CMMI Process Area Compliance with Formal Specification Based Software Development

Satish Mishra
Institute für Informatik
Humboldt University, Berlin
mishra@first.fraunhofer.de

Bernd-Holger Schlingloff
Institute für Informatik
Humboldt University,Berlin
Fraunhofer FIRST, Berlin
holger.schlingloff@first.fraunhofer.de

Abstract

The development of reliable systems is still a major challenge for software industry. Construction of such a system requires both process and product based quality assurance. Many process improvement models have been suggested in industry and found appropriate for achieving high quality products. Examples of such process improvement models are CMM/CMMI, Agile, SPICE, ISO 9000 family etc. However, implementation of these process improvement models often adds significant extra efforts. To minimize process implementation costs we propose a formal specification based product development model which integrates product and process quality.[1]

Formal specification methods have been in practice since decades, and have been successful in the development of safety-critical systems. Some formal methods are VDM, Z, LOTOS, CSP and CASL. In particular, we investigate the compliance of CMMI process area with the formal specification language CSP-CASL. CMMI is based on the notion of process area, which is a cluster of best practices with particular goals in a certain area. For each of the relevant process areas, we show how formal specifications can contribute to achieve the specific goals of that process area. This integration is a new result for achieving process compliance parallel with product development. We demonstrate our approach with an industrial case study.

1. Introduction

Currently, computational systems (hard- and software) are incorporating more and more functionality, which leads to an ever increasing complexity. For the development of such a complex computational system which nevertheless is reliable both product and process based quality assurance methods are necessary. CMMI (Capability Maturity Model Integration) [11][2] is a well-established process improvement model which has proved its benefits in hundreds of companies and thousands of projects. CMMI is a collection of best practices that cover all aspects of product development from inception of requirement through delivery until maintenance of the final product in an organization[11][2][22].

Formal specifications have established significant benefits in product based software quality improvement. Most formal specification methods have been demonstrated for the development of safety-critical systems, e.g. in aerospace or in the medical industry. A formal specification language consists of a well-defined syntax and a mathematical semantics. A formal method based on a specification language includes some transformation algorithm, which makes it appropriate for specifying, verifying and validating systems. Examples for formal specification languages which are being used in industry include VDM, Z, Larch [4][10][14] or, more recently, CASL [17]. These methods focus on the specification of structural properties, whereas CSP [9][21], CCS, LOTOS, StateCharts, or Temporal Logic [10] focus on behavioral properties.

In this paper, we describe the application of formal specification methods for the compliance of CMMI process areas. In this approach, the benefits of a broadly accepted standardized process improvement model can be combined with the advantages of formal specifications for product assurance. Here, we use CSP-CASL, which is a rather new algebraic / process algebraic specification language [20]. This choice is based on the fact that, it includes means for both structural and behavioral properties. CASL (Common Algebraic Specification Language) is appropriate for specification of structural properties of system and CSP (Com-

[1] A preliminary version of this approach has appeared in the workshop CS&P 2007, with general concept of formal specification in the implementation of CMMI. Here in particular, CMMI process area compliance is demonstrated with our own contribution on formal specification features.

municating Sequential Process) has been in existence since decades, particularly well suited for the specification of behavioral properties. For the compliance of CMMI process area CSP-CASL is used as an example of specification language. Our approach is rather open to any specification language. We discuss the contribution of the formal specification for the satisfaction of a process area and achievement of its associated components. Figure 1 depicts features of the specification language which are appropriate for the process areas compliance and specification based product development.

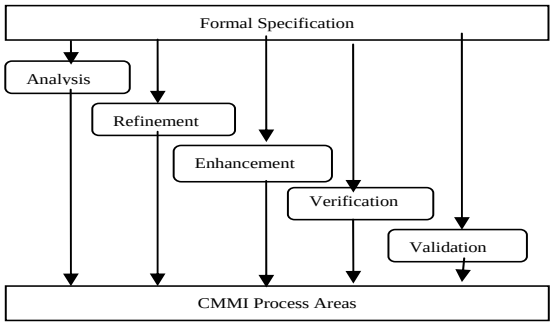

Figure 1. CMMI Process Area and Formal Specification.

