common "practice infrastructure" that would enable software developers to more quickly understand, compose and compare individual practices and entire methods. It could also form the basis for the appropriate governance of software organizations, while allowing their developers the freedom to use their preferred practices, composed with those of their organizations. Further, it would allow the evaluation and validation of comparable method and process elements, guide practical research to useful results and act as a common context for training and education.

6.2.2 The Kernel

In order to allow the broadest possible applicability, what needs to be standardized are not practices themselves, but a common *kernel* of underlying concepts and principles applicable across all methods that may be used to define various practices. Such a kernel may be specified as a *domain model* for software engineering, providing a common terminology of concepts and their relationships that may be used in the definition of practices. Kernel elements will include concepts of things to produce or have, such as system, requirement, team, etc. as well as concepts for things to do, such as activity, practice and method.

Further, the concepts in the kernel must be defined in a way that allows them to be *extensible* and *tailorable* to support the needs of a wide variety of practices and methods and to allow flexibility in the definition and application of those methods by each development team. Such extensibility may be provided through a specialization mechanism and/or "slots" or "templates" that are expected to be filled in when concepts are instantiated. However, it is of particular interest for responses to this RFP to provide innovative proposals for extensibility of the kernel.

The key is to provide a powerful enough extensibility mechanism that the kernel itself can stay focused and light, while still allowing it to be easily and naturally applied. This is what distinguishes what is being requested as a *kernel,* as opposed to a new "unified" method framework. The kernel should reflect the *essence* of software engineering in a way that can be widely accepted, and which does not change as new practices are built on it.

6.2.3 The Language

In addition to the kernel, the foundation requested by this RFP includes a standard *language* for specifying practices based on the kernel and for composing

methods from the practices. To support a model-driven approach to method engineering, the language should be a *modeling* language that can be used by a development team to both informally discuss and sketch their methods and then formalize those methods as they find appropriate.

As mentioned in Section 6.2.1, enactment is considered here to be the carrying out of a method in the context of a specific project effort. In a certain sense, the requested language will essentially be used to "script" methods for enactment, in an analogous way to how scripting languages are used in application frameworks. However, it must be possible for scripting in some cases to be very light, perhaps just specifying milestones tracked during the course of a method. But it also must be possible to add more detail on top of this, such as a practice that defines specific work products associated with method milestones or the inclusion of specific activities defined using more detailed practice scripts.

There is also an important distinction to be made between the rigorous scripting required for executable software and the more flexible scripting that must be allowed for methods. In the end, it is really the project team that enacts a method, and such teams are not "programmed" like computers. A method should be scripted in a way that is easy for practitioners to read and understand as guidelines for carrying out the method, while still giving them the freedom to use their judgment and do what makes sense within the context of their project.

Nevertheless, automated support for enactment can be very useful, especially for larger project teams, and the language must be defined precisely enough itself to allow for the automated support of methods formalized using the language. Such automation may involve the actual "execution" of method scripts, so that development team members can be given appropriate support for carrying out the process and their progress in doing so can be tracked.

However, the intent, in the end, is always to support the work of the development team, as the team sees most fit. It is the *method* that is being scripted, to whatever extent is appropriate, not the actual actions of the team. In fact, the very goal of the foundation being requested is to allow a development team to take control of its own development method, not to be controlled by it.

6.2.3.1 Users and usages of the Language

The Language has two major user types: practitioners and method engineers. This RFP primary target group is the practitioners, but method engineers also have an important role in defining methods.

Practitioners are primarily interested in the following usages of the language: read, evaluate, compare, compose, tailor, use and adapt. Method engineers are primarily involved in: define, compare and compose.

Practitioners want to use ready-to-use complete practices that each give a measurable value to the method, that is designed to be lean and that includes the verification of the work performed when using the practice. Practitioners must be able to reuse such complete practices, tailor them and adapt them based on retrospectives made by the team.

Method engineers create these complete practices by reusing practice fragments, which the method engineers glue together and fine tune to become a useful concrete practice. Method engineers also create method frameworks including packages of practices which are composed and from which a practitioner can select a subset to be used for a particular endeavor.

There will be practitioners at different competence levels. A very high percentage of the practitioners in the world are not interested in the definition of methods and practices. Their interest is to develop software only and thus to use the practices. Other more advanced practitioners are interested in having more detailed knowledge of the practices and they also want to have deeper understanding of the progress of their endeavor in producing the outcome of the project. Some, much fewer in today's situation, are interested in getting detailed support for the practices and its detailed activities.

