May 1

# SDN Engineering

2013

In the research paper "OpenFlow: Enabling Innovation in Campus Networks" Stanford has its roots in the Software Defined Networks epoch. This period of evolution in networking is marked by a software defined network management interface, opposing to the traditional approach of managing networks from a Command Line Interface where the device configuration remains relatively static. Networks can now be managed by customized software running on a controller device using the OpenFlow protocol. As SDN develops in the market place many commercial products are being brought to fruition. Products include UI software, vendor specific API and OpenFlow Networking Equipment. This paper covers engineering new Networking Applications as it relates to Capability Maturity Model Integration from Carnegie Mellon.

NCSU Software Engineering

## Table of Contents

## Introduction

In the research paper "OpenFlow: Enabling Innovation in Campus Networks" Stanford has its roots in the Software Defined Networks epoch. This period of evolution in networking is marked by a software defined network management interface, opposing to the traditional approach of managing networks from a Command Line Interface where the device configuration remains relatively static. Networks can now be managed by customized software running on a controller device using the OpenFlow protocol. As SDN develops in the market place many commercial products are being brought to fruition. Products include UI software, vendor specific API and OpenFlow Networking Equipment.

The following definition of SDN is used from the Open Networking Consortium; "Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow™ protocol is a foundational element for building SDN solutions. The SDN architecture is directly programmable, agile, centrally managed, programmatically configured, open standards-based and vendor-neutral." This growing trend fills a gap in the market place that traditional networking cannot; changing traffic patterns, "consumerization of IT", cloud services, and big data[1].

Current research in SDN stems from Stanford's research paper[Standford paper]. For comprehension research is split into the following categories; OpenFlow protocol, SDN Engineering, SDN Use Cases, and Software Engineering. OpenFlow is the protocol introduced

[1] https://www.opennetworking.org/

by Stanford University that enables communication between enabled Controllers and Networking devices. In succession there has been research conducted on Engineering of SDN applications. While sifting through related works it is found that majority of research in the area today revolves around SDN Use Cases.

In the SDN Engineering section CMMI-DEV paradigm is used to provide structure and a basis for engineering activities. These five process areas are Requirements Development, Technical Solutions, Product Integration, Verification, and Validation. As one goes through this section the CMMI-DEV Technical Report from SEI is a great reference[2]. A quick overview on CMMI is provided in Appendix, however it is recommended that one reads the following sections of the document: Using CMMI, Tying it Together, and Generic Goals and Practices.

As one goes through the document it becomes more apparent that the market is in a nascent stage. As the industry evolves it will drive research in new areas which will drive new scenarios and use cases. Big customers such as Service Providers, Big Data, Enterprise, and Federal customers will have final say in the success of emerging SDN technologies, but success in engineering and development efforts of SDN applications will reduce overheads; ultimately making the technology successful.

[2] http://cmmiinstitute.com/assets/reports/10tr033.pdf

## Industry Trends

While the Open Networking Foundation provides a definition of Software Defined Networks the definition still remains fluid. To understand this we must observe industry trends. In November 2007 Stanford introduced the inception of the idea with the whitepaper OpenFlow Standard v0.2[3]. Following on this idea “OpenFlow: Enabling Innovation in Campus Networks” was published in March 2008 which makes an argument for hardware vendors to open new features to the Academic community. Today, this idea has evolved into SDN where hardware functionality should be open to Software Developers utilizing it to drive innovation in networking.

The Stanford team declared that vendors should wait till the release of OpenFlow specification v1.0 before including support in their hardware. However, two OpenFlow products were released to market in September 2009 (an indication of possible markets for the technology), while v1.0 wasn’t released till December 2009. These two products are Open vSwitch[4] and the Toroki Lightswitch[5]. IBM entered the market in November 2011 with the OpenFlow enabled G8264 switch. The entrance of IBM is a significant step for SDN; signifying that a big company deems the technology worthy of commercializing.

---

[3] http://www.openflow.org/wp/
[4] http://openvswitch.org/- Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols.
[5] http://www.toroki.com/prd_toroki_ls4810.php - The LightSwitch 4810 is an OpenFlow enabled Ethernet switch. In addition to advanced L2 functions, LightSwitch 4810 supports the OpenFlow standard, enabling the deployment of user-defined policies in live production networks. The LightSwitch 4810 is useful for applications in data centers, for virtual machine mobility, high-security networks and next generation IP based mobile networks

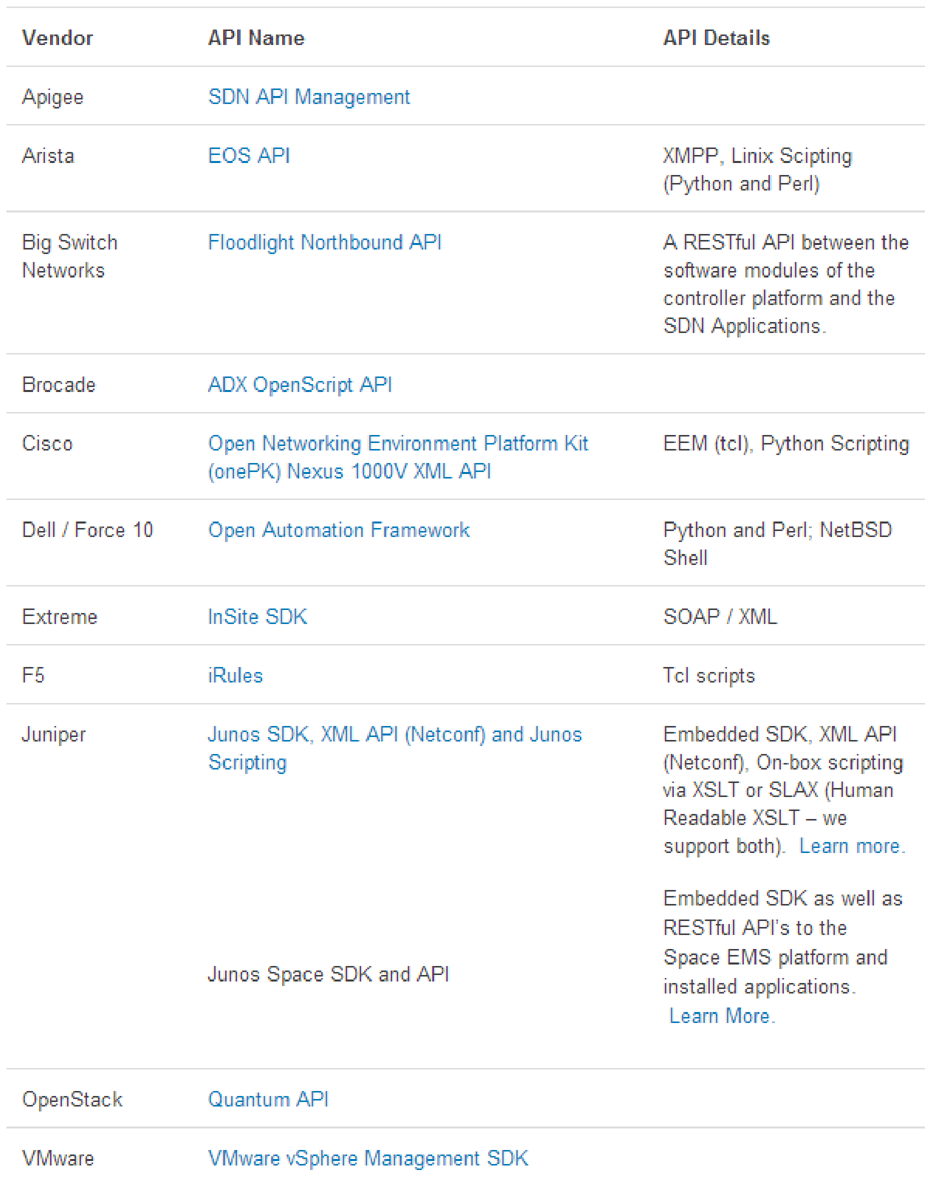
Support for OpenFlow in commercial vendor hardware has grown since. As of May 2013 the current list on the Open Networking Foundation site is as follows:

| | |
|---|---|
| 6WIND | 6WINDGate Networking Software |
| A10 Networks | AX Series |
| A10 Networks | AX Series |
| ADVA Optical Networking | FSP 3000 |
| ADVA Optical Networking | FSP 150 |
| Alcatel-Lucent / Nuage Networks | Virtualized Services Platform (VSP) |
| Alcatel-Lucent / Nuage Networks | Virtualized Services Platform (VSP) |
| Brocade | Brocade CER 2000 Series |
| Brocade | Brocade CES 2000 Series |
| Brocade | Brocade MLX Series |
| Brocade | Brocade ADX Series with Application Resource Broker (ARB) |
| Centec Networks (Suzhou) Co., Ltd. | GreatBelt (CTC5163) |
| Centec Networks (Suzhou) Co., Ltd. | V330 OpenFlow Switch Reference Design |
| Citrix | NetScaler for SDX |
| Cyan, Inc. | Blue Planet |
| Extreme Networks | BlackDiamond X8 |
| Extreme Networks | BlackDiamond 8000 Series |
| Extreme Networks | Summit X670 |
| HP | HP 5400 Switch Series |
| HP | HP 3800 Switch Series |
| HP | HP 3500 Switch Series |
| HP | HP 2920 Switch Series |
| IBM | IBM Programmable Network Controller |
| IBM | IBM RackSwitch™ G8264T |
| Infinera | DTN-X |
| Infoblox / FlowForwarding | LINC |

| | |
|---|---|
| Infoblox/ FlowForwarding | LINC |
| Intune Networks | iVX8000 |
| Lancope | StealthWatch System |
| Luxoft | Twister OpenFlow test automation framework |
| NoviFlow inc. | NoviWare 100 |
| NoviFlow Inc. | NoviKit 100 |
| Pica8, Inc. | Pica8 Open Switches |
| Plexxi | Plexxi Switch 1 |
| Plexxi | Plexxi Control |
| Qosmos | ixEngine |
| Radware | DefenseFlow |
| Tekelec | Diameter Signaling Router |
| Tekelec | Policy Server |
| Transmode | TM-Series |
| Turk Telekom / Argela | YakamOS |
| Turk Telekom / Argela | YakamOS |

While Cisco is not included in this list on April 9th they hosted a Webcast[6] in which they market onePK or One Platform Kit: "onePK is an easy-to-use toolkit for development, automation, rapid service creation and more. It enables you to access the valuable data inside your network via easy-to-use APIs.[7]" Cisco claims up to 700 APIs across their product lines. A current list of software support for OpenFlow and SDN is provided here.

[6] http://tools.cisco.com/gems/cust/customerQA.do?METHOD=E&LANGUAGE_ID=E&PRIORITY_CODE=&SEMINAR_CODE=S17939
[7] http://www.cisco.com/en/US/prod/iosswrel/onepk.html



| Vendor | API Name | API Details |
|---|---|---|
| Apigee | SDN API Management | |
| Arista | EOS API | XMPP, Linix Scipting (Python and Perl) |
| Big Switch Networks | Floodlight Northbound API | A RESTful API between the software modules of the controller platform and the SDN Applications. |
| Brocade | ADX OpenScript API | |
| Cisco | Open Networking Environment Platform Kit (onePK) Nexus 1000V XML API | EEM (tcl), Python Scripting |
| Dell / Force 10 | Open Automation Framework | Python and Perl; NetBSD Shell |
| Extreme | InSite SDK | SOAP / XML |
| F5 | iRules | Tcl scripts |
| Juniper | Junos SDK, XML API (Netconf) and Junos Scripting | Embedded SDK, XML API (Netconf), On-box scripting via XSLT or SLAX (Human Readable XSLT – we support both). Learn more. |
| | Junos Space SDK and API | Embedded SDK as well as RESTful API's to the Space EMS platform and installed applications. Learn More. |
| OpenStack | Quantum API | |
| VMware | VMware vSphere Management SDK | |

[8]

[8] http://www.sdncentral.com/comprehensive-list-of-sdn-apis/

## Literature Review

Industry trends indicate the field of Software Defined Networking to be in its early stages. While the idea was incepted in November 2007, the “Enabling Innovation in Campus Networks” publication [18] serves as the official whitepaper for OpenFlow. There have been many proposals to improve the OpenFlow protocol since. The SmartFlow paper proposes lightweight and low-level enhancements to improve OpenFlow [23]. The DevoFlow paper argues OpenFlow’s design imposes excessive overheads, and proposes a way to reduce unnecessary costs [6].

Since the introduction of OpenFlow the Software Defined Networks paradigm has been widely used and speculated. Cutting edge research on how to improve SDN engineering efforts has been proposed. Some research is focused around SDN controllers such as placement, scalability, and performance [9][25][27]. Other research is focused on improving overall SDN network design [5][11][29]. Three papers propose new description languages to improve SDN development efforts [7][16][34]. While description languages can relieve Software Development efforts there is additional research on SDN application development [2][8][20]. Once SDN applications have been developed it will need to be tested and verified. Some research focuses on these aspects of SDN engineering [1][15][36].

In addition to SDN engineering research several studies have been conducted with new and innovative approaches. Use cases include and aren’t limited to network management, monitoring, packet scheduling, mobile applications, equipment compatibility, MPLS, WLAN, security, and others. Two papers focus on performance [17][32]. Since the NetFPGA platform is based out of Stanford as well OpenFlow is implemented on a NetFPGA card [22]. OpenFlow can also support cloud based application ranging from mobile [10] to enterprise [3]. Two papers discuss WLAN use cases [31][33]. Use cases that prove useful to Service Providers include SDN routing [14] and MPLS [12][26]. Testing traffic [13], troubleshooting [35], and security [24][28] are important use cases in network management. Several other use cases are studied including multicasting [4][19][21][30].