In the next section, we give a brief overview of CMMI and subsequently in section three we present the required details of CSP-CASL. Section four describes a case study which will be referred to throughout the paper. Section five contains our main results about the contribution of formal specifications in the compliance of CMMI process areas. The last section contains some conclusions and suggestions for future work.

2. CMMI, Capability Maturity Model Integration

CMMI is a framework for assisting organizations to improve their development and maintenance processes for product development and maintenance[23]. CMMI is based on the notion of process area (PA). A process area is a cluster of related practices in an area. A typical process area is requirement management, which subsumes all practices necessary for dealing with demands which the software has to fulfill. When a process area is implemented, it satisfies several goals which are considered important for making improvements in that area. CMMI has 22 process areas which are considered important for the process improvement of an organization. CMMI offers two representations for its implementation, a continuous representation and a staged representation. The continuous representation offers more flexibility for process improvement. An organization can choose a focused process area, determine the dependent process areas, improve these at priority, and then concentrate on other process areas. In the staged representation, process areas are grouped together into capability maturity levels.

Since the notion of process area is independent from the representation, our results hold for any type of representation. In general, the domain of an organization can be divided into four groups: process management, project management, engineering and support. Each of these groups has a set of process areas for improving the capabilities of its processes. Associated with each process area is a set of goals which have to be satisfied as a measure for improvement in that process area.

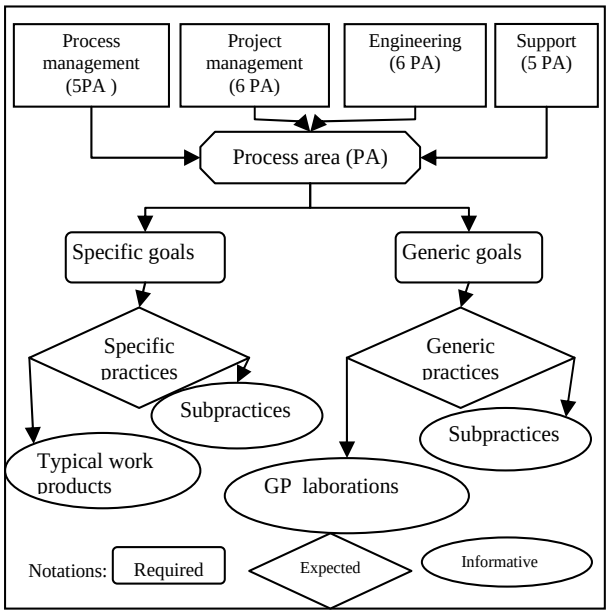

Figure 2. Details of Process Area and its components.

A process area is described by so-called model components. There are three types of model components; Required, Expected and Informative. Required model components describe what an organization must achieve to satisfy a process area. Expected model components describe what it may implement to achieve the required components. Informative model components provide details which help to initiate the approach followed by required and informative model component. Figure 2 shows groups of process areas and process area representation.

CMMI has a specific approach to describe the process areas. The description of a process area starts with introduction, purpose and relation with other process areas.

These are informative model components. Main characteristics of a process area are described by using specific goals (SG) and generic goals (GG). The specific goals are unique characteristics that must be present to satisfy the associated process area. A specific goal is a required model component. A specific practice (SP) is the description of an activity that is considered important in achieving the associated specific goal. A specific practice is an expected model component.

A generic goal is the required characteristics component to institutionalize the processes which implement a process area. Generic goals are called "generic" because the same goal applies to multiple process areas. A generic practice is the description of an activity that is considered important in achieving the associated generic goal. Thus, a generic practice is an expected model component. For analyzing the compliance of formal specification into process improvement models we only have to consider specific goals and its specific practices of the process areas. Generic goals are mainly for institutionalization and considered to be very subjective so compliance with formal specification does not make much sense.