6.3 Relationship to Existing OMG Specifications and activities

6.3.1 Software and System Process Engineering Metamodel (SPEM)

SPEM is an existing OMG specification that defines a framework, a metamodel and a UML profile to describe software and system engineering processes [SPEM2]. SPEM is specifically "targeted at process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for their development organizations or individual projects" [SPEM2, Subclause 6.2], and developers are primarily seen as end-users of the models defined by these. While SPEM specifically supports "deployment of just the method content and process needed by defining configurations of processes and method content" and "libraries of reusable process and method" [SPEM2, Subclause 6.2] through "clear separation of method content definitions from the development process application of method content" [SPEM2, Subclause 6.3.1], it does not provide a common kernel for that content. SPEM itself does not, therefore, provide the complete foundation requested by this RFP.

On the other hand, the SPEM 2.0 Profile [SPEM2, Clause 7] could provide a starting point for defining the language required by this RFP. Nevertheless, there are some fundamental issues that would need to be addressed in doing this.

For example, while the SPEM 2.0 Base Plug-in includes a concept of "Practice" [SPEM2, Subclause 18.3.8] that is similar to the way the term is used in this RFP, SPEM currently defines this concept simply as a kind of "guidance" in a method plug-in outside the core of the specification. In contrast, the language requested in this RFP is

required to have "practice" as a foundational language construct used in the creation of methods.

Further, the work that led to SPEM started some time ago, before the rise of much of the recent agile and lean movement. Over the past ten years, we have come to better understand software practices and how to represent them in support of practitioners doing their tasks. Recent experience indicates that newer, innovative language constructs will be necessary in order to meet the requirements in this RFP for the definition and use of the kernel in the creation of practices and methods (see, for example, the discussion of Semat in Section 6.4.2 below).

Given these considerations, this RFP does not require that the requested foundational kernel and language be based on or be a revision of SPEM. Nevertheless, the RFP allows for the possibility of a submission that does so build on SPEM. It would, of course, be incumbent on a submitter proposing using SPEM in this way to show how this would be effective in meeting the RFP requirements, just as it would be similarly incumbent on a submitter proposing a different approach.

6.3.2 Architecture Ecosystem SIG (AESIG)

The Architecture Ecosystem SIG [AESIG] is a special interest group organized under the OMG Architecture Board. The mission of the Architecture Ecosystem SIG is to work with OMG domain and platform task forces, other relevant OMG SIGs, external entities and related industry groups to facilitate the creation of a *common architectural ecosystem*. This ecosystem will support the creation, analysis, integration and exchange of *information* between modeling languages across different domains, viewpoints and from differing authorities.

The aim of the ecosystem is to allow languages to be more modular, and allow the domain architect to be able to flexibly tailor and integrate these modular languages to create viewpoints that are appropriate for their domain of interest. This ecosystem will be the natural successor to the current MOF and UML profiling standards. Depending on the RFPs that result from the AESIG effort, submissions to this RFP may wish to consider relevant results in their definition of the required language.

6.3.3 Case Management Process Modeling (CMPM) RFP

The Case Management Process Modeling RFP [CMPM] solicits proposals for a metamodel extension to BPMN 2.0 to support modeling of case management processes. Case Management focuses on actions to resolve a case – a situation to be managed toward objectives. Cases do not have predefined processes for achieving objectives. Humans make decisions based on observations, experience and the case file. Changes in the state of the case will result in new actions. A practice or discipline may adopt rules to guide decisions and make processes more repeatable. New modeling paradigms are required to facilitate all this.

Case management is typically suited to manage knowledge work, in particular work that is associated with innovation activities and initiatives. Integration with CMPM could be one possible approach considered by submitters to this RFP to support enactment of methods. There is an opportunity to seek alignment with the CMPM specification where possible. However, in order to allow for different approaches for a domain-specific language, this RFP is not mandating the foundation for this to be based on CMPM.

6.3.4 Structured Metrics Metamodel (SMM)

The SMM specification defines a metamodel for representing measurement information related to software, its operation, and its design. The specification is an extensible meta-model for exchanging software-related measurement information concerning existing software assets (designs, implementations, or operations). A standard for the exchange of measures is important given the role that measures play in software engineering and design. The SMM is part of the Architecture Driven Modernization (ADM) roadmap and fulfills the metric needs of the ADM roadmap scenarios as well as other information technology scenarios.