## SDN Engineering with CMMI-DEV

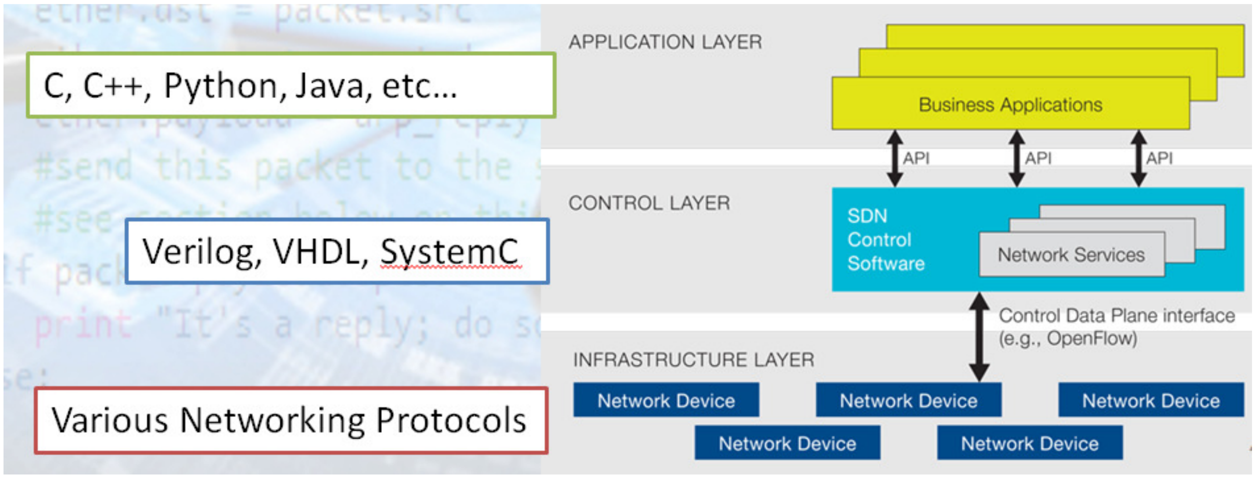
Most current research is focused around SDN use cases as seen in the Literature Review section. After understanding some of today's research it makes one wonder; how is this knowledge useful in real scenarios? While the information is not common knowledge, some generic real world uses cases include making improvements in: HyperScale and Enterprise Clouds, Enterprise Security, Enterprise Access Layer, Enterprise WAN, SP Mobile Access Layer, and SP Wired WAN. The Open Networking Foundation lists a few high level use cases[9]: operator network monetization, transforming private clouds, hybrid cloud services, and global provider data centers in core networks. Google's network is a great example of a production SDN; boasting close to 100% utilization in their networks with SDN Engineering and OpenFlow[10].

## SDN Advantages over Traditional Networking

These are all great examples, but it brings the question: what is wrong with existing networking infrastructure? Networking equipment has traditionally been configured via Command Line Interface and (SNMP) Simple Network Management Protocols. This results in static device configurations. Switching is based on Layer 2 learning and the Address Resolution Protocol. Enterprise networks use RIP, OSPF, and IGRP to manage routing tables. The internet relies on BGP and ISIS protocols. Service Providers use MPLS to take advantage of packet switching in hardware. These protocols are manually configured in a static configuration file. With the onset of OpenFlow and SDN applications can be made to replace static functionality. In

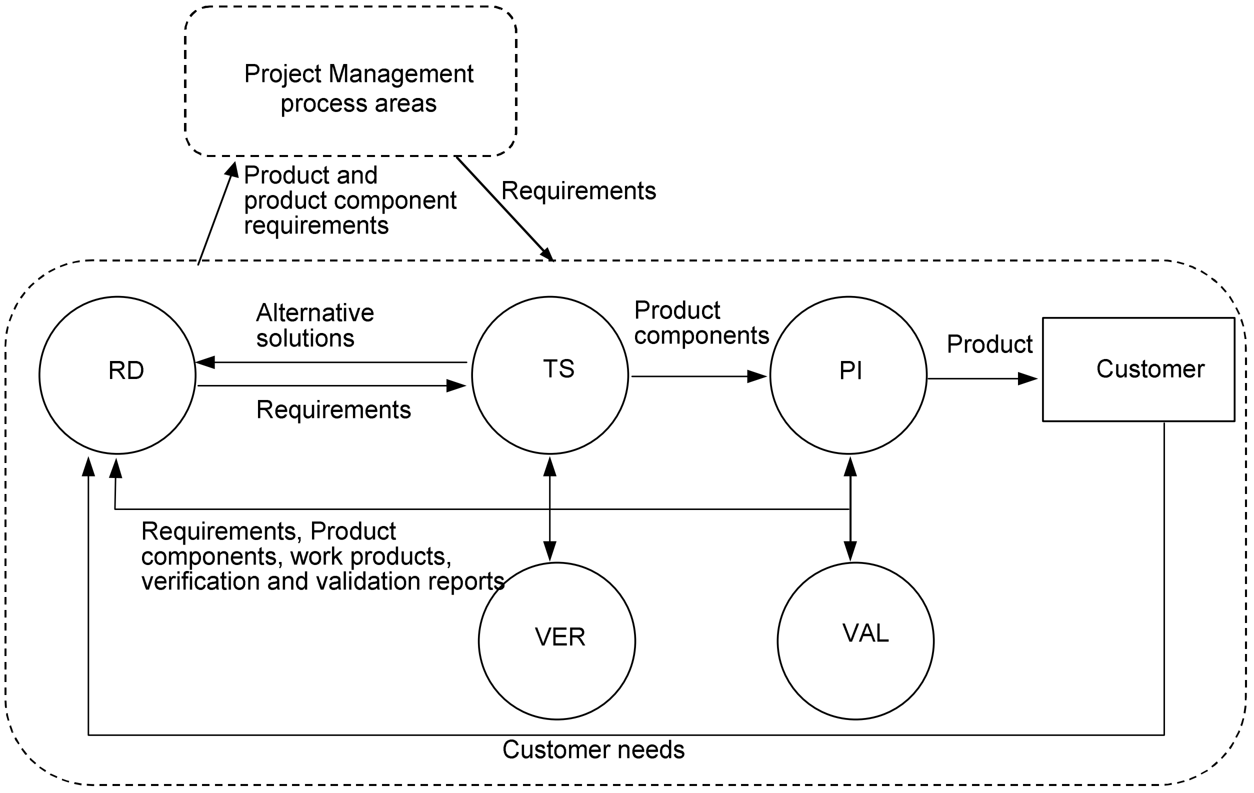
[9] https://www.opennetworking.org/sdn-resources/sdn-library/solution-briefs
[10] http://www.forbes.com/sites/eliseackerman/2012/04/18/google-unveils-secret-worldwide-networking-project/

SDN vendors create networking hardware that can be exposed to developers and network operators to better manage network resources. Hardware is written in low level languages such as Verilog, VHDL, and SystemC. This hardware can be exposed via API. Developers and network engineers alike can now write SDN applications in their favorite language while utilizing these APIs. The table below shows an abstract from ONF with some added technical details.



## SDN Engineering with CMMI

If this document is used to engineer SDN applications it is advisable to refer to the "Using CMMI" section of CMMI-DEV documentation. "Research has shown that the most powerful initial step to process improvement is to build organizational support through strong senior management sponsorship. To gain the sponsorship of senior management, it is often beneficial to expose them to the performance results experienced by others who have used CMMI to improve their processes [Gibson 2006]. For more information about CMMI performance results, see the SEI website at http://www.sei.cmu.edu/cmmi/research/results/." Once organizational support is established the CMMI team recommends making three selections to apply CMMI to the organization for process improvement: part of the organization, a model, and a representation. In this case one can use a Continuous representation when modeling the Engineering process areas and a Staged representation for the Engineering organization or entity. The Software Engineering Institute provides the following model for Engineering to be used in conjunction with the Project Management process areas.


Project Management process areas
Product and product component requirements
Requirements
Alternative solutions
Requirements
Product components
Product
RD
TS
PI
Customer
Requirements, Product components, work products, verification and validation reports
VER
VAL
Customer needs

PI = Product Integration
RD = Requirements Development
TS = Technical Solution
VAL = Validation
VER = Verification

Requirements development is the process of translating elicited customer/user stories into requirements that the Technical Solutions team can use to design and develop product components. Verification and validation can begin once the Product Integration process has finished.