3. CSP-CASL, Formal Specification Language

CSP-CASL is a specification language which combines two independent specification languages. The main advantage of this combined language is to provide a facility for describing data type development (CASL) in combination with description of processes (CSP). For the description of a system, processes are specified by CSP operators and communications between these processes are the values of CASL data types [20][9][21]. The basic syntax of CSP-CASL is an integration of the CSP syntax and the CASL syntax. Syntactically a CSP-CASL specification ccSp consists of a data part Sp, which is a CASL specification, an (optional) channel part Ch to declare channels, which are typed according to the data specification, and a process part P written in CSP, where CASL terms are used as communications. Thus, the generic format of a CSP-CASL specification is

$ccSpec\ ccSp = dataSpec\ Sp\ channel\ Ch\ process\ P$

The semantics of CSP-CASL is defined in three steps. In the first step each channel is encoded in CASL. In the second step CASL data types are evaluated, where families of processes are generated according to the data model. In the last step the evaluations are carried according to CSP operators. Detail of combined specification language and its components can be found in [20][1]. Here, we define features of a specification language required for the compliance of CMMI process areas.

3.1. Definition: Software Specification

We consider a practical approach to describe a software system with the help of the formal specification language CSP-CASL. A software system can be completely described by two types of behaviors; observational behavior and internal behavior. A description of the observational behavior is confined to a black box view of the system. Describing the internal behavior requires knowledge of design decisions inside the software. The desired properties are formulated in terms of the observational behavior, which is based on the internal behavior of the system. In this context we rewrite the syntax of CSP-CASL specification as follows

$ccSpec\ \ ccSp = dataSpec\ \ Sp_{Obs}\ \ And\ \ Sp_{Int}\ channel\ Ch\ process\ P\ end$

Here Sp_{Obs} and Sp_{Int} represent a CASL specification of the observational and internal behaviors of system, respectively. And is a CASL syntax construct which integrates observational and internal specification. Furthermore, the semantics of this definition is the same as mentioned in the preceding section. This representation allows us to deduce the concept of software enhancement, software refinement, verification and validation in a way which leads to CMMI process compliance in systematic manner.

3.2. Definition: Software Refinement

Software refinement is the process of preserving the correctness of program from abstract specification to detailed specification. Formally, based on CSP-CASL we define software refinement in a two step approach; data refinement and process refinement. A CSP-CASL specification (Spr, Spr_i, Pr) is the refinement of another specification (Sp, Sp_i, P) iff it satisfies the data refinement and process refinement conditions defined as follows.

Data Refinement

- $\Sigma(Sp) = \Sigma(Spr) \wedge \Sigma(Sp_i) \subseteq \Sigma(Spr_i)$
- $Mod(Spr) \subseteq Mod(Sp) \wedge Mod(Spr_i)_{|\Sigma(Sp_i)} \subseteq Mod(Sp_i)$

Here $Mod(Spr_i)_{|\Sigma(Sp_i)}$ represents the $Mod(Spr_i)$ restricted to the $Symbols(Sp_i)$; such that for all the symbols of $Symbols(Sp_i)$ there exist an injective mapping to the $Symbols(Spr_i)$ and forall $e \in Axioms(Sp_i) \bullet Model(Spr_i) \models e$.

Process enhancement For all $m \in Mod(Spr\ And\ Spr_i)$

- $Traces(Pr)_m \subseteq Traces(P)_m$
  ($Traces(P)_m$ is set of sequence of events of process P in perticular model m)

- Forall traces $t \in Traces(Pr) \bullet Failures(Pr/t) \subseteq Failures(P/t)$
  ($(Failures(Pe/t)$ is set of events after performing a trace t)

This definition of refinement is used to analyze stepwise development of system from abstract specification to implementation. Refinement is well suited to establish a traceability among development life cycle and verification of system.

