A Template for Documenting Prediction-Enabled Component Technologies

Paulo Merson

October 2003

Predictable Assembly from Certifiable Components Initiative

Technical Note
CMU/SEI-2003-TN-030

Contents

List of Figures

List of Tables

Abstract

Prediction-enabled component technology (PECT) is an approach to predicting the behavior of systems built from components with known properties. An important artifact produced by the PECT development process is the documentation of the technologies, tools, and theories as integral elements of the PECT, as well as the results and conclusions of the application of the PECT to a group of systems. This report suggests a template for documenting a PECT. The report also provides guidelines and a few examples to help PECT developers consolidate the broad range of information produced in the PECT development process into a single, organized volume.

1 Introduction

The Predictable Assembly from Certifiable Components (PACC) Initiative at the Software Engineering Institute (SEI) investigates the technology and methods for reliably predicting the runtime behavior of assemblies of components from their certifiable properties. Our approach to achieving predictable assemblies from certifiable components is based on the development of prediction-enabled component technologies (PECTs).

A PECT comprises a construction framework and one or more reasoning frameworks [Wallnau 03a]. The construction framework is associated with a specific component technology, which provides the means to construct assemblies. A reasoning framework is based on a computational theory and is used to analyze the behavior of the assemblies with respect to specific runtime properties.

Two previous applications of the PECT approach have been documented [Hissam 01, Hissam 02a]; a third, larger, application is under development. Based on previous experience, it became clear that the wide spectrum of information necessary to document a PECT should be captured in a standard, easy-to-reference format.

1.1 About This Report

1.1.1 Objective

The goal of this report is to provide a template for documenting a PECT, with instructions and examples that should help a PECT developer to capture all the relevant information about the construction framework and reasoning frameworks involved.

Wallnau lays out the key concepts and precepts that govern the development of a PECT [Wallnau 03a]. This report is complementary in the sense that it provides a format for compiling all the information involved in an actual PECT.

As described by Hissam, different stakeholders perform different activities in the PECT development process [Hissam 02b]. Specialized roles for a team of PECT developers include PECT designer, component technology specialist, analysis specialist, and PECT validator. All of these roles contribute to the PECT documentation, each one being responsible for different sections. The developers themselves use the documentation, along with all the other artifacts produced in the development of the PECT, to evolve or reuse that PECT.

A separate class of stakeholders, the PECT users, is also interested in the PECT description. They need much less information (for example, users are not concerned with interpretation or the details of the decision or validation procedures). The template and guidelines in this report address the needs of both PECT developers and PECT users.

1.1.2 Organization

Section 2 provides some general guidelines for documenting a PECT. Section 3 presents the template for a PECT Technical Reference Guide, which is the documentation by and for PECT developers. Each subsection corresponds to a section of that guide. Section 4 describes how to create a PECT User's Guide, which is a simplified version of the documentation that is suitable for PECT users. Finally, Section 5 presents plans for future work and related activities.

2 Guidelines for Documenting PECTs

The development of a PECT involves the application of diversified expertise (e.g., assembly development, runtime environment configuration, behavioral modeling and analysis, statistical validation) and produces a tremendous amount of information. It is vital to capture the essential information produced in an effective way.

When documenting a PECT, a developer should follow the general guidelines below:

- Use colloquial and clear language to communicate with the reader; avoid long, cluttered sentences.
- Be objective and succinct.
  - Assume the reader is familiar with the concepts behind predictable assembly from certifiable components. As a rule of thumb, assume the reader is familiar with the key technical concepts of PECT [Wallnau 03a]. In particular, the reader should understand the elements and relations of a PECT as pictured in Figure 1.
  - Do not try to explain the computational theory used in the reasoning frameworks. Assume the reader has the requisite basic knowledge or provide references.
  - When needed, use bulleted or numbered lists, tables, and pictures rather than narrative text to make the documentation easier to understand and faster to read.
- Make extensive use of examples. If an example describes a specific situation and there is a possible variation that is relevant, create another example to describe it. In the template, examples are placed in a separate section, so the number of examples should not affect the readability of the explanatory sections of the document.
- It is imperative that the PECT documentation be maintained under some sort of version or content management, so that changes are controlled, inconsistencies are avoided and the history of changes can be retrieved when necessary. The PECT documentation consolidates pieces of information produced by various players in the PECT development process. Therefore, it is likely that the contents will be created incrementally in a very dynamic fashion.
- If you are not completely sure about the validity of an assertion in the documentation—perhaps because it is something that will be decided later—record it anyway. However, use a question mark (“?”) or some other symbol to indicate that the point must still be clarified. Later, you can search for question marks and remove them, or restate the points as appropriate.
- Where possible, use the terminology associated with the Construction and Composition Language (CCL). CCL provides the means to express structural, behavioral, and