## Requirements Development (RD)

When beginning the SDN Engineering process the Target Market must be considered. Are networks using the SDN product Service Providers, Big Data, Enterprise, and/or Federal customers? Each category requires a separate set of requirements. E.g. Service Providers will want maximum utilization and maximizing throughput of their networks. Big Data will require low latency to their SAN. Enterprise customers may want customized applications, while Federal customers will require secure FIPS compliant hardware.

### SG1: Develop Customer Requirements

*Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.*

***SP1.1: Elicit Needs:*** *Elicit stakeholder needs, expectations, constraints, and interfaces for all phases of the product lifecycle.*
***SP1.2: Transform Stakeholder Needs into Customer Requirements:*** *Transform stakeholder needs, expectations, constraints, and interfaces into customer requirements.*

In Software Defined Networking stakeholders' needs, expectations, constraints, and interfaces will generally be elicited in terminology that may or may not reflect an entity that is savvy in technical networking terms. Once elicited it is important to translate these needs, expectations, constraints, and interfaces into Customer Requirements in the form of features, functions, and user stories.

### SG2: Develop Product Requirements

*Customer requirements are refined and elaborated to develop product and product component requirements.*

***SP2.1: Establish Product and Product Component Requirements:*** *Establish and maintain product and product component requirements, which are based on the customer requirements.*
***SP2.2: Allocate Product Component Requirements:*** *Allocate the requirements for each product component.*
***SP2.3: Identify Interface Requirements:*** *Identify interface requirements.*

Once Customer Requirements are created in the form of features, functions, and user stories one can establish terminology that an SDN engineering entity understands. This entails Specific Practices 2.1, 2.2, and 2.3. An SDN engineering team can entail software and networking bodies of knowledge: Customer, Product Manager, Software Engineer/Project Leader, Software Architect, Programmers, Testers, User, Organization General Manager, Network Engineer/Operator, etc… A Context Diagram and the Volere Shell (example in Appendix) are handy documentation templates that can be used to convey technical requirements.

A generic list of **network requirements** are not limited to: topology, hardware, bandwidth, latency, redundancy, scalability, security, and QoS**. Other Requirements** can include: Functional, Performance, Interface, Operational, Verification, Validation, Project, Process, Environmental , Design Requirements / Constraints, Algorithm or Mathematical Computation Inputs, Acceptance Testing , Documentation, Personnel-Related, Training, Logistics-Related, Packaging, Precedence and Criticality**.** Some **Non-functional/Quality Requirements** are: Portability, Reliability, Maintainability , Expandability, Safety, Security, Look-and feel, Usability, Performance, Availability, Scalability, Maintainability and Portability, Security, Cultural and political, Legal, and Constraints/Scope.

## SG3: Analyze and Validate Requirements

*The requirements are analyzed and validated.*

***SP3.1: Establish Operational Concepts and Scenarios:*** *Establish and maintain operational concepts and associated scenarios.*
***SP3.2: Establish a Definition of Required Functionality and Quality Attributes:*** *Establish and maintain a definition of required functionality and quality attributes.*
***SP3.3: Analyze Requirements:*** *Analyze requirements to ensure that they are necessary and sufficient.*
***SP3.4: Analyze Requirements to Achieve Balance:*** *Analyze requirements to balance stakeholder needs and constraints.*
***SP3.5: Validate Requirements:*** *Validate requirements to ensure the resulting product will perform as intended in the end user's environment.*

Specific Practices 3.1 to 3.5 are performed to ensure quality and consistency in the RD process area. The list of Technical Requirements in the Volere Shell can be analyzed to establish operational concepts and scenarios. Some Operational Concepts and Scenarios to consider in a SDN include protocols used, traffic composition, proprietary solutions, configurations, device limitations, multi-vendor environments, bottlenecks, and high latency. Concepts and scenarios can also be software driven.

To analyze and validate the requirements one can perform a high level review and low-level test[11]. The **High Level Review** involves understanding customer needs, investigating existing Standards/Guidelines, and Reviewing/Testing similar SDN applications. Standards can include: terminology and conventions, industry requirements, government standards, GUI standards, hardware and networking standards. The **low-level test** entails checking Specifications and Requirements for certain attributes and looking for terminology that can raise some flags. A checklist of **attributes** includes: Completeness, Accuracy, Clarity and precision, Consistency, Relevancy, Feasibility, Code independence, and Testability. **Terminology** that can raise flags include: Always, every, all, none, never, certainly, therefore, clearly, obviously, evidently, some, sometimes, often, usually, ordinarily, customarily, most, mostly, good, fast, cheap, efficient, small, stable, handled, processed, rejected, skipped, if, then , and else.

[11] CSC510 Lecture Notes

## Technical Solution (TS)

The average Software Development effort is estimated to be around 40% with a breakdown of: Analysis and Design (16%), Implementation (8%), Testing (16%)[12]. While SDN Engineering also entails networking aspects the development efforts are similar. Something to consider in this process area are the vendors, product lines, topology, requirements, and SDN/ OpenFlow Support. Since SDN is in its nascent stages in a product/market lifecycle product offerings and support will largely vary across vendors. If products are purchased on a component basis one will have to ensure compatibility. If purchased on a product line or vendor basis customer requirements may have to be sacrificed.

### SG1: Select Product Component Solutions

*Product or product component solutions are selected from alternative solutions.*

***SP1.1: Develop Alternative Solutions and Selection Criteria:*** *Develop alternative solutions and selection criteria.*
***SP1.2: Select Product Component Solutions:*** *Select the product component solutions based on selection criteria.*

Computer networks are made of various types of networking equipment; firewalls, servers, switches, routers, Intrusion Detection Systems, Intrusion Prevention Systems, WLAN APs, Controllers, SAN, multiple hosts, and servers. Accessories such as cabling, media, and other add-ons will also vary. In addition, variety in vendors and product lines add to the complexity. IEEE, IETF, and other compliancy specifications can drive multi-vendor devices to be compatible. In separating these components we can determine configurations and SDN

[12] CSC510 Lecture Notes

software engineering efforts for each component. By defining solutions on a component level there are more granularities which will carry into the design.

## SG2: Develop the Design

*Product or product component designs are developed.*

***SP2.1: Design the Product or Product Component:*** *Develop a design for the product or product component.*
***SP2.2: Establish a Technical Data Package:*** *Establish and maintain a technical data package.*
***SP2.3: Design Interfaces Using Criteria:*** *Design product component interfaces using established criteria.*
***SP2.4: Perform Make, Buy, or Reuse Analyses:*** *Evaluate whether the product components should be developed, purchased, or reused based on established criteria.*

Once solutions are defined on a component level basis one can precede to designing the component. A list of solutions include: protocols, interface specs, bandwidth, traffic engineering, connectivity, remote applications, etc. These solutions translate into the design. The design will entail component level configurations, and equipment accessories/appendages. Design decisions can also entail using traditional networking configurations vs the SDN approach. Some SDN attributes to keep in mind: API / SDK support from the hardware vendor, the hardware performance capability and its flexibility to developers, controller functionality and support.

When designing at a higher level network abstractions have different functionality; Core, Distribution, Access, and Edge. Consider how Use cases and Technical Requirements translate to behavior in the network. For SDN consider flow table definitions (filter criteria, actions) and engineer logic of the SDN application. There are various guides for designing network topologies.[13] In addition each vendor can provide generic topologies for their product lines. Numerous network design and modeling software is available: OpNet, Java Modeling Tools, GNS3, MatLab.