3.3. Definition: Software Enhancement

Software enhancement is process of adding new observable features or functionalities to the existing product by semantically preserving its existing features and functionalities. A CSP-CASL specification (Spe, Spe_i, Pe) is the enhancement of another specification (Sp, Sp_i, P) iff it satisfies the data enhancement and process enhancement conditions defined below.
Data Enhancement

- $\Sigma(Sp) \subseteq \Sigma(Spe) \wedge \Sigma(Sp_i) \subseteq \Sigma(Spe_i)$
- $Mod(Sp) = Mod(Spe)_{|\Sigma(Sp)} \wedge Mod(Sp_i) = Mod(Spe_i)_{|\Sigma(Sp_i)}$

Semantics of $Mod(Spe)_{|\Sigma(Sp)}$ and $Mod(Spe_i)_{|\Sigma(Sp_i)}$ is similar as in above subsection.
Process enhancement For all $m \in Mod(Spr\ And\ Spr_i)$

- $Traces(P)_m \subset Traces(Pe)_m$
- Forall traces $t \in Traces(Pr)_m \bullet Failures(Pe/t)_m \subseteq Failures(P/t)_m$

This definition of enhancement is used to analyze the change request or upgrade in software specification. This definition helps to track the changes in the software system components for CMMI process area compliance.

3.4. Definition: Test Case and Test Verdict

Testing is practical approach to validate the correctness of the system. This requires set of data as input to the system and some way to interpret the resultant data from the system. We define a CSP-CASL based testing framework based on some established research [24][15][16]. For CSP-CASL specifications a test case TC is a trace of a CSP-CASL process. The test verdict from the specification (Sp, Sp_i, P) is based on the following definition
Pass: For all $m \in Mod(Sp\ And\ Sp_i) \bullet Traces(TC)_m \subseteq Traces(P)_m \wedge \forall tr : Traces(TC)_m \bullet tr =< t1, t2..tn > tr \notin Failures(P/ < t1, t2..tn - 1 >)_m$
Fail: For all $m \in Mod(Sp\ And\ Sp_i) \bullet Traces(TC)_m \subseteq Traces(P)_m$ and there exist $tr \in Traces(TC)_m \bullet tr =< t1, t2..tn > tr \in Failures(P/ < t1, t2..tn - 1 >)_m$
Inconclusive: not in the above two conditions
Based on the given definition we consider each trace of the system as a test case which leads us to setup validation environment for CMMI process area compliance.

4. Case study: Medical Embedded Device

We work with an industrial partner to apply this approach on a Medical Embedded Device (MED). This MED is human heart supporting device for controlling and monitoring the status of heart patient. This system is developed in such a way that patients can be monitored and controlled remotely. Since; patient's data is extremely sensitive information it has to be properly encrypted before sending and receiving through the system. This data communication protocol is common to all type of medical devices produced by this company. For handling the complexity and integrity of data we proposed to apply formal methods and CMMI to keep better control on the quality of the product.

4.1. Informal Description

The Communication Protocol is a small part of MED which we use here to visualize the approach of product based process quality. The Communication Protocol describes the protocol of data transmission between patient and server. Each communication has to be encrypted in such a way that it encapsulates identification number, acknowledgement and actual data. After receiving and sending the data protocol confirms the integrity by sending/receiving an acknowledgement. Formally, this requirement is given in further section.

4.2. Formal Requirement Specification of Communication Protocol

Given below in Figure 3 is the formal requirement specification of the Communication Protocol. In the first part the data part is described via the observational data structure which is further used as a part of the processes described in CSP.

Here the system is specified abstractly so it does not include any internal behavior. In the next section we give a refined specification of this version obtained by a systematic refinement process.

4.3. Formal Design specification of Communication Protocol

Following in Figure 4 is the refined specification of proposed case study. Here, some of the design decisions are fixed according to required property.

```
Spec CommunicationProtocol_AbstractSpec
Data
  Sort SendData, RecvData, RecdMsg
  Sort DevId, ComAck, RecdAck, EncData
  Ops EncryptMsg :  SendData x DevId x ComAck
  →EncrMsg
  DecryptMsg :  EncrMsg → RecdMsg
  ExtractAck  :  RecdMsg → ComAck
  ExtractDevId :  RecdMsg → DevId
  ExtractData  :  RecdMsg → RecvData
Channel
  ComCh : EncData; DataCh : Sort
Process
  SendControl(ComAck,DevId,SendData) = ComCh ?
   EncrMsg:{ EncryptMsg(ComAck,DevId,SendData)}
   → ComCh ! EncrMsg : {SendMsg()}→
   RecvControl() → IF      RecdAck=TRUE SKIP
        ELSE SendControl(ComAck,DevId,SendData)

  RecvControl() = ComCh? EncrData : {RecvData()}→
     DataCh ! RecdMsg : {DecryptMsg(EncrData)
    → ValidateRecdData(RecdData) → DataCh !
     RecdAck :{ExtractAck(RecdMsg) → DataCh !
     RecdDevId  :{ExtractDevId(RecdMsg)} → DataCh
     ! RecdData :  {ExtractData(RecdMsg)}
     IF RecdAck= TRUE SKIP ELSE  RecvControl()
End
```