6.4 Related non-OMG Activities, Documents and Standards

6.4.1 ISO/IEC 15288, ISO/IEC 24744 and ISO/IEC 12207

ISO/IEC 24744, Software Engineering Metamodel for Development Methodologies (SEMDM), establishes a formal framework for the definition and extension of development methodologies for information-based domains (IBD). This standard is intended to be used by method engineers while defining or extending development methodologies. ISO/IEC 12207 and ISO/IEC 15288 are related standards that only deal with process aspects of methodology and also focus on the method engineer. This RFP, on the other hand, is focused on the end-to-end practices, and the enactment stage conducted by software practitioners as described above in Section 6.2.

The ISO/IEC 24744 metamodel has been used by method engineers within the Situational Method Engineering (SME) community to define light-weight and flexible method approaches. The metamodel builds upon a small set of agreed core elements. The essential process components are stages, producers, work units and work products. The community has recently come to acknowledge two different characteristics of method enactment: (1) *tailored process*, essentially a static viewpoint, and (2) *performed process*, a more dynamic performance. This could provide a basis for a submission to this RFP.

6.4.2 Software Engineering Method and Theory (Semat)

Semat is an effort to “refound software engineering based on a solid theory, proven principles and best practices.” The Semat *Grand Vision* (the call for action statement) is supported by three dozen well-known individuals who have made significant

contributions to the software community, by a dozen major corporations and by more than 1400 supporters from all over the world.

The work that has started within the Semat community has been a major influence on this RFP, and the fundamental organization of the requested foundation into a kernel and a language has been adopted from this work. However, the need for the kernel and language being requested by this RFP has been justified in Sections 6.1 and 6.2 above independently of the wider Semat vision. One reason for issuing this OMG RFP is specifically to solicit participation beyond the current Semat community. Nevertheless, the ongoing experience of the Semat group provides some insight into ways in which a kernel and language such as requested in this RFP may need to go beyond what his been traditionally available for method and process definition.

For example, it was realized early in the Semat effort that, rather than just providing the means to define work products to be produced, it would be necessary to capture in the kernel a conceptualization of tracking project state at a higher level than the specific work products produced by any specific practice or method. This led to the introduction of *Abstract-Level Project Health Attributes,* or "Alphas", that "subsume and encapsulate work products at a higher level of abstraction." An Alpha is defined as "a property of the current state of a software project, satisfying the following criteria:

- It is relevant to an assessment of the project's health, that is to say, of the degree to which the project satisfies its stated objectives (such as deadlines, costs, quality).
- It can be determined, directly or indirectly, in terms of the current state of the project's work products. The indirect case means that the definition of an alpha may involve work products as well as previously defined alphas."

Of course, it is not required that a submission to this RFP adopt this concept of an Alpha in order to structure the proposed kernel. However, the concept is an example of how, at least in the experience of the Semat community, some innovation will be necessary in order to create a proper kernel that goes beyond previous approaches to method content definition.

The Semat effort will continue outside the scope of this RFP, in its current capacity as a self-organizing community. The usefulness of the result of the RFP to the general software development community, though, is not dependent on the future work of Semat, or even its continuation at all. Nevertheless, it is expected that an adopted OMG specification for the requested kernel and language will be highly synergistic, by intent, with the continued realization of the Semat vision.

6.4.3 Eclipse Process Framework (EPF) and Unified Method Framework (UMF)

The open-source Eclipse Project Framework (EPF) project (http://www.eclipse.org/epf) under the Eclipse Foundation represents an important community in existence today that claims to have implemented tool support and practices according to the existing SPEM specification [SPEM]. EPF aims at producing a customizable software process

engineering framework, with exemplary process content and tools, supporting a broad variety of project types and development styles.

EPF implements the Unified Method Framework (UMF)[1]. The UMF framework defines a method plugin architecture and provides a set of core infrastructure method elements, i.e. the core method plugins. The framework and its core method elements can be perceived as a practice-agnostic "EPF kernel" that can be used to define a wide variety of practices.

EPF comes with a set of defined practices[2] that is intended to be used by process engineers to learn about the practices in order to make decisions about which practices to include in a process configuration. EPF is an implementation of the SPEM 2.0 standard that has tried in this way to address some of the goals of this RFP. It may be possible to use this experience to develop a submission for this RFP along similar lines.

6.5 Mandatory Requirements

6.5.1 The Kernel

6.5.1.1 Domain model

The Kernel shall be represented as a domain model of a small number (expected to be closer to 10 than a 100) of essential concepts of software engineering and their relationships. The Kernel shall be expressed in the Language. (In the following when referring to the Language, we mean the language requested in this RFP.)