[13] http://www.networkcomputing.com/netdesign/series.htm#netsys

Once the Design decisions have been made documentation can be consolidated into a Technical Data Package. The package should consist of the various other technologies used in the design of the SDN product. The typical TDP includes specifications, drawings, and a bill of characteristics. “All applicable technical data such as engineering drawings, associated lists, product and process specifications and standards, performance requirements, quality assurance provisions, and packaging details”[14] should be included here.

## SG3: Implement the Product Design

*Product components, and associated support documentation, are implemented from their designs.*

***SP3.1: Implement the Design:*** *Implement the designs of the product components.*
***SP3.2: Develop Product Support Documentation:*** *Develop and maintain the end-use documentation.*

After the design had been created the Engineering team must start the development lifecycle.

This will entail piecing the networking equipment together, configuration, and/or developing SDN software for the network. If the SDN route is chosen a development decision must be made on which lifecycle model should be used: Linear, Iterative, Incremental, Evolutionary (Spiral), Agile, Component assembly, Concurrent process, Formal, or Cleanroom. An important factor to remember is that the costs of defects and changes goes up throughout the development lifecycle. If Project Management areas have not been institutionalized the seven process areas must be considered: Integrated Project Management (IPM), Project Monitoring and Control (PMC), Project Planning (PP), Quantitative Project Management (QPM), Requirements Management

[14] http://www.expertglossary.com/technology/definition/technical-data-package-tdp

(REQM), Risk Management (RSKM), Supplier Agreement Management (SAM). A parallel Specific Practice is the documentation of all development efforts in the form of: Test Plans, Operating Procedures, QA Procedures, User Manuals, Installation Instructions, etc.

## Product Integration (PI)

On approaching the Product Integration process area the Technical Requirements have been translated to Technical Solutions at the Component level. These Technical Solutions have been used to design and develop each component in the network. Now one must engage in piecing the components together.

### SG1: Prepare for Product Integration

*Preparation for product integration is conducted.*

***SP1.1: Establish an Integration Strategy:*** *Establish and maintain a product integration strategy.*
***SP1.2: Establish the Product Integration Environment:*** *Establish and maintain the environment needed to support the integration of the product components.*
***SP1.3: Establish Product Integration Procedures and Criteria:*** *Establish and maintain procedures and criteria for integration of the product components.*

Protocols such as OSPF, BGP, MPLS, and many others require remote connectivity to be functional. The SDN application may also require remote functionality. To fulfill remote requirements the network must be integrated. An innovative Product Integration strategy is a key to rapidly deploying a network. Pitfalls include hostile environments; heavy networking equipment, remote locations, malfunctioning equipment, organization within a rack/datacenter, accessibility to device and people, etc. A good technique to use is establishing a base internal network with ICMP(ping) connectivity to other devices.

### *SG2: Ensure Interface Compatibility*

*The product component interfaces, both internal and external, are compatible.*

***SP2.1: Review Interface Descriptions for Completeness:*** *Review interface descriptions for coverage and completeness.*
***SP2.2: Manage Interfaces:*** *Manage internal and external interface definitions, designs, and changes for products and product components.*

Excluding Ethernet there is a long list of form factors which aren’t usually compatible between vendors. In addition some form factors are only compatible with certain types of cabling. Consult vendor manuals and configuration guides to ensure Interface compatibility.

## SG3: Assemble Product Components and Deliver the Product

*Verified product components are assembled and the integrated, verified, and validated product is delivered.*

***SP3.1: Confirm Readiness of Product Components for Integration:*** *Confirm, prior to assembly, that each product component required to assemble the product has been properly identified, behaves according to its description, and that the product component interfaces comply with the interface descriptions.*
***SP3.2: Assemble Product Components:*** *Assemble product components according to the product integration strategy and procedures.*
***SP3.3: Evaluate Assembled Product Components:*** *Evaluate assembled product components for interface compatibility.*
***SP3.4: Package and Deliver the Product or Product Component:*** *Package the assembled product or product component and deliver it to the customer.*

Depending on the agreement with the customer the product may be verified and validated before or after purchasing/delivery. The customer may require a Proof of Concept verification and validation before putting the network into production.

## Verification (VER)

Before deploying a network into production the SDN product will need to be demonstrated as a Proof of Concept and thoroughly tested/verified.

### SG1: Prepare for Verification

*Preparation for verification is conducted.*

***SP1.1: Select Work Products for Verification:*** *Select work products to be verified and verification methods to be used.*
***SP1.2: Establish the Verification Environment:*** *Establish and maintain the environment needed to support verification.*
***SP1.3: Establish Verification Procedures and Criteria:*** *Establish and maintain verification procedures and criteria for the selected work products.*

Verification can begin with testing for connectivity while more formal methods should include comparing measures with functional requirements. Hardware will need to be tested: Power On Self-Test, equipment software up to date. Line cards, management modules, Power Supply Units, Switch Fabrics, and interfaces should be up and running. Other nuances include traffic behavior, control/data traffic as expected, number of packets dropped, etc.

The separate testing strategy should be used for SDN software. The strategy should be included in the Test Plan which defines a type or a combination of types: unit, integration, functional, acceptance, regression, or beta testing. Determine if black-box or white-box testing will be used. Also determine if static or dynamic testing should be used.

### SG2: Perform Peer Reviews

*Peer reviews are performed on selected work products.*

***SP2.1: Prepare for Peer Reviews:*** *Prepare for peer reviews of selected work products.*
***SP2.2: Conduct Peer Reviews:*** *Conduct peer reviews of selected work products and identify issues resulting from these reviews.*
***SP2.3: Analyze Peer Review Data:*** *Analyze data about the preparation, conduct, and results of the peer reviews.*

If peer reviews are being conducted for verification assign roles: coordinator, reader, moderator, author, recorder, and inspector. Follow the process(in appendix) of planning, overview, preparation, inspection meeting, rework, and follow up.[15] The inspection meeting entails an introduction, review, solutions, and conclusion.

## SG3: Verify Selected Work Products

Selected work products are verified against their specified requirements.

***SP3.1: Perform Verification:*** *Perform verification on selected work products.*
***SP3.2: Analyze Verification Results:*** *Analyze results of all verification activities.*

Here one will follow the Test Plan developed in Specific Goal one.

[15] CSC510 Lecture Notes

## Validation (VAL)

Once the SDN engineered product has been verified it will be presented to the customer as outlined in Product Integration Specific Goal number three. The verified product will now need to be validated by the customer.

### SG1: Prepare for Validation

*Preparation for validation is conducted.*

***SP1.1: Select Products for Validation:*** *Select products and product components to be validated and validation methods to be used.*
***SP1.2: Establish the Validation Environment:*** *Establish and maintain the environment needed to support validation.*
***SP1.3: Establish Validation Procedures and Criteria:*** *Establish and maintain procedures and criteria for validation.*

In most cases the SDN product that has been designed and developed will be the product targeted for validation. The verified product will usually be in the customer's production environment before it is validated. The procedures and criteria may be defined by the customer; usually determined by satisfaction of requirements and use cases.

### SG2: Validate Product or Product Components

*The product or product components are validated to ensure they are suitable for use in their intended operating environment.*

***SP2.1: Perform Validation:*** *Perform validation on selected products and product components.*
***SP2.2: Analyze Validation Results:*** *Analyze results of validation activities.*

If the final product satisfies the requirements and functions as the customer's intended use case it will be validated.