analysis-specific information, and is used to create specifications of components, assemblies, and runtime environments [Wallnau 03b]. CCL permeates most of the work in the development of a PECT and provides a common vocabulary for communicating ideas.

- As more PECTs are documented, it is likely and desirable that the template itself will evolve to incorporate new ways to organize information. Propose modifications to the template if you encounter deficiencies in the current version. The PACC Initiative at the SEI fosters the improvement of all artifacts related to PECT development.

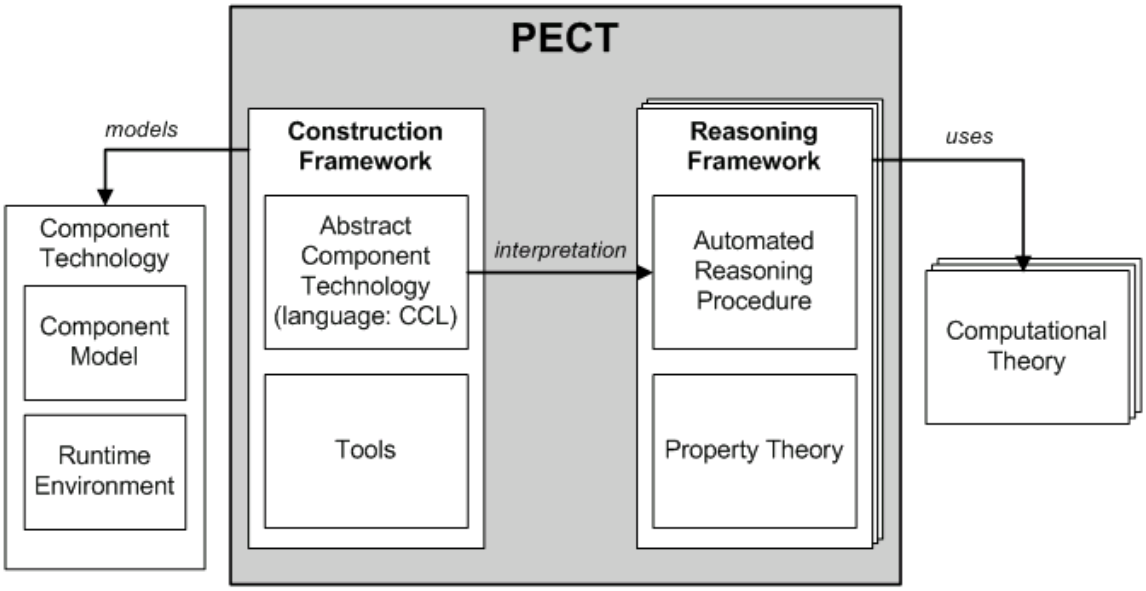

Figure 1: Logical Structure of PECT

3 Template for a PECT Technical Reference Guide

The Technical Reference Guide is a single volume that contains all the relevant technical information about a PECT. It is created, maintained, and used by PECT developers. Detailed descriptions of each section and subsection that should compose the Technical Reference Guide of a PECT are presented in sections 3.1 through 3.5 of this report. Note that Section 3.1 of this report corresponds to Section 1 of the Technical Reference Guide, Section 3.2.2 corresponds to Section 2.2 of the guide, Section 3.3.3.1 corresponds to Section 3.3.1 of the guide, and so on.