6.5.1.2 Key conceptual elements

The Kernel shall define the key conceptual elements that all software engineering endeavors have to monitor, sustain and progress, covering at least the following kinds of concepts (the specific grouping used here is not required):

a. *System:* Concepts related to the system being produced, for example: software, platform, etc.

b. *Functionality:* Concepts related to the required function of the system being produced, for example: requirements, needs, opportunities, stakeholders, etc.

c. *People:* Concepts related to the people required to create a system with the required functionality, for example: project, team, role, etc.

d. *Way of Working:* Concepts related to the way an organized team carries out its work to create a system with the required functionality, for example: method, practice, goal, etc.

[1] http://epf.eclipse.org/wikis/epfpractices/practice.bus.mdev.base/guidances/concepts/umf_62CAA5FA.html
[2] http://epf.eclipse.org/wikis/epfpractices/index.htm

6.5.1.3 Generic activities

The Kernel shall define the generic activities that a team will need to undertake to successfully engineer and produce a software system, covering at least the following kinds of activities (the specific grouping used here is not required):

a. *Interacting with stakeholders:* Activities related to necessary interactions with stakeholders, for example: exploring possibilities, understanding needs, ensuring satisfaction, handling change, etc.

b. *Developing the system:* Activities related to actually constructing a system, for example: specifying, shaping, implementing, testing, deploying and operating the system.

c. *Managing the project:* Activities related to managing a project, for example: steering the project, supporting the project team, assessing progress and concluding the project.

6.5.1.4 Kernel elements

The definition of each element of the Kernel shall include the following:

a. A concise description of the meaning of the element and its use in software engineering, intuitively understandable to a practitioner.

b. The relationships of the element to other elements in the Kernel.

c. The various different states the element may take over time, including initial/entry and final/exit criteria as appropriate for the element.

d. How the element is applied in practice, including how it may be instantiated, tailored or extended to support the work of a specific project team using specific practices.

e. How different ways of applying the element may be compared to each other and guidance on deciding among the alternatives.

f. Appropriate metrics that can be used to assess progress, quality, etc.

6.5.1.5 Scope and coverage

The Kernel shall be sufficient to allow for the definition of practices and methods supporting projects of all sizes and a broad range of lifecycle models and technologies used by significant segments of the software industry.

6.5.1.6 Extension

The Kernel shall also allow for extension, both in terms of addition of new elements and providing additional detail on existing elements that provide for practice-specific work products.

a. The Kernel shall allow for project and organization specific extensions.

b. The Kernel shall be tailorable to specific domains of application and to projects involving more than software, e.g., to serve as a basis for future extensions for systems engineering.

6.5.2 The Language

6.5.2.1 Language Definition

6.5.2.1.1 MOF metamodel

The Language shall have an abstract syntax model defined in a formal modeling language. The submission is expected to reflect this requirement in a description or mapping to the OMG architectural framework based on MOF.

6.5.2.1.2 Static and operational semantics

The Language shall have formal static and operational semantics defined in terms of the abstract syntax.

6.5.2.1.3 Graphical syntax

The Language shall have a graphical concrete syntax that formally maps to the abstract syntax. The submission is expected to reflect this requirement in a description following the Diagram Definition specification [DD] unless arguments are given for choosing something else.

6.5.2.1.4 Textual syntax

The Language shall also have a textual concrete syntax that formally maps to the abstract syntax.

6.5.2.1.5 SPEM 2.0 metamodel reuse

Proposals shall reuse elements of the SPEM 2.0 metamodel where appropriate. Where an apparently appropriate concept is not reused, proposals shall document the reason for creating substitute model elements.

6.5.2.2 Language Features

6.5.2.2.1 Ease of use

The Language shall be designed to be easy to use for practitioners at different competency levels:

a. Those that have very little modeling experience and quickly and intuitively need to understand and learn how to use the Language.

b. Intermediate users who are more advanced and willing to describe what kind of outcome they expect of their work.

c. Advanced users that can work with all aspects of the Language to model their complete software endeavor.

6.5.2.2.2 Separation of views for practitioners and method engineers

The Language shall provide features to express two different views of a method: the method engineer's view and the practitioner's view. The primary users of methods and practices are practitioners (developers, testers, project leads, etc.).