## Conclusion and Future Research Work

In this paper the concept of Software Defined Networking has been introduced, followed by the inception of OpenFlow into industry, and current research trends. Upon covering these topics the paper integrates SDN Engineering into SEI's Capability Maturity Model Integration paradigm which entails the five CMMI-DEV engineering process areas.

Current SDN market is in its nascent stages with little documentation of SDN use cases in combination with software engineering paradigms. As the Literature Review section shows most current research is focused on use cases of SDN. As the SDN industry develops the service provider, big data, enterprise, and federal customers will ultimately have final say in the success and determining the direction of SDN. Success of SDN will drive new research in new use cases and scenarios. Efficiency in SDN engineering will lower overheads/costs and ultimately will lead to success of SDN technologies.

## Works Cited

[1] Al-Shaer, Ehab, and Saeed Al-Haj. "FlowChecker: configuration analysis and verification of federated openflow infrastructures." *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*. ACM, 2010.

[2] Anwer, Muhammad Bilal, et al. "Switchblade: a platform for rapid deployment of network protocols on programmable hardware." *ACM SIGCOMM Computer Communication Review* 40.4 (2010): 183-194.

[3] Benson, Theophilus, et al. "Cloudnaas: a cloud networking platform for enterprise applications." *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011.

[4] Bozakov, Zdravko, and Panagiotis Papadimitriou. "AutoSlice: automated and scalable slicing for software-defined networks." *Proceedings of the 2012 ACM conference on CoNEXT student workshop*. ACM, 2012.

[5] Casado, Martín, et al. "Fabric: a retrospective on evolving SDN." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[6] Curtis, Andrew R., et al. "DevoFlow: scaling flow management for high-performance networks." *SIGCOMM-Computer Communication Review* 41.4 (2011): 254.

[7] Foster, Nate, et al. "Frenetic: a high-level language for OpenFlow networks."*Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM, 2010.

[8] Hassas Yeganeh, Soheil, and Yashar Ganjali. "Kandoo: a framework for efficient and scalable offloading of control applications." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[9] Heller, Brandon, Rob Sherwood, and Nick McKeown. "The controller placement problem." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[10] Jain, Raj. "OpenADN: mobile apps on global clouds using software defined networking." *Proceedings of the third ACM workshop on Mobile cloud computing and services*. ACM, 2012.

[11] Jarschel, Michael, et al. "Modeling and performance evaluation of an OpenFlow architecture." *Proceedings of the 23rd International Teletraffic Congress*. ITCP, 2011.

[12] Kempf, James, et al. "Openflow mpls and the open source label switched router." *Proceedings of the 23rd International Teletraffic Congress*. ITCP, 2011.

[13] Khurshid, Ahmed, et al. "VeriFlow: verifying network-wide invariants in real time." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[14] Kotronis, Vasileios, Xenofontas Dimitropoulos, and Bernhard Ager. "Outsourcing the routing control logic: better internet routing based on SDN principles." *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012.

[15] Kuzniar, Maciej, et al. "A SOFT way for openflow switch interoperability testing." *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012.

[16] Liang, Junxue, Zhaowen Lin, and Yan Ma. "OF-NEDL: an openflow networking experiment description language based on XML." *Web Information Systems and Mining*. Springer Berlin Heidelberg, 2012. 686-697.

[17] Luo, Yan, et al. "Accelerating OpenFlow switching with network processors."*Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2009.

[18] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks."*ACM SIGCOMM Computer Communication Review* 38.2 (2008): 69-74.

[19] Mendonca, Marc, Katia Obraczka, and Thierry Turletti. "The case for software-defined networking in heterogeneous networked environments." *Proceedings of the 2012 ACM conference on CoNEXT student workshop*. ACM, 2012.

[20] MM, Othman Othman, and Koji Okamura. "Design and Implementation of Application Based Routing Using OpenFlow."

[21] Nakagawa, Yukihiro, Kazuki Hyoudou, and Takeshi Shimizu. "A management method of IP multicast in overlay networks using openflow." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[22] Naous, Jad, et al. "Implementing an OpenFlow switch on the NetFPGA platform." *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2008.

[23] Németh, Felicián, et al. "Towards SmartFlow: case studies on enhanced programmable forwarding in OpenFlow switches." *ACM SIGCOMM Computer Communication Review* 42.4 (2012): 85-86.

[24] Porras, Philip, et al. "A security enforcement kernel for OpenFlow networks."*Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[25] Rothenberg, Christian Esteve, et al. "Revisiting routing control platforms with the eyes and muscles of software-defined networking." *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012.

[26] Sharafat, Ali Reza, et al. "Mpls-te and mpls vpns with openflow." *ACM SIGCOMM Computer Communication Review*. Vol. 41. No. 4. ACM, 2011.

[27] Soliman, Mourad, et al. "Source routed forwarding with software defined control, considerations and implications." *Proceedings of the 2012 ACM conference on CoNEXT student workshop*. ACM, 2012.

[28] Stabler, Greg, et al. "Elastic IP and security groups implementation using OpenFlow." *Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date*. ACM, 2012.

[29] Stephens, Brent, et al. "PAST: scalable ethernet for data centers." *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012.

[30] Sukoco, Heru, and Koji Okamura. "Grouping packet scheduling for virtual networks by genetic algorithm." *Proceedings of the 6th International Conference on Future Internet Technologies*. ACM, 2011.

[31] Suresh, Lalith, et al. "Demo: programming enterprise WLANs with odin." *ACM SIGCOMM Computer Communication Review* 42.4 (2012): 279-280.

[32] Tanyingyong, Voravit, Markus Hidell, and Peter Sjödin. "Improving PC-based OpenFlow switching performance." *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2010.

[33] Vestin, Jonathan, et al. "CloudMAC: towards software defined WLANs." *ACM SIGMOBILE Mobile Computing and Communications Review* 16.4 (2013): 42-45.

[34] Voellmy, Andreas, and Paul Hudak. "Nettle: Taking the sting out of programming network routers." *Practical Aspects of Declarative Languages*. Springer Berlin Heidelberg, 2011. 235-249.

[35] Wundsam, Andreas, et al. "Ofrewind: enabling record and replay troubleshooting for networks." *USENIX ATC*. 2011.

[36] Zeng, Hongyi, et al. "Automatic test packet generation." *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012.

## Appendices

### 22 Process Areas

Process Management
Organizational Process Definition (OPD)
Organizational Process Focus (OPF)
Organizational Performance Management (OPM)
Organizational Process Performance (OPP)
Organizational Training (OT)

Management
Integrated Project Management (IPM)
Project Monitoring and Control (PMC)
Project Planning (PP)
Quantitative Project Management (QPM)
Requirements Management (REQM)
Risk Management (RSKM)
Supplier Agreement Management (SAM)

Engineering
Product Integration (PI)
Requirements Development (RD)
Technical Solution (TS)
Validation (VAL)
Verification (VER)