Each version of the Technical Reference Guide will contain a title and revision notice that uniquely identify the document. Revision information may include a version number and a list of modifications containing the date, description, and author. If the document is maintained by a version control system, revision history information can be retrieved from the version control tool.

Additional sections and subsections may be included in the Technical Reference Guide to provide information not covered in the template. For instance, other appendices might be added as needed.

Figure 2 outlines the template for a PECT Technical Reference Guide. A PECT consists of a construction framework and one or more reasoning frameworks. Each reasoning framework is documented in a separate section (3, 4, etc.) using the same template.

Template for a PECT Technical Reference Guide

Table of Contents

List of Figures

List of Tables

1. Introduction

2. Construction Framework

    2.1 Component Technology

    2.2 Constraints

        2.2.1 Constructive Constraints

        2.2.2 Analytic Constraints – *XXX*

        2.2.3 Analytic Constraints – *YYY*

        . . .

    2.3 Required Properties

        2.3.1 Properties Required for Construction

        2.3.2 Required Properties – *XXX*

        2.3.3 Required Properties – *YYY*

        . . .

3. Reasoning Framework *XXX*

    3.1 Property Theory Concepts

    3.2 Interpretation

    3.3 Prediction

        3.3.1 Predicted Properties

        3.3.2 Decision Procedure

    3.4 Validation Procedure

    3.5 Reasoning Framework Infrastructure

    3.6 Results and Conclusions

4. Reasoning Framework *YYY*

    4.1 Property Theory Concepts

    . . .

Appendix – Examples

References

Figure 2: PECT Technical Reference Guide Template ("XXX" and "YYY" represent the name of the reasoning frameworks)

3.1 Introduction (Section 1 of the Technical Reference Guide)

The introduction section provides the motivation and goals of this PECT, as well as the context in which it was developed (e.g., research investigation, software development project). Many of the goals and requirements of a PECT can be expressed in terms of the properties that it can predict. Summarize them here and point the reader to Section 3.3.1 (and 4.3.1, 5.3.1, etc.) of the Technical Reference Guide where the predicted properties are described in full detail.

Give the key characteristics that distinguish this PECT, without further details. For example, a PECT that has a reasoning framework that is suitable for performance analysis may be characterized as supporting *aperiodic tasks* using *sporadic server*, tasks with *variable execution time*, *distribution* using an *Ethernet* local area network, *asynchrony*, and *blocking*. If the PECT has an acronym or a name that follows a naming convention that is related to key properties, explain the PECT designation. For example, in "λ_{ABA}," "ABA" stands for "average case latency," "blocking," and "asynchrony," and lambda represents "latency" [Hissam 02a].

Describe very briefly the reasoning frameworks used by this PECT. You can simply indicate the theories used and what they are used for.

Comment about the relationship of this PECT with other PECTs, or the relationship of the reasoning frameworks in this PECT with other reasoning frameworks that were or will be developed. For example, indicate if the present work is an evolution from previous work, or if it is part of a "roadmap" that defines a planned series of PECTs.

3.2 Construction Framework (Section 2 of the Technical Reference Guide)

This section of the Technical Reference Guide describes the component technology and runtime environment, specifies the constraints that apply in construction, and identifies the properties required from construction elements.

3.2.1 Component Technology (Section 2.1 of the Technical Reference Guide)

Identify and briefly describe the software component technology used to build components in this PECT. Characterize the component model: the application programming interfaces (APIs) that can be used, constraints imposed on the implementation of components,[1] the

[1] These refer to well-formedness rules imposed by the original component model (for example, the Enterprise JavaBeans (EJB) component model establishes that the session bean class of a stateless session bean must have a public constructor that takes no parameters [Sun 01]). In addition to constraints inherent to the component model, the PECT will impose other constraints that are described in Section 2.2 of the Technical Reference Guide.

deployment process, the life cycle of components at runtime, and other characteristics. Describe the key characteristics of the runtime environment that is used to deploy and test assemblies, including the interaction mechanisms (for example, inter-thread communication, inter-process communication, signals) and services provided by the runtime environment. At the implementation level, indicate any reference components (a skeleton of components or super classes) that are available for deriving new components, as well as library components or services that can be used to create assemblies. Point the reader to bibliographical references where detailed documentation about the component technology can be found.