The proposal shall be accessible to both practitioners and method engineers, but should target the practitioners first and foremost. Extensions should support method engineers to effectively define, compose and extend practices, *without complicating* its usage by the practitioners.

6.5.2.2.3 Specification of kernel elements

The Language shall have features for specifying Kernel elements, including:

a. Formal and informal descriptions of the content and meaning of an element.

b. The relationship of the element of other elements.

c. States the element may take over time and the events that cause transitions among those states.

d. How the element is instantiated, including provisions for practice-specific tailoring of the element, and the basis for comparing different instantiations.

e. Metrics defined to assess various attributes of the use of the element.

6.5.2.2.4 Specification of practices

The Language shall have features for specifying practices in terms of Kernel elements, including:

a. Description of the particular cross-cutting concern addressed by the practice and the goal of the application of the practice.

b. The Kernel elements relevant to the practice and how they are instantiated for use in the practice, including any practice-specific tailoring of the elements.

c. Any work products required by and produced by the practice.

d. The expected progress of work under the practice, including progress states, the rules for transition between them and their relation to the states of relevant Kernel elements used in the practice. (For example, describing a practice that involves iterative development requires describing the starting and ending states of every iteration.)

e. Verification that the goal of the practice has been achieved in it application, particularly in terms of measurements of metrics defined for its elements.

6.5.2.2.5 Composition of practices

The Language shall have features for the composition of practices, to describe existing and new methods, including:

a. Identifying the overall set of concerns addressed by composing the practices.

b. Merging two elements from different practices that should be the same in the resulting practice, even if they have different contents defined in the practices being composed. (For example, a use case practice may have a work product called Use Case, with a name, a basic flow etc. A testing practice may have a work product called Testable Requirement with an identifier and a description. In the method resulting from composing these two practices, these two work products should be merged into one, where the name of the Use Case is the identifier of the Testable Requirement and the basic flow of the Use Case is the description of the Testable Requirement).

c. Separating two elements from different practices that should be different in the resulting practice, even though they may superficially seem to be the same. (For example, in a testing practice there may be a work product called Plan and in an iterative development practice there may also be a work product called Plan. In the method resulting from composing these two practices these two work products must be different – e.g., the Testing Plan vs. the Development Plan.)

d. Modifying an existing method by replacing a practice within that method by another practice addressing a similar cross-cutting concern.

6.5.2.2.6 Enactment of methods

The semantic definition of the Language shall support the enactment by practitioners of methods defined in the Language, for the purposes of

a. Tailoring the methods to be used on a project

b. Communicating and discussing practices and methods among the project team

c. Managing and coordinating work during a project, including modifications to the methods over the course of the project by further tailoring the use of the practices in the method

d. Monitoring the progress of the project

e. Providing input for tool support for practitioners on the project.

6.5.3 Practices

The focus of this RFP is on the Kernel and the Language that will be used to describe software engineering practices. Submissions are expected to demonstrate non-normative examples using the Kernel and the Language as defined in the submission.

6.5.3.1 Examples of Practices

a. Submissions shall provide working examples to demonstrate the use of the Kernel and Language to describe practices. Preferably these examples should be drawn from existing and well-known practices.

b. Submissions shall provide working examples to demonstrate the composing of practices into a method.

c. Submissions shall provide working examples to demonstrate how a method can be enacted.

d. Submission shall include a capability to demonstrate the operational execution of methods as a proof of concept.

It is expected that the example practices are well-structured and suited to demonstrate how well the proposed Kernel and Language can be used to define good-quality practices. Each example of practice shall:

a. be described on its own, independent from any other practice

b. be either explicitly defined as a continuous activity or have a clear beginning and end states

c. bring defined value to its stakeholders

d. be assessable; in other words, its description must include criteria for its assessment when used

e. include, whenever applicable, quantitative elements in its assessment criteria; correspondingly, the description must include suitable assessing metrics.

6.5.3.2 Existing Practices and Methods

Respondents shall provide a guideline for how existing SPEM-based practices and methods, and possibly other representations, can be migrated to the new proposed specification.

6.6 Optional Requirements

None

6.7 Issues to be discussed

a. Submissions shall include the definition of the Kernel along with any alternative options considered for the names and definitions of Kernel elements, and the reasons for not using such alternatives.

b. Submissions not based on SPEM 2.0 shall discuss why they did not use SPEM and clearly describe and demonstrate the main differentiators.