Figure 3. CSP-CASL Specification of the Case Study.

```
Spec CommunicationProtocol_RefinedSpec
Data
            #  Integrate data part from requirement #
            Then Axioms
            Length(DeviceId)=8; Length(ComAck)=2
            Then
            Isorts GenAck, GenData
            IOps
            FormatAck   : ComAck x SendData → GenAck
            FormatData   : SendData → GenData
            Length(GenAck)=8
Channel
            ComCh : EncData;         DataCh : Sort
Process
   SendControl(ComAck,DevId,SendData) =  DataCh !
    GenAck : {FormatAck  (ComAck x SendData)} →
    DataCh ! GenData : { FormatData(SendData)} →
    ComCh ? EncrMsg :  { EncryptMsg(GenAck,DevId,GenData)}→
    ComCh !  EncrMsg : {SendMsg()}→ RecvControl() →
    IF   RecdAck=Ok SKIP ELSE
    SendControl(ComAck,DevId,SendData)

   RecvControl() = ComCh? EncrData : {RecvMsg()}→
     DataCh ! RecdMsg : {DecryptMsg(EncrData)
     → ValidateRecdData(RecdData) → DataCh !
     RecdAck :{ExtractAck(RecdMsg)}  → DataCh !
     RecdDevId : {ExtractDevId(RecdMsg)}
     → DataCh ! RecdData : {ExtractData(RecdMsg)}
     IF RecdAck=Ok SKIP ELSE  RecvControl()
End
```

Figure 4. CSP-CASL Refined specification of the Case Study.

At the initial steps, axioms are added to guarantee the length of acknowledgement and device ID. Further internal structures are added which guides the system to generate proper acknowledgement and data for the encryption. The external behavior is not changed with this new signature, only it helps to make the system more concrete. This internal function is called in process before processing the data into the encryption algorithm. Subsequently, we will refer to the above two specifications in order to show the compliance of specific goals and specific practices of CMMI process area.

5. CMMI Process Areas Compliance Evaluation with Formal Specification

To evaluate the Formal Specification based Contribution (FSC) in the compliance of the CMMI process area we have developed the following grading scheme.

Fully Contributed (FC):A process area is satisfied as FC if 90-100% of its specific goals are achieved using formal specification. A specific goal is achieved as FC when 90-100% of its specific practices can be performed by formal specification.

Largely Contributed (LC):A process area is satisfied as LC if 60-89% of its specific goals are achieved using formal specification. A specific goal is achieved as LC when 60-89% of its specific practices can be performed by formal specification.

Partially Contributed (PC):A process area is satisfied as PC if 30-59% of its specific goals are achieved using formal specification. A specific goal is achieved as PC when 30-59% of its specific practices can be performed by formal specification.

Not Contributed (NC):A process area is NC if less than 29% of its specific goals can be achieved using formal specification. A specific goal is NC when less than 29% of its specific practices can be performed by formal specification.

The main advantage of the formal specification starts with a precise and unambiguous description of the requirements. Formally specified requirements provide a basis for the software development lifecycle. Preciseness of specification can help to automate most of the software development lifecycle elements. We experiment with these features of formal specification in the development of case study and then lead towards process mapping of CMMI. We found that formal specification based development approach provide significant aspects in the satisfaction of CMMI process areas. After the experiment with this case study we have concluded that there are six process areas which can be satisfied up to great extend with formal specification.

This is good advantage of formal specification based development. If we consider compliance of CMMI process area we find many tools which are required for process area

compliance, for example project management tools, quality assurance, testing, debugging, time management, configuration management tools, etc. In most of these cases separate tools are required for the each process area. The formal specification based development approach is very unique where process areas can be satisfied parallel to the development of product. Following is a list of process areas which can be achieved with formal specification based development.