A component technology can be described in terms of a component model and a component runtime environment [Bachmann 00]. These two elements could be explained in separate subsections. However, we perceive that the pieces of information that characterize the component model and the runtime environment are highly interconnected and it is more practical to describe both in the same section.

3.2.2 Constraints (Section 2.2 of the Technical Reference Guide)

This subsection lists the constraints that apply to the creation of assemblies for this PECT, excluding the constraints that are innate to the component model—that is, constraints that apply to any system that uses that component model. The constraints in this subsection are specific to this PECT and are divided into constructive and analytic constraints. Wallnau has written about the distinction between constructive and analytic constraints [Wallnau 03a]. Some constraints may refer to the required properties defined in Section 2.3 of the Technical Reference Guide.

3.2.2.1 Constructive Constraints (Section 2.2.1 of the Technical Reference Guide)

Specify here the constraints that are imposed on assemblies so that they can be effectively constructed—that is, compiled, linked, deployed, and executed. All assemblies should meet the constructive constraints, regardless of the reasoning framework that is used to analyze those assemblies.

The constraints should be described in a way that guides the creation of a CCL specification—that is, they should provide the well-formedness rules for the characterization of pins, reactions, runtime services, gateways, and CCL elements [Wallnau 03b]. For example, if you want to point out that every component in a system must be invoked by some other component or activated by a clock, you could write “all components in an assembly shall have at least one sink pin that interacts with the source pin of another component or a clock.”

3.2.2.2 Analytic Constraints – Reasoning Framework (Sections 2.2.2, 2.2.3, etc. of the Technical Reference Guide)

Specify here the analytic constraints that are imposed by a specific reasoning framework to ensure that the assembly is analyzable. Use CCL terms for analytic constraints; for example,

"the mean value of the interarrival interval between messages sent to the sink pin of a sporadic server component shall be greater than the replenishment period of that component."

In the title of the subsection, indicate the reasoning framework to which the constraints apply. Thus, if there are two or more reasoning frameworks in the PECT, there will be more than one "Analytic Constraints" section. Number them sequentially and use a name that designates the reasoning framework; for example,

2.2.2 Analytic Constraints – Performance

2.2.3 Analytic Constraints – Model Checking

2.2.4 Analytic Constraints – Reliability

3.2.3 Required Properties (Section 2.3 of the Technical Reference Guide)

List here the properties that are required from components used in this PECT. The first subsection gives the properties required from all assemblies, regardless of the reasoning frameworks that are used to analyze them. In the subsequent subsections, the properties demanded by specific reasoning frameworks are specified.

3.2.3.1 Properties Required for Construction (Section 2.3.1 of the Technical Reference Guide)

A required property is necessarily associated with a constructive element through a CCL property annotation. For each property, indicate the referent element, a property name, a range of valid values and, if applicable, the units of measure. The information can be presented in the form of a table, such as the one shown in Table 1. The first row reflects the fact that a clock (a runtime service) must be configured with a predefined period value; otherwise it cannot be instantiated by the runtime.

A property is an n-tuple, that is, a property name can be associated with one or more values, and each value can have a different format and range. The second row in Table 1 is an example of a multi-valued property that is the period of a special type of clock called a random clock, which generates pulses at variable intervals.

The value of some properties (for example, execution time of a reaction) results from a certification process that uses empirical or formal evidence to obtain a certain level of trust. Other properties are simply configurable parameters of the components (for example, the period of a clock). As appropriate, you may indicate which properties pertain to each category.

Table 1: Example of Required Property

Referent	Property Name	Valid Values	Unit
Clock source pin	period	Positive integer	millisecond
Random clock source pin	randomPeriod	• Positive integer for mean value • Positive integer for standard deviation • Name of the distribution to be used (“exponential,” “uniform,” or “poisson”)	• millisecond • millisecond • string

3.2.3.2 Required Properties – Reasoning Framework (Sections 2.3.2, 2.3.3, etc., of the Technical Reference Guide)

List here the additional properties that are required by a given reasoning framework. Again, these are properties that are annotated in the CCL specification and can be described here in a table format. Table 2 shows an example.