6.8 Evaluation Criteria

6.8.1 Objective

Submissions will be evaluated based on the following general criteria.

a. Support of the principal goal for the improvement of software and system products and methods.

b. Inclusion of only the essentials of software engineering.

c. Grounding of Kernel and Language on a solid theoretical basis

d. Ability to be applied to different scales of projects, from small to very large projects.

e. Justification by a clear rationale.

f. Ability to support agile/lean software development methods as well as traditional prescriptive methods. (It is imperative that this flexibility should not lead to increasing the complexity in the Language or the Kernel, rather, it is an integral part of them.)

6.8.2 Kernel and Language

The proposed Kernel and Language will be evaluated based on the following criteria.

a. Applicability to all software engineering efforts.

b. Ability to contribute in a positive and perceptible way to the quality of software products. Although the evaluation may take multiple years, it should be evaluated by comparing different approaches and making an objective assessment.

c. Applicability to all software engineers, regardless of their backgrounds, and their methodological camps (if any).

d. Precision of the definition of Kernel elements.

e. Applicability of Kernel elements, through precise guidelines that projects can apply.

f. Suitability of Kernel elements for quantitative evaluation of their application.

g. Comprehensiveness in capturing the Essence of Software Engineering, providing a Kernel with its essential elements that supports the crucial ways of working of software engineering teams.

h. Ability of experienced practitioners to use the ideas presented in the Kernel to execute a software endeavor without the need for further explicit guidance. (That is the Kernel with its essential elements should be useful in itself without the composition of practices.)

i. Ease of use by software practitioners. Descriptions written in the Language should be easy to understand by all its users. The Language should be designed for the developer community, not just process engineers and academics.

j. Coverage of relevant practices and their composition in today's methods.

k. The ability of guidance on practices and methods, as defined in the Language, to be queried, such that a practitioner can easily discover relevant guidance and have it presented in new and informative ways.

l. The descriptions are built in terms of the essential elements of the Kernel helping one of the principal goals: to avoid reinventing practices and methods.

m. Support for simulating the application of methods and practices, thus providing insight into the dynamics of the method.

n. Support for applying methods and practices (as described through the Language) in real projects.

o. Support for the comparison of methods and practices to see which are suitable for a given situation.

p. Support for assessing whether a project claims to apply a given method or practice (as described through the Language) actually does. (This problem is sometimes

referred to as "closing the gap" between what project teams say they do and what they actually do.)

q. Support for the measurement of practices and methods, both to enable performance evaluation and to guide evaluation and validation in research.

r. Ease with the way in which practices defined through the Language can be taught.

s. Support for quantitative assessment of all the relevant artifacts.

t. Support for mechanisms enabling the evolution of the Kernel.

u. The ability to add practices, levels of detail and lifecycle models.

6.8.3 Practices

Examples of practices will be used to evaluate the proposal. Specifically, to evaluate how well they demonstrate positive qualities of the proposed solution in terms of the above Kernel and Language evaluation criteria.

6.9 Other information unique to this RFP

NONE

6.10 RFP Timetable

The timetable for this RFP is given below. Note that the TF or its parent TC may, in certain circumstances, extend deadlines while the RFP is running, or may elect to have more than one Revised Submission step. The latest timetable can always be found at the OMG Work In Progress page at http://www.omg.org/schedules under the item identified by the name of this RFP. Note that "<month>" and "<approximate month>" is the name of the month spelled out; e.g., January.

Event or Activity	Actual Date
Preparation of RFP by TF	*ADTF – December, 2010 – June, 2011*
RFP placed on OMG document server	*"Four week rule" – May 23, 2011*
Approval of RFP by Architecture Board *Review by TC*	*June 23, 2011*
TC votes to issue RFP	*June 24, 2011*
LOI to submit to RFP due	*November 22, 2011*
Initial Submissions due and placed on	*February 22, 2012*

OMG document server ("Four week rule")	
Voter registration closes	*February 22, 2012*
Initial Submission presentations	*March 21, 2012*
Preliminary evaluation by TF	*March 21, 2012*
Revised Submissions due and placed on OMG document server ("Four week rule")	*August 15 , 2012*
Revised Submission presentations	*September 12, 2012*
Final evaluation and selection by TF *Recommendation to AB and TC*	*December 5, 2012*
Approval by Architecture Board *Review by TC*	*December 6, 2012*
TC votes to recommend specification	*December 7, 2012*
BoD votes to adopt specification	*December 7, 2012*

Appendix A References and Glossary Specific to this RFP

A.1 References Specific to this RFP

The following documents are referenced in this document:

[AESIG] "Architecture Ecosystem SIG", http://www.omgwiki.org/architecture-ecosystem/doku.php

[DD] "Diagram Definition, Version 1.0 – FTF Beta 1", OMG Document ptc/2010-12-18, http://www.omg.org/spec/DD/1.0/Beta1/

[BPMN2] "Business Process Model and Notation, Version 2.0", OMG Document formal/2011-01-03, http://www.omg.org/spec/BPMN/2.0/

[CMPM] "Case Management Process Modeling (CMPM) RFP", OMG Document bmi/09-09-23, http://www.omg.org/cgi-bin/doc?bmi/09-09-23

[Semat] "Software Engineering Method and Theory – A Vision Statement", http://www.semat.org/pub/Main/WebHome/SEMAT-vision.pdf

[SMM] "Architecture-Driven Modernization (ADM): Software Metrics Meta-Model (SMM) – FTF Beta 1", OMG Document ptc/2009-03-03, http://www.omg.org/spec/SMM/1.0/Beta1

[SPEM1] "Software Process Engineering Metamodel (SPEM) Specification, Version 1.0", OMG Document formal/02-11-14, http://www.omg.org/cgi-bin/doc?formal/02-11-14

[SPEM2] "Software & Systems Process Engineering Metamodel (SPEM) Specification, Version 2.0", OMG Document formal/2008-04-01, http://www.omg.org/spec/SPEM/2.0/

A.2 Glossary Specific to this RFP

Here is a list of some key terms specific to this RFP as they have been used in the RFP text. It is expected that a submission will contain a more comprehensive glossary of terms, and also might update these definitions if deemed necessary.

Method - A *method* is a systematic way of doing things in a particular discipline. For the purpose of this RFP, the relevant discipline is software engineering.

Practice - A *practice* is a general, repeatable approach to doing something with a specific purpose in mind, providing a systematic and verifiable way of addressing a particular aspect of the work at hand. It should have a clear goal expressed in terms of the results its application will achieve and provide guidance on what is to be done to achieve the goal and to verify that it has been achieved. Such practices may include specific approaches for software design, coding, testing at various levels, integration, organizing and managing the development team, etc. Practices are being defined using elements from the Kernel (see below).

Kernel - The Kernel includes essential elements of software engineering. The Kernel represents a *domain model* for software engineering that provides a common terminology of concepts and their relationships that may be used in the definition of software engineering practices. The Kernel is defined in terms of the Language (see below).

Language - In this document the Language is the standard modeling language requested by this RFP for specifying practices based on the Kernel and for composing methods from the practices. It can be used by a development team to both informally discuss and sketch their methods and then formalize those methods as they find appropriate. The Language has an abstract syntax, static and operational semantics. It also has a concrete lexical syntax and a concrete graphical syntax.

Appendix B General Reference and Glossary

B.1 General References

The following documents are referenced in this document:

[BCQ] OMG Board of Directors Business Committee Questionnaire, http://doc.omg.org/bc/07-08-06

[CCM] CORBA Core Components Specification, *http://www.omg.org/technology/documents/formal/components.htm*

[CORBA] Common Object Request Broker Architecture (CORBA/IIOP), *http://www.omg.org/technology/documents/formal/corba_iiop.htm*

[CSIV2] [CORBA] Chapter 26

[CWM] Common Warehouse Metamodel Specification, *http://www.omg.org/technology/documents/formal/cwm.htm*

[DAIS] Data Acquisition from Industrial Systems, *http://www.omg.org/technology/documents/formal/dais.htm*

[EDOC] UML Profile for EDOC Specification, *http://www.omg.org/techprocess/meetings/schedule/UML_Profile_for_EDOC_FTF.html*

[EJB] "Enterprise JavaBeans™", *http://java.sun.com/products/ejb/docs.html*

[FORMS] "ISO PAS Compatible Submission Template". *http://www.omg.org/cgi-bin/doc?pas/2003-08-02*

[GE] Gene Expression, *http://www.omg.org/technology/documents/formal/gene_expression.htm*

[GLS] General Ledger Specification , *http://www.omg.org/technology/documents/formal/gen_ledger.htm*

[Guide] The OMG Hitchhiker's Guide,, *http://www.omg.org/cgi-bin/doc?hh*

[IDL] ISO/IEC 14750 also see [CORBA] Chapter 3.