5.1. Requirement Management(RM)

The process area Requirements Management provides guidelines for addressing demands of product features and product component features. In addition to this, it also provides guidelines for removing inconsistencies between requirements and other work products. The contribution of formal specification for this process area and its components is shown in Table 1.

Table 1. RM process area and FSC

Specific goals and specific practices	FSC
SG 1 Manage Requirements	LC
SP 1.1 Obtain an Understanding of Requirements	FC
SP 1.2 Obtain Commitment to Requirements	LC
SP 1.3 Manage Requirements Changes	LC
SP 1.4 Maintain Bidirectional Traceability of	LC
SP 1.5 Identify Inconsistencies	LC

A precise and unambiguous semantics of formal specification based development is basis for the compliance of this process area. Once a requirement is formally specified and we follow the formal specification based development approach then all the model components of this process area can be easily achieved by specification and refinement features of formal specification. Table 2 shows selected part of the process from the case study. This formal specification demonstrates the refinement relation among requirement, design document and considered test case. This is simple example from specified formal specification of the case study. In the same manner our experiment has lead us to present the results in Table 1.

Table 2. Refinement relation

In requirement	EncrMsg → SendMsg→ RecvMsg→ CheckAck→ TRUE
In design	FormatAck→ GenData →EncrMsg →SendMsg →RecvMsg→ CheckAck →TRUE \ { FormatAck , GenData } (Hiding internal functions makes equivalent to requirement)
In test case	EncrMsg→ SendMsg→ RecvMsg→ CheckAck →TRUE

5.2. Product Integration(PI)

The process area Product Integration guides the integration of component's functions according to the requirements and the integration of components into a complete product. Contribution of formal specification into this process area, specific goals and the specific practices is shown in the Table 3.

Table 3. PI process area and FSC

Specific goals and specific practices	FSC
SG 1 Prepare for Product Integration	LC
SP 1.1 Determine Integration Sequence	LC
SP 1.2 Establish the Product Integration Environment	LC
SP 1.3 Establish Product Integration Procedures and Criteria	LC
SG 2 Ensure Interface Compatibility	PC
SP 2.1 Review Interface Completeness Descriptions	LC
SP 2.2 Manage Interfaces	LC
SG 3 Assemble Product Components and Deliver the Product	PC
SP 3.1 Confirm Readiness of Product Components for Integration	PC
SP 3.2 Assemble Product Components	PC
SP 3.3 Evaluate Assembled Product Components	PC
SP 3.4 Package and Deliver the Product and Component	PC

Formal specification has been proposed for component based development, e.g., in [6]. In particular, CSP-CASL provides significant features for component based development, such as giving a structural and architectural approach to requirements engineering [17]. In addition to this, the advantage of CSP-CASL for product line based development has been studied in [18]. Process algebra [12] has very powerful features for mastering the complexity of processes via parallel and sequential composition. An unambiguous definition of interfaces and its behaviors reduces the complexity of implementing in the system. Formal specification based development guarantees various quality aspects for the products which are composed of from many components.

5.3. Requirement Development(RD)

This process area describes customer requirements, product requirements and product component requirements. The process area compliance is presented in Table 4.

A formal specification based unambiguous and precise description of the system is appropriate for the compliance of SG 1 and SG 2. Compliance of SG 3 requires other features of formal specification like refinement, verification and validation. In Section 4 we present part of case study as requirement specification and design specification as refinement of requirement specification. The relation between these two specifications can be proved by our refinement approach. This type of property is only possible if the

Table 4. RD process area and FSC

Specific goals and specific practices	FSC
SG 1 Develop Customer Requirements	FC
SP 1.1 Elicit Needs	LC
SP 1.2 Develop the Customer Requirements	FC
SG 2 Develop Product Requirements	FC
SP 2.1 Establish Product and Product Component Requirements	FC
SP 2.2 Allocate Product Component Requirements	FC
SP 2.3 Identify Interface Requirements	LC
SG 3 Analyze and Validate Requirements	LC
SP 3.1 Establish Operational Concepts and Scenarios	LC
SP 3.2 Establish a Definition of Functionality	LC
SP 3.3 Analyze Requirements	LC
SP 3.4 Analyze Requirements to Achieve Balance	PC
SP 3.5 Validate Requirements	FC

initial requirement is stated in some formal language. We apply the same approach for the development of our case study. The process of stepwise refinement is continued until all design decisions are fixed. The refinement also establishes consistency between requirement, design and test cases. Formal specification has also shown significant benefits when generating frameworks for the validation and verification of products and product components.