Table 2: Example of Required Property for Performance Reasoning

Referent	Property Name	Valid Values	Unit
Reaction	fixedExecutionTime	• Real number for the mean • Real number for standard deviation	• millisecond • millisecond

In the title of the subsection, indicate the reasoning framework with a name. If there are two or more reasoning frameworks in the PECT, create multiple subsections numbered sequentially; for example,

2.3.2 Required Properties – Performance
2.3.3 Required Properties – Model Checking
2.3.4 Required Properties – Reliability

3.3 Reasoning Framework – Name (Sections 3, 4, etc., of the Technical Reference Guide)

This section of the Technical Reference Guide describes a specific reasoning framework, presenting the property theory concepts and describing interpretation, prediction, and the validation procedure.

Use the name or designation of the reasoning framework in the title of the section. If there are two or more reasoning frameworks in the PECT, create multiple sections numbered sequentially. Subsections of each reasoning framework section are numbered accordingly; for example,

3 Reasoning Framework – Performance
  3.1 Property Theory Concepts
  …
4 Reasoning Framework – Model Checking
  4.1 Property Theory Concepts
  …
5 Reasoning Framework – Reliability
  5.1 Property Theory Concepts
  …

3.3.1 Property Theory Concepts (Section 3.1 of the Technical Reference Guide)

Use this subsection to explain the technical terms that are used in the reasoning framework, associating them to PECT concepts and construction elements as necessary. Besides prose, another possible format is a glossary that lists the terms and gives short descriptions. Do not try to describe concepts in full detail; provide the reader with minimal information about the concepts and terminology used in that reasoning framework, and use bibliographical references to complement that information.

In addition to the definition of key terms, use this subsection to explain concepts and procedures that are peripheral to the reasoning framework (e.g., if your validation procedure uses Monte Carlo simulation, you may want to use this subsection to explain what that is).

More than in other parts of the Technical Reference Guide, this subsection requires you to write from the reader's point of view. That demands careful reflection about who readers are and why they are reading the document. The reader can be a PECT developer, who is a specialist in one area and contributes a part of the document but also needs to read and understand the parts he or she did not write. The reader can be a PECT developer who will create a new PECT based on an old one; he or she will rely on the documentation of the old PECT to reuse artifacts and reapply concepts. In any case, assume that readers have basic knowledge about the computational theory and use common sense to discern the right amount of information to provide here.

3.3.2 Interpretation (Section 3.2 of the Technical Reference Guide)

Describe how elements in a CCL specification are mapped to elements in the property theory. In particular, list how properties of construction elements map to element attributes in the post-interpretation model. For instance, a reaction (specified in CCL) may translate to a subtask in a performance analysis reasoning framework, and the "execution time" property of a reaction (specified in a CCL property annotation) may translate to the "execution time" property of the subtask in the reasoning framework. A table with the construction elements in one column and the reasoning framework elements in the other is an appropriate format, although the relation is not always one to one.

An interpretation must provide an unambiguous (and automated) translation from a well-formed assembly specified in CCL to the correspondent, analyzable model in the property theory. Thus, a desirable piece of documentation is the syntax-directed translation, which specifies, for each syntactic production, semantic rules in terms of actions on syntactic elements and their properties.

Explain the algorithms and transformations used in the interpretation. They should reflect the actual code written for interpretation, which follows the syntax-directed translation but may also include some preprocessing, rewrite rules, and optimizations. Pseudocode is a suitable format for presenting the key parts of the algorithms.

An efficient way to explain what interpretation does is to provide examples of the translation of entire assemblies specified in CCL into elements in the property theory. Therefore, make use of examples that illustrate various situations and configurations. (Examples should be in the appendix of the Technical Reference Guide.)

3.3.3 Prediction (Section 3.3 of the Technical Reference Guide)

This subsection lists the properties that the reasoning framework can predict and describes the decision procedure used to generate the predictions.