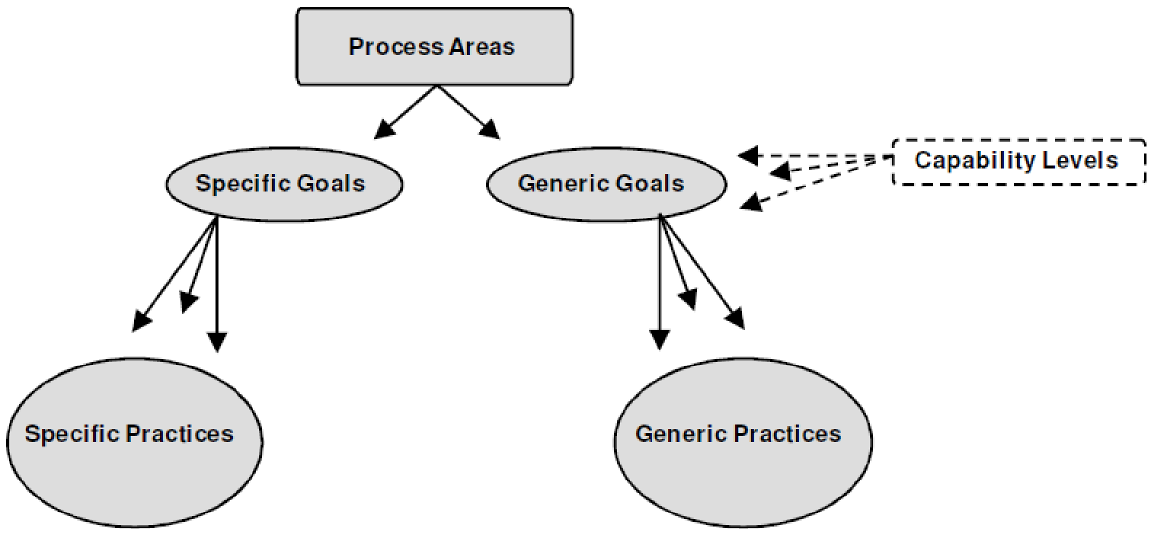
Support
Causal Analysis and Resolution (CAR)
Configuration Management (CM)
Decision Analysis and Resolution (DAR)
Measurement and Analysis (MA)
Process and Product Quality Assurance (PPQA)

## Continuous and Staged Representation

| *Level* | *Continuous Representation Capability Levels* | *Staged Representation Maturity Levels* |
|---|---|---|
| Level 0 | Incomplete | |
| Level 1 | Performed | Initial |
| Level 2 | Managed | Managed |
| Level 3 | Defined | Defined |
| Level 4 | | Quantitatively Managed |
| Level 5 | | Optimizing |

## Continuous Representation

Continuous representation uses capability levels to characterize the state of the organization's processes relative to an individual process area.



Capability Level 0: Incomplete

An *incomplete process* is a process that either is **not performed** or is **partially performed**. **One or more of the specific goals of the process area are not satisfied** and **no generic goals exist** for this level since there is no reason to institutionalize a partially performed process.

Capability Level 1: Performed

A *performed process* is a process that accomplishes the needed work to produce work products; the specific goals of the process area are satisfied. Although capability level 1 results in important improvements, those improvements can be lost over time if they are **not institutionalized**. The application of institutionalization (the CMMI generic practices at capability levels 2 and 3) helps to ensure that improvements are maintained.

Capability Level 2: Managed

A *managed process* is a performed process that is **planned and executed** in accordance with policy; employs skilled people having adequate resources to produce **controlled outputs**; involves relevant stakeholders; is **monitored, controlled, and reviewed**; and is **evaluated** for adherence to its process description. The process discipline reflected by capability level 2 helps to ensure that existing practices are retained during times of stress.

Capability Level 3: Defined

A *defined process* is a managed process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines; has a **maintained**

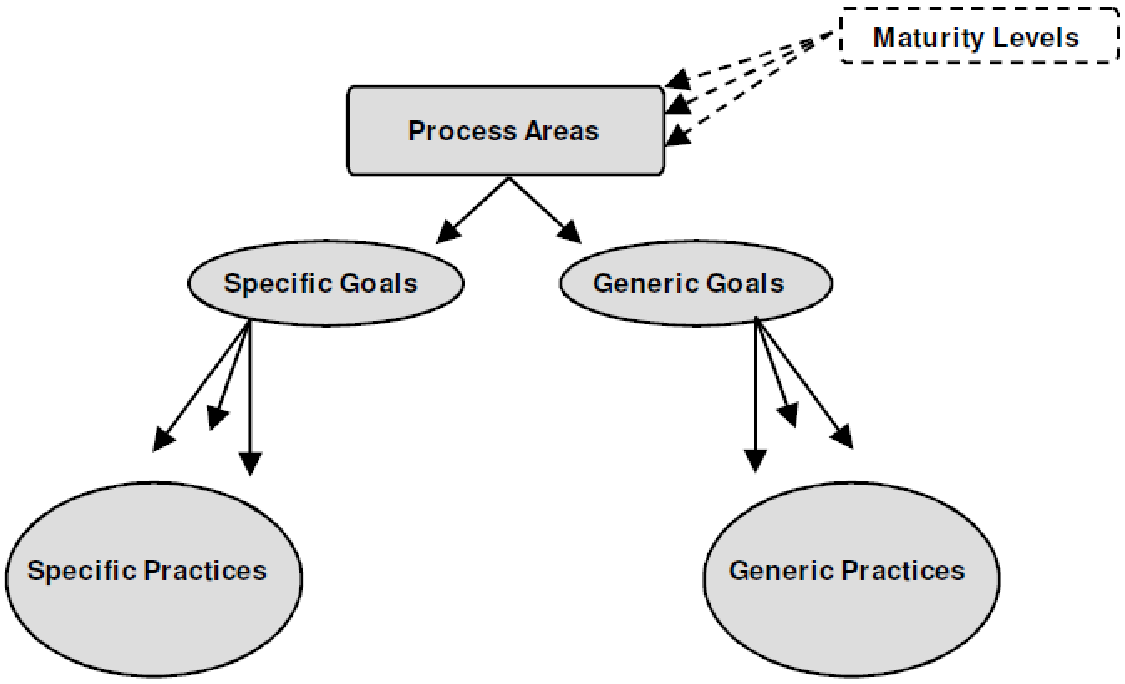
**process description**; and contributes **process related experiences** to the organizational process assets.

A critical distinction between capability levels 2 and 3 is the scope of standards, process descriptions, and procedures. At capability level 2, the standards, process descriptions, and procedures can be quite different in each specific instance of the process (e.g., on a particular project). At capability level 3, the standards, process descriptions, and procedures for a project are **tailored** from the organization's set of standard processes to **suit a particular project or organizational unit** and therefore are more consistent, except for the differences allowed by the tailoring guidelines.

Another critical distinction is that at capability level 3 processes are typically **described more rigorously** than at capability level 2. A defined process clearly **states** the purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria. At capability level 3, processes are **managed more proactively** using an understanding of the interrelationships of the process activities and detailed measures of the process and its work products.

## Staged Representation

Staged representation uses maturity levels to characterize the overall state of the organization's processes relative to the model as a whole.



<u>Maturity Level 1: Initial</u>

At maturity level 1, processes are usually **ad hoc and chaotic**. The organization usually does **not provide a stable** environment to support processes. Success in these organizations depends on the **competence and heroics of the people** in the organization and **not on the use of proven processes**. In spite of this chaos, maturity level 1 organizations often produce products and services that work, but they frequently **exceed the budget and schedule** documented in their plans. Maturity level 1 organizations are characterized by a **tendency to overcommit, abandon their processes** in a time of crisis, and be **unable to repeat** their successes.

Maturity Level 2: Managed

At maturity level 2, the projects have ensured that processes are planned and executed in **accordance with policy**; the projects employ **skilled people** who have **adequate resources** to produce **controlled outputs**; involve relevant stakeholders; are **monitored, controlled, and reviewed**; and are **evaluated** for adherence to their process descriptions. The process discipline reflected by maturity level 2 helps to ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans.

Also at maturity level 2, the **status of the work products are visible to management at defined points** (e.g., at major milestones, at the completion of major tasks). **Commitments are established** among relevant stakeholders and are revised as needed. Work products are appropriately **controlled**. The work products and services satisfy their specified process descriptions, standards, and procedures.

Maturity Level 3: Defined

At maturity level 3, processes are well **characterized and understood**, and are described in standards, procedures, tools, and methods. The organization’s set of standard processes, which is the basis for maturity level 3, is **established and improved over time**. These standard processes are used to establish **consistency across the organization**. Projects establish their defined processes by tailoring the organization’s set of standard processes according to tailoring guidelines. (See the definition of “organization’s set of standard processes” in the glossary.)

A critical distinction between maturity levels 2 and 3 is the **scope** of standards, process descriptions, and procedures. At maturity level 2, the standards, process descriptions, and procedures can be quite different in each specific instance of the process (e.g., on a particular project). At maturity level 3, the standards, process descriptions, and procedures for a project are **tailored** from the organization's set of standard processes to **suit a particular project or organizational unit** and therefore are **more consistent** except for the differences allowed by the tailoring guidelines.

Another critical distinction is that at maturity level 3, processes are typically described **more rigorously** than at maturity level 2. A defined process clearly **states** the purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria. At maturity level 3, processes are **managed more proactively** using an understanding of the interrelationships of process activities and detailed measures of the process, its work products, and its services.

At maturity level 3, the organization **further improves its processes** that are related to the maturity level 2 process areas. Generic practices associated with generic goal 3 that were not addressed at maturity level 2 are applied to achieve maturity level 3.

## Maturity Level 4: Quantitatively Managed

At maturity level 4, the organization and projects establish **quantitative objectives** for quality and process performance and use them as criteria in managing projects. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers. Quality and process performance is understood in statistical terms and is managed throughout the life of projects.

For selected subprocesses, **specific measures of process performance** are collected and statistically analyzed. When selecting subprocesses for analyses, it is critical to understand the relationships between different subprocesses and their impact on achieving the objectives for quality and process performance. Such an approach helps to ensure that subprocess monitoring using statistical and other quantitative techniques is applied to where it has the most overall value to the business. Process performance baselines and models can be used to help set quality and process performance objectives that help achieve business objectives.

A critical distinction between maturity levels 3 and 4 is the **predictability of process performance**. At maturity level 4, the performance of projects and selected subprocesses is controlled using statistical and other quantitative techniques, and predictions are based, in part, on a statistical analysis of fine-grained process data.

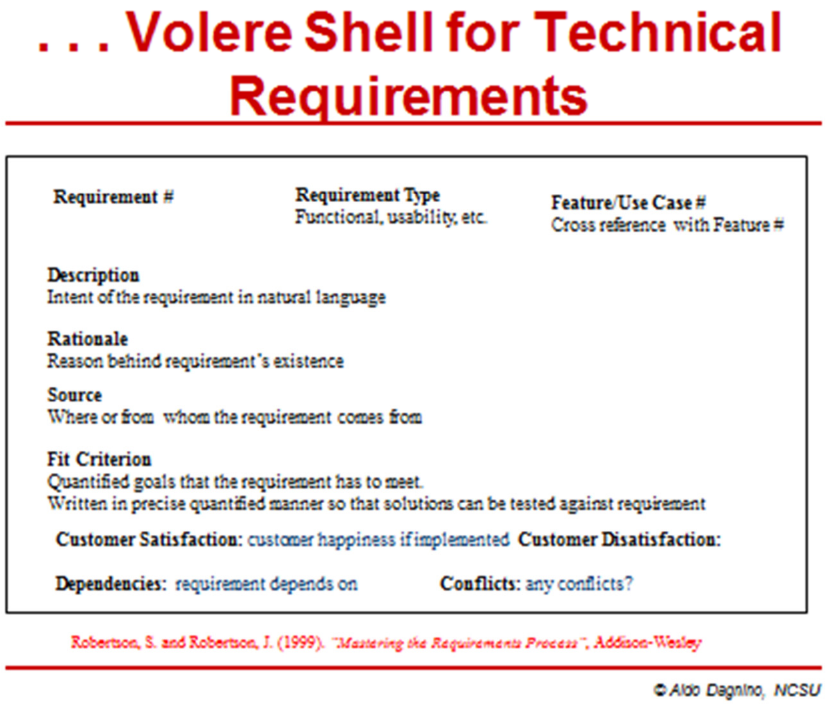
Maturity Level 5: Optimizing

At maturity level 5, an organization **continually improves its processes based on a quantitative understanding** of its business objectives and performance needs. The organization uses a **quantitative approach** to understand the variation inherent in the process and the causes of process outcomes.

Maturity level 5 focuses on **continually improving process performance** through incremental and innovative process and technological improvements. The organization's quality and process performance **objectives are established**, **continually revised** to reflect changing business objectives and organizational performance, and used as **criteria in managing process improvement**. The effects of deployed process improvements are measured using **statistical and other quantitative techniques** and compared to quality and process performance objectives. The project's defined processes, the organization's set of standard processes, and supporting technology are targets of measurable improvement activities.

A critical distinction between maturity levels 4 and 5 is the **focus on managing and improving organizational performance**. At maturity level 4, the organization and projects focus on understanding and controlling performance at the subprocess level and using the results to manage projects. At maturity level 5, the organization is concerned with **overall organizational performance using data collected from multiple projects**. **Analysis** of the data identifies shortfalls or gaps in performance. These gaps are used to drive organizational process improvement that generates measureable improvement in performance.

## Volere Shell



### . . . Volere Shell for Technical Requirements

**Requirement #**

**Requirement Type**
Functional, usability, etc.

**Feature/Use Case #**
Cross reference with Feature #

**Description**
Intent of the requirement in natural language

**Rationale**
Reason behind requirement's existence

**Source**
Where or from whom the requirement comes from

**Fit Criterion**
Quantified goals that the requirement has to meet.
Written in precise quantified manner so that solutions can be tested against requirement

**Customer Satisfaction:** customer happiness if implemented **Customer Disatisfaction:**

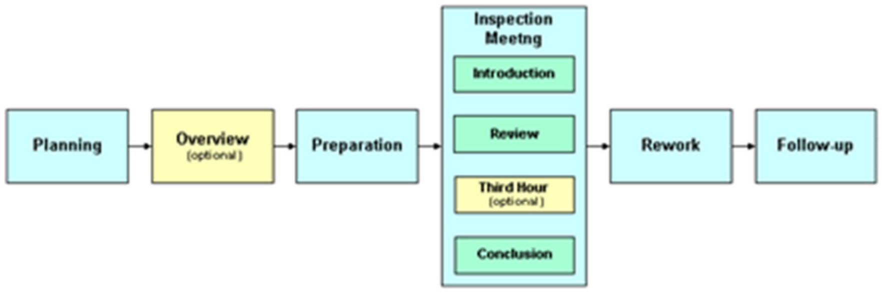
**Dependencies:** requirement depends on **Conflicts:** any conflicts?

Robertson, S. and Robertson, J. (1999). *"Mastering the Requirements Process"*, Addison-Wesley

© Aldo Dagnino, NCSU

## Peer Review Process

### Process



© Aldo Dagnino, NCSU