5.4. Technical Solutions(TS)

This process area provides guidance for design, development and implementation of the given requirements. The main focus of this process area is to evaluate and select a solution, to develop a detailed design of the selected solution and to implement the design as a product or product component. Table 5 shows formal specification based scale of compliance for this process area.

Table 5. TS process area and FSC

Specific goals and specific practices	FSC
SG 1 Select Product Component Solutions	LC
SP 1.1 Develop Alternative Solutions and Selection Criteria	LC
SP 1.2 Select Product Component Solutions	LC
SG 2 Develop the Design	PC
SP 2.1 Design the Product or Product Component	LC
SP 2.2 Establish a Technical Data Package	PC
SP 2.3 Design Interfaces Using Criteria	PC
SP 2.4 Perform Make, Buy, or Reuse Analyses	PC
SG 3 Implement the Product Design	PC
SP 3.1 Implement the Design	PC
SP 3.2 Develop Product Support Documentation	PC

The specification language CSP-CASL is well suited for specifying industrial applications [1]. Stepwise refinement allows leading the requirement at a level where all the design decisions are fixed. This refined specification reaches very near to implementation and gives possibility to generate the implementation code. The whole approach shows that formal specification based development is well suited for the compliance of SG 1, SG 2, SG 3 and most of its specific practices. Below in Table 6, we show the aspect of refinement which is provable with our definitions given in Section 4.

Table 6. Refinement in SDLC

Requirement	Design	Implementation
Sort ComAck	ComAck = FormatAck(ComAck x SendData)	language based code

5.5. Validation

The purpose of the activities in this process area is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. The contribution of formal specification into this process is as follows in Table 7.

Table 7. Validation process area and FSC

Specific goals and specific practices	FSC
SG 1 Prepare for Validation	FC
SP 1.1 Select Products for Validation	LC
SP 1.2 Establish the Validation Environment	FC
SP 1.3 Establish Validation Procedures	FC
SG 2 Validate Product or Product Components	FC
SP 2.1 Perform Validation	FC
SP 2.2 Analyze Validation Results	LC

The formal specification based system development approach makes major contributions to this process area; since test case generation, evaluation and execution has been extensively experimented with formal specification. We have developed our own testing framework for CSP-CASL based test generation and execution [24][7][13]. Test case selection is based on the definitions given in the Subsection 3.4. In our consideration each trace acts like a test case which has to refine to be executable on the implementation. Steps of refinement should be similar refinement steps applied on specification. These are the basic consideration for our validation framework, this makes formal specification very appropriate for the compliance of SG 1 and SG 2.

5.6. Verification

The Verification process area ensures that the products which are the result of the processes under improvement meet their specified requirements. Formal specification based process compliance is shown in Table 8.

Formal specification methods have mainly two ways to contribute to this process area, model checking and theorem

Table 8. Verification process area and FSC

Specific goals and specific practices	FSC
SG 1 Prepare for Verification	LC
SP 1.1 Select Work Products for Verification	LC
SP 1.2 Establish the Verification Environment	LC
SP 1.3 Establish Verification Procedures	LC
SG 2 Perform Peer Reviews	NC
SP 2.1 Prepare for Peer Reviews	NC
SP 2.2 Conduct Peer Reviews	NC
SP 2.3 Analyze Peer Review Data	NC
SG 3 Verify Selected Work Products	PC
SP 3.1 Perform Verification	LC
SP 3.2 Analyze Verification Results	PC

proving. Model checking is the process of building a model of a system and checking whether desired properties hold in this model [5][25]. Theorem proving is the process of finding a proof of a property from the axioms of a system, where the property and the system are expressed in the formal specification language [15][8]. An enormous amount of work has been done in the case of verification and established significant presence in industry[19]. We claim formal specification is well suited for this process area.