3.3.3.1 Predicted Properties (Section 3.3.1 of the Technical Reference Guide)

List the properties that this reasoning framework can predict. Here you should explain what each property means rather than give predicted values for some execution of the prediction. Defer the presentation of actual results to Section 3.6 of the Technical Reference Guide. Indicate whether the predicted property is a claim that evaluates to true or false, or is a quantifiable measure, in which case you should describe units of measure and desired accuracy.

In general, a reasoning framework can provide a prediction for any well-formed assembly, but in some cases, a property can be predicted only if the assembly possesses some specific characteristics. For example, a given performance reasoning framework may predict the

probability of queue overflow only when the assembly contains aperiodic tasks that process a FIFO queue. Use this subsection to document any relation of that nature. Also note situations where a predicted property is derived from another predicted property.

3.3.3.2 Decision Procedure (Section 3.3.2 of the Technical Reference Guide)

Describe the automated procedure used to generate the predictions. Use pseudocode, state diagrams, UML diagrams, or other symbology that clarifies how the decision procedure operates.

Characterize the decision procedure as being analytic, simulation based, or hybrid. Analytic decision procedures are often strongly associated with formulae and algorithms in the property theory. Describe here such formulae and algorithms and how they are adapted and combined in the context of this reasoning framework.

3.3.4 Validation Procedure (Section 3.4 of the Technical Reference Guide)

Discuss the rationale for the choice of property theory and decision procedure. Then, provide the reasons for trust in the validity of the predictions provided by the reasoning framework. If predicted results are compared against well-defined measures obtained through direct observation, describe how this empirical evidence is used to achieve confidence in the results. Also, explain what is required to achieve statistical trust.

If predicted results are obtained through mathematical proof, they are irrefutable in principle. Nevertheless, it is important to comment on the level of confidence in which the decision procedure and the tools used to generate predictions are held.

Present the reasons for trusting that the specification of an assembly in CCL is a valid abstraction of the actual assembly implementation. Explain how construction rules and constraints are verified. Also, if applicable, describe why interpretation, as defined in Section 3.2 of the Technical Reference Guide, is correct.

Use this section to describe how validation is performed rather than to present actual validation results. In Section 3.6 of the Technical Reference Guide, the prediction results are presented along with validation data (e.g., statistical analysis).

A separate issue in the validation is the confidence in the required properties of components (i.e., certification). The goal of a PECT is not to establish trust in certified properties, so you do not need to explain here how component properties were certified.

3.3.5 Reasoning Framework Infrastructure (Section 3.5 of the Technical Reference Guide)

Describe the infrastructure that was built to support the reasoning framework. Provide a short user guide to the tools and artifacts involved in the generation and validation of results, indicating filenames, parameters, scripts, input and output artifacts, and other elements. Include a flowchart or activity diagram that depicts the sequence of steps needed to produce assemblies with validated predictions from certified components.

It is likely that diverse kinds of tools and processes will be required by different reasoning frameworks, so the contents of this subsection may vary significantly and may or may not include the following:

- tools used in the decision procedure, either off the shelf or developed specifically for this reasoning framework
- measurement infrastructure, comprising instrumentation APIs and other artifacts used to collect empirical evidence
- statistical analysis tools
- assembly-generation tool, in case a representative sample of assemblies is created automatically
- simulation tools
- parsers, compilers, and interpretation translators

3.3.6 Results and Conclusions (Section 3.6 of the Technical Reference Guide)

Report the results of applying this PECT to a specific set of assemblies—a sample space. Define the validation goal used to determine whether results were successful. In the case of empirical validation, the goal is typically expressed in nominal statistical tolerance intervals. Characterize the assemblies used to produce the results and tell why they were chosen. Present all findings and/or nominal results of the experiment in the most appropriate format, which may include measures, execution traces, and other findings. Consolidate the results and analyze them to produce final conclusions.

This subsection is optional in the sense that a PECT can be developed and documented without a specific set of assemblies in mind. In fact, the PECT can be applied to a different set of assemblies, producing different results and conclusions. In that case, this subsection should be subdivided accordingly; for example,

3.3.6.1 Results and Conclusions – Assemblies in Set A_1

3.3.6.1 Results and Conclusions – Assemblies in Set A_2

3.3.6.1 . . .