[IDLC++] IDL to C++ Language Mapping, *http://www.omg.org/technology/documents/formal/c++.htm*

[Inventory] Inventory of Files for a Submission/Revision/Finalization, *http://doc.omg.org/smsc/2007-09-05*

[MDAa] OMG Architecture Board, "Model Driven Architecture - A Technical Perspective*", http://www.omg.org/mda/papers.htm*

[MDAb] "Developing in OMG's Model Driven Architecture (MDA)," *http://www.omg.org/docs/omg/01-12-01.pdf*

[MDAc] “MDA Guide” (*http://www.omg.org/docs/omg/03-06-01.pdf*)

[MDAd] “MDA "The Architecture of Choice for a Changing World™"”, *http://www.omg.org/mda*

[MOF] Meta Object Facility Specification, *http://www.omg.org/technology/documents/formal/mof.htm*

[MQS] “MQSeries Primer”, *http://www.redbooks.ibm.com/redpapers/pdfs/redp0021.pdf*

[NS] Naming Service, *http://www.omg.org/technology/documents/formal/naming_service.htm*

[OMA] “Object Management Architecture™”, *http://www.omg.org/oma/*

[OTS] Transaction Service, *http://www.omg.org/technology/documents/formal/transaction_service.htm*

[P&P] Policies and Procedures of the OMG Technical Process, *http://www.omg.org/cgi-bin/doc?pp*

[PIDS] Personal Identification Service, *http://www.omg.org/technology/documents/formal/person_identification_service.htm*

[RAD] Resource Access Decision Facility, *http://www.omg.org/technology/documents/formal/resource_access_decision.htm*

[RFC2119] IETF Best Practices: Key words for use in RFCs to Indicate Requirement Levels, (*http://www.ietf.org/rfc/rfc2119.txt*).

[RM-ODP] ISO/IEC 10746

[SEC] CORBA Security Service, *http://www.omg.org/technology/documents/formal/security_service.htm*

[TOS] Trading Object Service, *http://www.omg.org/technology/documents/formal/trading_object_service.htm*

[UML] Unified Modeling Language Specification, *http://www.omg.org/technology/documents/formal/uml.htm*

[UMLC] UML Profile for CORBA, *http://www.omg.org/technology/documents/formal/profile_corba.htm*

[XMI] XML Metadata Interchange Specification,
http://www.omg.org/technology/documents/formal/xmi.htm

[XML/Value] XML Value Type Specification,
http://www.omg.org/technology/documents/formal/xmlvalue.htm

B.2 General Glossary

Architecture Board (AB) - The OMG plenary that is responsible for ensuring the technical merit and MDA-compliance of RFPs and their submissions.

Board of Directors (BoD) - The OMG body that is responsible for adopting technology.

Common Object Request Broker Architecture (CORBA) - An OMG distributed computing platform specification that is independent of implementation languages.

Common Warehouse Metamodel (CWM) - An OMG specification for data repository integration.

CORBA Component Model (CCM) - An OMG specification for an implementation language independent distributed component model.

Interface Definition Language (IDL) - An OMG and ISO standard language for specifying interfaces and associated data structures.

Letter of Intent (LOI) - A letter submitted to the OMG BoD's Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements.

Mapping - Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.

Metadata - Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM.

Metamodel - A model of models.

Meta Object Facility (MOF) - An OMG standard, closely related to UML, that enables metadata management and language definition.

Model - A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA) - An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Normative - Provisions that one must conform to in order to claim compliance with the standard. (as opposed to non-normative or informative which is explanatory material that is included in order to assist in understanding the standard and does not contain any provisions that must be conformed to in order to claim compliance).

Normative Reference – References that contain provisions that one must conform to in order to claim compliance with the standard that contains said normative reference.

Platform - A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM) - A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM) - A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Request for Information (RFI) - A general request to industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's Technology Committee subgroups.

Request for Proposal (RFP) - A document requesting OMG members to submit proposals to an OMG Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing Task Force.

Task Force (TF) - The OMG Technology Committee subgroup responsible for issuing a RFP and evaluating submission(s).

Technology Committee (TC) - The body responsible for recommending technologies for adoption to the BoD. There are two TCs in OMG – the *Platform TC* (PTC) focuses on IT and modeling infrastructure related standards; while the *Domain TC* (DTC) focuses on domain specific standards.

Unified Modeling Language (UML) - An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile - A standardized set of extensions and constraints that tailors UML to particular use.

XML Metadata Interchange (XMI) - An OMG standard that facilitates interchange of models via XML documents.