6. Conclusions

In this paper, we have examined the compliance of CMMI process area with formal specification based development. A case study is presented to illustrate the applicability of a specification language for achieving goals of the process areas. Out of 22 process areas from CMMI, six process areas can be satisfied with a formal specification based development approach. We have also shown the possibility of automation in process compliance which subsequently reduces the effort for the implementation of a process model. In this research, we concentrated on CSP-CASL as a formal specification language. Although we believe this language to be particularly well-suited, most of our results hold for other specification formalisms as well. This approach is based on very generic features of specification languages, which gives flexibility for the specification language selection.

What is left open in this paper is the quantitative analysis of cost and benefits in practical examples and comparison with other formal specification language features for the process compliance. This will give us confidence to compare with UML (Unified Modeling Language) which has already been experimented for the process compliance and product development together.

References

[1] M. R. A. Gimblett and H. Schlingloff. Towards a formal specification of electronic payment systems in csp-casl. WADT, 2004.

[2] D. M. Ahern. Cmmi distilled a practical introduction to integrated process improvemen. Addison-Wesley.

[3] D. J. Anderson. Stretching agile to fit cmmi level 3 the story of creating msf for cmmi process improvement. Microsoft Corporation ADC05, 2005.

[4] Y. Cheon and G. T. Leavens. The larch/smalltalk interface specification language. ACM Transactions on Software Engineering, 1994.

[5] G. Craigen D. Formal methods in critical systems. IEEE Transactions on Software Engineering 11.

[6] P. F. Elsa Estevez. Algebraic specifications and refinement for component-based development using raise. JCTS, 2000.

[7] M.-C. Gaudel. Testing can be formal, too. LNCS 915, 1995.

[8] A. R. Henzinger. Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering 22.

[9] C. A. R. Hoare. Communicating sequential processes. commun. ACM 21(8):666–677, 1978.

[10] http://www.rbjones.com/rbjpub/cs/csfm02.htm. Formal specification languages.

[11] http://www.sei.cmu.edu/CMMI/. Software engineering institute, cmmi department. Carnegie Mellon, Pittusburgh, PA.

[12] A. P. "J. A. Bergstra and S. Smolka". Handbook of process algebra. Elsevier, 2001.

[13] A. B. Jan Tretmans. Automatic testing with formal methods. In Proceedings of the 7th European International Conference on Software Testing.

[14] D. Kreuz. Formal specification of corba services using object-z formal engineering methods. Proceedings of the Second IEEE International Conference on Formal Engineering Methods, 1998.

[15] P. R. J. M. C. Gaudel. Testing algebraic data types and processes: A unifying theory. Formal Aspect of Computing 10(5-6), 1998.

[16] P. D. L. Machado. Testing from structured algebraic specifications. LNCS 1816, 2000.

[17] P. D. M. Michel Bidoit. The common algebraic specification language users/reference manual. LNCS 2960, 2004.

[18] S. Mishra. Specification based software product line testing:a case study. Concurrency, Specification and Programming, 2006.

[19] Y. C. Nagoya F, Shaoying Liu. A tool and case study for specification-based program review. COMPSAC 2005,29th Annual International, Volume 1.

[20] M. Roggenbach. Csp-casl a new integration of process algebra and algebraic specification. Theoretical Computer Science, 2006.

[21] A. W. Roscoe. The theory and practices of concurrency. Prentice Hall, 1998.

[22] W. Royce. Software project management a unified framework. Addison Wesley.

[23] H. S. S. Mishra. Using formal specifications in the implementation of cmmi. 16th Int. Conf on Concurrency, Specification and Programming, Lagow, Poland, 2007.

[24] M. R. Temesghen Kahsai and B.-H. Schlingloff. Specification-based testing for refinement. SEFM, 2007.

[25] Y.Isobe and M.Roggenbach. A generic theorem prover of csprefinement. Proceedings of TACAS, 2005.