3.4 Examples (Appendix of the Technical Reference Guide)

Identify and describe all examples used in the Technical Reference Guide. Rather than inserting partial examples in various sections, consider creating more complete examples in this section, and then add references to them in the appropriate parts of the document.

Typically this section would present examples of CCL specifications of assemblies, along with the results of interpretation and prediction for those assemblies.

3.5 References (of the Technical Reference Guide)

A reasoning framework makes use of a computational theory that is probably a body of knowledge fully documented in books and papers. Likewise, there may be books, manuals, and Web pages about the component technology used in the construction framework. In this section, list the bibliographical entries that were used as sources of information in the development of the PECT.

In addition to simply listing a title, provide comments explaining which chapters, sections, or pages are most relevant with respect to this reasoning framework, what specific information was obtained from that reference, and other helpful comments.

4 Template for a PECT User's Guide

The Technical Reference Guide described in Section 3 encompasses information produced and used by PECT developers. It gives details that may not be relevant or easily understood by the users of a PECT. If the PECT is going to be made available to users that are not part of the PECT development team, a separate document should be produced to bundle the information that describes the basics of the PECT, how it is used, and the benefits and results that it can produce.

The structure of the PECT User's Guide is similar to the Technical Reference Guide, but the contents should reflect the different needs and background of PECT users and developers. In particular, a PECT user is less interested in the mechanisms used to generate predictions—that is, the theory behind a reasoning framework and how it is applied to components and assemblies. Furthermore, if the PECT comprises two or more reasoning frameworks, it is probably more practical from the user's perspective to produce two or more user's guides, one for each reasoning framework. That way, each reasoning framework appears as a separate product that can be applied independently.

Figure 3 shows the template for the PECT User's Guide. The guidelines presented in Section 3 of this report can be used at large in the creation of the corresponding sections of that guide, keeping in mind the characteristics of the readers of the two documents. Table 3 lists side by side the sections in the Technical Reference Guide and in the User's Guide, with comments about noticeable differences.

Template for a PECT User's Guide

Table of Contents

List of Figures

List of Tables

1. Introduction

2. Construction Framework

    2.1 Component Technology

    2.2 Constraints

    2.3 Required Properties

3. Reasoning Framework *XXX*

    3.1 Property Theory Concepts

    3.2 Predicted Properties

    3.3 Validation Procedure

    3.4 Reasoning Framework Infrastructure

    3.5 Results and Conclusions

Appendix – Examples

References

Figure 3: PECT User's Guide Template ("XXX" and "YYY" represent the name of the reasoning frameworks)

Table 3: Technical Reference Guide and User's Guide Compared

Section of the Technical Reference Guide	**Section of the User's Guide**	**Comment**
1. Introduction	1. Introduction	similar, but omitting details of the development of the PECT in the User's Guide
2. Construction Framework	2. Construction Framework	similar
2.1 Component Technology	2.1 Component Technology	similar
2.2 Constraints 2.2.1 Constructive Constraints 2.2.2 Analytic Constraints – Reasoning Framework	2.2 Constraints	The Constraints section of the User's Guide should present the constructive and analytic constraints together. For a PECT user, the distinction is not important.

2.3 Required Properties 2.3.1 Properties Required for Construction 2.3.2 Required Properties – Reasoning Framework	2.3 Required Properties	The Required Properties section of the User's Guide should list together the properties required for construction and required by the reasoning framework. For the PECT user, the distinction is less relevant.
3. Reasoning Framework	3. Reasoning Framework	A PECT user is less interested in details of the reasoning framework and more interested in the results it can produce. Therefore, this section and its subsections in the User's Guide will be much simpler than in the Technical Reference Guide.
3.1 Property Theory Concepts	3.1 Property Theory Concepts	In the User's Guide, document only the concepts that are relevant to a PECT user, such as concepts reflected in the required properties or predicted properties.
3.2 Interpretation	N/A	The User's Guide should not describe interpretation.
3.3 Prediction 3.3.1 Predicted Properties 3.3.2 Decision Procedure	3.2 Predicted Properties	In the User's Guide, listed simply list the predicted properties in Section 3.3.1 of the Technical Reference Guide.
3.4 Validation Procedure	3.3 Validation Procedure	The PECT user may be interested in the reasons for trusting the predicted results. Therefore, in the User's Guide, describe the aspects of validation that may concern a PECT user. In addition, you may use the validation of actual results as a reason for trust.
3.5 Reasoning Framework Infrastructure	3.4 Reasoning Framework Infrastructure	In the User's Guide, emphasize the instructions on the tools that the PECT user has to execute to perform prediction.
3.6 Results and Conclusions	3.5 Results and Conclusions	similar
Examples	Examples	In the User's Guide, create examples that illustrate the use of the PECT.
References	References	similar

Table 3: Technical Reference Guide and User's Guide Compared (cont.)

5 Future Work

This report represents our current best thinking regarding a complete and structured model for documenting a PECT. It is the result of our experience with PECTs. Our objective now is to apply the template in current and future PECT developments. The process, as well as concepts and even terminology, is rapidly evolving in this initial stage of the technology. The Technical Reference Guide and the User's Guide are core artifacts produced by the PECT process and we, as PECT developers, will review the process and its artifacts every time it is performed.

The Technical Reference Guide template is currently being applied to document PECTs that contain performance reasoning frameworks and are being developed in the context of the robotics model problem investigation for ABB Ltd.

We expect that the consolidated PECT documentation to be produced in the near future will provide not only insight to enhance the template, but also comprehensive examples to PECT developers who use the template.

References

[Bachmann 00] Bachmann, F.; Bass, L.; Buhman, C.; Comella-Dorda, S.; Long, F.; Robert, J.; Seacord, R.; & Wallnau, K. *Volume II: Technical Concepts of Component-Based Software Engineering, 2nd Edition* (CMU/SEI-2000-TR-008, ADA379930). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr008.html>.

[Hissam 01] Hissam, S.; Moreno, G.; Stafford, J.; & Wallnau, K. *Packaging Predictable Assembly with Prediction-Enabled Component Technology* (CMU/SEI-2001-TR-024, ADA3399793). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tr024.html>.

[Hissam 02a] Hissam, S.; Hudak, J.; Ivers, J.; Klein, M.; Larsson, M.; Moreno, G.; Northrop, L.; Plakosh, D.; Stafford, J.; Wallnau, K.; & Wood, W. *Predictable Assembly of Substation Automation Systems: An Experiment Report, Second Edition* (CMU/SEI-2002-TR-031, ADA411970). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tr031.html>.

[Hissam 02b] Hissam, S. & Ivers, J. *PECT Infrastructure: A Rough Sketch* (CMU/SEI-2002-TN-033, ADA413548). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tn033.html>.

[Sun 01] Sun Microsystems. *Enterprise JavaBeans Specification, Version 2.0.* Palo Alto, CA: Sun Microsystems, August 14, 2001.

[Wallnau 03a] Wallnau, K. *Volume III: A Technology for Predictable Assembly from Certifiable Components* (CMU/SEI-2003-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr009.html>.

[Wallnau 03b] Wallnau, K. & Ivers, J. *Snapshot of CCL: A Language for Predictable Assembly* (CMU/SEI-2003-TN-025). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tn025.html>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
A Template for Documenting Prediction-Enabled Component Technologies	F19628-00-C-0003

6. AUTHOR(S)

Paulo Merson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TN-030

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Prediction-enabled component technology (PECT) is an approach to predicting the behavior of systems built from components with known properties. An important artifact produced by the PECT development process is the documentation of the technologies, tools, and theories as integral elements of the PECT, as well as the results and conclusions of the application of the PECT to a group of systems. This report suggests a template for documenting a PECT. The report also provides guidelines and a few examples to help PECT developers consolidate the broad range of information produced into the PECT development process in a single, organized volume.

14. SUBJECT TERMS	15. NUMBER OF PAGES
prediction-enabled component technology, PECT, predictable assembly from certifiable components, PACC, software development, software components	32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL